Explain the three-tier architecture. Give examples.

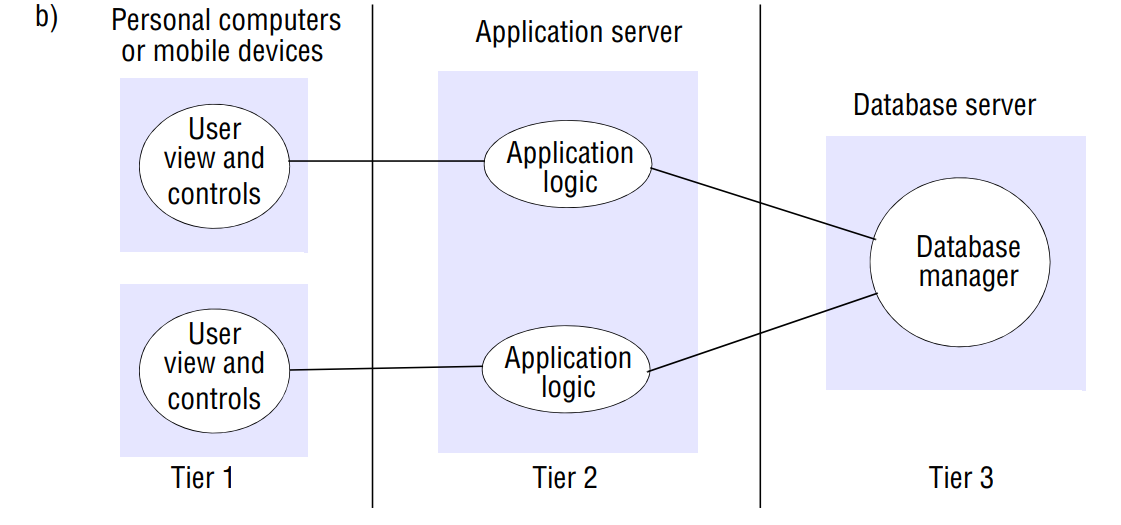
Tiered architecture consists of:

1 • the presentation logic, which is concerned with handling user interaction and updating the view of the application as presented to the user;

2 • the application logic, which is concerned with the detailed application-specific processing associated with the application (also referred to as the business logic, although the concept is not limited only to business applications);

3 • the data logic, which is concerned with the persistent storage of the application, typically in a database management system.

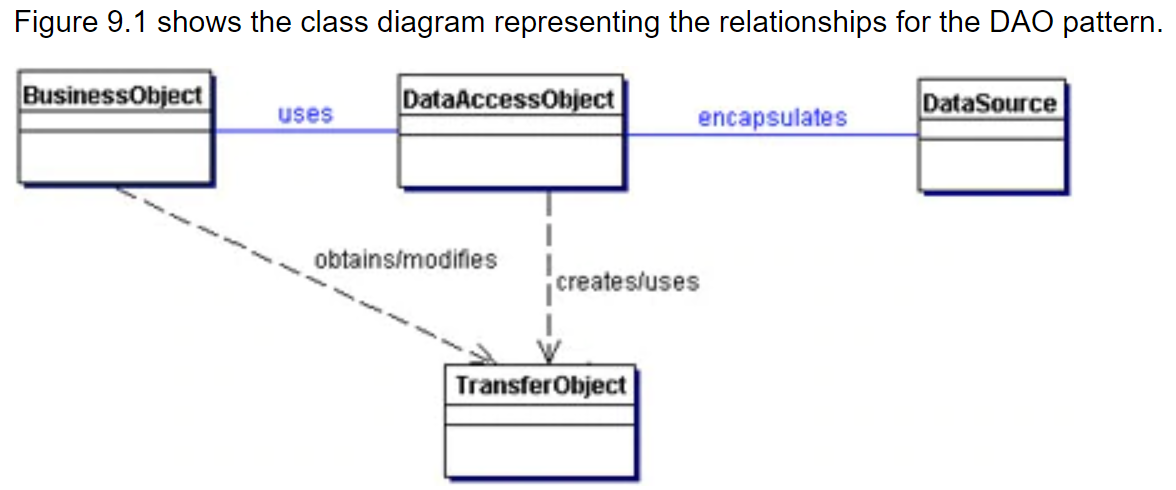
In the three-tier solution, there is a one-to-one mapping from logical elements to physical servers and hence, for example, the application logic is held in one place, which in turn can enhance maintainability of the software. Each tier also has a well-defined role; for example, the third tier is simply a database offering a (potentially standardized) relational service interface. The first tier can also be a simple user interface allowing intrinsic support for thin clients (as discussed below). The drawbacks are the added complexity of managing three servers and also the added network traffic and latency associated with each operation.

