Some Important University Exam Questions with their answers.

Q1. Explain the use and feature of COM in visual Basic Programming.

Ans:- VB is a very high-level programming language, it is easily understandable as it uses many English keywords in the programming language itself. It makes it very easy for one to create a GUI and implement actions for components on the interface without having to write much code.

One may use VB in creating applications for windows and the web. Using Visual Studio as the IDE makes errors in programming very forgiving, it is able to detect errors and point out suggestions with a good (debugging) interface and able to model relationships of objects within graphically. There is a lot more to the IDE that would make a complete programming beginner want to adopt the language, but the language itself helps make your code more powerful while being so compact.

The Component Object Model: Technical Overview

The Component Object Model (COM) is a software architecture that allows applications to be built from binary software components. COM is the underlying architecture that forms the foundation for higher-level software services, like those provided by OLE. OLE services span various aspects of commonly needed system functionality, including compound documents, custom controls, inter-application scripting, data transfer, and other software interactions.

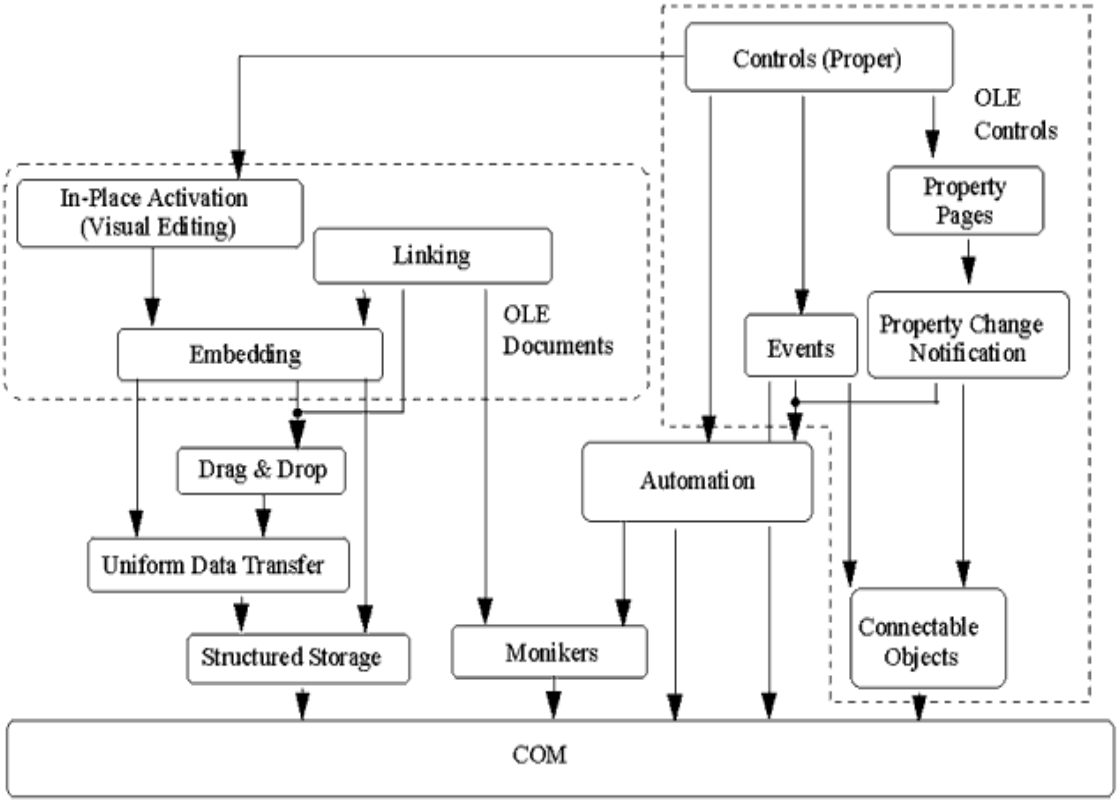


Figure : OLE technologies build on one another, with COM as the foundation.

These services provide distinctly different functionality to the user. However they share a fundamental requirement for a mechanism that allows binary software components, derived from any combination of pre-existing customers' components and components from different software vendors, to connect to and communicate with each other in a well-defined manner. This mechanism is supplied by COM, a software architecture that does the following:

- Defines a binary standard for component interoperability
- Is programming-language-independent
- Is provided on multiple platforms (Microsoft® Windows®, Windows 95, Windows NT™, Apple® Macintosh®, and many varieties of UNIX®)
- Provides for robust evolution of component-based applications and systems
- Is extensible by developers in a consistent manner
- Uses a single programming model for components to communicate within the same process, and also across process and network boundaries
- Allows for shared memory management between components
- Provides rich error and status reporting
- Allows dynamic loading and unloading of components

It is important to note that COM is a general architecture for component software. Although Microsoft is applying COM to address specific areas such as controls, compound documents, automation, data transfer, storage and naming, and others, any developer can take advantage of the structure and foundation that COM provides.

How does COM enable interoperability? What makes it such a useful and unifying model? To address these questions, it will be helpful to first define the basic COM design principles and architectural concepts. In doing so, we will examine the specific problems that COM is meant to solve, and how COM provides solutions for these problems.

Q2. Describe Sharing and Scalability of DCOM.

Ans:- Microsoft's distributed COM (DCOM) extends the Component Object Model (COM) to support communication among objects on different computers—on a LAN, a WAN, or even the Internet. With DCOM, your application can be distributed at locations that make the most sense to your customer and to the application.

Because DCOM is a seamless evolution of COM, the world's leading component technology, you can take advantage of your existing investment in COM-based applications, components, tools, and knowledge to move into the world of standards-based distributed computing. As you do so, DCOM handles low-level details of network protocols so you can focus on your real business: providing great solutions to your customers.

Sharing feature of DCOM:- Most application level protocols require some kind of lifetime management. The component needs to get notified when a client machine suffers a catastrophic hardware failure or the network connection between client and component breaks for an extended period of time.

