Introduction to SQL Server

Every organization needs to maintain information related to employees, customers, business partners, or business transactions. Organizations build business applications with a user-friendly interface to store and manipulate this information and to generate reports. For this, they need a platform to store and maintain this information in an efficient way. Various Database Management Systems (DBMS) and Relational Database Management Systems (RDBMS), such as SQL Server, Oracle, and Sybase, provide the platforms for storing and maintaining this information.

SQL Server is a database engine introduced by Microsoft. It provides an environment used to create and manage databases. It allows secure and efficient storage and management of data. In addition, it provides other components and services that support the business intelligence platform to generate reports and help in analyzing historical data and predicting future trends. As a database developer, it is important for you to identify the role of a database server in an organization. You can design a database effectively if you know all the components and services of SQL Server. In addition, you need to understand the basics of SQL, a language that is used to query and manage data.

Role of a Database Server

Organizations have always been storing and managing business data. Earlier, organizations used to store data on paper. With an increase in the usage of computers, organizations started maintaining the same information in computers. Data was stored in an organized way, and it was also easy to retrieve data faster than before. As data retrieval became easy and fast, organizations started using business or Web applications to support the business operations.

Consider a scenario. The Human Resource department of an organization uses an application to manage the employee data. The users need to add the details of new employees. For this, the application provides an interface to enter the employee details. These details are validated for accuracy based on business rules. For example, a business rule is defined to check that the date of joining of the new employee is less than or equal to the current date. If the data meets the requirements, it is saved in the data store. Based on the preceding scenario, a business application can have the following elements:

- The User Interface (UI) or the presentation element through which data is entered.
- The application logic or the business rule element, which helps in validating the entered data.

- The data storage or the data management element, which manages the storage and retrieval of data.

These elements form the base of the models or architectures used in application development. All these elements can exist on the same computer as a single process or on different computers as different processes. Depending on the placement of these elements, the application architecture can be categorized as:

- Single-tier architecture
- Two-tier architecture
- Three-tier architecture
- N-tier architecture

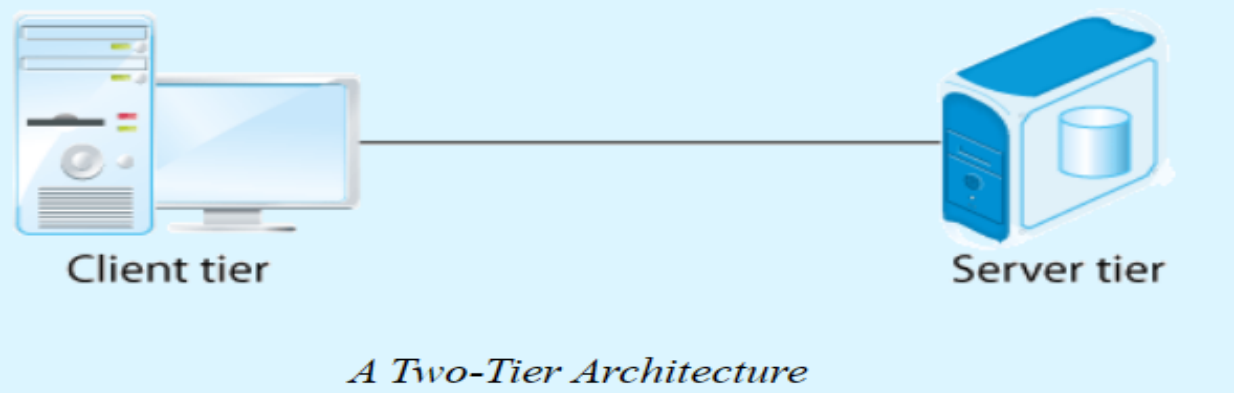
Single-Tier Architecture

In a single-tier architecture, all elements of a business application are combined as a single process. If multiple users need to work on this application then it needs to be installed on the computer of every user. This type of architecture has one disadvantage. In case some errors are identified in the application then after rectifying the same, the application has to be installed again on the system of every user. This is a time-consuming process.

Two-Tier Architecture

To solve the problems of single-tier application, two-tier architecture was introduced. In two-tier architecture, the application is divided into two parts. One part handles the data, while the other provides the user interface. Therefore, this architecture is called two-tier

architecture. These two parts can be located on a single computer or on separate computers over a network. The part that handles the UI is called the client tier. The part that implements the application logic and validates the input data based on the business rules is called the server tier, as shown in the following figure.



