

11.3: Two-tier and Three-tier Architectural Distinctions

Though many people may not realize it, when a group of tech-savvy individuals develops software, one of the most important decisions at the start is the architecture: “architecture is a key feature which decides that the system will meet its performance and other quality objectives” (Hayat and Akram, 2007). Several main components come into play when it comes to deciding the type of architecture, including performance, availability, complexity, and cost.

Performance depends on the quantity of simultaneous users that can be processed by the database system to ensure smooth processing. Typically, most applications limit the number of users accessing different services of the application by only providing them with restricted data. While choosing an architecture, it is quite important to keep in mind the performance component. Another significant factor is the cost. The selected architecture must be affordable for any organization just starting. Availability is the accessibility of the system to the community of users. Complexity is the extent of difficulty of the system in terms of the user’s and developer’s viewpoint. If the developers feel they want a lower level of difficulty, they can make the user’s level of difficulty rise due to lack of user interface organization/improvements. However, to lower the user’s level of difficulty, the developers may have to increase their effort and continuously improve, revise, and contemplate new ways to make the user interface easier and more accessible.

In two-tier architectures, the interfaces run on a client, which means the client communicates with the server directly, and because it runs on a client, another data layer or structure is then stored on a server, as seen in Figure 11.3.1 below. The client's responsibility covers user interface logic from the presentation, which deals with how objects in the business get shown to other users of the software as well as data processing and rules (algorithms) that dictate the information that gets exchanged between the database system and the user interface. Meanwhile, database servers access and process requests from the clients. This architecture allows for optimization of the processing time in the database servers and is highly recommended for small workgroups. However, clients or servers that use two-tier architecture tend only to support a relatively small number of users, because the transaction of information is usually low, and security is not necessarily the top priority. Because of this, two-tier architectures are not used for applications with large user bases due to their limited flexibility.

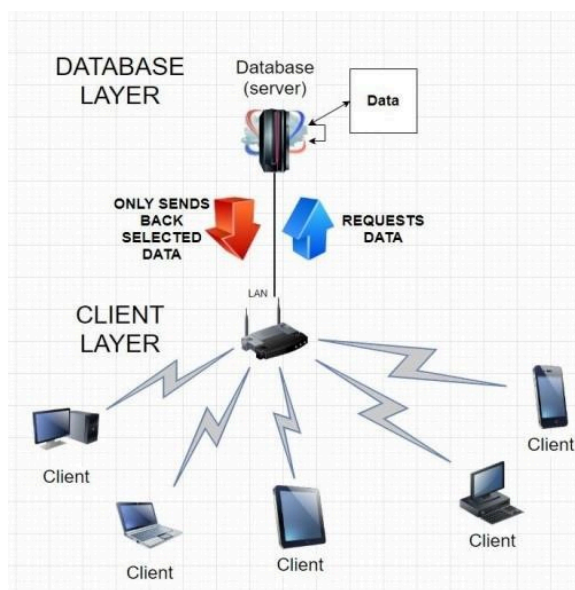


Figure 11.3.1: Visual of two-tier architecture (two layers) (Made via Draw.io, Martin Nguyen, 2020)

In general, three-tier (n-tier) architecture includes another server layer, as seen in Figure 11.3.2. This extra layer makes it much easier to scale the system, which makes performance much better, resulting in a highly flexible and reusable structure. The extra layer can have several purposes, such as an application server, or it can be used as local storage and hold local databases. In contrast, the other servers hold more enclosed data. Utilizing a three-tier architecture can lead to better performance, and make it easier to transport application code to other platforms, reducing dependency on commonly domain-specific language. Three-tier systems can often be found in Web-based applications—any program accessible using HTTP.

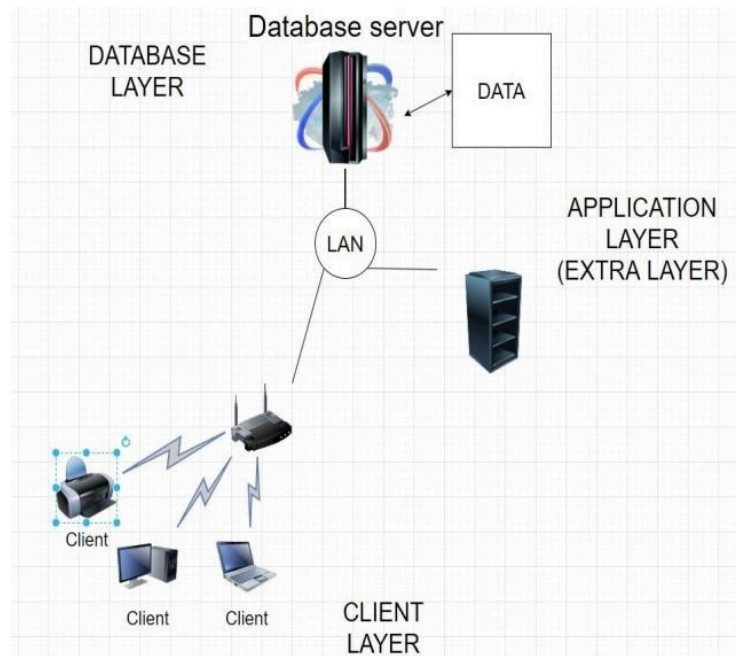


Figure 11.3.2: Visual of three-tier (three or nth layers) (Made via Draw.io, Martin Nguyen, 2020)

11.3: Two-tier and Three-tier Architectural Distinctions is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.