A common approach to this problem is to send keep-alive message at periodic intervals (pinging). If the server does not receive a ping message for a specified time, it declares the client "dead."

DCOM uses a per machine keep-alive message. Even if the client machine uses 100 components on a server machine, a single ping message keeps all the clients connections alive. In addition to consolidating all the ping messages, DCOM minimizes the size of these ping messages by using delta pinging. Instead of sending 100 client identifiers, it creates meta-identifiers that represent all 100 references. If the set of references changes, only the delta between the two references sets is transmitted. Finally, DCOM piggybacks the ping message onto regular messages. Only if the entire client machine is idle with respect to a given server machine does it send periodic ping messages (at a 2-minute interval).

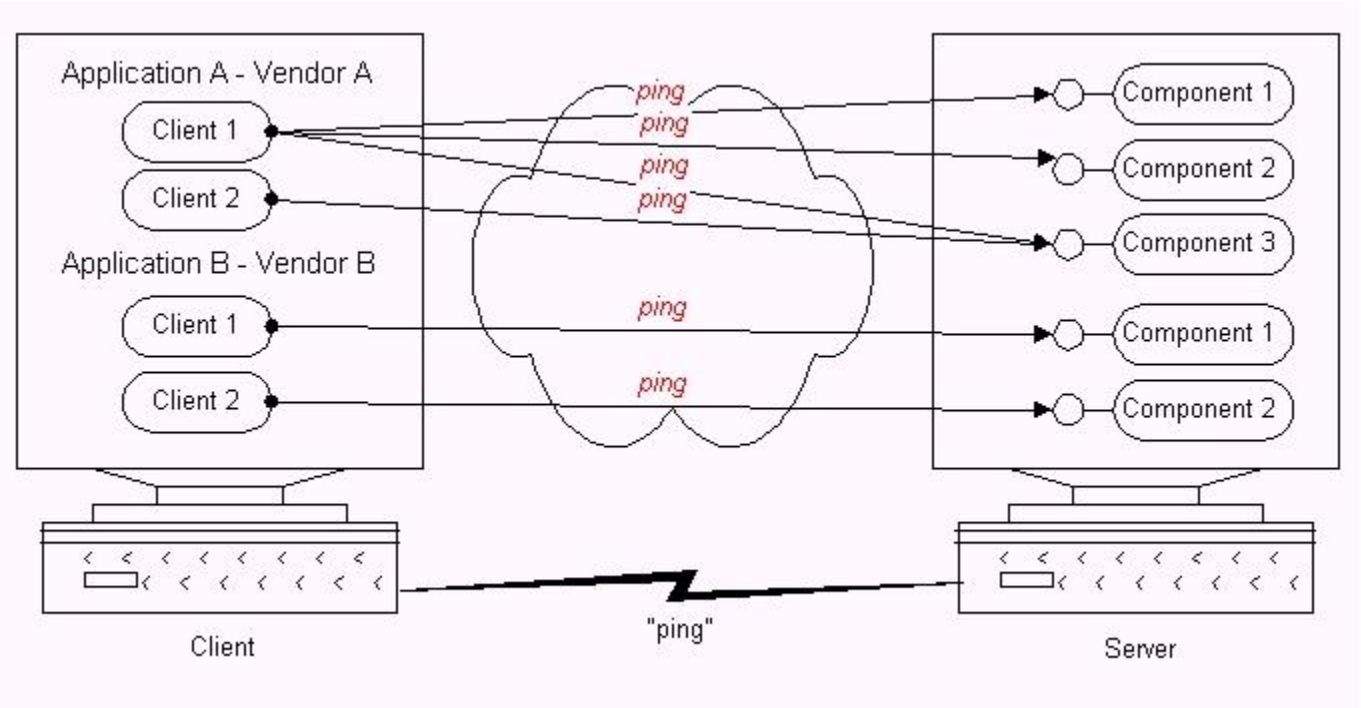


Figure: - Consolidated lifetime management

DCOM allows different applications (even from different vendors) to share a single, optimized lifetime management and network failure detection protocol, reducing bandwidth significantly. If 100 different applications with 100 different custom protocols are running on a server, this server would normally receive one ping message for each of those applications from each of the connected clients. Only if these protocols somehow coordinate their pinging strategies can the overall network overhead be reduced. DCOM automatically provides this coordination among arbitrary COM-based custom protocols.

Scalability Feature of DCOM:- A critical factor for a distributed application is its ability to grow with the number of users, the amount of data, and the required functionality. The application should be small and fast when the demands are minimal, but it should be able to handle additional demands without sacrificing performance or reliability. DCOM provides a number of features that enhance your application's scalability.

Q3. List out various components of COM/DCOM.

Ans:- DCOM is an extension of the Component Object Model (COM). COM defines how components and their clients interact. This interaction is defined such that the client and the component can connect

without the need of any intermediary system component. The client calls methods in the component without any overhead whatsoever. Figure 1 illustrates this in the notation of the Component Object Model:

Figure : - COM Components: Working in the same process

In today's operating systems, processes are shielded from each other. A client that needs to communicate with a component in another process cannot call the component directly, but has to use some form of inter-process communication provided by the operating system. COM provides this communication in a completely transparent fashion: it intercepts calls from the client and forwards them to the component in another process. Figure 2 illustrates how the COM/DCOM run-time libraries provide the link between client and component.

