**We have**

**1) Two tier architecture:**  
**2) Three tier architecture:**  
**3)N- tier architecture:**

**Two tier architecture:**  
Two tier architecture is similar to a basic **client-server** model. The application at the client end directly communicates with the database at the server side. API’s like ODBC,JDBC are used for this interaction. The server side is responsible for providing query processing and transaction management functionalities. On the client side, the user interfaces and application programs are run. The application on the client side establishes a connection with the server side in order to communicate with the DBMS.  
An advantage of this type is that maintenance and understanding is easier, compatible with existing systems. However this model gives poor performance when there are a large number of users.

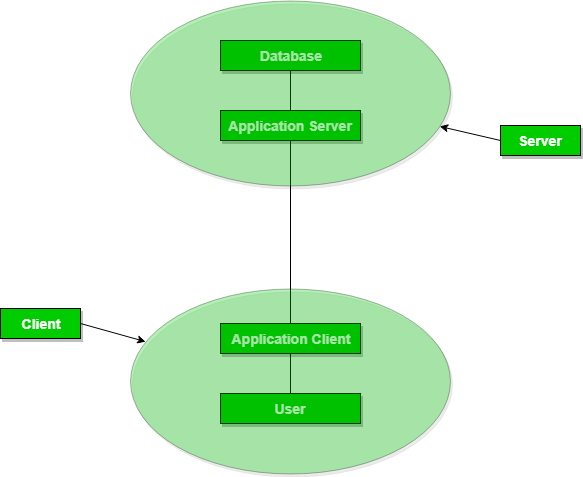


**Three Tier architecture:**  
In this type, there is another layer between the client and the server. The client does not directly communicate with the server. Instead, it interacts with an application server which further communicates with the database system and then the query processing and transaction management takes place. This intermediate layer acts as a medium for exchange of partially processed data between server and client. This type of architecture is used in case of large web applications.  
**Advantages:**

* **Enhanced scalability** due to distributed deployment of application servers. Now, individual connections need not be made between client and server.
* **Data Integrity** is maintained. Since there is a middle layer between client and server, data corruption can be avoided/removed.
* **Security** is improved. This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.

**Disadvantages**:  
Increased complexity of implementation and communication. It becomes difficult for this sort of interaction to take place due to presence of middle layers.

**Three-Tier Architecture**



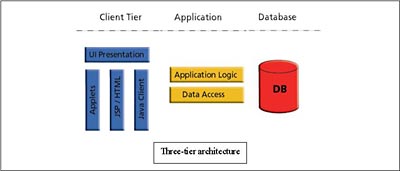
**3-tier application architecture**

A 3-tier application architecture is a modular client-server architecture that consists of a presentation tier, an application tier and a data tier. The data tier stores information, the application tier handles logic and the presentation tier is a graphical user interface ([GUI](https://searchwindevelopment.techtarget.com/definition/GUI)) that communicates with the other two tiers. The three tiers are logical, not physical, and may or may not run on the same physical server.

Presentation tier - This tier, which is built with [HTML5](https://whatis.techtarget.com/definition/HTML5), cascading style sheets ([CSS](https://www.theserverside.com/definition/cascading-style-sheet-CSS)) and [JavaScript](https://www.theserverside.com/definition/JavaScript), is deployed to a computing device through a web browser or a web-based application. The presentation tier communicates with the other tiers through application program interface ([API](https://searchapparchitecture.techtarget.com/definition/application-program-interface-API)) calls.

Application tier - The application tier, which may also be referred to as the logic tier, is written in a programming language such as [Java](https://www.theserverside.com/definition/JavaScript) and contains the [business logic](https://whatis.techtarget.com/definition/business-logic) that supports the application’s core functions. The underlying application tier can either be hosted on distributed servers in the cloud or on a dedicated in-house server, depending on how much processing power the application requires.

Data tier - The data tier consists of a database and a program for managing read and write access to a database. This tier may also be referred to as the storage tier and can be hosted on-premises or in the cloud. Popular database systems for managing read/write access include [MySQL](https://searchoracle.techtarget.com/definition/MySQL), PostgreSQL, Microsoft SQL Server and [MongoDB](https://searchdatamanagement.techtarget.com/definition/MongoDB).



