CS608 Lecture Notes

Visual Basic.NET Programming

OOP – Business Objects/System Architecture and Client/Server Concepts

Part (I)

(Lecture Notes 4A)

Professor: A. Rodriguez

Chapter 1 Introduction to Business Objects

1.1 Review of a General Object

1.1.1 Regular Objects (Revisited)

- Objects allow programs to be based on real world entities, such as a car, person, employee, customer, inventory part etc.
- The mechanism VB provides to implement Objects is the ***Class Module***.
- A ***Class Module*** is a plan or template that specifies what ***Data***, ***Methods*** and ***Events*** will reside inside the ***objects*** that are created from the ***Class***.
- Combining Data & the Methods that operate on such Data into a single package is a feature known as **Data *Encapsulation*** or ***Data hiding***

1.1.2 Object's Data, Behavior & Interface

Object's Data

- *Objects* should completely encapsulate or protect it's data and contain all the code required to manipulate that data (*Let/Get/Methods*)
- The ***Data***, are the **Private variables** that are declared in the variable section of the ***Class Module***.
- ***Objects*** handle themselves and have a life of their own, thus **behaving** like real world objects.

Object's Behavior

- Since we control via the ***Methods*** what we can do to an Object, An Object can only do or **behave** as per what the Methods dictates or allows the Object to do.
- Therefore, the ***Behavior*** of an object is the ***services*** that the object provides to client programs via its **public Methods**. For example, if we create a Customer Object, and we include methods to shop, rent, charge to credit, travel etc. Then shopping, renting and charging to a credit card are all behavior of a Customer.

Object's Interface

- Objects need to provide an ***Interface*** to allow client programs to access the ***Behavior*** or services.
- The Interface is composed of the Object's:
  - **Properties:** (Let & Get)
  - **Methods:** (*Sub Procedures*, *Functions*
  - **Events:** *Event-Handlers*

- Therefore the *interfaces* of an Object are the Properties & Methods of an Object which allow the user to interface with it.
- Characteristics of the Interface are:
  - ***Relationships between Objects*** – Relationship between communicating with other Objects.
  - ***Multiple Interfaces*** – Object may contain multiple interface or methods to allow it to have different behavior.

Interface – Relationships or Object to Object communication

- Objects can make references or communicate with other Objects.
- In other words and Object may need information from another Object so it may do the following:
  - Call the methods of another Object:
    ex. **MyObject.CustomerObject.PrintCustomer()**
  - Call Methods which take other Objects as Parameter Arguments:
    ex. **MyObject.PrintCustomer(CustomerObject)**
  - The Object may contain data members which are Objects:
    ex. *(The following declaration is being made from inside the clsMyObject class*)
    Private CustomerObject As clsCustomer

- This Object-to-Object dependency is known as the Objects ***Relationships***. Therefore in our Objects, we may need to create *Properties* & *Methods* that allow Objects to interact with each other.
- This Object communication is part of the Interface.

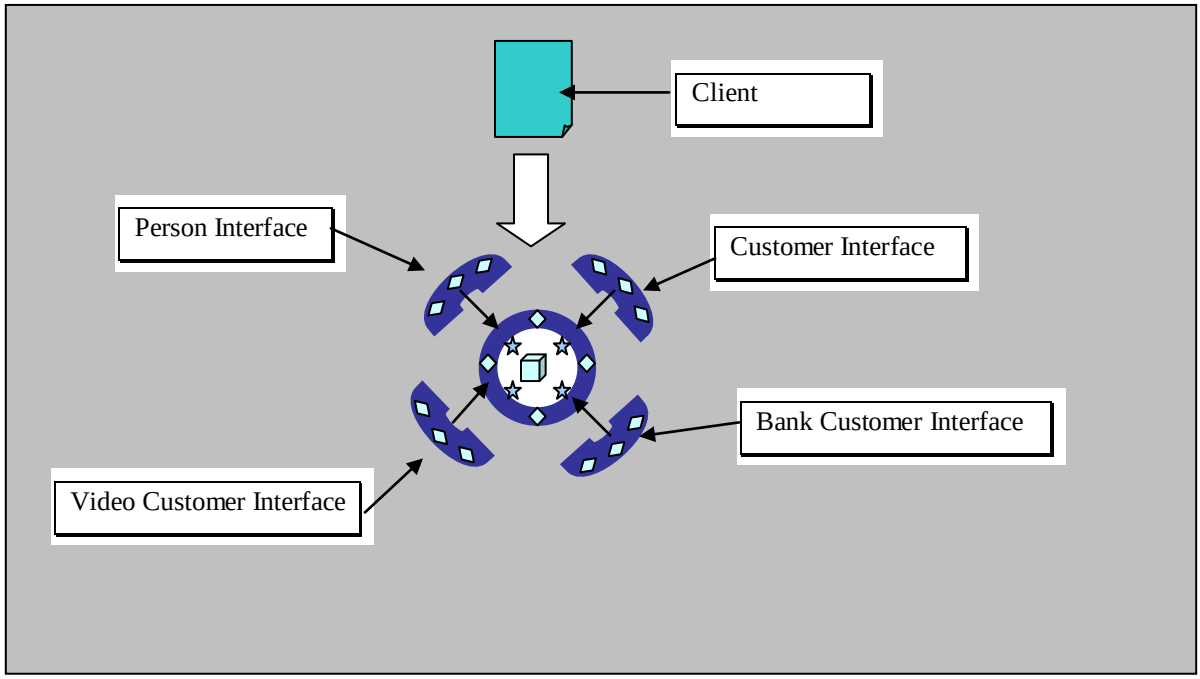
Interface - Multiple Interface

- Sometimes one Interface is not enough.
- Objects may represent more than one real world entity.
- For instance, a ***Customer Object*** may not only be a store customer but a ***Person*** Object as well. Or a Customer Object may have methods for it to be a Video Store Customer & methods for it to be a Bank Customer, thus multiple interfaces.
- Therefore a *Customer Object* may need to act as a *Person Object* from time to time, a Video Store Customer, and a Bank Customer.
- In this case, we create the Customer Object with a set of *Methods* & *Properties* with make up its ***Customer Interface***.
- Add a set of *Methods* & *Properties* with make up its ***Video Customer Interface***
- Add a set of *Methods* & *Properties* with make up its ***Bank Customer Interface***
- Finally, we add to the Customer Object a set of *Methods* & *Properties* with make up its ***Person Interface***.

1.1.3 Summary

- The key properties of an objects are:
  - **Data:** The private variables
  - **Interface:** Property Let & Get/Methods, Event-handlers
    - Characteristics of the Interface:
      - *Relationships between Objects*
      - *Multiple Interfaces.*
  - **Behavior:** The services provided by the object's methods or what it can do.

- In summary, we can see that Objects have three key items: 1) **Data**, 2) an **interface** & 3) **behavior**:



1.2 Introduction Business Objects

1.2.1 What are Business Objects?

- OK, now that we understand the theory of classes and Objects, let understand Business Objects.
- ***Business Objects*** are Objects, which represent **real world business concepts**.
- ***Business Objects*** are Objects that model and simulate the real world and business processes.
- ***Business Objects*** have the following characteristics:

1. They manage their own *data* & *database access*.
2. They contain all the ***Data***, ***Methods*** and ***Events*** needed to model their real world counterpart.
3. Can **evolve** & **gain new data, properties & methods to support more functionality**.
4. When they **evolve**, the **previous interface remains the same so that legacy (older) applications need not be changed and accommodate the new upgrade application**. **In other words the old and newly upgraded application can both be supported.**

Why are Business Objects Important?