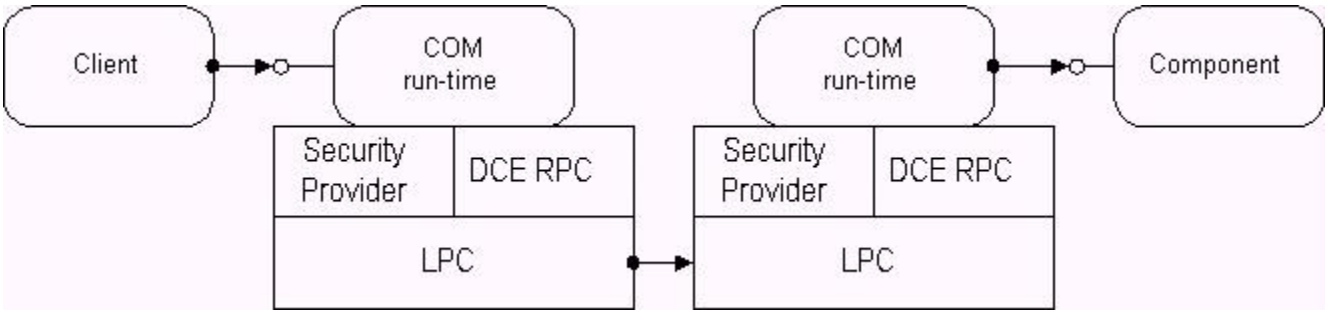


Figure : - COM Components: Working in different processes

When client and component reside on different machines, DCOM simply replaces the local Inter-process communication with a network protocol. Neither the client nor the component are aware that the wire that connects them has just become a little longer.

Figure 3 shows the overall DCOM architecture: The COM run-time provides object-oriented services to clients and components and uses RPC and the security provider to generate standard network packets that conform to the DCOM wire-protocol standard.

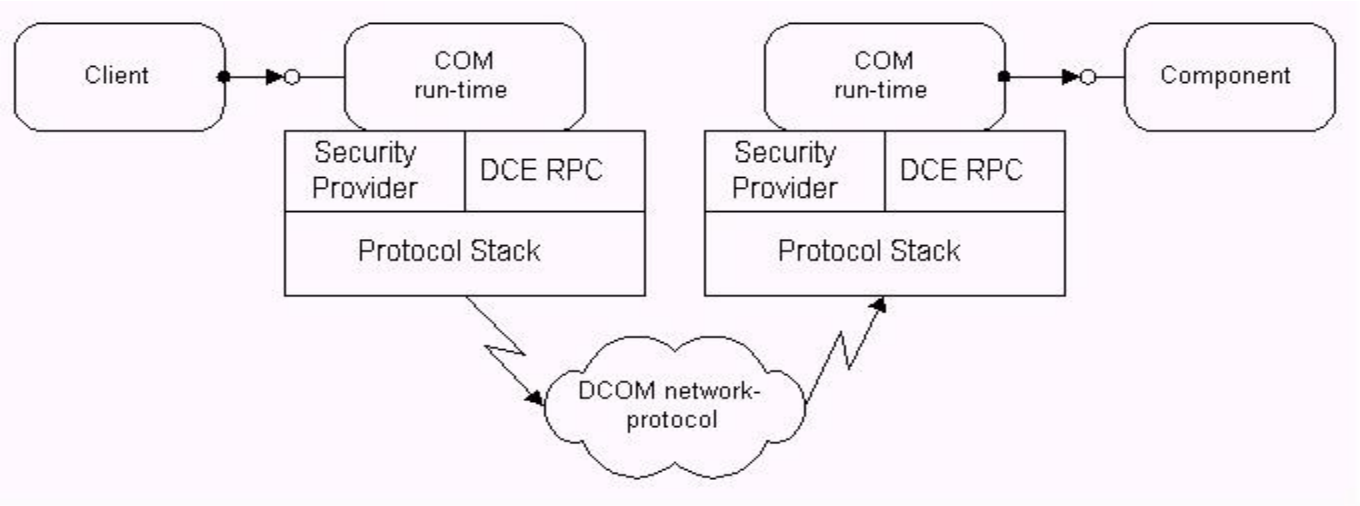


Figure : – DCOM Components: Working on different machines

Q4. Explain process of scalable Client/Server application development.

Ans:- *Scalability is one of the most critical conditions for the success of a web application.*

Looking at the websites of the frontline companies, we don't always realise what kind of workload they are to withstand. Neither do we think about the importance of high productivity and usability for the success of the company. Therefore, when the newcomers start building their first websites, they don't know about some problems they may run into later:

- An increase in the number of visitors reduces the performance of web applications and blows up the failures;
- Expansion of the product range negatively affects the page load time; updating the inventory of an e-commerce shop becomes problematic;
- Changing the code structure becomes dangerous and overcomplicated; adding a new product or service take too much time and becomes expensive, and the possibility of carrying out the A/B tests reduces.

Scalability:- The website scalability definition is the ability of a system, network, or process to cope with the increase in workload when adding resources (usually hardware).

Scalability can be assessed through the ratio of the increase in system performance to the rise in resources used. Also, scalability means the ability to add extra resources while keeping the structure of the central node intact.

In a system with poor scalability, adding resources leads only to a slight increase in performance, and after reaching a certain threshold, boosting the resources does not have any effect at all.

Below are some of the features of client/server architecture:

1. **Service:** Client/Server can be established as a relationship between different processes that run on separate machines. On one end the server process is a provider of service and on the other end the client is a consumer of services.

