Introduction to Multi-Tier Configurations

A 3-tier application is an application program that is organized into three major parts, each of which is distributed to a different place or places in a network. The three parts are:

- The workstation or presentation interface
- The business logic
- The database and programming related to managing it

In a typical 3-tier application, the application user's workstation contains the programming that provides the graphical user interface (GUI) and application-specific entry forms or interactive windows.

LEARN MORE

- Software Quality Resources
- Software Development Fundamentals

(Some data that is local or unique for the workstation user is also kept on the local hard disk.)

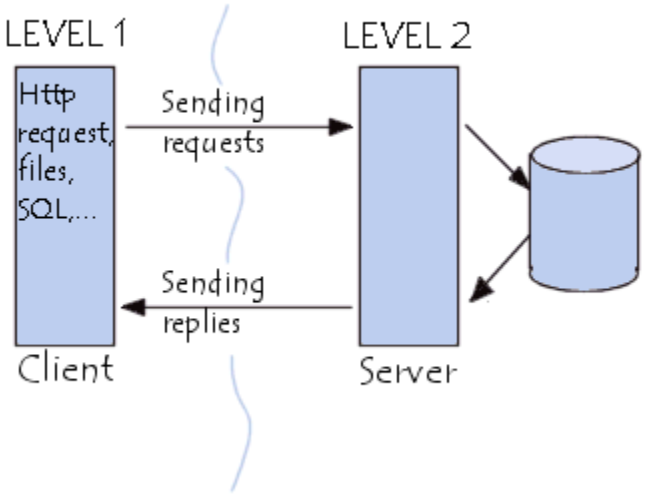
Business logic is located on a local area network (LAN) server or other shared computer. The business logic acts as the server for client requests from workstations. In turn, it determines what data is needed (and where it is located) and acts as a client in relation to a third tier of programming that might be located on a mainframe computer.

The third tier includes the database and a program to manage read and write access to it. While the organization of an application can be more complicated than this, the 3-tier view is a convenient way to think about the parts in a large-scale program.

A 3-tier application uses the client/server computing model. With three tiers or parts, 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.

Introduction to 2-Tier Architecture

2-tier architecture is used to describe client/server systems where the client requests resources and the server responds directly to the request, using its own resources. This means that the server does not call on another application in order to provide part of the service.



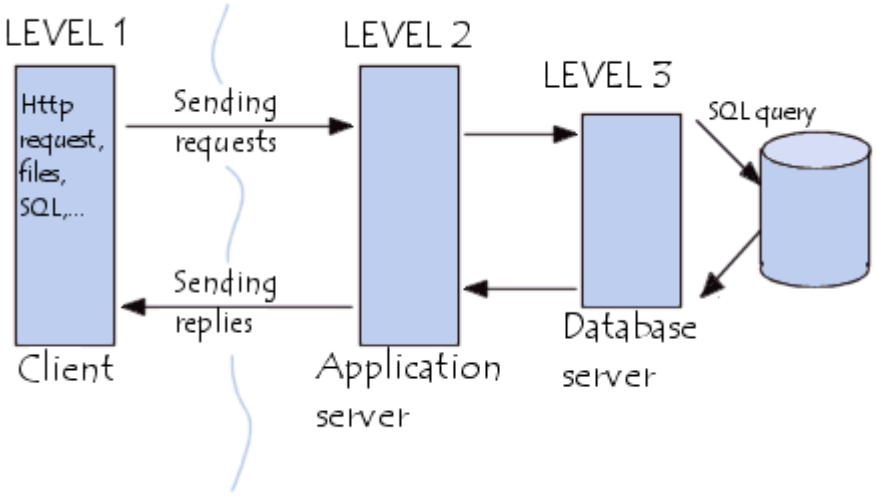
Introduction to 3-Tier Architecture

In 3-tier architecture, there is an intermediary level, meaning the architecture is generally split up between:

A client, i.e. the computer, which requests the resources, equipped with a user interface (usually a web browser) for presentation purposes

The application server (also called middleware), whose task it is to provide the requested resources, but by calling on another server

The data server, which provides the application server with the data it requires



Comparing both types of architecture

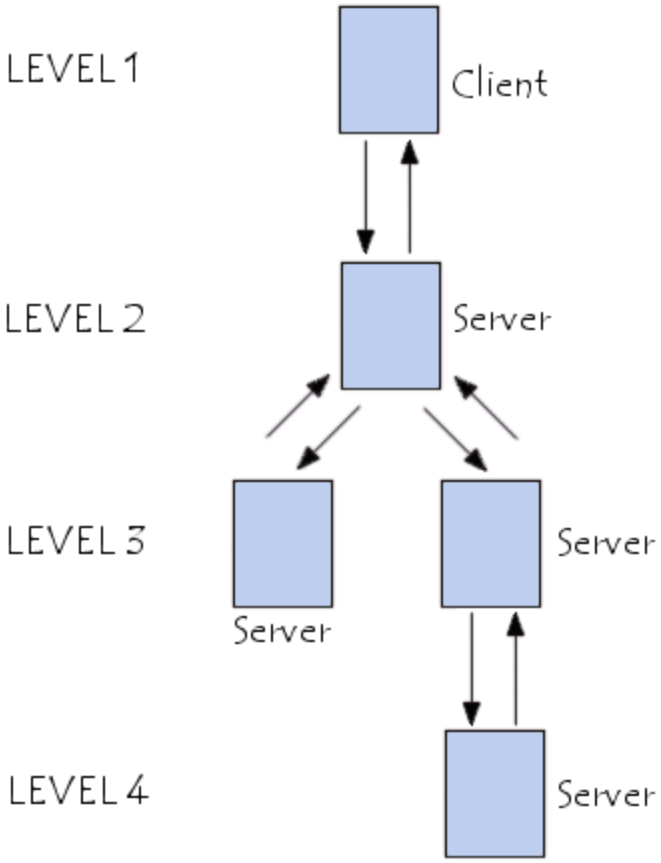
2-tier architecture is therefore a client-server architecture where the server is versatile, i.e. it is capable of directly responding to all of the client's resource requests.

In 3-tier architecture however, the server-level applications are remote from one another, i.e. each server is specialised with a certain task (for example: web server/database server). 3-tier architecture provides:

- A greater degree of flexibility
- Increased security, as security can be defined for each service, and at each level
- Increased performance, as tasks are shared between servers

Multi-Tiered Architecture

In 3-tier architecture, each server (tier 2 and 3) performs a specialized task (a service). A server can therefore use services from other servers in order to provide its own service. As a result, 3-tier architecture is potentially an n-tiered architecture



http://en.kioskea.net/contents/cs/cs3tier.php3

The 3-Tier Architecture - is it hardware or software?

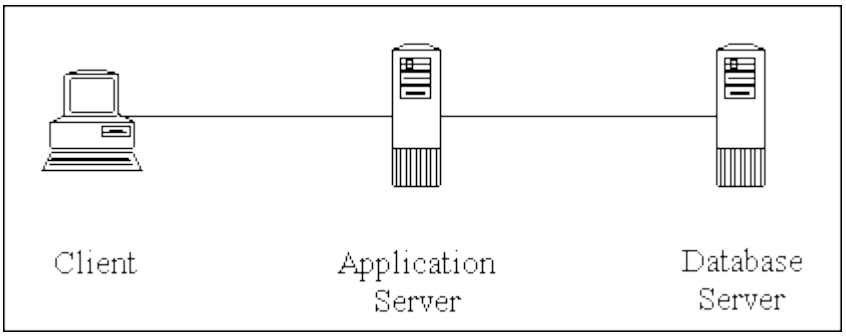
Tony Marston - 24th April 2002

The use of the term '3-Tier architecture' can sometimes be confusing as it may mean different things to different people. In my long years of experience I have come across two different meanings - one hardware oriented (physical), the other software oriented (logical). Perhaps the next time you get into a discussion with someone on this topic it may help to check that you are both talking about the same view.