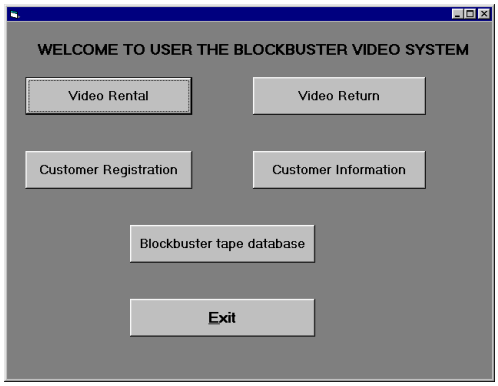
- Business Objects are important, because using them yields better software development, since they model business process.
- And most important: **Code reuse & Scalability!**

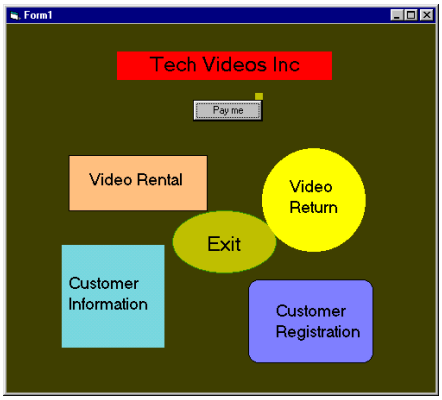
Chapter 2 Business Objects & Client/Server Architecture

2.1 Application Architecture for Business Objects

2.1.1 Traditional Visual Basic Applications

- Traditional Applications in VB is usually ***Event-Driven*** application.
- These are the applications written in CS101 & CS508
- The applications use ***Forms*** and code is written in Event-Handlers to respond to events caused by the user interactions with the Form & its graphical ***Controls***.
- In traditional VB applications, all the code and processing lived primarily in the User Interface or UI, inside Event-Handlers.
- In traditional VB application, programmers put most of their code behind the ***Controls*** on their ***Forms***. Also ***Standard Modules*** are used to contain general code that are called from the ***Forms***. Nevertheless, processing is entirely built around the GUI or ***User Interface*** (UI).
- In some cases, such as the material so far covered as review in this course, you went a bit further than traditional programming as follows:
  - Added class ***Objects*** to represent real world entities.
  - You then created standard modules where the bulk of the code was located.
  - In most cases the methods in the modules were called from the forms.
  - Although our design was an improvement over traditional programming, most of the ***processing was still built around the UI***.



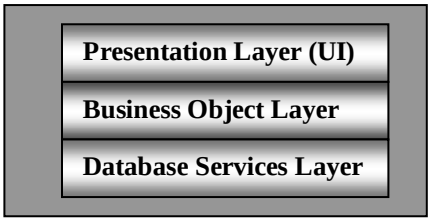


- Through the use of Data Access Technologies traditional applications enable us to interact with ***databases*** and other ***data sources*** such as files etc. But again the ***processing was still built around the UI***.
- Finally, traditional VB programming architecture does not take into consideration by default ***Distributed*** or **Network Programming**.
- What we are saying is that a program written based on traditional VB programming models are not very scalable. It's true that Object improves the program, but overall the program code is based on the User Interface thus not scalable.

2.1.2 Component-Based Scalable Logical Architecture (CSLA)

The Basic CSLA Architecture

- Last course you were introduced to the ***CSLA*** architecture.
- This is a development architecture used to develop powerful ***component-based applications***.
- CSLA is design to work well whether it's placed entirely on a single workstation or whether it is spread across a number of machines on a ***network***.
- This means that this architecture is ***highly scalable***.
- This logical architecture break the application development process into three layers or tiers:

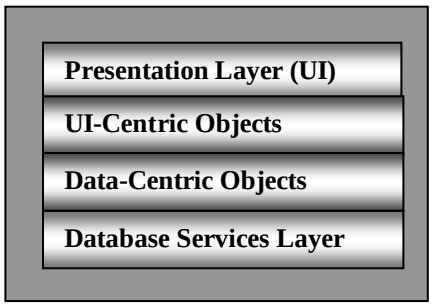


  - **Presentation or User-Interface Layer: -** Front-end or client that users will interact with. It can be Forms, Web Forms or any type of Graphical Inteface
  - **Business Object Layer: -** *ActiveX DLL* or *EXE* components that will contain Objects of the application. All the business logic or rules should be in this layer.
  - **Database Service Layer: -** Database Management system that will actually store the data. This layer actually represents the database itself such as Oracle, MS SQL and MS Access

- Note that the ***presentation layer*** or **UI** acts as a client to the ***Business Layer***.
- The ***Business Objects Layer*** relies on the ***Data Services layer*** to manage the data processing, storing, retrieving and manipulating data from the database.

An n-Tier Architecture

- We can further enhance this architecture by splitting the ***Business Object Layer*** and dividing into Business Objects that are more related to the ***Presentation Layer (UI)*** and those who are more related to the ***Data Services Layer*** as follows:

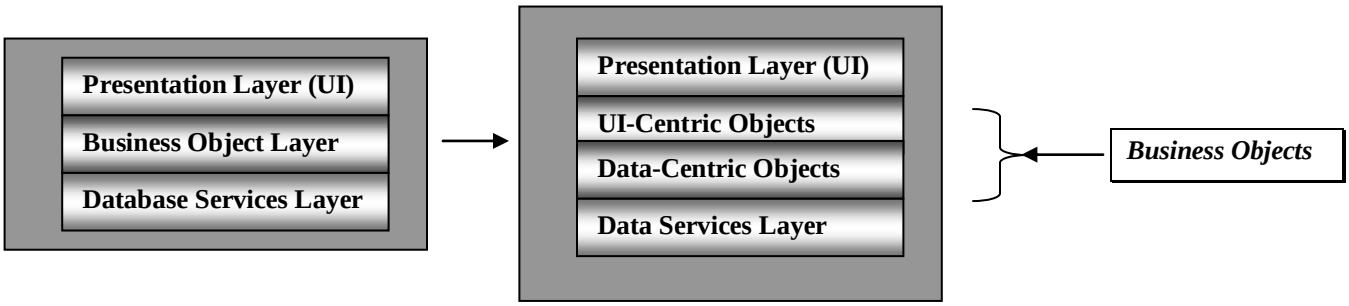


  - ***UI-Centric Business Object Layer: -*** components or objects that work more closely with the UI.
  - ***Data -Centric Business Objects Layer: -*** components or Objects that work more closely with accessing & storing the data to the database system.

- The idea behind splitting the business processing is to keep processing as physically close to where is needed. That means processing needed by the ***Presentation Layer*** should be physically closer to that layer. Therefore the ***UI-Service Business Objects Layer*** should be closer to the ***Presentation Layer***. In addition, processing that works mostly with data should be physically closer to the ***Database Service Layer***, therefore the ***Data-Service Business Objects Layer*** should be next to the ***Database Service Layer***.

The Complete CSLA Architecture

- The ***CSLA*** architecture is ***highly scalable***, and can be used to develop powerful ***component-based applications***, which is designed to work well whether it's placed entirely on a *single workstation* or whether it is spread across a number of *machines* on a ***network***
- The architecture consisted of three layers or the much more scalable four layer architecture:



- Using the CSLA Architecture we can develop scalable application suitable for a ***Distributed*** or **Network** environment.
- What is Distributed Application:

  - ***Distribute Application*** **-** The distribution of applications, business logic & resources across multiple processing platforms.
    - Distributed processing implies that processing will occur on more than one processor in order for a transaction to be completed. In other words, processing is distributed across two or more machines and the processes are most likely not running at the same time, i.e. each process performs part of an application in a sequence.
    - Often the data used in a distributed processing environment is also distributed across platforms.

2.2 Business Objects & Client/Server Technology

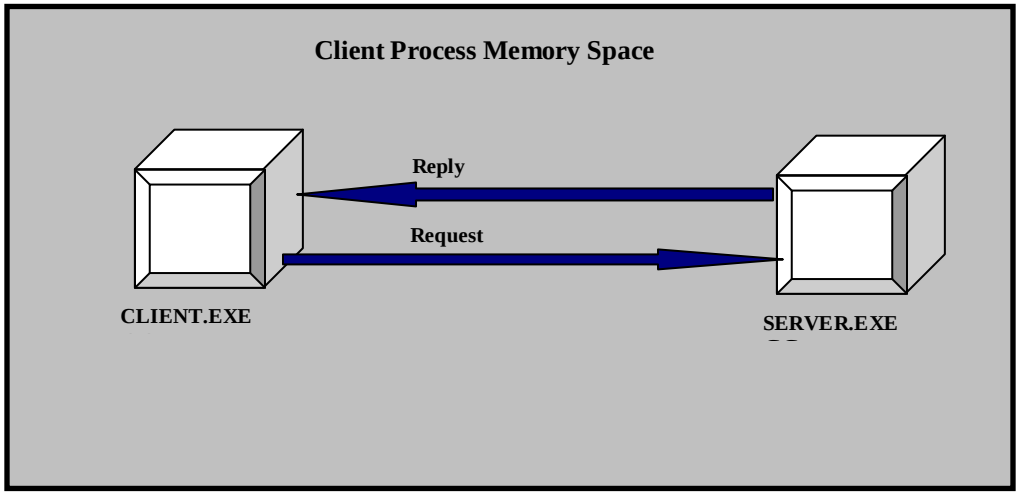
2.2.1 Client/Server Architecture

Introduction

- Client/Server Technology is the standard that has been used when creating network applications and systems.
- The Client/Server definition is as follows:

  - ***Client*** **–** Process or executable program that makes requests to another process (executable)
  - ***Server*** **–** Process or executable program that services or complies with the request made by a client process, thus the name server.