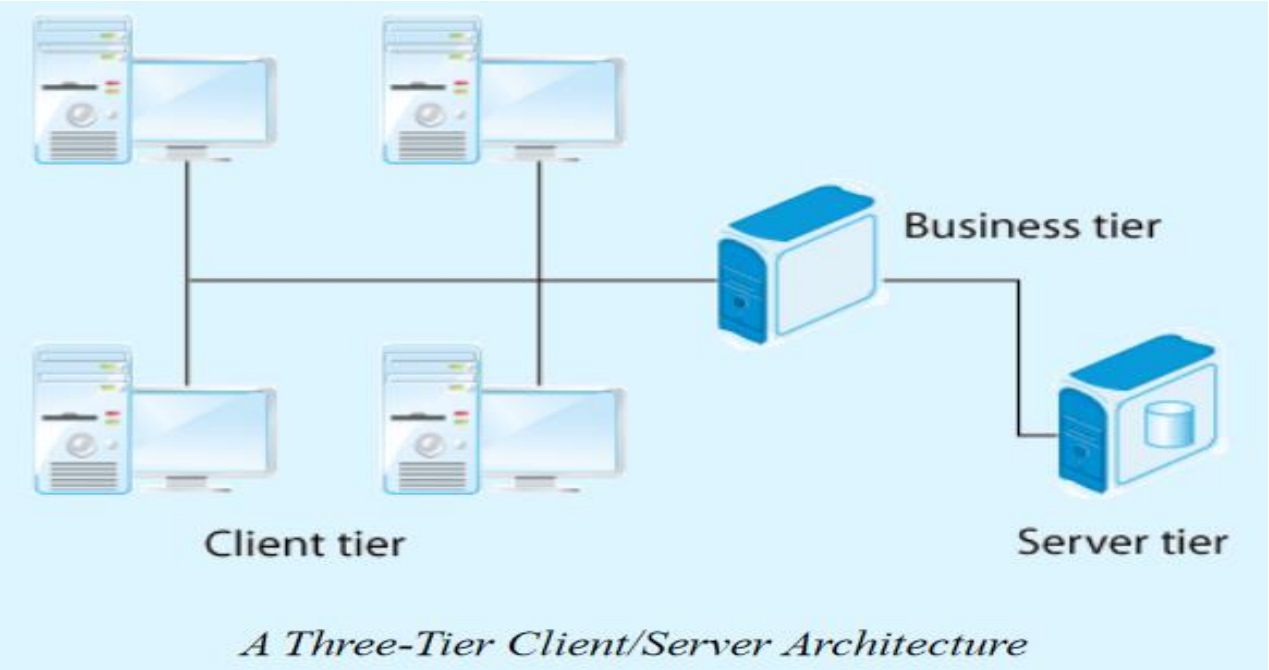
A Two-Tier Architecture

In the preceding architecture, the maintenance, upgrade, and general administration of data is easier, as it exists only on the server and not on all the clients. A two-tier architecture is also called the client-server architecture. A client sends the request for a service and a server provides that service. Most RDBMSs, such as Microsoft Access, SQL Server, and Oracle, support client-server architecture. RDBMS provides centralized functionality while supporting many users.

Three-Tier Architecture

When implementing complex business solutions in a two-tier architecture, the tier on which the business logic is implemented becomes over loaded. As a result, it takes more time to execute. Therefore, to provide further flexibility, the two-tier architecture can be split into three tiers. In three-tier architecture, the first tier is the client tier. The second or the middle tier is called the business tier. The third tier is called the server tier. The server tier contains a

database server that manages the data. In this architecture, an additional tier called a business tier has been added between the client tier and the server tier, as shown in the following figure.



A Three-Tier Client/Server Architecture

The business tier consists of all the business rules. It consists of the application logic that implements business rules and validates the data. The advantage of a three-tier application is that it allows you to change the business rules without affecting the other two tiers. For example, in a banking application for loans, the user tier is the frontend used by the customer to specify the loan details. The server tier can consist of an RDBMS in which the data is stored. The business tier lies between the other two tiers and consists of business rules, such as the loan limit and the interest rate charged to a customer. If there is a change in the rate of interest, only the middle tier component needs to be modified.