**Benefits of a 3-tier app architecture**

The benefits of using a 3-layer architecture include improved [horizontal scalability](https://searchcio.techtarget.com/definition/horizontal-scalability), performance and availability. With three tiers, each part can be developed concurrently by different team of programmers coding in different languages from the other tier developers. Because the programming for a tier can be changed or relocated without affecting the other tiers, the 3-tier model makes it easier for an enterprise or software packager to continually evolve an application as new needs and opportunities arise. Existing applications or critical parts can be permanently or temporarily retained and encapsulated within the new tier of which it becomes a component.

3-tier Architecture

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

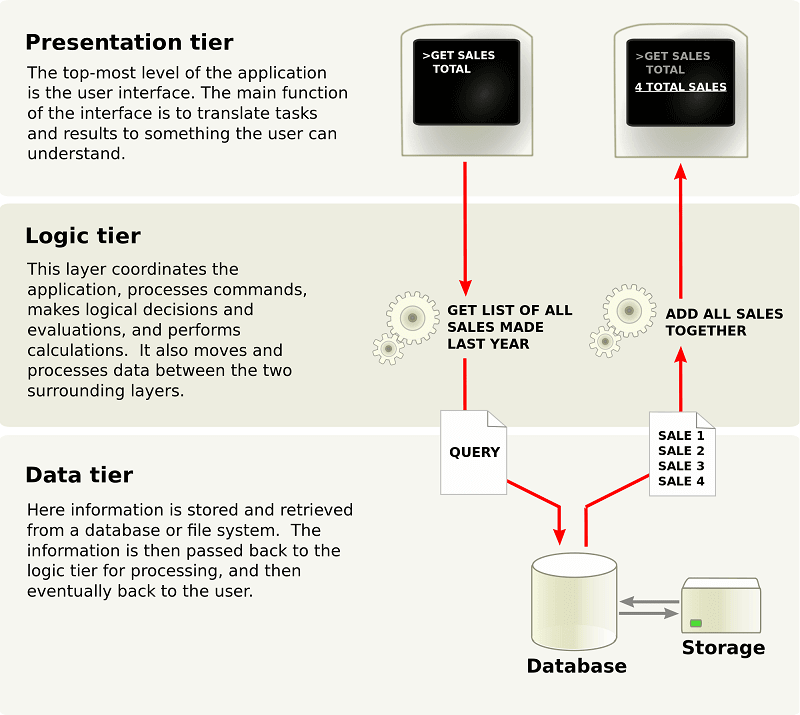


* **Database (Data) Tier** − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
* **Application (Middle) Tier** − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
* **User (Presentation) Tier** − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

N-tier architecture would involve dividing an application into [three different tiers](https://msdn.microsoft.com/en-us/library/bb384398.aspx).  These would be the

1. logic tier,
2. the presentation tier, and
3. the data tier.



The separate physical location of these tiers is what differentiates n-tier architecture from the model-view-controller framework that only separates presentation, logic, and data tiers in concept.  N-tier architecture also differs from MVC framework in that the former has a middle layer or a logic tier, which facilitates all communications between the different tiers.  When you use the MVC framework, the interaction that happens is triangular; instead of going through the logic tier, it is the control layer that accesses the model and view layers, while the model layer accesses the view layer.  Additionally, the control layer makes a model using the requirements and then pushes that model into the view layer.

This is not to say that you can only use either the MVC framework or the n-tier architecture.  There are a lot of software that brings together these two frameworks.  For instance, you can use the n-tier architecture as the overall architecture, or use the MVC framework in the presentation tier.

### What are the Benefits of N-Tier Architecture?

There are several benefits to using n-tier architecture for your software.  These are scalability, ease of management, flexibility, and security.

* **Secure:** You can secure each of the three tiers separately using different methods.
* **Easy to manage:** You can manage each tier separately, adding or modifying each tier without affecting the other tiers.
* **Scalable:** If you need to add more resources, you can do it per tier, without affecting the other tiers.
* **Flexible:** Apart from isolated scalability, you can also [expand each tier](https://www.codeproject.com/Articles/430014/N-Tier-Architecture-and-Tips#Tier%20and%20Layer%20Relationship) in any manner that your requirements dictate.