- The diagram bellow illustrates the Client/Server definitions. Note how the client process makes a request to the Server Process. The server process handle the request for the client:



Client/Server Interaction

- Note that the name Client and Server have been adopted for many other systems in the field of computer science. For example, the word *Server* is sometimes referred to a physical server computer on a network. And the word *Client*, to represent a computer workstation on a user's desk.
- These representations are also valid since the client workstation is making request from a server machine, but can be quite misleading.
- It's important to understand that the word ***Client*** and ***Server*** as referred in a Client/Server Architecture, represent individual running processes or executables. Where the ***Client*** executable makes request to the ***Server*** executable, and the server executable processes and return the answer to the request to the client.
- What this means is that we can have a ***Client*** process run in the same physical machine as the ***Server*** process. The physical Hardware System or Computer has nothing to do with the location of the Client and Server process.
- So don't get confused with the computer called a server or client and the Client/Server process we are referring to.

Client/Server and Database or DBMS

- **IMPORTANT!** It turns out that 95% of the time, the SERVER part of the Client/Server architecture is a **Database Management System (DBMS)** or some sort of **Database application**.

Client/Server Operating Systems

- The *Client/Server architecture* is so popular and used throughout the field of Computer Science, that it is also used to implement Network Operating Systems.
- For example, most of today's popular networks Operating System Architectures are in reality ***Client/Server***.
- Take for example a popular network operating systems architecture, such as Microsoft Windows NT/2000/2003
- In the Microsoft Network Operating System architectures, the word Workstation (Client) and Server as used and most computer professionals refer to the workstation as the computer used by the user and the server as the backend computer. But in reality the computers have nothing to do with it. In fact what really makes a computer the client or workstation and the server is not the machine itself but the Operating systems running on these machines.
- Let's look at the architecture in detail:

    - **Workstations (Client) Computer -** In this architecture, the word workstation or client is referred to the computer being used by a user. But the truth is as follows:

        - These computers run a CLIENT or desktop Operating System (OS) such as ***Windows 2000 Professional***, or ***Windows NT Workstation***, ***Windows XP*** or ***Windows 98*** etc.
        - It is these Operating Systems that are really clients' process, because the OS is an Operating system which makes request from a Server Operating Systems, which we will discuss below.
        - *Windows 2000/2003 Professional, Windows XP* and *Windows NT Workstation* are client Operating Systems, thus the computer they run on are called Clients or Workstations.
        - So as you can see, the computer has nothing to do with it, it is the Client OS running on these machines.

    - **Server Computer -** In this architecture, the word server is referred to the computer in the backend that contain all the user's data and other programs which can be distributed or used by many workstations. But the truth is as follows:

        - These computers run a SERVER Operating System (OS) such as ***Windows 2000 Server***, ***Windows 2003 Server***, or ***Windows NT Server***.
        - These Operating Systems that are really Server Operating Systems, because the OS is an Operating system which manages or handles request from Client Operating Systems.
        - Windows 2000/2003 Server and Windows NT Server are server Operating Systems, thus the computer they run on are referred to as Servers.
        - So as you can see, the computer has nothing to do with it, it is the Server OS running on these machines.

- So, once again it's important that you understand that the word ***Client*** and ***Server*** as referred in a Client/Server Architecture, represent individual running processes or executables, which can be Operating systems as well and not the physical computers.

Server Operating Systems & Database

- Well, guess what? It turns out that the *Server Operating Systems* contain a Database.
- This database is used to store all the objects which make up the network resources, such as Computers, Users, Groups, Global Policies, etc.
- In Windows 2000/20003 this Database is called: **ACTIVE DIRECTORY**
- The ***Active Directory*** is nothing more than a Database stored on Server Operating System that contains records with all the objects in the network and their relationship etc.
- This Database gets replicated among all the Servers, which run the *Server Operating Systems* so that they are all in Sync or CONSISTENT.
- Like I stated in the last section, the SERVER portion of a Client/Server is usually a Database. Well, even in *Operating Systems* the server contains some sort of **Database Application**.

Client/Server Architecture in Application Development

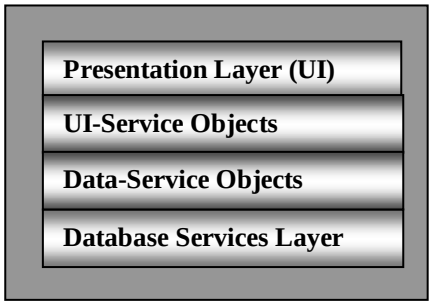
- OK, now lets focus again on the Client/Server architecture when it applies to application development or how applications are implemented in the field of Computer Science.
- The *Client/Server Architecture* is composed of four basic architectures. They are as follows:

    - Single-Tier
    - Two -Tier
    - Three-Tier
    - Web-Based

- In the next section we will look at them in detail

2.2.2 Business Objects & Client/Server Architecture

- Up to this point we can create ***Components*** to neatly package our ***Business Objects***, in addition, have The ***CSLA Architecture*** to develop powerful & scalable component-based applications:



- But, we need to ask ourselves the questions, in a network application, where do we place our business object or components?

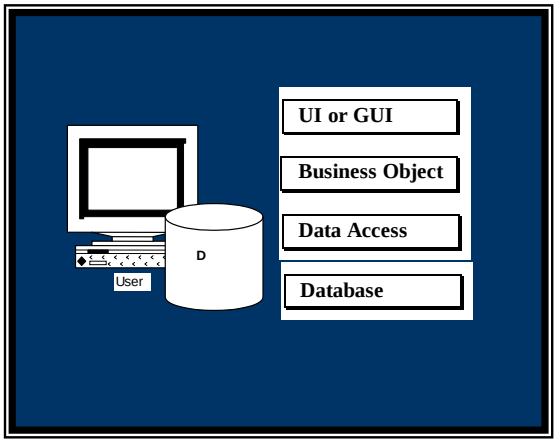
2.2.3 Single-Tier Client/Server

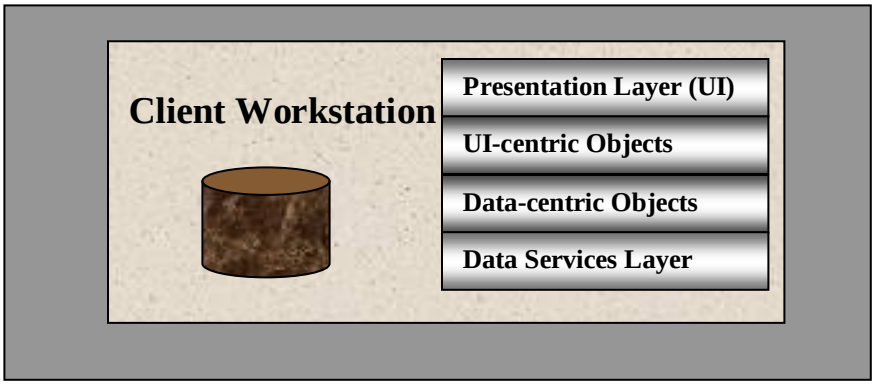
Single-tier Architecture

- A Single-Tier is simply a program or executable that runs on a computer, the program provides the User Interface (UI), application logic and data.
- In other words all of the ***CSLA Architecture*** layers are contained within this application.
- For example, Excel is an example of a single-tier application, since it provides the presentation, application logic & data services in one package.
- Note that in a Single-Tier application that contains a database, the database is usually a desktop database like Microsoft Access, where the data is store in a file, not a separate executable or stand alone process.
- Applications that use Microsoft Access are really Single-Tier applications, since the Microsoft Access database is really a file that can reside in the same computer or on a ***mapped drive*** located on a network server. Don't think because the database file resides on a mapped drive, that the database is a separate program or a server component. Although a mapped driver physically resides on a server machine on the network, to the client machine thinks the drive is local, thus is actually only one machine involved.
- Applications that use Microsoft Access are Single-tier, but those who use Microsoft SQL Server or Oracle Server are not they are n-tier applications as we will see in the next sections.