2. **Shared resources:** A server has the ability to service multiple clients at simultaneously and regulate their overall access to shared resources.
3. **Functional modules:** The client/server also acts as a functional module that has well-defined interfaces embedded in it. Most of the functions which are performed by a client and a server can be implemented by a set of software modules, hardware components, or both. They can also run on dedicated machines, if needed.
4. **Asymmetrical Protocols:** Between clients and servers, there exists a m:1 relationship. Clients are the one who initiate the dialog by requesting a service. Servers on the other end passively await requests from the clients.
5. **Transparency of location:** The server is a process that houses on the same machine as the client or on an entirely different machine across a network. Client/server software creates a veil across the location of the server from the clients by redirecting the service calls when needed. A program can either be a client, a server or both.
6. **Mix-and-Match:** A stable client/server software works independent of hardware or OS software platforms, enabling us to mix-and-match client and server program.
7. **Message based exchanges**: Client/server is a system that is loosely coupled and interacting through a message-passing mechanism. The message acts as the delivery mechanism for the service requests and replies.
8. **Encapsulation of Services:** The server can be termed as a “Specialist”. In the first step, message tells a server what service is requested, in the next step the server than determines how to get the job done. One can upgrade the server without affecting the clients as long as the published message interface is not deterred.
9. **Scalability:** You can scale the client/server horizontally or vertically. Horizontal scaling will add or remove client workstation with only a slight performance impact. Vertical scaling will migrate it to a larger and faster server machine or multi-server.
10. **Integrity:** In the client/server architecture, the server code and data is centrally maintained resulting in cheaper utilization maintenance and the guarding of shared data integrity. Simultaneously, the clients remain personal and independent.
11. **Robustness:** The server should not expect client code to be error free.
12. **Authentication:** More often than usual, a few clients should be permitted to use a particular service.
13. **Secrecy:** The data has to be encrypted in order to keep it private when it is sent over public networks.
14. **Availability**: Denial of service (DoS) attacks are particularly problematic. They can involve very large numbers of requests coming from a number of coordinated sites. The server is expected to identify such attacks and ignore the messages involved in the attack.
15. **Timeliness:** Servers are expected to respond within tight time deadlines in order to ensure QoS. This can sometimes be problematic, at times where the load on the server can be very variable.

Conclusion:- Client/server systems have become the computing and application architecture for business organizations across the globe. Speaking in technical terms, a client/server system puts application processing close to the user; helping improve the performance.

Q5. Describe role of multi-tier architecture for application implementation.

Ans:- It is also called “N-Tier Architecture”. The multi-tier architecture is an industry-proven software architecture model. It is suitable to support enterprise level client-server applications by providing solutions to scalability, security, fault tolerance, reusability, and maintainability.

N-tier application architecture provides a model by which developers can create flexible and reusable applications. By segregating an application into tiers, developers acquire the option of modifying or adding a specific layer, instead of reworking the entire application. A three-tier architecture is typically composed of a presentation tier, a domain logic tier, and a data storage tier.

While the concepts of layer and tier are often used interchangeably, one fairly common point of view is that there is indeed a difference. This view holds that a layer is a logical structuring mechanism for the elements that make up the software solution, while a tier is a physical structuring mechanism for the system infrastructure.

The three tiers, or layers, involved include:

1. A Presentation Layer that sends content to browsers in the form of HTML/JS/CSS. This might leverage frameworks like React, Angular, Ember, Aurora, etc.
2. An Application Layer that uses an application server and processes the business logic for the application. This might be written in C#, Java, C++, Python, Ruby, etc.
3. A Data Layer which is a database management system that provides access to application data. This could be MSSQL, MySQL, Oracle, or PostgreSQL, Mongo, etc.

As a simple example, suppose you are looking to find movie times in your area using a web application. First, the presentation layer displays a web page with some fields for you to enter, like the date you want to view the movie and your zip code. This information is then passed to the application layer, which formats a query and passes it to the database layer. The database system runs the query and returns the results (a list of movies available within your geographic area) to the application layer, which formats it into a web page. The page is then sent back to the browser, where the presentation layer displays it on a laptop or other device.

Here are the benefits of separating an application into tiers:

It gives you the ability to update the technology stack of one tier, without impacting other areas of the application.

It allows for different development teams to each work on their own areas of expertise. Today's developers are more likely to have deep competency in one area, like coding the front end of an application, instead of working on the full stack.

You are able to scale the application up and out. A separate back-end tier, for example, allows you to deploy to a variety of databases instead of being locked into one particular technology. It also allows you to scale up by adding multiple web servers.

It adds reliability and more independence of the underlying servers or services.

It provides an ease of maintenance of the code base, managing presentation code and business logic separately, so that a change to business logic, for example, does not impact the presentation layer.

With 3-tier architecture, you have the ability to utilize new technologies as they become available. This ensures your product is ready to adapt; ready for the future. You have the opportunity to redesign your product or application and actually look not only to today's needs but into the future. Stay ahead of the game and maintain a competitive advantage.

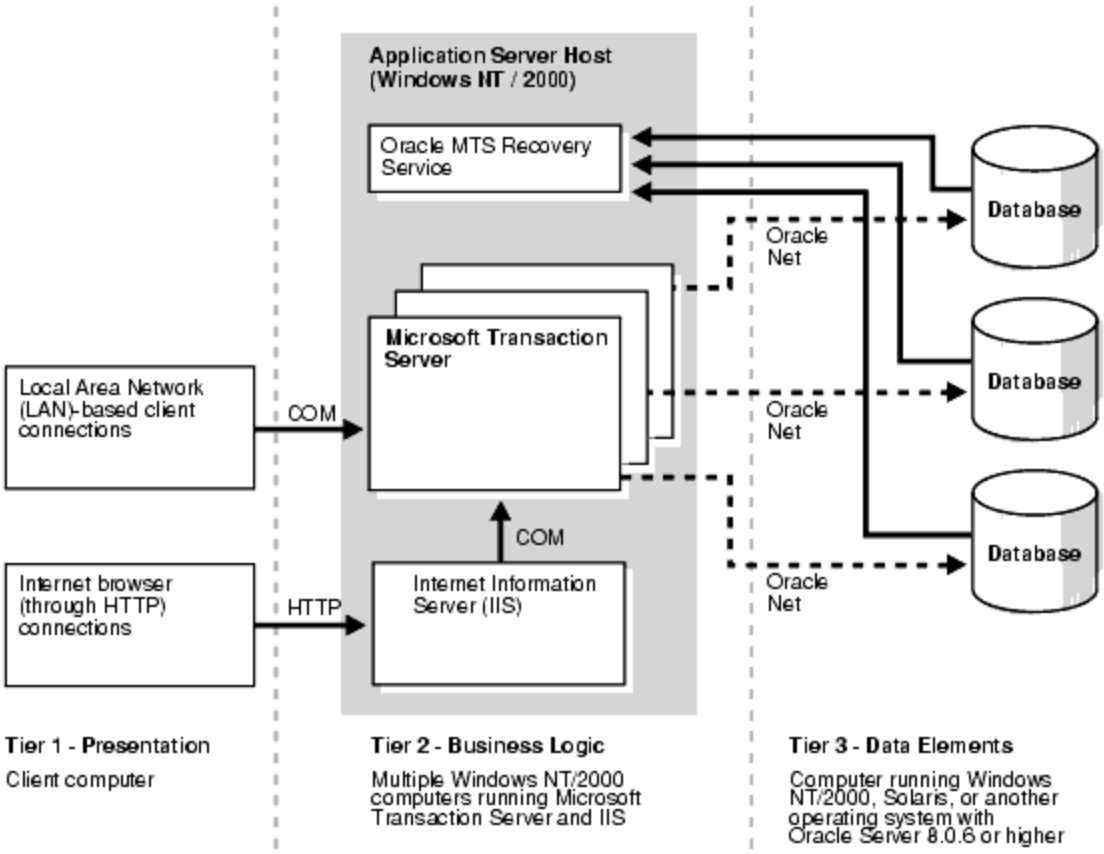
Q6. How does COM/DCOM supports for database transactions ?

Ans:- Microsoft Transaction Server is a proprietary component object model (COM) transaction processing system that runs on an Internet or network server. Microsoft Transaction Server deploys and manages application and database transaction requests on behalf of a client computer. Microsoft Transaction Server provides the following:

- An ActiveX/distributed component object model (DCOM) programming model to develop distributed applications and a runtime environment in which to deploy these applications
- Atomicity, Consistency, Isolation, and Durability (ACID) properties to components in transactions
- Access to performance enhancing features such as component caching and database connection pooling

Microsoft Transaction Server is a component of the three-tiered, server-centric architecture model. This enables the presentation, business logic, and data elements of applications to be clearly separated onto different computers connected in a network.

Following figure shows the Microsoft Transaction Server(COM/DCOM based Component) and Database Server Integration



Q7. Differentiate between distributed computing and standalone computing.

Ans:- By local system I guess you mean single node and that is also called centralized system. A distributed one is having multiple nodes potentially.

DISTRIBUTED COMPUTING: In decentralized systems, every node makes its own decision. The final behavior of the system is the aggregate of the decisions of the individual nodes. Note that there is no single entity that receives and responds to the request.

Example –
Google search system. Each request is worked upon by hundreds of computers which crawl the web and return the relevant results. To the user, the Google appears to be one system, but it actually is multiple computers working together to accomplish one single task (return the results to the search query).

Characteristics of Distributed System – : Concurrency of components: Nodes apply consensus protocols to agree on same values/transactions/commands/logs.

Lack of a global clock: All nodes maintain their own clock.

Independent failure of components: In a distributed system, nodes fail independently without having a significant effect on the entire system. If one node fails, the entire system sans the failed node continue to work.

Scaling –
Horizontal and vertical scaling is possible.

Components of Distributed System –
Components of Distributed System are, Node (Computer, Mobile, etc.),Communication link (Cables, Wi-Fi, etc.),

Architecture of Distributed System –

peer-to-peer – all nodes are peer of each other and work towards a common goal

client-server – some nodes are become server nodes for the role of coordinator, arbiter, etc.

n-tier architecture – different parts of an application are distributed in different nodes of the systems and these nodes work together to function as an application for the user/client

Limitations of Distributed System –

Difficult to design and debug algorithms for the system. These algorithms are difficult because of the absence of a common clock; so no temporal ordering of commands/logs can take place. Nodes can have different latencies which have to be kept in mind while designing such algorithms. The complexity increases with increase in number of nodes. Visit this link for more information

No common clock causes difficulty in the temporal ordering of events/transactions

Difficult for a node to get the global view of the system and hence take informed decisions based on the state of other nodes in the system

Advantages of Distributed System –

Low latency than centralized system – Distributed systems have low latency because of high geographical spread, hence leading to less time to get a response

Disadvantages of Distributed System –

Difficult to achieve consensus

Conventional way of logging events by absolute time they occur is not possible here

Applications of Distributed System –

Cluster computing – a technique in which many computers are coupled together to work so that they achieve global goals. The computer cluster acts as if they were a single computer

Grid computing – All the resources are pooled together for sharing in this kind of computing turning the systems into a powerful supercomputer; essentially.

Use Cases – SOA-based systems, Multiplayer online games,

Organisations Using – Apple, Google, Facebook.

Standalone Systems:-

A standalone computer is exactly what its name implies: a computer that stands on its own. Any tasks or data associated with that computer stay inside it and are not accessible from anywhere else. Any peripherals, such as printers, must be directly connected to it in order to work.

Network

A standalone's counterpart is the network. Basically, a network is a group of separate computers connected together. A peer-to-peer network is the simplest form, because it simply hooks computers together in a circular fashion. Other methods, such as client/server, are controlled by a central area called a hub.

Standalone Advantages

One advantage of a standalone computer is damage control. For example, if something goes wrong, only the standalone will be affected. Simplicity is another advantage, because it takes a lot less expertise to manage one computer than it does to setup or troubleshoot several. Standalone computers can also be more convenient. For example, printing on a network may require you to walk some distance from the computer to the printer. Inversely, any peripherals on a standalone have to be in arm's reach. Finally, a standalone does not affect other computer users. With a network, one user may waste space by watching movies or listening to music. In turn, everyone else using the network may see slower computer performance.

Standalone Disadvantages

Standalone computers have drawbacks. First of all, users are restricted to a single computer. On a network, users can access their files from any connected computer. Second, the same software cannot be installed simultaneously. While a network allows everything to be changed at once, a standalone requires that any new programs must be set up one-by-one, which is much more time-consuming. Third, it is much cheaper to connect every computer to one printer than to buy a printer for each standalone computer. Finally, standalones are harder to monitor. On a network, certain software can be used to simultaneously view each user's activity.

Q8. Describe the features of ActiveX and ADO.

Ans:- ActiveX Data Objects (ADO) is an application program interface from Microsoft that lets a programmer writing Windows applications get access to a relational or non-relational database from both Microsoft and other database providers. For example, if you wanted to write a program that would provide users of your Web site with data from an IBM DB2 database or an Oracle database, you could include ADO program statements in an HTML file that you then identified as an Active Server Page. Then, when a user requested the page from the Web site, the page sent back would include appropriate

data from a database, obtained using ADO code. Microsoft introduced ADO in October 1996, positioning the software as a successor to Microsoft's earlier object layers for accessing data sources, including RDO (Remote Data Objects) and DAO (Data Access Objects).

Like Microsoft's other system interfaces, ADO is an object-oriented programming interface. It is also part of an overall data access strategy from Microsoft called Universal Data Access. Microsoft says that rather than trying to build a universal database as IBM and Oracle have suggested, finding a way to provide universal access to various kinds of existing and future databases is a more practical solution. In order for this to work, Microsoft and other database companies provide a "bridge" program between the database and Microsoft's OLE DB, the low-level interface to databases.

OLE DB is the underlying system service that a programmer using ADO is actually using. A feature of ADO, Remote Data Service, supports "data-aware" ActiveX controls in Web pages and efficient client-side caches. As part of ActiveX, ADO is also part of Microsoft's overall Component Object Model (COM), its component-oriented framework for putting programs together.

ADO evolved from an earlier Microsoft data interface, Remote Data Objects (RDO). RDO works with Microsoft's ODBC to access relational databases, but not non-relational databases such as IBM's ISAM and VSAM.

Q9. Explain Common Language Platform dot net.

Ans:- Common Language Runtime (CLR) is a managed execution environment that is part of Microsoft’s .NET framework. CLR manages the execution of programs written in different supported languages.

CLR transforms source code into a form of bytecode known as Common Intermediate Language (CIL). At run time, CLR handles the execution of the CIL code.

Developers write code in a supported .NET language, such as C# or VB.Net. The .NET compiler then converts it into CIL code. During run time, the CLR converts the CIL code into something that can be understood by the operating system. Alternately, the CIL code can be transformed into native code by using the native image generator (NGEN).

The language compilers store metadata that describes the members, types and references in the compiled code. The CLR uses the metadata to lay out instances in memory, locate and load classes, enforce security, set runtime context boundaries, and generate native code.

CLR allows for the easy use of different supported languages to achieve a common goal. This makes it flexible for developers to choose their own programming language, provided it is supported by the .NET framework. With CLR, .NET can manage the execution of all supported languages by transforming them to bytecode and then into the native code for the chosen platform.

Using NGEN makes later runs faster because CLR will not have to transform the bytecode into native code each time. Although other implementations of CLI can run on platforms other than Windows, Microsoft’s CLI implementation is only meant to run on the Windows platform.

CLI languages are computer programming languages that are used to produce libraries and programs that conform to the Common Language Infrastructure (CLI) specifications. With some notable

exceptions, most CLI languages compile entirely to the Common Intermediate Language (CIL), an intermediate language that can be executed using the Common Language Runtime, implemented by .NET Framework, .NET Core, and Mono. Some of these languages also require the Dynamic Language Runtime (DLR).

As the program is being executed, the CIL code is just-in-time compiled (and cached) to the machine code appropriate for the architecture on which the program is running. This step can be omitted manually by caching at an earlier stage using an "ahead of time" compiler such as Microsoft's ngen.exe and Mono's "-aot" option.

Q 10. Define the terms: (a) CORBA (b) OLE (c) IDL (d) DAO

Ans:- **(a) CORBA**: - The Common Object Request Broker Architecture (CORBA) is a specification developed by the Object Management Group (OMG). CORBA describes a messaging mechanism by which objects distributed over a network can communicate with each other irrespective of the platform and language used to develop those objects.

There are two basic types of objects in CORBA. The object that includes some functionality and may be used by other objects is called a service provider. The object that requires the services of other objects is called the client. The service provider object and client object communicate with each other independent of the programming language used to design them and independent of the operating system in which they run. Each service provider defines an interface, which provides a description of the services provided by the client.

CORBA enables separate pieces of software written in different languages and running on different computers to work with each other like a single application or set of services. More specifically, CORBA is a mechanism in software for normalizing the method-call semantics between application objects residing either in the same address space (application) or remote address space (same host, or remote host on a network).

(b) OLE:- Object Linking & Embedding (OLE) is a proprietary technology developed by Microsoft that allows embedding and linking to documents and other objects. For developers, it brought OLE Control Extension (OCX), a way to develop and use custom user interface elements. On a technical level, an OLE object is any object that implements the IOleObject interface, possibly along with a wide range of other interfaces, depending on the object's needs.

OLE allows an editing application to export part of a document to another editing application and then import it with additional content. For example, a desktop publishing system might send some text to a word processor or a picture to a bitmap editor using OLE. The main benefit of OLE is to add different kinds of data to a document from different applications, like a text editor and an image editor. This creates a Compound File Binary Format document and a master file to which the document makes reference. Changes to data in the master file immediately affect the document that references it. This is called "linking" (instead of "embedding").

(c) IDL:- IDL, short for Interactive Data Language, is a programming language used for data analysis. It is popular in particular areas of science, such as astronomy, atmospheric physics and medical imaging.[citation needed] IDL shares a common syntax with PV-Wave and originated from the same codebase, though the languages have subsequently diverged in detail. There are also free or costless implementations, such as GNU Data Language (GDL) and Fawlty Language (FL).

IDL is vectorized, numerical, and interactive, and is commonly used for interactive processing of large amounts of data (including image processing). The syntax includes many constructs from Fortran and some from C.

(d) DAO:- In computer software, a data access object (DAO) is a pattern that provides an abstract interface to some type of database or other persistence mechanism. By mapping application calls to the persistence layer, the DAO provides some specific data operations without exposing details of the database. This isolation supports the single responsibility principle. It separates what data access the application needs, in terms of domain-specific objects and data types (the public interface of the DAO), from how these needs can be satisfied with a specific DBMS, database schema, etc. (the implementation of the DAO).

Although this design pattern is equally applicable to most programming languages, most types of software with persistence needs, and most types of databases, it is traditionally associated with Java EE applications and with relational databases (accessed via the JDBC API because of its origin in Sun Microsystems' best practice guidelines[1] "Core J2EE Patterns" for that platform).

It is a object/interface, which is used to access data from database of data storage. WHY WE USE DAO: it abstracts the retrieval of data from a data resource such as a database. The concept is to "separate a data resource's client interface from its data access mechanism."

Q11. What do you mean by client/server system ? Explain the advantages of client-server software development.

Ans:- In Computer science, client-server is a software architecture model consisting of two parts, client systems and server systems, both communicating over a computer network or on the same computer. A client-server application is a distributed system made up of both client and server software. Client server application provide a better way to share the workload. The client process always initiates a connection to the server, while the server process always waits for requests from any client.

In client server computing, the clients requests a resource and the server provides that resource. A server may serve multiple clients at the same time while a client is in contact with only one server. Both the client and server usually communicate via a computer network but sometimes they may reside in the same system.

Advantages of Client Server Computing:- The different advantages of client server computing are:

1) Centralization : Unlike P2P, where there is no central administration, here in this architecture there is a centralized control. Servers help in administering the whole set-up. Access rights and resource allocation is done by Servers.
2) Proper Management : All the files are stored at the same place. In this way, management of files becomes easy. Also it becomes easier to find files.
3) Back-up and Recovery possible : As all the data is stored on server its easy to make a back-up of it. Also, in case of some break-down if data is lost, it can be recovered easily and efficiently. While in peer computing we have to take back-up at every workstation.
4) Up gradation and Scalability in Client-server set-up : Changes can be made easily by just upgrading the server. Also new resources and systems can be added by making necessary changes in server.
5) Accessibility : From various platforms in the network, server can be accessed remotely.

6) As new information is uploaded in database , each workstation need not have its own storage capacities increased (as may be the case in peer-to-peer systems). All the changes are made only in central computer on which server database exists.
7) Security : Rules defining security and access rights can be defined at the time of set-up of server.
8) Servers can play different roles for different clients.

Q12. What are the functions of a middleware in a three-tier architecture ? Explain two popular middleware standards.

Ans:- Middleware is software which lies between an operating system and the applications running on it. Essentially functioning as hidden translation layer, middleware enables communication and data management for distributed applications. It is sometimes called plumbing, as it connects two applications together so data and databases can be easily passed between the "pipe." Using middleware allows users to perform such requests as submitting forms on a web browser or allowing the web server to return dynamic web pages based on a user's profile.

Common middleware examples include database middleware, application server middleware, message-oriented middleware, web middleware and transaction-processing monitors. Each programme typically provides messaging services so that different applications can communicate using messaging frameworks like simple object access protocol (SOAP), web services, representational state transfer (REST) and JavaScript object notation (JSON). While all middleware performs communication functions, the type a company chooses to use will depend on what service is being used and what type of information needs to be communicated. This can include security authentication, transaction management, message queues, applications servers, web servers and directories. Middleware can also be used for distributed processing with actions occurring in real time rather than sending data back and forth.

Two Popular middleware standards are: -

1. Red Hat JBoss Enterprise Application Platform (EAP)

Red Hat JBoss Enterprise Application Platform (EAP) is a mouthful of a name, and it packs a powerful punch as well. Version 7 of the software was released last year, providing a cloud-compatible, flexible solution for enterprises looking to increase their agility.

Benefits of JBoss EAP
•Solid architectural foundation, with quick start-up times and low memory usage
•Out-of-the-box integration with DevOps tools such as Maven, Jenkins and Arquillian
•"JBoss Migration Center" at JBoss.org makes it simpler to move your existing applications
•High-quality support, consistently outperforming the other two solutions in customer surveys
2. IBM WebSphere
IBM promotes its solution WebSphere Application Server as a high-performance, feature-rich option – the Ferrari to Red Hat's Ford Focus, if you will. But are the advantages worth the higher price tag?
Benefits of IBM WebSphere
•Rapid installation, deployment and development
•Support for on-premises, cloud-based and hybrid solutions
•Integration with other IBM products in the cloud, such as the Watson artificial intelligence and the dashDB SQL database service
•Flexible, robust architecture that can scale to match demand

Q13. Compare and contrast between COM/DCOM and CORBA.

Ans:-

Basic Characteristics	COM / DCOM	CORBA
Inheritance of a base interface	Every object implements IUnknown	Every interface inherits from CORBA.object
Unique identification of a remote server object	Through its interface pointer	Through an object reference (objref)
Unique identification of an interface	Using the concept of Interface IDs (IIDs)	Using the interface name
Unique identification of a named implementation of a server object	Using the concept of Class IDs (CLSIDs) the mapping of which is found in the registry	By the mapping to a name in the implementation repository
Reference generation of the remote server object	Performed on the wire protocol by the Object Exporter	Performed on the wire protocol by the Object Adapter
Handling of common tasks like object registration, skeleton instantiation, etc.	Either explicitly performed by the server program or handled dynamically by the DCOM run-time system	Performed implicitly by the constructor
Underlying remoting protocol	Object Remote Procedure Call (ORPC)	Internet Inter-ORB Protocol (IIOP)
Activation of a server object	Mainly by using CoCreateInstance()	Mainly by binding to a naming or a trader service
Mapping of object name to its implementation	Handled by the registry	Handled by the implementation repository
Storage of type information	Type library	Interface repository
Client side stub	Called a proxy	Called a proxy or stub
Server side stub	Called a stub	Called a skeleton
Definition of parameters passed between the client and server objects	Defined in the interface at the interface definition file. Depending on what the IDL specifies, parameters are passed either by value or by reference	All interface types are passed by reference. All other objects are passed by value including highly complex data types
Definition of complex types	Complex types that will cross interface boundaries must be declared in the IDL	Complex types that will cross interface boundaries must be declared in the IDL

Support of distributed garbage collection	On the wire by a pinging mechanism which garbage collects remote object references and encapsulates them in the IOXIDResolver interface	No
Platform support	Any platform as long as there is a COM service implementation for that platform	Any platform as long as there is a CORBA ORB implementation for that platform
Programming language support	Since the specification is at the binary level, diverse programming languages can be used	Since it is just a specification, diverse programming languages can be used, as long as there are ORB libraries suitable for coding in a specific language.

Q14. Difference between OLE and COM.

Ans:- OLE(Object Linking and Embedding): This is a method of linking parts of one document to parts of another. For example, having a PowerPoint slide with an Excel chart embedded into it. When the Excel spreadsheet is updated, the chart should update too. When you reopen PowerPoint, magically it has! (This example is a linked object.) Embedded objects are the same only the Excel spreadsheet doesn't exist in an external file, the data for the spreadsheet is contained within the PowerPoint file.

You can embed Excel, Word and PowerPoint documents into each other with linked objects. Other applications were written specifically to support being embedded into Word, such as Microsoft Equation Editor.

OLE Version 1 was built on DDE(Dynamic Data Exchange), which used window messages to notify applications when source data changed, and typically passed data around by using HGLOBAL global memory handles.

OLE Version 2 was built on COM(Component Object Model).

COM is an language neutral, object-oriented component model and ABI(Application Binary Interface) based on DCE(Distributed Computing Equipments) RPC(Remote Procedure Call). As an RPC system, it supported remote calls between processes on the same machine, and later, with DCOM, on different machines. Initially COM was used as part of the architecture of MAPI(Messaging Application Programming Interface) which uses the COM object model but not the COM registration services before being formally launched on its own as a general object model complete with registry and object activation and other services. (Monikers and structured storage for example.)

OLE Automation has nothing to do with OLE - it's a branding connection only. OLE Automation is a Visual Basic-compatible subset of COM which supports basic data types only (for example no unsigned integers or structs) but including objects (COM interfaces).

OLE Controls visual components primarily targeted at Visual Basic users from VB 4 onwards, but the visual elements are provided using the embedding facilities of OLE 2. They can also be hosted (in theory, if properly written) by anything capable of hosting an OLE 2 embedded object, and were often used in C++ applications too. They typically use OLE Automation compatible interfaces for programming at runtime.

ActiveX control is a marketing term for COM objects, from the time when Microsoft were attempting to popularize the technology for extending web applications.

Q15. Write short notes on following terms:-

(a) Message queuing (b) Clustering (c) DLL (d) Queued Components.

Ans:- **(a) Message queuing:-** In computer science, message queues and mailboxes are software-engineering components used for inter-process communication (IPC), or for inter-thread communication within the same process. They use a queue for messaging – the passing of control or of content. Group communication systems provide similar kinds of functionality.

Message queuing has been used in data processing for many years. It is most commonly used today in electronic mail. Without queuing, sending an electronic message over long distances requires every node on the route to be available for forwarding messages, and the addressees to be logged on and conscious of the fact that you are trying to send them a message. In a queuing system, messages are stored at intermediate nodes until the system is ready to forward them. At their final destination they are stored in an electronic mailbox until the addressee is ready to read them.

•Messaging means that programs communicate by sending each other data in messages rather than calling each other directly.

•Queuing means that messages are placed on queues in storage, allowing programs to run independently of each other, at different speeds and times, in different locations, and without having a logical connection between them.

(b) Clustering:- Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups.

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labelled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

(c) DLL: - A DLL is a library that contains code and data that can be used by more than one program at the same time. For example, in Windows operating systems, the Comdlg32 DLL performs common

dialog box related functions. Therefore, each program can use the functionality that is contained in this DLL to implement an Open dialog box. This helps promote code reuse and efficient memory usage.

The following list describes some of the files that are implemented as DLLs in Windows operating systems:

•ActiveX Controls (.ocx) files: An example of an ActiveX control is a calendar control that lets you select a date from a calendar.

•Control Panel (.cpl) files: An example of a .cpl file is an item that is located in Control Panel. Each item is a specialized DLL.

•Device driver (.drv) files: An example of a device driver is a printer driver that controls the printing to a printer.

(d) Queued Components:- The COM+ queued components service enhances the COM programming model by providing an environment in which a component can be invoked either synchronously (real-time) or asynchronously (queued). A component need not be aware of whether it is employed in a real-time or a queued context.

The COM+ queued components service consists of the following parts:

•Recorder (for the client or send side) : The recorder marshals the client's security context into the message and records all of the client's method calls. In its role as proxy for the server component, the recorder selects interfaces from the queuable interfaces in the COM+ catalog.

•Listener (for the server or receive side): The queued components listener retrieves the message from the queue and passes it to the queued components player.

•Player (for the server or receive side): The player unmarshals the client's security context at the server side and then invokes the server component and makes the same method calls. The method calls are not played back by the player until the client component completes and the transaction that recorded the method calls commits.