In short, with n-tier architecture, you can adopt new technologies and add more components without having to rewrite the entire application or redesigning your whole software, thus making it easier to scale or maintain.  Meanwhile, in terms of security, you can store sensitive or confidential information in the logic tier, keeping it away from the presentation tier, thus making it more secure.

Other benefits include:

* **More efficient development.** N-tier architecture is very friendly for development, as different teams may work on each tier.  This way, you can be sure the design and presentation professionals work on the presentation tier and the database experts work on the data tier.
* **Easy to add new features.** If you want to introduce a new feature, you can add it to the appropriate tier without affecting the other tiers.
* **Easy to reuse.** Because the application is divided into independent tiers, you can easily reuse each tier for other software projects.  For instance, if you want to use the same program, but for a different data set, you can just replicate the logic and presentation tiers and then create a new data tier.

### How It Works and Examples of N-Tier Architecture

When it comes to n-tier architecture, a [three-tier architecture](http://tutorials.jenkov.com/software-architecture/n-tier-architecture.html) is fairly common.  In this setup, you have the presentation or GUI tier, the data layer, and the application logic tier.

**The application logic tier.**  The application logic tier is where all the “thinking” happens, and it knows what is allowed by your application and what is possible, and it makes other decisions.  This logic tier is also the one that writes and reads data into the data tier.

**The data tier.** The data tier is where all the data used in your application are stored.  You can securely store data on this tier, do transaction, and even search through volumes and volumes of data in a matter of seconds.

**The presentation tier.**  The presentation tier is the user interface.  This is what the software user sees and interacts with.  This is where they enter the needed information.  This tier also acts as a go-between for the data tier and the user, passing on the user’s different actions to the logic tier.

Just imagine surfing on your favorite website.  The presentation tier is the Web application that you see.  It is shown on a Web browser you access from your computer, and it has the CSS, JavaScript, and HTML codes that allow you to make sense of the Web application.  If you need to log in, the presentation tier will show you boxes for username, password, and the submit button.  After filling out and then submitting the form, all that will be passed on to the logic tier.  The logic tier will have the JSP, Java Servlets, Ruby, PHP and other programs.  The logic tier would be run on a Web server.  And in this example, the data tier would be some sort of database, such as a MySQL, NoSQL, or PostgreSQL database.  All of these are run on a separate database server.  Rich Internet applications and mobile apps also follow the same three-tier architecture.

And there are n-tier architecture models that have [more than three tiers](http://stackoverflow.com/questions/10761440/example-of-4-tier-for-n-tier-architecture).  Examples are applications that have these tiers:

* **Services**– such as print, directory, or database services
* **Business domain** – the tier that would host Java, DCOM, CORBA, and other application server object.
* **Presentation tier**
* **Client tier** – or the thin clients

One good instance is when you have an enterprise service-oriented architecture.  The enterprise service bus or ESB would be there as a separate tier to facilitate the communication of the basic service tier and the business domain tier.

### Considerations for Using N-Tier Architecture for Your Applications

Because you are going to work with several tiers, you need to make sure that network bandwidth and hardware are fast.  If not, the application’s performance might be slow.  Also, this would mean that you would have to pay more for the network, the hardware, and the maintenance needed to ensure that you have better network bandwidth.

Also, use as fewer tiers as possible.  Remember that each tier you add to your software or project means an added layer of complexity, more hardware to purchase, as well as higher maintenance and deployment costs.  To make your n-tier applications make sense, it should have the minimum number of tiers needed to still enjoy the scalability, security and other benefits brought about by using this architecture.  If you need only three tiers, don’t deploy four or more tiers.

**log troubleshooting -  server and application log files**

Sometime we need to record something after we give the software to our customers, so when something bad happened, we can find it out without the debug environment. Sometime we want to get some trace information in the release mode of the program. In these circumstances, a log file is useful.

------------------------------------------------------------------------------------

import org.apache.log4j.\*;

public class LogClass {

private static org.apache.log4j.Logger log = Logger.getLogger(LogClass.class);

public static void main(String[] args) {

log.setLevel(Level.WARN);

log.trace("Trace Message!");

log.debug("Debug Message!");

log.info("Info Message!");

log.warn("Warn Message!");

log.error("Error Message!");

log.fatal("Fatal Message!");

}

}