Business Objects Placement in a Single-Tier

- As for In a Single-Tier, the point is that the ***User-Interface (UI)***, ***Business Logic or Objects*** and ***Data processing*** are all on the same computer.

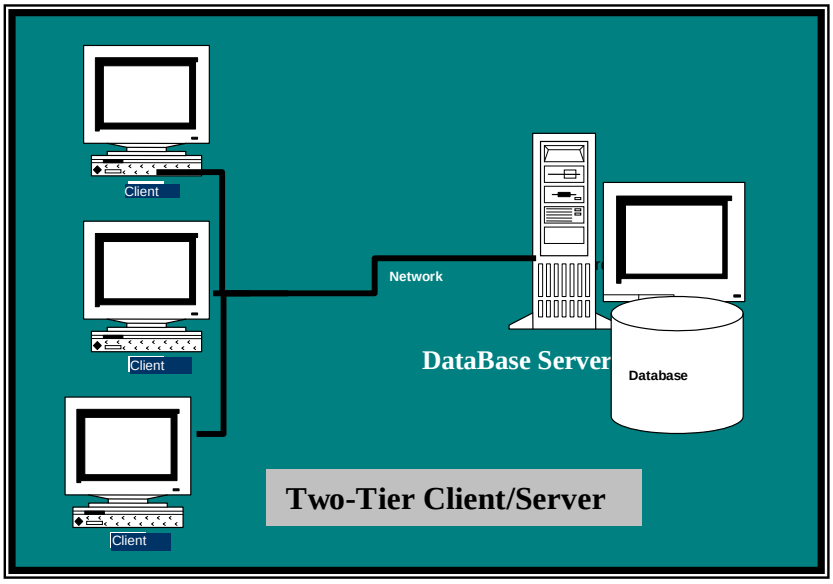




2.2.4 Two-Tier Client/Server

Two-tier Architecture

- The *Two-Tier Client/Server* is the most common architecture.
- It consists of one or more client computers, connecting to a server, which processes the client's request. Thus the term, Client/Server.
- Two-tier consist of a client process running on a workstation, and a separate machine running a Database Server Application or Database Management System. This machine is usually called a **Database Server** because it runs a DBMS such as ***Microsoft SQL Server*** or ***Oracle Server***. These Database Server Applications usually are installed in a Server Operating System such as **Microsoft Windows 2000/2003** and **Windows NT Server**.
- The diagram below illustrates these architecture:

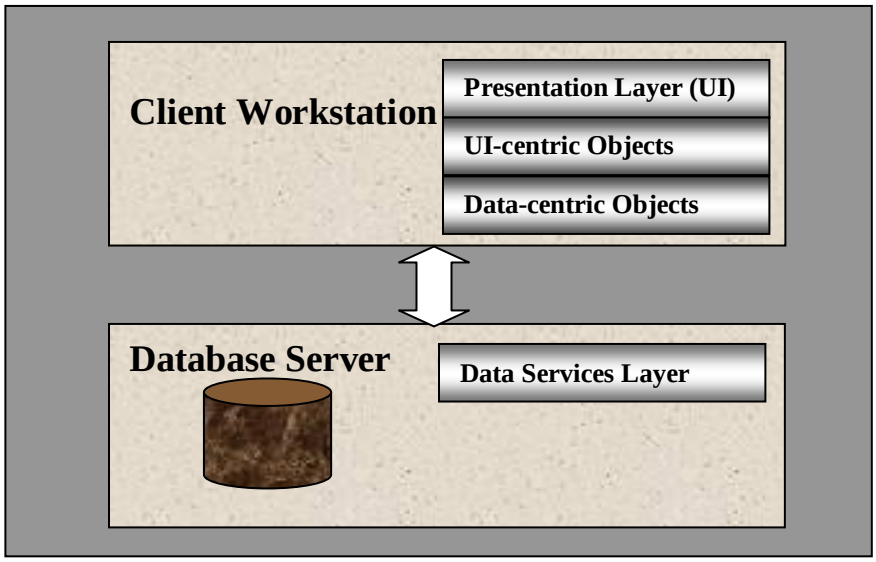


Business Objects Placement in a Two-Tier

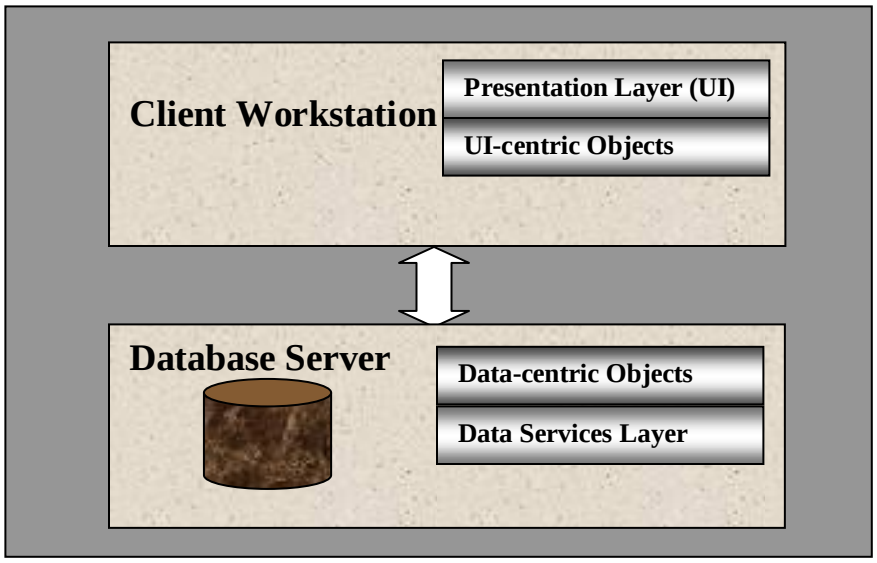
- There are two types of Two-tier Client/Server and they are based on how much processing is done by the client machine or server machine:
  - ***Thin-Client/Fat-Server:*** Client has almost no processing; Usually UI related code only and all processing/Business logic is done on the Database Server.
  - ***Fat-Client/Thin-Server:*** Client has almost all processing, business logic while server has little functionality.

- Placement of Business Object really depends on which type of Two-tier architecture we are designing:

1. ***Fat-Client/Fat-Server:*** Client does most of the processing. User Interface and all data processing/***Business logic*** are done on client. The Data Service Layer is the actual Database Server, which handles the storage or services the data. *Note that this will be the architecture we will use to create our class projects*.



2. ***Thin-Client/Fat-Server:*** Client has less processing, usually UI & UI-Centric ***Business Objects***. The ***Data-Centric Business Objects,*** which handles the data access code, resides on the Data Service Layer or Database Server.