The Hardware (physical) view

When viewed from the hardware perspective the 3-Tier architecture consists of 3 device layers, as shown in Figure 1:

Figure 1 - 3 Tier Architecture (hardware view)



In this configuration all Forms (.FRM files) are executed on the Client, while all Services (.SVC) and Reports (.RPT) are executed on the Application Server. It should be noted here that Services and Reports can be executed on an Application Server because they do not have any dialog with the user. It is also advisable that they be compiled as 'self-contained' components in order to avoid any problems with global objects. To quote from the Uniface documentation:

A component running remotely executes in the environment of its Application Server. This means that it accesses the DOL or UOBJ.TEXT located in the installation directory for that Application Server. This does not necessarily contain the same global objects available to the client application. When a service or report that is not self-contained runs in the Application Server environment, the difference in global objects can lead to problems that are difficult to detect.

In order for this configuration to work the Application Server requires special software (also known as Application Server or ASV). This communicates with the Client through a Message Daemon. Synchronous communication provides a separate instance of the ASV for each Client, while Asynchronous communication allows a single ASV to communicate with multiple Clients.

This configuration has the following characteristics:

All service and report components are held on the application server.

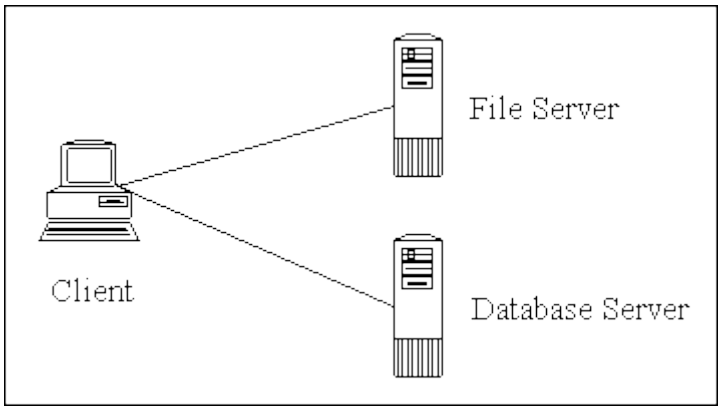
All service and report components are executed on the application server.

The advantage of this configuration is that the service and report components execute at the speed of the Server device, not the Client. The network traffic between the Client and the Application Server may also be a lot less than the network traffic between the Application Server and the Database Server.

A False view

I have also come across a configuration which has a File Server instead of an Application Server, as shown in Figure 2, but this is not a true 3-Tier configuration.

Figure 2 - with File Server instead of Application Server



This configuration has the following characteristics:

All components (form, service and report) are held on the file server.

All components are executed on the client device.

The advantage of this configuration is that when components need to be updated the updates need only be applied to the File Server instead of all the Client devices.

The disadvantage is that when a component needs to be accessed by a Client device it needs to be copied from the File Server into the Client's memory before it can be executed.

