Data Access and Business Logic Layers

Most applications reside in the data and business logic layers of distributed networks. To design and implement those layers, one must understand each layer and how the data are exchanged between those layers.

The Data Access Layer

A data access layer (DAL) provides access to data stored in the storage like a structured query language (SQL) database. In a typical three-tier architecture, DAL is located in the middle layer as a sublayer to business logic, as shown below:

![Diagram of three-tier architecture with presentation tier, business logic layer, data access layer, and data tier]

Some system architects consider the business logic and data access layers to be the same from the business-object point of view; they prefer to put data access logic into business objects. In some situations, like distributed systems with one database, it is nice to combine the two logics and keep everything in one package. However, when a system needs to support multiple database systems—for example, both Oracle and SQL database servers—the entire middle tier (including the business logic objects) must be changed.

Separating the data access layer from the business logic layer allows one to switch databases without modifying the business logic layer. Any other changes in the data access layer would not affect the business logic layer because the data access logic is only in the data access layer.

Accessing Data

To access data from a database, database commands such as select, insert, update, and delete may be used. These commands can be packed as a single
Data Access and Business Logic Layers

DAL method, which can then be called from business logic layer.

When using object-oriented programming, database commands do not need to be used directly in lines of code. Instead, a class can be created, with the database commands written inside a method within the class, which then puts the retrieved data into the class' attributes.

An object is created as a new instance of a class, having all the properties of that class. When that object's attribute is called (object->attribute or object.attribute), the attribute tells the method to do the work, issue the database command, and put the data into that attribute, thus providing the data automatically.

By creating a class with methods that issue database commands, which put the data into the class's attributes, data can be accessed by directly referencing an object's attributes.

Business Logic Layer

The business logic layer deals with everything between the user interface (presentation layer) and a database (data layer). The business logic layer contains business rules or processing logic for specific business instances. The business logic layer should be separated from other layers so that changes do not affect other layers. The business logic layer design and development depend on the specific distributed system.

The business logic layer is often broken into two sublayers: the up-layer and the low-layer, as shown below:
Data Access and Business Logic Layers

The business logic up-layer communicates with the presentation layer and can deal with user interface directly, including references or knowledge regarding the controls in the user interface. The advantage for separating the up-layer and low-layer is reducing code duplication, especially when there are numerous applications in the presentation layer. In this situation, there are multiple user interface controls (such as Windows.NET or Java) that connect to the business logic layer. It is fine for the up-layer to communicate with the data access layer directly. However, communication with the presentation layer should only be from the up-layer. The presentation layer is hidden from the low-layer.

The low-layer classes should not have dependency on or access to objects or controls in the user interface. For example, in .NET programming, whenever a method in the up-layer needs to call a method in the low-layer, the control values like `ListView.Item`, should be passed as parameter values like a string (the value of the item), instead of the object of the Item. If it is passed by reference to the object, it may access the user interface from the low-layer directly. Most communications between the business logic and data access layer are done through the low-layer.