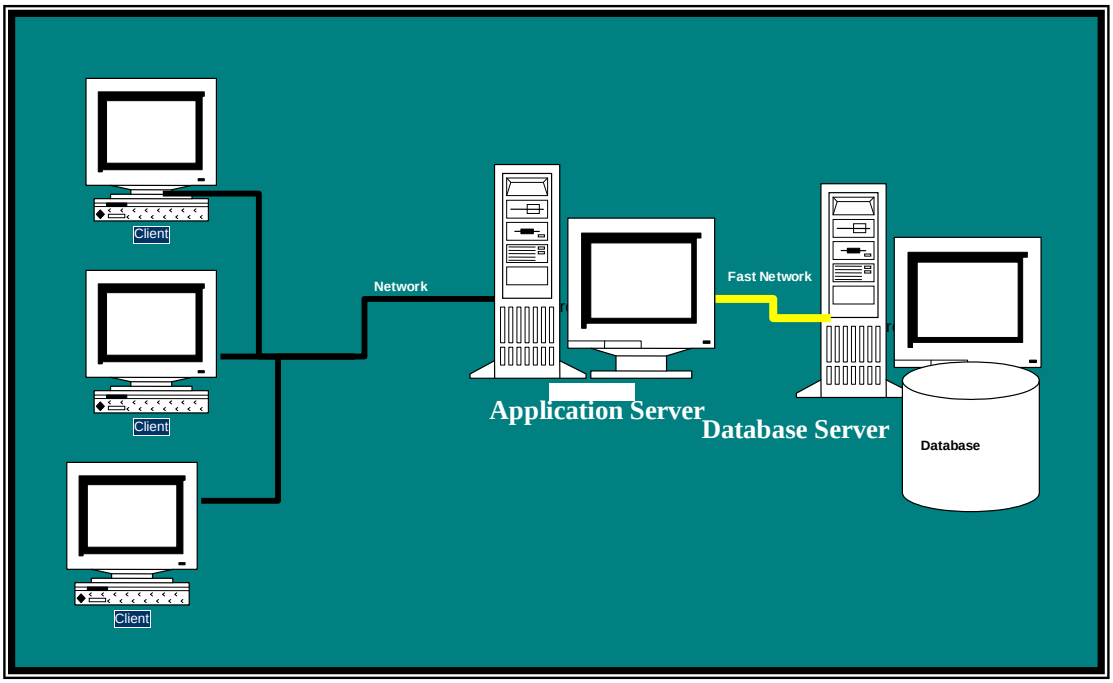
2.2.5 Three-Tier Client/Server

Three-tier Architecture

- The problem with a ***Two-Tier Client/Server*** is that it is not powerful enough or flexible to handle many large applications
- The two-tier solution has the disadvantage that client workstations maintain a dialog or connection with the central database server, therefore **network traffic is high**.
- Also, the central database **server** can become a performance **bottleneck** when many clients try to access the server at the same time.
- Three-Tier Client/Server helps address these issues by putting another physical server between the client and the database known as an ***Application Server***.

Application Server:

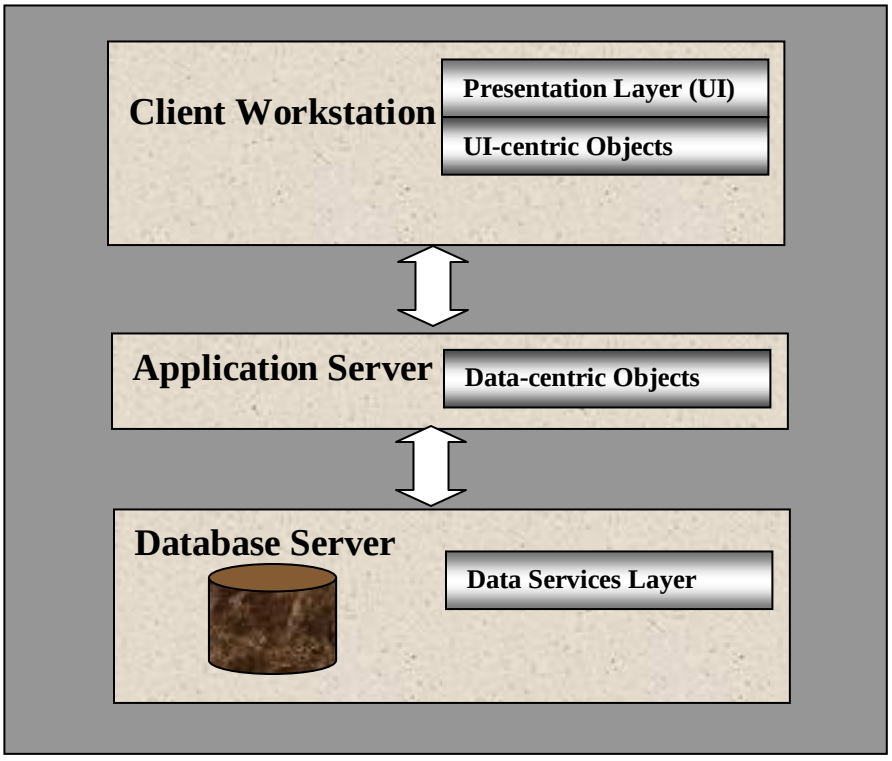
- The application server servers as a middle-man between the client and the database servers.
- This central Application Server can be fine-tuned and configured to more **efficiently handle network traffic** thus diminishing the **load on the database server**, and allowing the database servers to concentrate on what they do best, database access.
- Also, having an Application Server means that we can now move a substantial part of an application from the workstation to the Application Server. This means that Business Objects, such as the ***Data-Centric Business Objects*** can now be placed and managed by the Application Server. The application server handles all connection and manages the use of these business objects. This is known as ***Distributed Processing***.
- **Microsoft Transaction Server** is an example of an Application Server.



- Some of the benefits of Three-Tier Client/Server:
  - Centralization of Shared Resources
  - Rich & Flexible Client Interface to the data
  - Spreading workload across various machines
  - Can dedicate a fast network connection between Application Server and Database Server, thus better performance
  - Putting processing in the most efficient location possible.

Business Objects Placement in a Three-Tier

- Usually in a *Three-Tier* configuration, the ***User-Interface (UI)*** and usually the ***Business Objects*** are kept in the **client**, the ***Data-Service Objects*** that handle database access resided in the ***Application Server***, and the processing of data or Database Access layer resides on the Database Server.



2.2.6 Web-Base Client/Server (Internet/Intranet)

Web-Based Architecture

- Applications written for the World-Wide-Web have taken a different approach, by extending the three-tier concept.
- Client machines don't run a client program, but put as little as possible on the client machine.
- Instead of a client program, the client runs an HTML Browser Application such as Microsoft Internet Explore and Netscape Navigator which handle presenting the data to the user or ***User-Interface (UI)***.
- The client code, actually resides on a Web Server which stores the actual HTML code that are requested by the client Browser. It is important to really understand what's going on here compared to the Two & Three-tier architectures, and why the Web-Based Client/Server configuration is so popular:

Client or Browser:

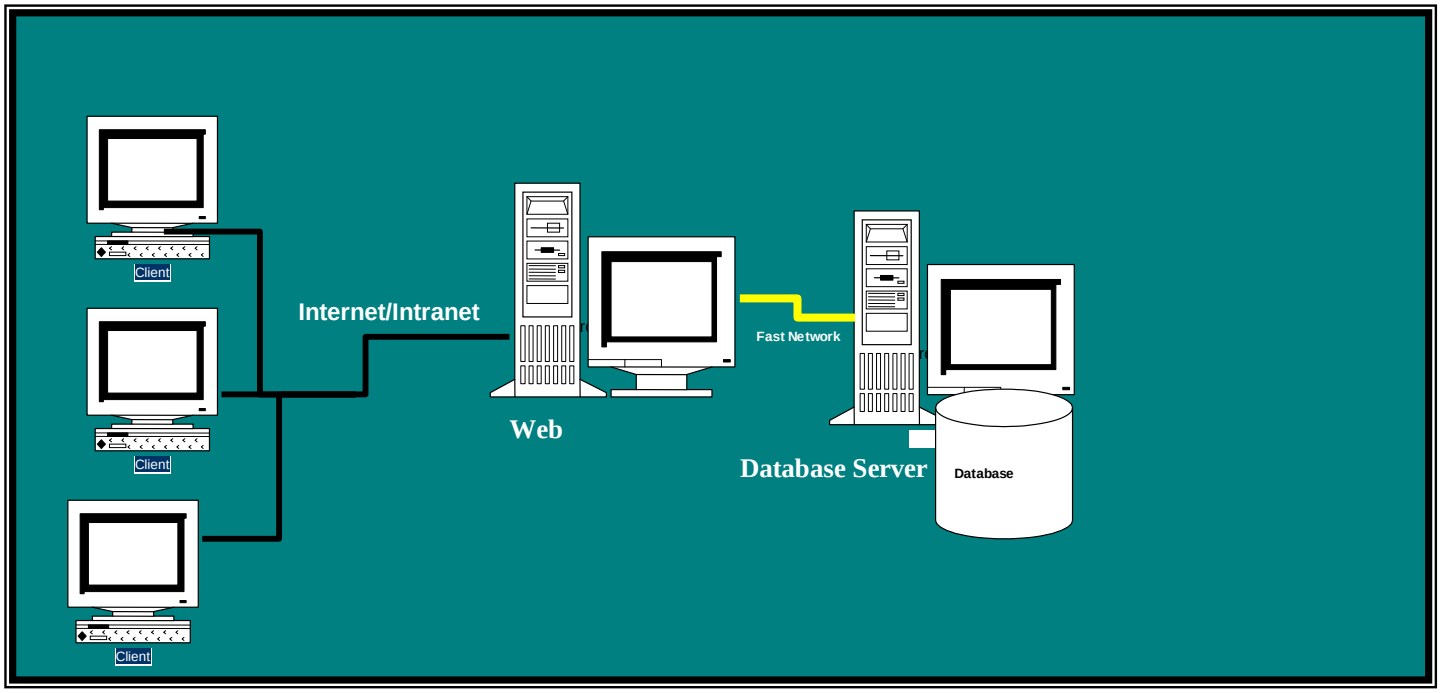
- There is only one client program, or program which runs on the client machine, which is the Browser application.
- This means, that programmers don't have to worry about writing a complex client program and all it's interaction with the operating system etc. The programmer only concern is writing HTML code to solve the problem at hand, not other Operating System related programming.
- Programming in HTML is easier; although for complex processing scripting languages are used such as VBScript, JavaScript etc (More on this in later lectures). But writing complex combinations of HTML & scripting language is still easier than having to write an entire client application.
- Since the browser is common to all application, once a user learns to use a browser, learning new application is a snap. Thus a smaller learning curve, unlike standard client applications which were all different and required users to learn a different interface each time.
- Also, there is not installation of client application involved since there is only one client application, the Browser!