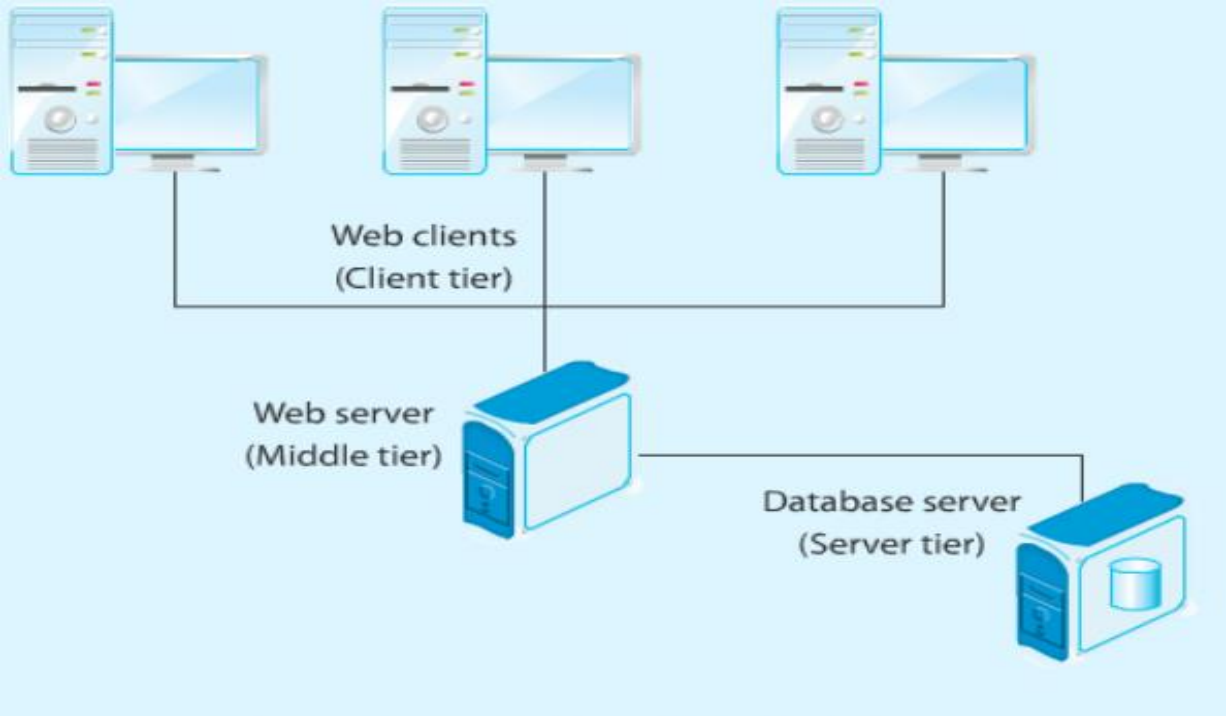
N-Tier Architecture

As the business complexities increased, the business tier became larger and unmanageable. This led to the evolution of n-tier architecture, where the business services model was divided into smaller manageable units. N-tier architecture is also called a multi-tier architecture. In this architecture, one component near the client tier is responsible to do the client side validation and send the data to the presentation tier. Therefore, it is possible to keep the UI-centric processing component on a computer near the client. The UI-centric processing component is responsible for processing the data retrieved from and sent to the presentation tier. In addition, you may have another component near the database server to manipulate and validate the data. You can keep the data-centric processing components on another computer near the database server, so that you gain significant performance benefits. Data-centric processing components are responsible for accessing the data tier to retrieve, modify, and delete data to and from the database server. The n-tier architecture consists of the following layers:

- Client tier
- UI-centric processing components
- Data-centric processing objects
- Database server

The banking application, when further expanded, can represent an example of n tier architecture. The client tier would consist of the user interface, which would include the user interface controls, such as forms, menus, and toolbars. The server tier would consist of data-handling including saving data to the database server. The business

logic would include the rules and guidelines for different types of accounts, interest rates, fixed deposits, ATMs, and loan rules. All of these would form the middle tier. However, there would be some rules that need to be implemented both on the user interface and on the database. You can place these rules either on the UI-centric processing components or data-centric processing components, based on the functionality. Applications that follow multi-tier architecture can be used across various locations. For example, in Web applications, an application is stored on the Web server. The clients access the application from any location through a browser. The clients make requests to the Web server and receive responses. The following figure shows the architecture of the Web applications.



The Architecture of the Web Applications

Depending on the type of business rules, the applications can be implemented on any of the tiers, such as Web clients, Web server, or the database server. To provide support to applications where users

can send requests simultaneously, the database server needs to be fast, reliable, and secure. SQL Server is one such complete database platform that provides a fast, reliable, and secure RDBMS. It also helps in data analysis with integrated BI tools. The BI tools are used to prepare reports that are analyzed further to make efficient business decisions.

SQL Server Editions

SQL Server supports various editions that provide different features targeting different business requirements. SQL Server has the following editions:

- **Enterprise**:

The SQL Server Enterprise edition includes the core database engine and supports the advanced services such as table partitioning, parallel indexing, and indexed views. The Enterprise edition is designed to provide high performance and security, optimized productivity, and high scalability of data.

- Standard:

The SQL Server Standard edition has core database engine and basic business intelligence capabilities. It differs from the Enterprise Edition in that it does not support all the security and data warehousing features of the Enterprise edition. The Standard edition provides easy integration, improved manageability, and easy extension of data across various platforms, such as .Net or Linux.

❑Business Intelligence:

The BI edition has been introduced in SQL Server 2012. It consists of all the Standard Edition capabilities and supports the business intelligence tools, such as Master Data Services and Data Quality Services. The Business Intelligence edition provides rapid exploration of data and greater consistency, manageability, and security of data.