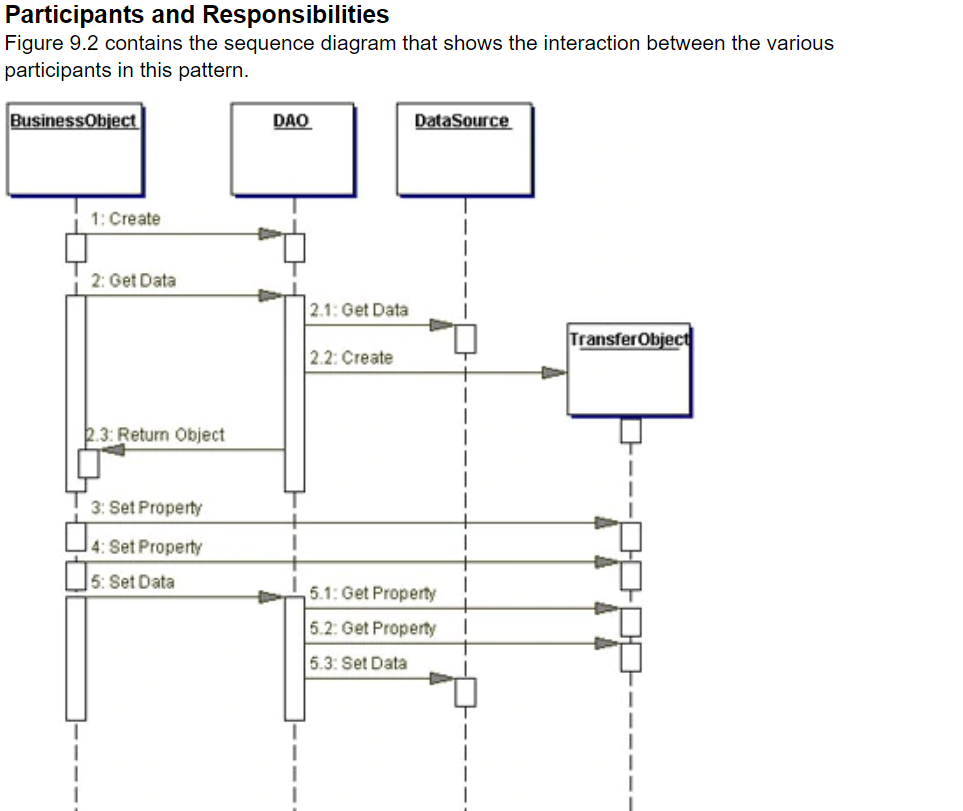


**DAO**

**Use a Data Access Object (DAO) to abstract and encapsulate all access to the data source. The DAO manages the connection with the data source to obtain and store data**

The DAO completely hides the data source implementation details from its clients. Because the interface exposed by the DAO to clients does not change when the underlying data source implementation changes





## Consequences

**Enables Easier Migration**

**Reduces Code Complexity**

**Centralizes All Data Access into a Separate Layer**

**Adds Extra Layer** 

**Transfer Object** is a simple POJO class having getter/setter methods and is serializable so that it can be transferred over the network. It does not have any behavior. Server Side business class normally fetches data from the database and fills the POJO and send it to the client or pass it by value.

**Use a Transfer Object to encapsulate the business data. A single method call is used to send and retrieve the Transfer Object. When the client requests the enterprise bean for the business data, the enterprise bean can construct the Transfer Object, populate it with its attribute values, and pass it by value to the client.**

, the client makes a single remote method invocation to the enterprise bean to request the Transfer Object instead of numerous remote method calls to get individual attribute values. The enterprise bean then constructs a new Transfer Object instance, copies values into the object and returns it to the client. The client receives the Transfer Object and can then invoke accessor (or getter) methods on the Transfer Object to get the individual attribute values from the Transfer Object

all calls to the Transfer Object instance are local calls instead of remote method invocations.

## Consequences

**Simplifies Entity Bean and Remote Interface**   
**Transfers More Data in Fewer Remote Calls**   
**Reduces Network Traffic**, **Reduces Code Duplication** 