Web Server:

- The job of the web server is to provide the browsers with the requested HTLM pages. It is simply an HTML page distributor.
- Writing the client program means writing HTML code which will then reside on a Web Server and the server distributes the pages as requested by the client browsers.

Database Server:

- In the web server architecture, database servers are also used to store data requested by the clients.



Characteristics of Web Architecture

- We don't need Business Objects or Object-Oriented Programming to write an application on the Web Architecture. A matter of fact, originally, web applications were simply an HTML page with text/graphics that is loaded from the server to the client browser. The browser simply translated and displayed the text/graphics.
- But this limits how powerful a Web App can be, since there is **no processing** on the client, applications cannot perform any processing functionalities and are not powerful. To resolve this problem computer scientist developed the following technologies:

  - **Client- Side Scripting:** Browser built-in *Interpreters* (Translators) for scripting languages such as ***VBScript*** and ***JavaScript*** for client side processing.
    - **Advantage:**
      - Processing on the client
      - Web application can do business logic and programs can be powerful
    - **Disadvantage:**
      - Learn new language, VBScript or JavaScript
      - Powerful workstation to handle the processing

  - **Server-Side Scripting:** Web Server Built-in *Interpreters* (Translators) and Technology such as ***Active-Server Pages*** (uses VBScript or JavaScript) and ***CGI Script*** (can use VBScript, JavaScript, C/C++, Pearl & others).
    - **Advantage:**
      - No Processing on the Client
      - Less powerful Client computer
    - **Disadvantage:**
      - Learn new language, VBScript or JavaScript
      - And, learn a Active Server Pages or worst CGI Script

- In resent times, the Server-Side Scripting technology has been in the fore front of the Web development industry. The advantages of being able to have as little processing in the client and allowing the important processing to be done at the server where is closer to the database has made it appealing to write most web applications using Server-Side Scripting.
- Technologies like Active Server Pages & Java Server Side Scripting has grown in popularity

Internet/Intranet

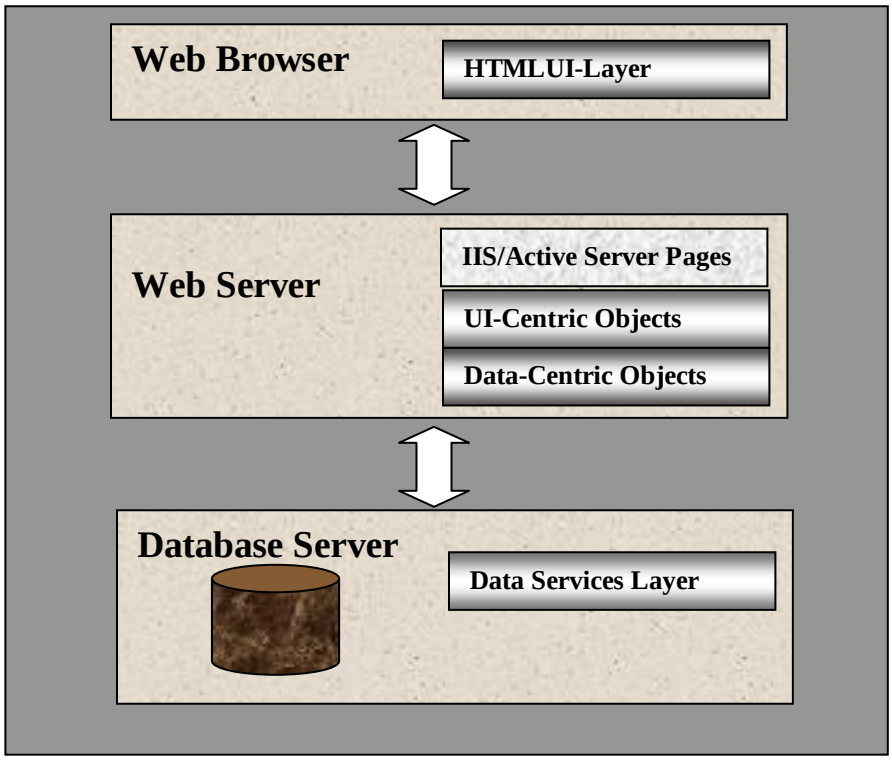
- The Web-Based Client/Server configuration is the standard for the ***Internet***.
- The Internet is a combination of thousands or Web-Based Client/Server architecture all connected together.
- Every time you log on to the Web, you are using a Web Browser to connect to a Web Server that has an address such as Microsoft.com that you enter in the Web Brower.
- The Web-Based architecture has become so popular that industry & governments are now adopting this architecture to develop their internal business applications.
- This lead to the terminology ***Intranet***.
- ***Intranet*** refers to a Web-Based Client/Server architecture that is private or internal to an organization. It is not public like the Internet.

Business Objects Placement in a Web Based Client/Server

- Until recently, in a basic Web-Server architecture, business objects were not needed or used when creating smart web based applications. Programmers got the job done using HTML & scripting languages such as VBScript & JavaScript using Client-Side or Server-Side scripting.
- But these applications were not powerful and lacked the flexibilities, robustness and reusability of their non-web based Object-Oriented Distributed Applications Client/Server counterparts.
- With the processing demands of today e-commerce business & other business related processing, Web-Based Client/Server architecture has evolved and Object-Oriented methodologies are being implemented.
- With the new trend of ***Object-Oriented Programming***, we can create more powerful Web Apps using ***Business Objects*** to handle the business logic and get the benefits of code ***reusability*** and ***scalability***.
- But, how do we fit our Business Objects into this architecture? As stated earlier, the technologies for Web Apps are *scripting*-related languages technologies, which and not full-blown Object-Oriented Languages like Visual Basic, C++ or Java.
- Business Objects are written in these full-blown languages (VB, C++ or Java)
- So, how do we fit our business Objects into the Web Architecture, if the Web Scripting Languages are not OOP languages? The answer is:

1) Write the Business Objects in VB, C++, or Java.
2) Package the Business Objects in an ***ActiveX Component (DLL or EXE)***
3) Have Active-Server Pages or Java Server Side Technologies in the *Web Server* interact with the ***Business Objects*** in the ***ActiveX Component***.
4) Place the Business Objects as follows:

   - In a *Web-Based* configuration, the ***User-Interface (UI)*** is the Web Browser on the client, and the ***UI-Business Objects*** and the ***Data-Service Objects*** that handle database access resided in the Web Server. As usual, the ***Data Service layer*** or database itself resides on the Database Server.



