**Single-architecture:**

In a **single-tier** database architecture, the database and application reside on the same system. There is no strict separation between data storage, business logic, and the user interface. Common in small applications, local databases, or inserted systems.

**Implementation Steps:**

1. Install the database on a local system.

2. Develop the application that directly connects to the database.

3. Deploy the system without requiring a network for database access.

**Advantages:**

-Simple to implement and maintain.

-Fast data access since no network communication is involved.

-Ideal for personal or standalone applications.

**Challenges:**

-Poor scalability - difficult to handle large user loads.

-Limited security - if a local system is compromised, all data is exposed.

-Hard to integrate with other applications.

**Comparison with other architectures:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Single-tier Vs. Two-tier** | **Single-tier Vs. Three-tier** |
| **Performance** | Fast (no network overhead ) | Fast but lack of scalability |
| **Scalability** | Poor | Very limited |
| **Security** | Lower | Lower than three-tier |
| **Maintenance** | Easier | Easier but not modular |

**Two-Tier Architecture:**

In a **two-tier** architecture, the application and database reside on separate systems. The client (frontend) communicates directly with the database server without an intermediate layer. Common in client-server applications like traditional desktop apps with backend databases.

**Implementation Steps:**

1. Set up a database server on a remote system.

2. Develop a client-side application that directly connects to the database.

3. Establish a network connection between the client and database.

**Advantages:**

-Better performance than single-tier when handling multiple users.

-Centralized database for easier backup and management.

-Supports moderate scalability.

**Challenges:**

-More network overhead than single-tier.

-Direct client-database interaction can lead to security risks.

-Hard to scale beyond a certain number of users.

**Comparison with other architectures:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Two-tier Vs. Single-tier** | **Two-tier Vs. Three-tier** |
| **Scalability** | Slightly slower | Faser than three-tier |
| **Security** | More scalable | Less scalable |
| **Performance** | Better security | Less secure than three-tier |
| **Maintenance** | Harder than single-tier | Easier to maintain |

**Three-Tier Architecture**

In a **three-tier** architecture, the system is divided into three layers:

Presentation Layer (UI): The frontend user interface.

Application Layer (Business Logic): Processes requests and connects the UI with the database.

Database Layer: Stores and manages data.

Common in enterprise applications, web apps, and cloud-based systems.

**Advantages:**

-High scalability - supports large-scale applications.

-Better security - the client doesn’t directly interact with the database.

-Modularity - easier to maintain and upgrade.

-Supports multiple clients - web, mobile, and desktop apps can connect to the same backend.

**Challenges:**

-More complex to implement and manage.

-Higher latency due to multiple layers of processing.

-Requires more resources and infrastructure.

**Comparison with other architectures:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Three-tier Vs. Single-tier** | **Three-tier Vs. Two-tier** |
| **Scalability** | Slower | Slower |
| **Security** | Highly scalable | More scalable |
| **Performance** | Most secure | More secure |
| **Maintenance** | More modular | More modular |

**Final comparison & Use cases**

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Best for** | **Not suitable for** |
| **Single-tier** | Local applications, standalone system | Multi-user or distributed systems |
| **Two-tier** | Small business, client-server application | Large, enterprise-scale applicatios |
| **Three-tier** | Web apps, enterprise software, cloud-based apps | Simple or resource-limited appplication |