Features of SQL Server

The components of SQL Server help improve the database management and developer productivity by the following features of SQL Server:

❑ **Built-in support for *Extensible Markup Language (XML)*data**:

Allows you to store and manage XML data in variables or columns of the XML data type. The XML feature of SQL Server enables you to write code to retrieve data from the database in the form of XML. In addition, it allows you to read an XML document and store the XML data in the database.

❑ CLR integration:

Allows you to use the CLR features of .NET Framework in the SQL Server database. It enables you to use the code written in any of the .NET supported languages for implementing complex programming logics in the database. For example, you need to write a code for the verification of the credit card number entered by the user. It will be complex to write the code for the verification of the credit card number in T-SQL. However, the same code can be written effectively using a .NET programming language. Therefore, you can write the code

in a .NET programming language and use that code in SQL Server to verify the credit card information.

- **Scalability:**

Allows you to distribute the data in large tables into several filegroups. This enables SQL Server to access all the filegroups simultaneously and retrieve the data quickly. This makes the database scalable and helps improve the performance of queries.

- **Service-oriented architecture:** Provides distributed and asynchronous application framework for large-scale applications. This allows the database clients to send requests to the database server even if the server is not available to process the request immediately.

- **Support for Web services:** Allows you to provide direct access to the data from the Web services by implementing the HTTP endpoints. For example, sales executives of an organization need to access the data on the database server through their Personal Desktop Assistant (PDA) devices. However, providing direct access from a PDA device to the database server involves a high cost. Therefore, organization can implement Web services through which each sales executive can log the sales details online from anywhere using any device.

- **High level of security**: Implements high security by enforcing policies for log on passwords. Administrators can also manage permissions on database objects granted to different users.

- **High availability:** Ensures that the database server is available to all users at all times. This reduces the downtime of the server. In SQL Server, high availability is implemented with the help of database mirroring, failover clustering, and database snapshots.

- **Support for data migration and analysis:** Provides tools to migrate data from different data sources to a common database. In addition, it allows building the data warehouse on this data that can support BI applications for data analysis and decision-making.
- **Intellisense:** Provides the feature of auto completion of code written to create or manipulate database objects in the Query Editor window.
- **Policy-based management:**

Used to define a set of policies for configuring and managing SQL Server. For example, you can define a policy to set a naming convention for tables and stored procedures. When a user tries to create a table, the table name must map with the naming convention defined in the policy, else an error will be raised.

- **Resource governor:**

Used to manage the workload of SQL Server by allocating and managing the server resources, such as CPU time and memory. The resource pool represents the server resource. You can specify the minimum and maximum values of the CPU and memory utilization in a resource pool. These resources are used for running and performing various assigned tasks in SQL Server.

Types of SQL Statements

As a database developer, you need to manage the database to store, access, and modify data. SQL is the core language used to perform these operations on the data. SQL, pronounced as 'sequel', is a language that is used to manage data in an RDBMS. This language was developed by IBM in the 1970s. It follows the International Organization for Standardization (ISO) and American National Standards Institute (ANSI) standards. Most database systems have created customized versions of the SQL language. For example, Transact-SQL (T-SQL) is a scripting language used in SQL Server for programming. The applications may have different user interfaces but have a common way to communicate with SQL Server by sending T-SQL statements to the server. The SQL statements can be divided into the following categories:

❑ **Data Definition Language (DDL)**: Is used to define the database, data types, structures, and constraints on the data. Some of the DDL statements are:

- **CREATE**: Used to create a new database object, such as a table.
- **ALTER**: Used to modify the database objects.
- **DROP**: Used to delete the objects.

❑ **Data Manipulation Language (DML):**

Is used to manipulate the data in database objects. Some of the DML statements are:

- **INSERT**: Used to insert a new data record in a table.
- **UPDATE**: Used to modify an existing record in a table.

• **DELETE**: Used to delete a record from a table.

❑ Data Control Language (DCL):

Is used to control the data access in the database. Some of the DCL statements are:

- **GRANT**: Used to assign permissions to users to access a database object.
- **REVOKE**: Used to deny permissions to users to access a database object.

❑ Data Query Language (DQL):

Is used to query data from database objects. SELECT is the DQL statement that is used to select data from the database in different ways and formats. SQL is not a case-sensitive language. Therefore, you can write the statements in any case, lowercase or uppercase. For example, you can use the SELECT statement in lowercase as 'select' or in title case as 'Select'.

Services by implementing the HTTP endpoints. For example, sales executives of an organization need to access the data on the database server through their Personal Desktop Assistant (PDA) devices. However, providing direct access from a PDA device to the database server involves a high cost. Therefore, organization can implement Web services through which each sales executive can log the sales details online from anywhere using any device.