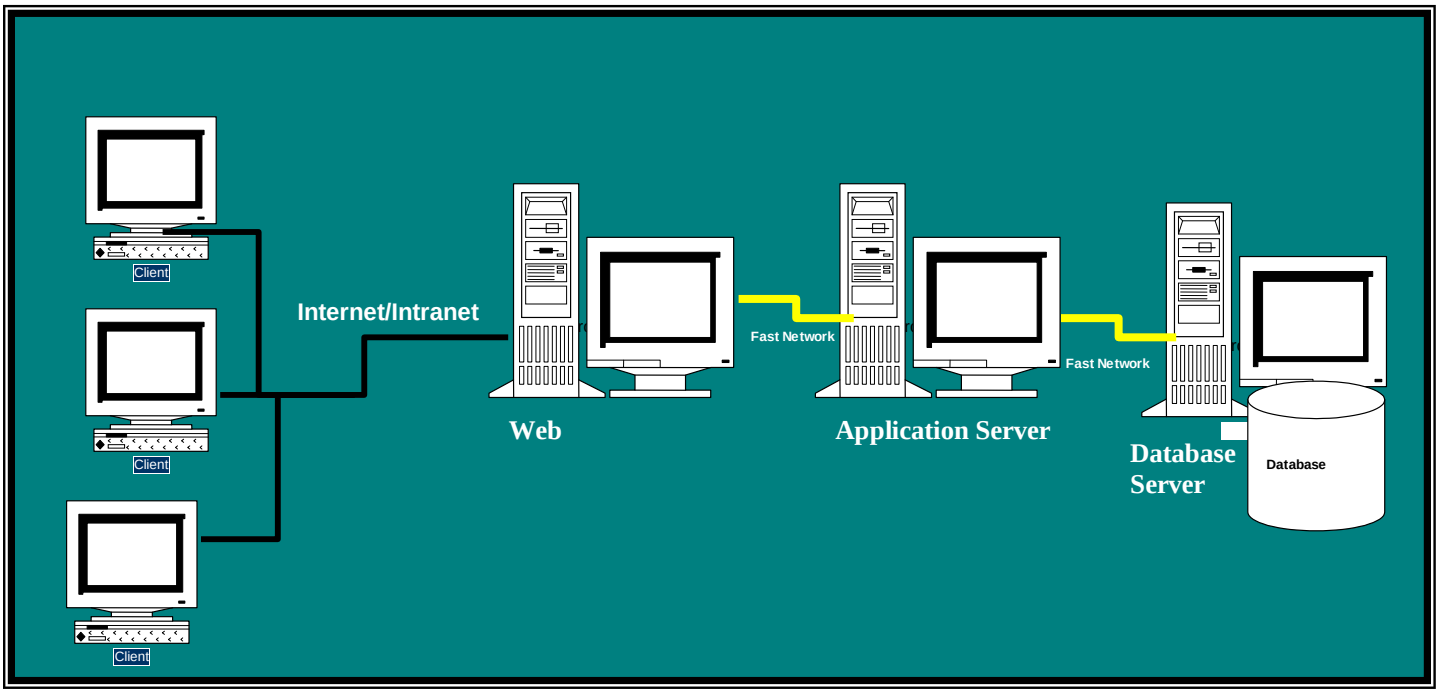
2.2.7 N-Tier Web-Base Client/Server

Multi-tier Web-Base Architecture

- As the demand for faster processing and the eCommerce growth, the Web-Based Client/Server architecture has evolved into a combination of the ***Three-Tier*** & ***Web-Based Client/Server*** to form the ***n-tier Web Based Client/Server Architecture***.
- In this architecture, the **Application Server** Tier is added, thus yielding all the benefits of managing faster connection, business object management etc.

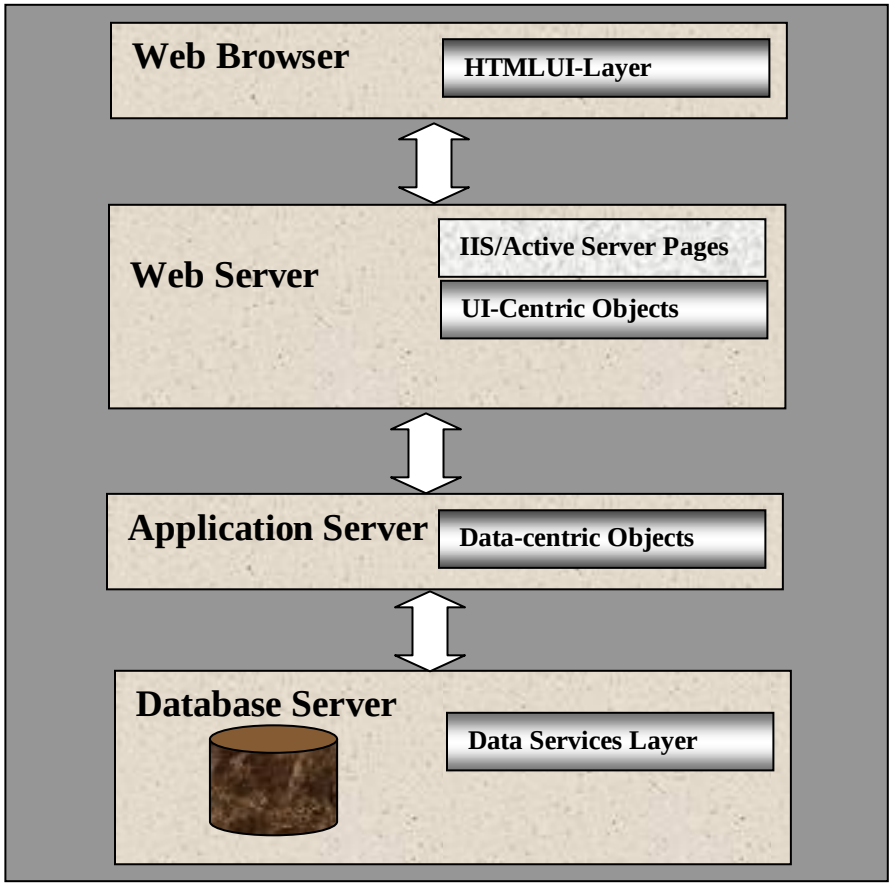
Application Server:

- The application server is used to manage the connections and data access to the database servers.
- This server has all the advantages listed for the Three-Tier Client/Server configuration.
- You can add several Application Servers or Server Farms for load balancing.



Business Objects Placement in a Web Based Client/Server

- In this configuration, the ***User-Interface (UI)*** is the Web Browser on the client, and the User Interface Related Business Objects or ***UI-Centric Business Objects*** are kept at the Web Server closer to the HTML code.
- On the other hand, the ***Data-Centric Business Objects*** are placed and managed in the Application Server.
- As usual, the ***Data Service layer*** or database itself resides on the Database Server



2.2.8 Sample Client/Server Applications

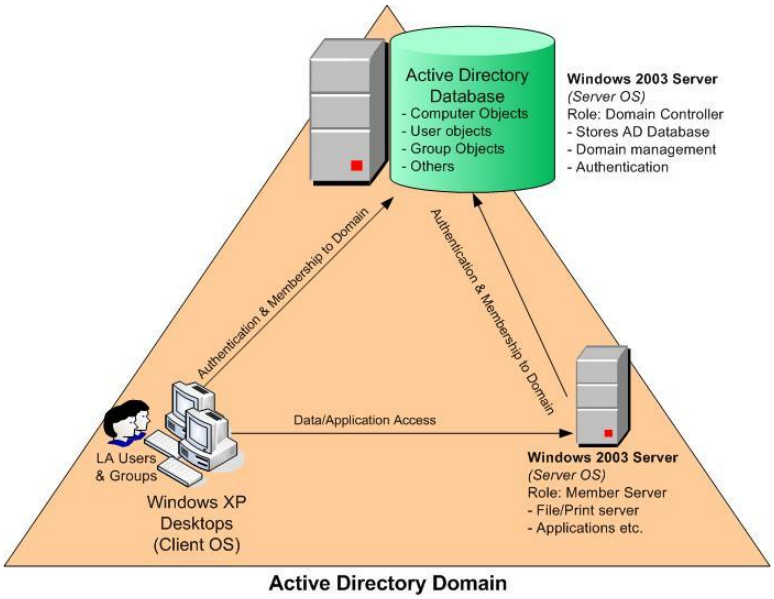
- At this point I will illustrate some popular Client/Server systems.