The Software (logical) view

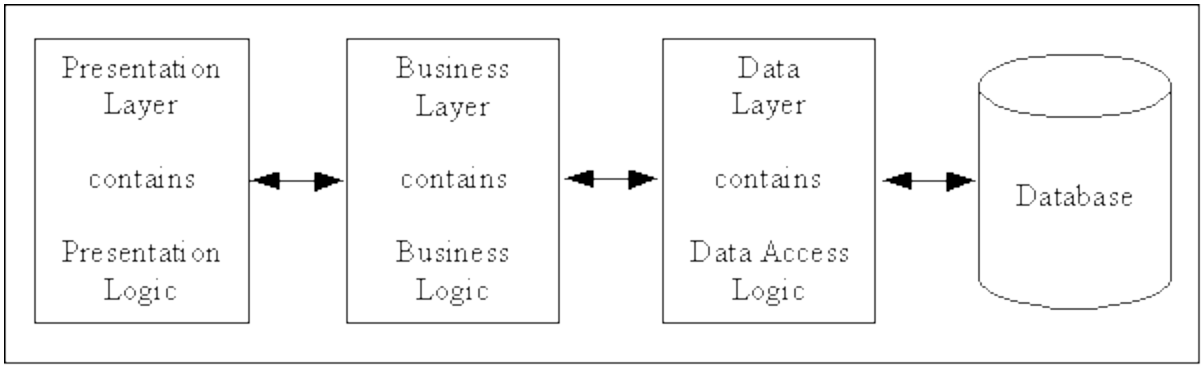
Software is comprised of code that can be broken down into 3 distinct areas:

- Presentation logic = User Interface, displaying data to the user, accepting input from the user.
- Business logic = Data Validation, ensuring the data is kosher before being added to the database.
- Data Access Logic = Database Communication, accessing tables and indices, packing and unpacking data.

This is described in an earlier article entitled UNIFACE and the N-Tier Architecture, where all this code is contained within a single component you have what is known as a 1-Tier system. When utilizing UNIFACE with a database driver all the data access logic is contained within the database driver, therefore this is known as a 2-Tier system. It is possible to change one database driver for another without having to make any changes to any application code.

If the Presentation Logic and Business Logic are split off into separate components you end up with a 3-Tier system, as shown in Figure 3.

Figure 3 - 3 Tier Architecture (software view)



These are some advanced topics that I want to expose you to, however, you will not need to know this material for any exams or assignments.

In this system the component types within each layer are as follows:

- Presentation Layer = forms or server pages.
- Business Layer = services and reports.
- Data Layer = database driver.

The service components can come in the following flavours:

- Session Services = may contain as many entities as is required by the Presentation Layer component.
- Entity Services (formerly known as Object Services) = these deal with only one entity. They come between Session Services and the Data Layer, and are entirely optional.

One of the advantages of this 3-Tier system is that all business logic is centralised in one layer. A component in the Business Layer can be accessed by any number of components in the Presentation Layer, therefore any changes to business logic can be made in one place and be automatically inherited by all other components without having to duplicate the change in those other components.

Note that the Presentation Layer components do not access the database - all data is provided by the Business Layer in the form of XML streams. Any changes made in the Presentation Layer need to be passed back to the Business Layer before they can be applied to the database. This particular architecture was made possible in UNIFACE 7.2.06 with the appearance of the following commands:

- xmlsave - transfer data from the component's structure into an XML stream.
- xmlload - transfer data from an XML stream into the component's structure.
- retrieve/reconnect - reconnect any occurrences created by xmlload with the database to determine whether these occurrences are updates or additions.

More details on this are contained in an earlier article entitled 3 Tiers, 2 Models, and XML Streams which also comes with some sample code. Note that this software also demonstrates the use of two different application models - one for the Presentation Layer and one for the Business Layer. This provides a much greater degree of flexibility than the single model approach.

The use of XML streams in this way utilises what are known as 'disconnected record sets' as the occurrences within the Presentation Layer components are not connected to the database. Although this is not essential in a Client/Server application it does not prevent their use. However, disconnected record sets are important with Web applications as the Web is entirely stateless. This method means that the Business layer components can be used by a Presentation Layer which can be either Client/Server or Web (or even both at the same time).

Does 3-Tier Software require 3-Tier Hardware?

The simple answer is No. It is possible to have all types of component (forms, services and reports) stored and executed on the Client device. Indeed, this is how I built and tested my sample software.

However, as this architecture has all its business logic contained within service components it is an ideal candidate for being deployed on 3 layers of hardware.

Derived from the following websites

Tony Marston
24th April, 2002

http://www.tonymarston.net/uniface/3tierhardsoft.html