Client/Server Operating Systems (OS)

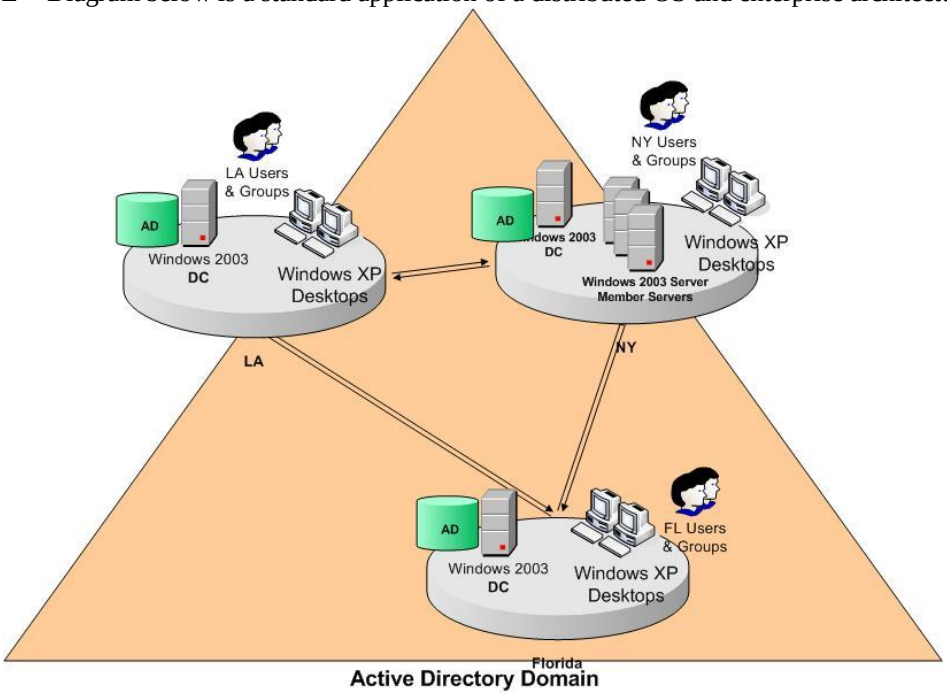
Microsoft Windows XP and Microsoft Windows 2003 Server

- The Microsoft operations systems are based on the Client/Server concept as follows:
  - **Client OS** – Windows XP, Windows 2000 Professional
  - **Server OS** – Windows 2003 Server, Windows 2003 Advanced Server etc.
  - **Client App** – Active Directory Users & Computers (Example of AD Management Console)

- Diagram below is a standard configuration:



- Diagram below is a standard application of a distributed OS and enterprise architecture:



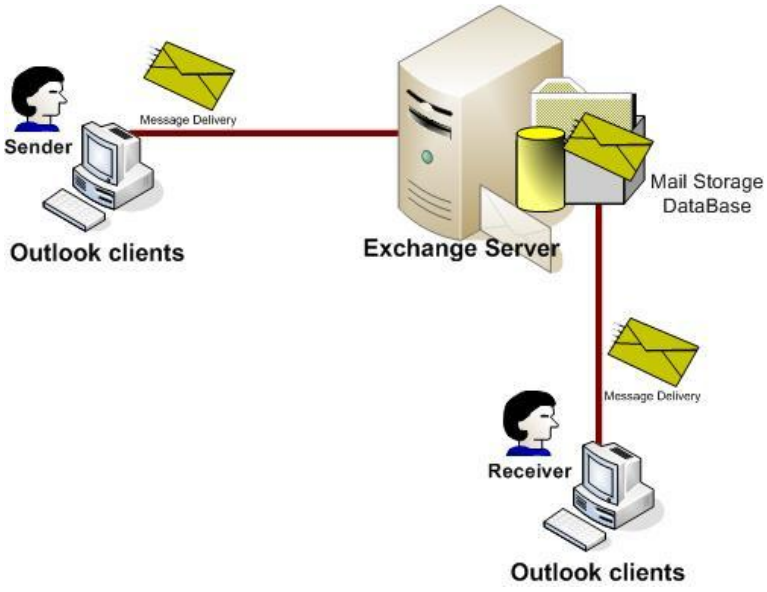
Client/Server Email Systems (Messaging)

Microsoft Exchange 2003 Server and Microsoft Outlook 2003

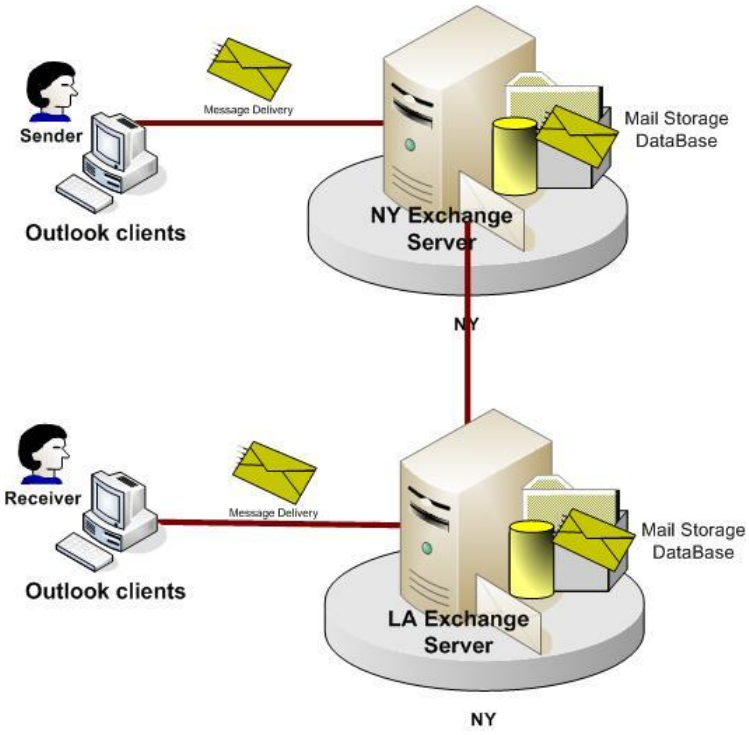
- The Microsoft operations systems are based on the Client/Server concept as follows:

  - **Client** – Outlook 2003
  - **Client** – Exchange Administrator Console
  - **Server**– Exchange Server 2003

- The Microsoft Exchange and Outlook client concept:



- The Microsoft Exchange and Outlook client deployment:



Client/Server System Management Applications

System Management Software

- Management of Servers and Desktops is a very important aspect of an IT department.
- Example of managing the server and desktop environments include:

  - Building new Servers and Workstations (Installing OS manually or using an image (GHOST)). These include:
    - New Desktop Hardware refresh – replacing old workstations with new ones and may include new OS
    - Upgrading existing desktops to new OS
    - Daily management of workstation images (OS images to automatically install new OS on workstations)

  - Installing, updating, applications to Workstations, such as regular applications, virus protections, spyware protection, firewalls, new SECURITY PATCHES etc.

- These tasks can be a nightmare for large or global companies. Especially when new OS or applications need to be deployed.
- System Management software helps IT organization tackle this problem
- Characteristics of System Management Software:

  - Imagine a company of 10,000 users which need a new OS upgrade, example from Windows 2000 professional to Windows XP. How would you do this? Would you send Technicians around with WinXP CDs to do the install and configuration? This may take months, even a year.
  - Using a System Management program, you can do the following:
    - Create images of a standard desktop and STORE YOUR OS IMAGES in the Management Server.
    - Deploy OS IMAGES or installation of a new OS to many desktops simultaneously from the Management SERVER
    - PUSH applications from the MANAGEMENT SERVER to all desktops
    - Create application packages to PUSH and deploy, such as virus protection clients, spyware, word processing etc.

  - System Management Applications, usually have the following features:
    - Maintain Server & Workstation Inventory.
    - Management of OS images (Servers and Workstations)
    - Deployment of Applications remotely
    - Reporting

  - Using a System Management program allows IT Organizations to automate the management of servers and desktops

Microsoft SMS Server 2003 (System Management Server)

- The Microsoft SMS 2003 Server is Microsoft's version of a client/server System Management systems.
- It is a Client/Server concept as follows:

  - **Client:** SMS 2003 Client, SMS Management Console
  - **Client:** SMS Management Console
  - **Server:** SMS 2003 Server

- SMS MUST BE INSTALLED IN A MICROSOFT CLIENT/SERVER NETWORK OPERATING SYSTEM!
- As shown in previous example, the Microsoft operations systems are based on the Client/Server concept as well:

  - **Client OS** – Windows XP, Windows 2000 Professional
  - **Server OS** – Windows 2003 Server, Windows 2003 Advanced Server etc.

- Diagram below is a standard **SMS** configuration for deployment of OS image distribution or deploying the installation of a new desktop remotely: