Characteristics of DBMS

DBMS stands for Data Base Management System. It is a set of computer programs that are used for the creation and modification of a database. It is a software integrity package. The Data Base Management System also acts as an intermediate between the end user and the Database. It also establishes an environment for multiuser to create, access, and manipulate the data in the Database.

Characteristics of DBMS

Some well-known characteristics are present in the DBMS (Database Management System). These are explained below.

1. Real World Entity

The reality of DBMS (Database Management System) is one of the most important and easily understandable characteristics. The DBMS (Database Management System) is developed in such a way that it can manage huge business organizations and store their

business data with security.

* In DBMS (Database Management System), the entities look like real-world entities.
* For example, if we want to create a student database, we need some entity. Any student stores their data.
* In the Database, then, it should be the real-world entity. The most commonly used properties in the student database are name, age, gender, roll number, etc.

2. Self-explaining nature

* In DBMS (Database Management System), the Database contains another database, and another database also contains metadata.
* Here the term metadata means data about data.
* For example, in a school database, the total number of rows and the table's name are examples of metadata.
* So the self-explaining nature means the Database explains all the information automatically itself. This is because, in the Database, all the data are stored in a structured format.

3. Atomicity of Operations (Transactions)

* Here, atomicity means either the operation should be performed or not performed. i.e., it should complete the operation on 0% or 100%.
* Here DBMS (Database Management System) provides atomicity as a characteristic. This is the most important and useful characteristic of the DBMS (Database Management System). You can completely understand the atomicity with the help of the below example.
* For example, every bank has its own Database, and the Database contains all the information about its customers. Let transaction is the most common atomic operation of the bank. If Sona wants to transfer 1000 rupees to the Archita account, it is possible with the help of the atomicity feature of the Database. If there is a problem in the Archita account, if there is a problem in the atomicity of the Database, then the money will be deducted from the Sona account but not credited to the Archita account.
* The Database has the feature of atomicity then; such transactions have not occurred at all, and if the transaction fails, then the money will automatically return to the sender account.
* Basically, for a successful transaction, the total operation depends on the Database. If the Database works perfectly, the transaction will be successful, and if the Database fails, the whole banking server will be down.

4. Concurrent Access without Anomalies

* Here the term anomalies mean multiuser can access the Database and fetch the information without any problem.
* For a better understanding, let's take the example of a bank again. Let Sonu give his ATM card to his sister Archita and tell her to withdraw 5000 from the ATM. At the same time, Sonu transferred 2000 rupees to his brother Monu. At the same time, both operations perform successfully. Initially, Sonu had 10000 rupees in his bank account. After both transactions, i.e., transfer and withdraw, when Sonu checks his bank balance, it shows 3000 rupees. This error-free updation of bank balance is possible with the help of the concurrent feature of the Database.
* Thus here we see that concurrent is a great feature of the Database.

5. Stores Any Kind of Structured Data

* The Database has the ability to store the data in a structured format.
* In most of the websites, we see that only student database examples are given for a better understanding, but the important fact is that the Database has the ability to store an unlimited amount of data.
* DBMS has the ability to store any type of data that exists in the real world, and these data are structured way. It is another type of very important characteristic of DBMS.

6. Integrity

* Here the term integrity means the data should be correct and consistent in nature. Let's understand this by taking an example.
* Let's say there is a bank named ABC bank, and ABC bank has its own Database for the storage of its customer data. If we try to enter the account details of ABC bank and the account details are not available in the bank, then the Database gives the incorrect output. However, if a customer changes their address but the new address is not updated in the Database, it is called data inconsistency.
* So the data available in the Database should be correct as well as consistent.
* If someone's account has zero balance and later the customer deposits 6000 rupees in his account, if the new account balance is not updated in the Database, it creates a problem for the customer.

7. Ease of Access (The DBMS Queries)

* The file and folder system was used to store the data before the DBMS came to the market.
* Searching for the student's name was a very difficult task at that time. This is because every search operation is done manually in the file and folder system. But when DBMS comes into the market, it is very easy to access the Database.
* In DBMS, we can search any kind of stored data by applying a simple search operation query. It is so much faster than manual searching.
* In DBMS, there is a CRUD operation ( here CRUD means Create, Read, Update & Delete) by which we can implement all the types of query in the Database.

8. SQL and No-SQL Databases

* There are two types of databases (not DBMS): SQL and No-SQL.
* The SQL databases store the data in the form of Tables, i.e., rows and columns. The No-SQL databases can store data in any form other than a table. For instance: the very popular MongoDB stores the data in the form of JSON (JavaScript Object Notation).
* The availability of SQL and No-SQL databases allows us to choose the method of storing the data as well.
* There should not be any debate between SQL and No-SQL databases. The one that we require for a particular project is better for that project, while the other might be better for some other use.
* This is a characteristic of DBMS because DBMS allows us to perform operations on both kinds of databases. So, we can run queries and operations on SQL as well as No-SQL databases.

9. ACID Properties

* The DBMS follows certain properties to maintain consistency in the Database. These properties are usually termed ACID Properties.
* However, we have already talked about some of these properties, but it is very important to mention the ACID properties as a whole.
* ACID stands for Atomicity, Consistency, Isolation, and Durability.
* We have already talked about atomicity and consistency. Atomicity means the transaction should either be 0% or 100% completed, and consistencymeans that the change in data should be reflected everywhere in a database.
* Isolation means that multiple transactions can occur independently without the interference of some other transactions.
* Durability means that the chances of a successful atomic transaction, i.e., a transaction that has been 100% completed, should reflect in the Database.

10. Security

* The Database should be accessible to the users in a limited way.
* The access to make changes to a database by the user should be limited, and the users must not be given complete access to the entire Database.
* Unauthorized users should not be allowed to access the Database.
* Authentication: The DBMS has authentication for various users that directly refers to the limit to which the user can access the Database. Authentication means the process of laughing in of the user only with the rights that he/she has been authorized to. For instance, in any organization, the admin has access to make changes to the Database of the organization as some new employee might have joined the organization or someone might have left it. However, the employees have access only to their personal profiles and can make changes to them only. They cannot access the Database of any other employee or the organization as a whole.

**Explain the three level schema architecture in DBMS?**

The three-schema architecture divides the [**database**](https://www.tutorialspoint.com/Introduction-to-Databases) into three-level used to create a separation between the physical database and the user application. In simple terms, this architecture hides the details of physical storage from the user.

The database administrator ([**DBA**](https://www.tutorialspoint.com/explain-the-role-of-the-database-administrator-in-dbms)) responsible is to change the structure of database storage without affecting the user’s view. It deals with the data, the relationship between them and the different access methods implemented on the database. The logical design of database is called a schema

This architecture contains three layers of database management system, which are as follows −

* External level
* Conceptual level
* Internal level

External/ View level

This is the highest level of database abstraction. It includes a number of external schemas or user views. This level provides different views of the same database for a specific user or a group of users. An external view provides a powerful and flexible security mechanism by hiding the parts of the database from a particular user.

Conceptual or Logical level

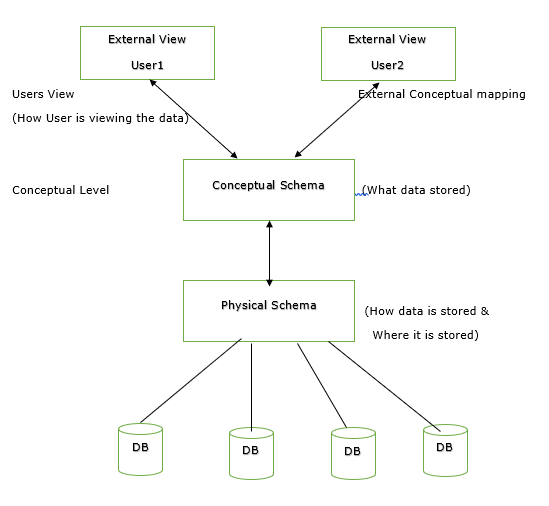
This level describes the structure of the whole database. It acts as a middle layer between the physical storage and user view. It explains what data to be stored in the database, what the data types are, and what relationship exists among those data. There is only one conceptual schema per database.

Internal or Physical level

This is the lowest level of database abstraction. It describes how the data is stored in the database and provides the methods to access data from the database. It allows viewing the physical representation of the database on the computer system.

The interface between the conceptual and internal schema identifies how an element in the conceptual schema is stored and how it may be accessed. It is one which is closest to physical storage. The internal schema not only defines different stored record types, but also specifies what indices exist, how stored fields are represented.

The three level schema architecture in DBMS is given below −



The three-schema architecture is commonly known as the three-tier architecture or ANSI/SPARC architecture. These terms and framework are for describing a particular database system structure. The physical database and the user applications are kept separate using this three-tier architecture. This type of architecture has three, which are what keeps the user and the database separate.

The framework of this type of architecture includes an external schema, a conceptual level, an internal schema, and then the database itself. The various levels of the database are mapped, which leads to the transformation of response and request in the database. Smaller DBMS are challenging to map since they are more time-consuming. In the external/conceptual mapping, the transformation of the request to the conceptual level from the external level is necessary. In the conceptual/ internal mapping, the request received from the conceptual level is transformed to the internal level by the DBMS.

***1.    Internal Level:***

* The internal level consists of an internal schema which is used to describe the physical structure of the database in which the data is stored.
* This schema is often called as the physical schema.
* The internal level uses the physical data model. This data model describes the method by which the data should be stored in a block.
* Complex low-level data structures are described in detail in this schema.

***2.    Conceptual Level:***

* The conceptual schema elaborates the database and its data along with the design at a conceptual level.
* The conceptual level is often known as the logical level.
* Conceptual level is used for describing the entire database with its structure.
* This level also describes where the data which is stored in the entire database and the relation about the data as well.
* The data structure’s implementation and other internal details are hidden at the conceptual level.
* Administrators and programmers work on the database at the conceptual level.

***3.    External Level:***

* A database contains various schemas at this level, and these schemas may also be called as subschemas. These are used to describe the possible different views in the database.
* View schema is another name for the external schema.
* External level describes the user and his interaction with the data in the database.
* **Physical Level:** At the physical level, the information about the location of database objects in the data store is kept. Various users of DBMS are unaware of the locations of these objects.In simple terms,physical level of a database describes how the data is being stored in secondary storage devices like disks and tapes and also gives insights on additional storage details.
* **Conceptual Level:**At conceptual level, data is represented in the form of various database tables. For Example, STUDENT database may contain STUDENT and COURSE tables which will be visible to users but users are unaware of their storage.Also referred as logical schema,it describes what kind of data is to be stored in the database.
* **External Level:** An external level specifies a view of the data in terms of conceptual level tables.  Each external level view is used to cater to the needs of a particular category of users. For Example, FACULTY of a university is interested in looking course details of students, STUDENTS are interested in looking at all details related to academics, accounts, courses and hostel details as well. So, different views can be generated for different users. The main focus of external level is data abstraction.

**Difference between Schema and Instance in DBMS**

**“Schema”** and **“Instance”** are key ideas in a database management system (DBMS) that help organize and manage data.

**Instances**

* An Instance is the state of an operational database with data at any given time.
* It contains a snapshot of the database.
* The instances can be changed by certain CRUD operations, such as like addition, and deletion of data.
* It may be noted that any search query will not make any kind of changes in the instances.

**Example:**  
In School database, Teacher table is there, suppose the table has 50 records so the **instance** of the database has 50 records for now and tomorrow we are going to add another 50 records so tomorrow the **instance** has a total of 100 records. This is called an instance.

**Schema**

Schema is the overall description of the database. The basic structure of how the data will be stored in the database is called schema.

Schema is of three types: Logical Schema, Physical Schema and view Schema.

* **Logical Schema –** It describes the database designed at a logical level.
* **Physical Schema –** It describes the database designed at the physical level.
* **View Schema** – It defines the design of the database at the view level.

**Example:**  
In our school database, teacher table requires the name, dob, and doj in their table so we design a structure as:

**Teacher table**  
name: String  
doj: date  
dob: date

Above given is the schema of the table teacher.

## ****Difference Between Schema and Instance****

| **Schema** | **Instance** |
| --- | --- |
| It is the overall description of the database. | It is the collection of information stored in a database at a particular moment. |
| The schema is same for the whole database. | Data in instances can be changed using addition, deletion and updation. |
| Does not change Frequently. | Changes Frequently. |
| Defines the basic structure of the database i.e. how the data will be stored in the database. | It is the set of Information stored at a particular time. |

## Conclusion

* The schema is the **blueprint** of the database, while the instance is the **actual data** that is in the database.
* The schema is the **database’s design**, and the instance is the **data it contains.**
* When you **change** the schema, you change the **structure of the database**. When you **change** the instance, you change the **data content**.

**Data Independence**

Data independence means a change of data at one level should not affect another level.

In simple words, we can say that Data independence is a property of a database that allows the **User** or **Database Administrator** to change the schema at one level without affecting the data or schema at another level.

**Purpose:**The purpose of data independence is to enhance the security of the system, save time and reduce costs needed once the information is changed or altered.

Two types of data independence are present in this architecture:

* **Physical Data Independence:** Any change in the physical location of tables and indexes should not affect the conceptual level or external view of data. This data independence is easy to achieve and implemented by most of the DBMS.
* **Conceptual Data Independence:** The data at conceptual level schema and external level schema must be independent. This means a change in conceptual schema should not affect external schema. e.g.; Adding or deleting attributes of a table should not affect the user’s view of the table.

But this type of independence is difficult to achieve as compared to physical data independence because the changes in conceptual schema are reflected in the user’s view.

## Difference between Logical Data Independence and Physical Data Independence

|  |  |
| --- | --- |
| **Physical Data Independence** | **Logical Data Independence** |
| Physical data independence is used to change the internal schema without requiring a change in the logical schema. | Logical data independence is making sure that if you add any new field or delete any existing field we do not need to change the application program. |
| Physical data independence is easy to attain in comparison to logical data independence. | It is difficult to attain logical data independence compared to physical data independence. |
| Physical data independence provides feasibility if we want to shift the database or want to change the file organization structure. | Logical data independence helps us to change the data definition and the structure of the data without having changes in the physical schema. |
| Physical data independence deals with the internal structure of the schema. | Logical data independence deals with conceptual schema. |
| Examples of changes in Physical independence are Changing the compression techniques, hashing algorithms, SSD, location of the database, etc. | Examples of changes in logical independence are Adding, deleting, or modifying the entity or relationship. |

DBMS architecture:

DBMS architecture describes the structure and how the users are connected to a specific database system. Architecture affects the performance of the database.

Database Management System (DBMS) architecture is crucial for efficient data management and system performance. It involves the database's design, development, and maintenance, determining how users interact with and access the system. In client/server architecture, numerous PCs and workstations connect through a network, facilitating a large user base and secure data access. The choice of architecture depends on factors such as database size, user count, and inter-user relationships.

**Types of DBMS Architecture**

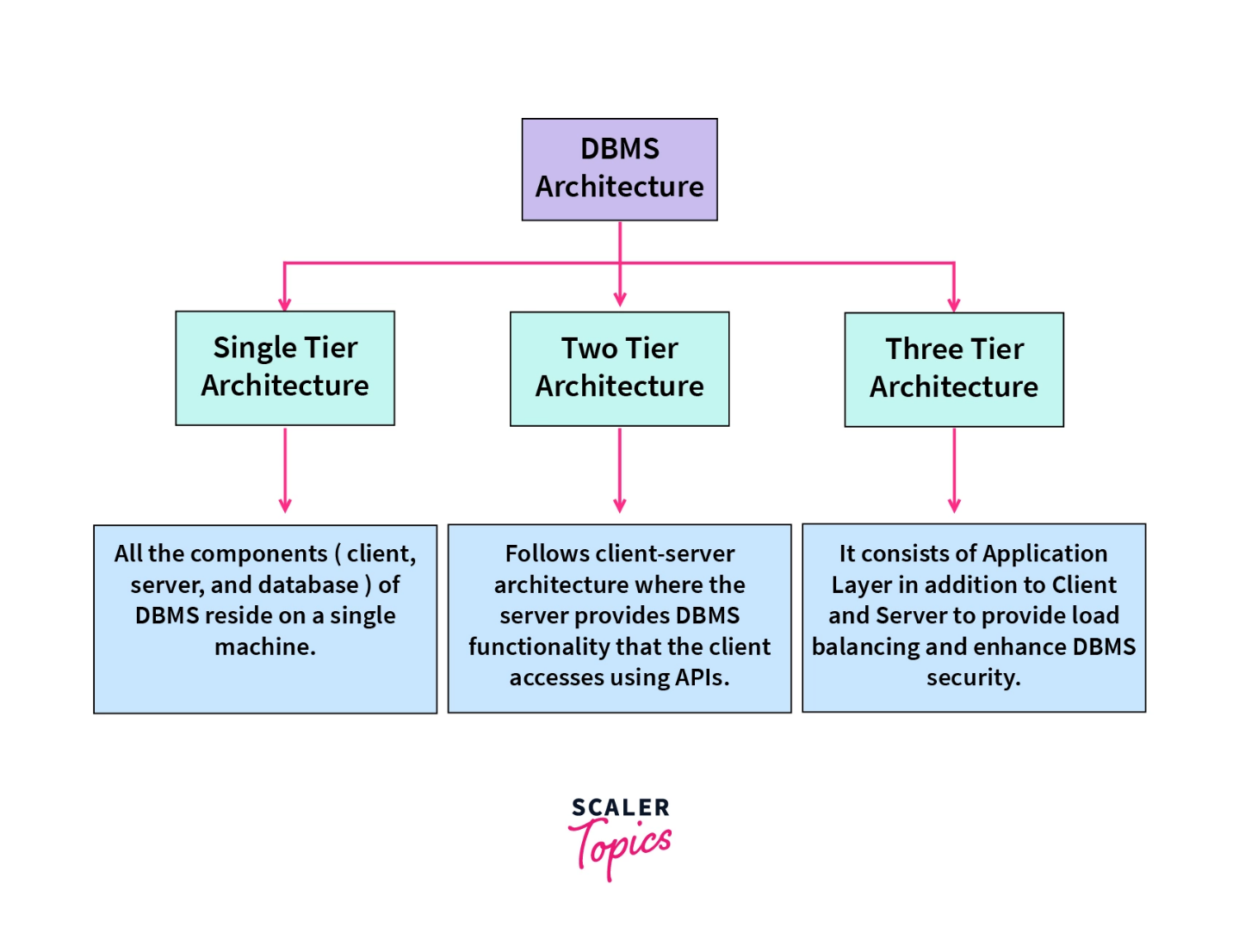
Database management systems are divided into multiple levels of abstraction for proper functioning. These modules/layers describe the functioning and the design of the DBMS.

Since a database management system is not always directly accessible by the user or an application, we can maintain it with the help of various architectures based on how the user is connected to the database. These architectures follow a **tier-based classification**, i.e., the DBMS architecture is classified depending upon how many layers are present in the structure of the DBMS.

Hence, an **n-tier DBMS Architecture** divides the whole DBMS into related but **n independent layers** or levels, i.e., a one-tier architecture divides the DBMS into a single layer, a two-tier DBMS architecture divides the DBMS into two layers, a three-tier in three layers, and so on. When the layers are increased in the architecture, the level of abstraction also increases, resulting in an increase in the security and the complexity of the DBMS structure. All these layers are independent, i.e., any modification performed in a particular layer does not affect the other layer present in the architecture.

Now, let’s look at the most common DBMS architectures:

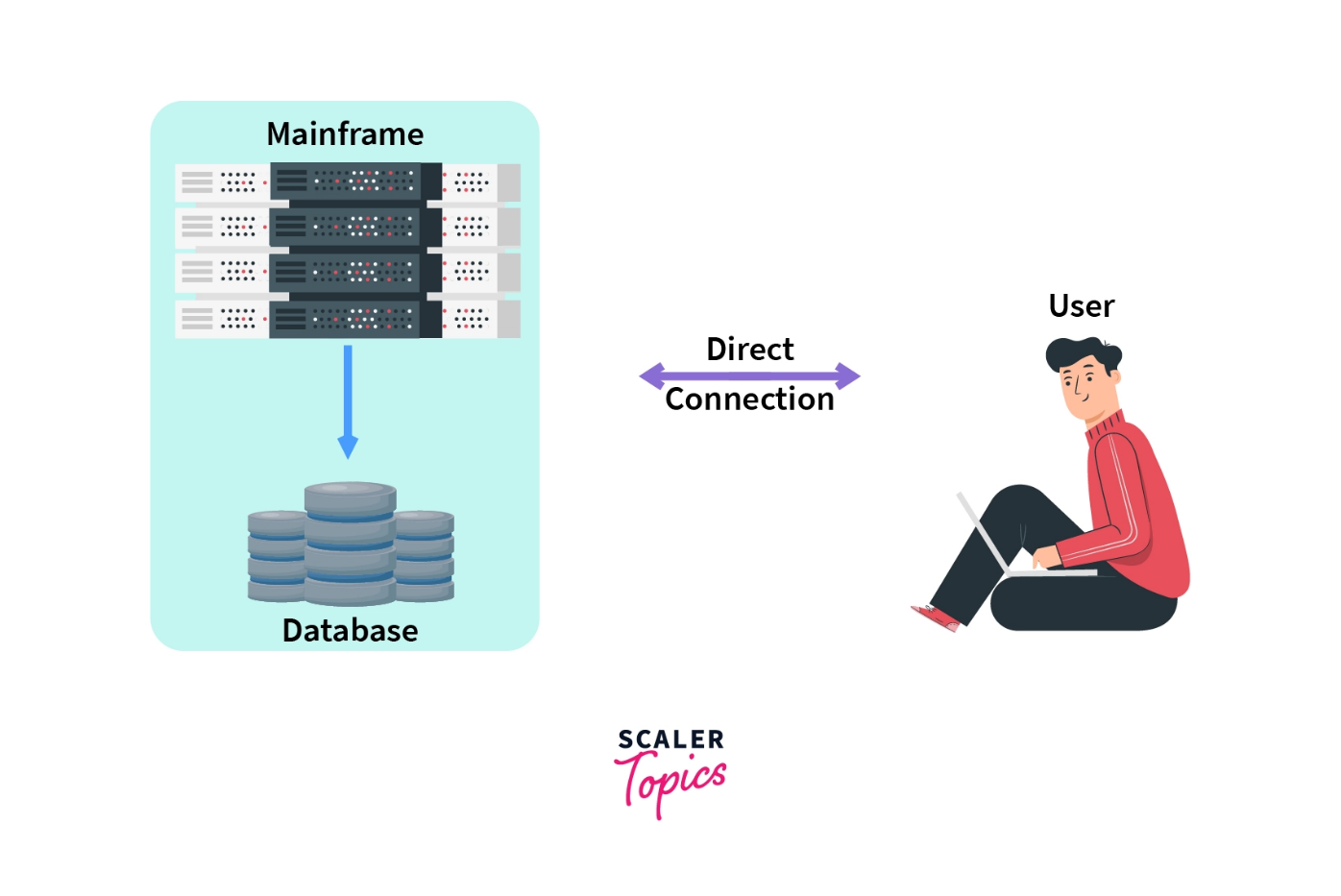
* Single Tier Architecture (One-Tier Architecture)
* Two-Tier Architecture
* Three-Tier Architecture



Now, let's explore these three different ways to structure the database management systems.

**1. Single Tier Architecture**

* In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
* Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
* The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

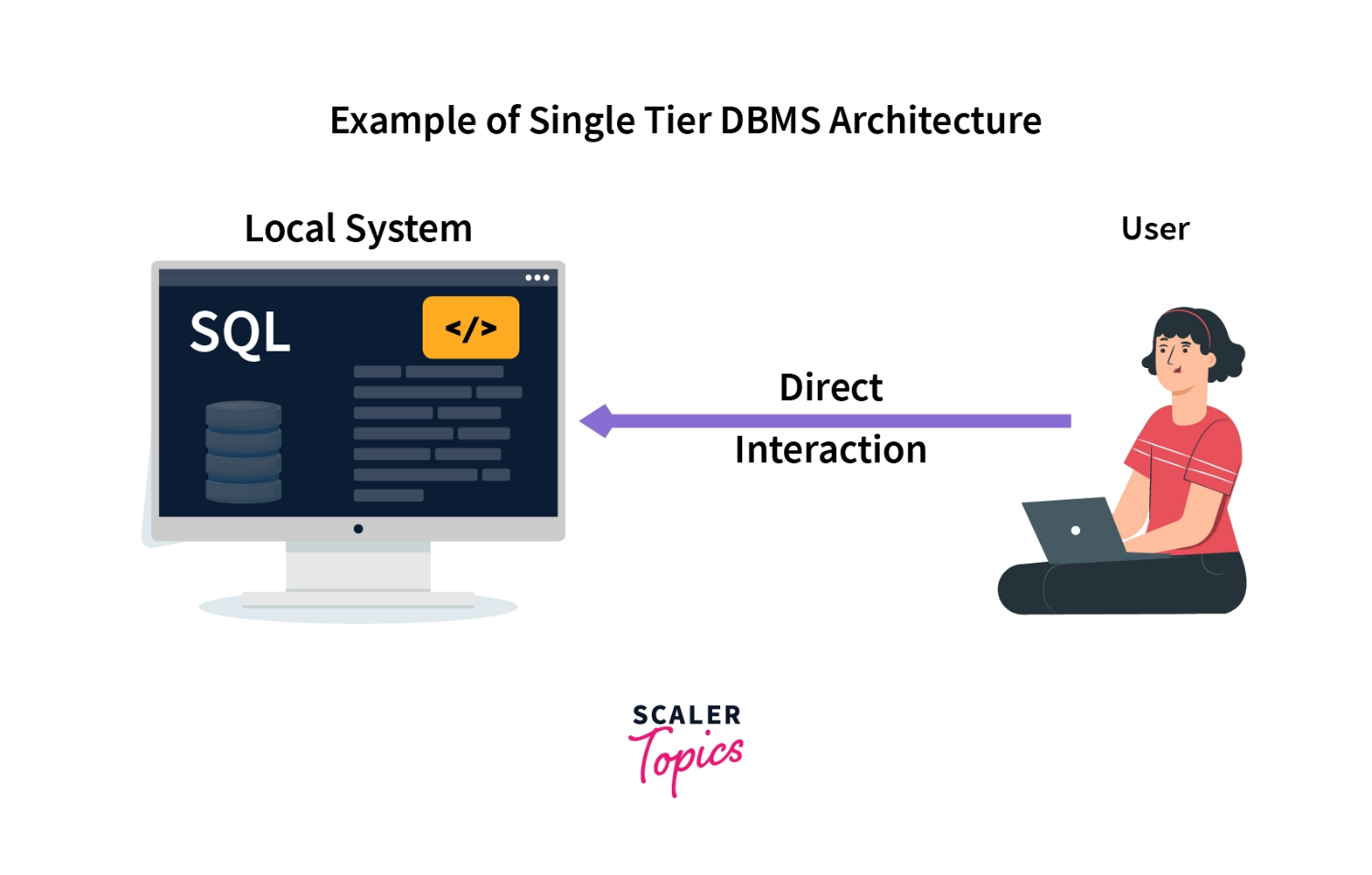


Single Tier DBMS Architecture is used whenever:

* The data isn't changed frequently.
* No multiple users are accessing the database system.
* We need a direct and simple way to modify or access the database for application development.

**Example of Single Tier DBMS Architecture:**

Structure Query Language (SQL), we set up our SQL server and the database on our local system. This SQL server enables us to directly interact with the relational database and execute certain operations without requiring any network connection. SQL queries is an example of Single-Tier DBMS architecture.



**Highlights:**

**1**- Simplest DBMS architecture.

**2**- All the components of DBMS, i.e., the server, database, and client, reside on a single system.

**3**- The user can directly access the database.

**4**- Used when data isn't changing frequently.

**5**- Suitable for programmers, database designers, and single-user access.

**2. Two Tier Architecture**

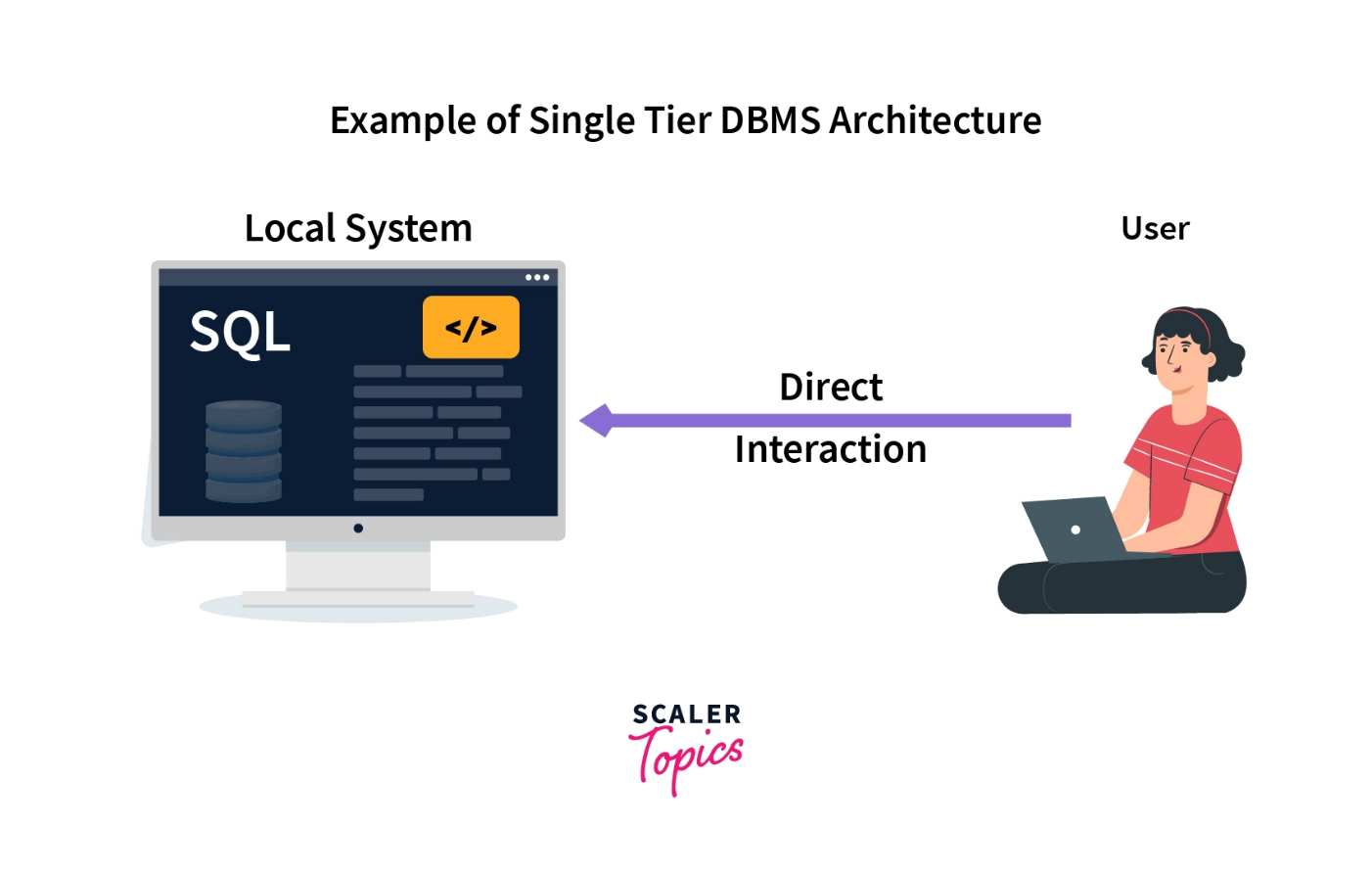
* The 2-Tier architecture is the same as the basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database on the server side. For this interaction, APIs like ODBC, and JDBC are used.
* The user interfaces and application programs are run on the client side.
* The server side is responsible to provide the functionalities like query processing and transaction management. To communicate with the DBMS, the client-side application establishes a connection with the server side.

The main advantages of having a two-tier architecture over a single tier are:

* Multiple users can use it at the same time. Hence, it can be used in an organization.
* It has high processing ability as the database functionality is handled by the server alone.
* Faster access to the database due to the direct connection and improved performance.
* Because of the two independent layers, it's easier to maintain.

**Example of Two Tier DBMS Architecture:**

Consider a situation where you went to a bank to withdraw some cash. After entering the withdrawal amount and the account details on the withdrawal slip, the banker will go through the server-side database via his credential (**API call**) and will check whether there is enough balance present or not. This client-server model is an example of Two-Tier DBMS architecture.



**Note -** The main disadvantages of Two-Tier DBMS Architecture are:

* **Scalability -** As the number of clients increases, the load on the server increases. Thereby declining the performance of the DBMS and, in turn, the client-side application.
* **Security -** The Direct connection between the client and server systems makes this architecture vulnerable to attacks.

**Highlights:**

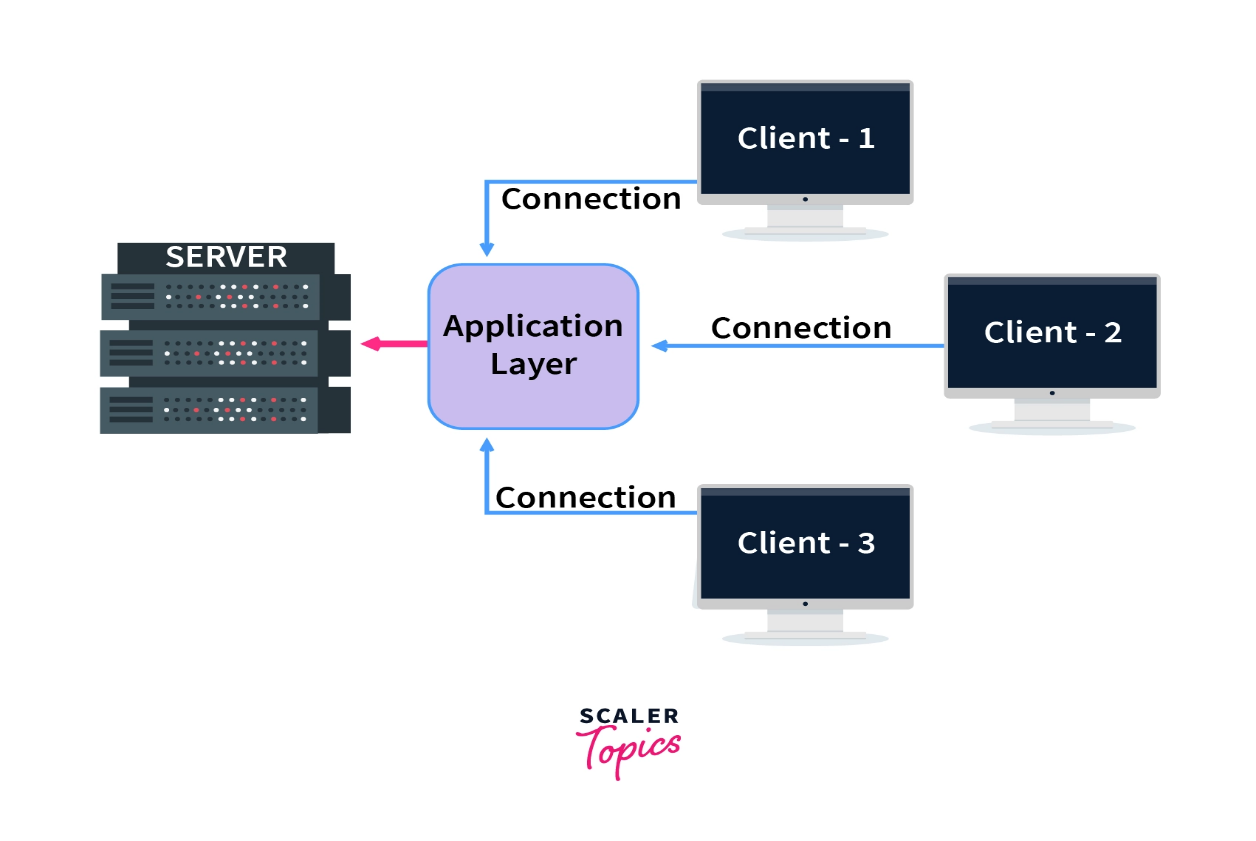
**1**- Similar to a client-server architecture.

**2**- Faster access, Easier to maintain, and can handle multiple users simultaneously.

**3**-Used when we wish to access DBMS via applications and APIs.

**4**-Has scalability and security issues because of direct client-server connection.

**3. Three Tier Architecture**

* The 3-Tier architecture contains another layer between the client and server. In this architecture, the client can't directly communicate with the server.
* The application on the client end interacts with an application server which further communicates with the database system.
* The end-user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application. The 3-Tier architecture is used in the case of the large web application.
* The main advantages of Three Tier DBMS Architecture are:
* **Scalability -** Since the database server isn't aware of any users beyond the application layer and the application layer implements load balancing, there can be as many clients as you want.
* **Data Integrity -** Data corruption and bad requests can be avoided because of the checks performed in the application layer on each client reques.
* **Security -** The removal of the direct connection between the client and server systems via abstraction reduces unauthorized access to the database.

**Note -** In Three Tier DBMS Architecture, an additional layer (Application Layer) is added between the Client and the Server. This increases the number of layers present between the DBMS and the end-users, making the implementation of the DBMS structure complex and difficult to maintain.

**Highlights:**

**1**- Most widely used DBMS architecture.

**2**- Follows Client-Application-Server architecture.

**3**- Enhanced security, data integrity, and scalability.

**4**-Has complexity and maintenance issues because of the extra layer.

**Summary:**

* DBMS architecture describes the structure and how the users are connected to a specific database system.
* Architecture affects the performance of the database.
* In the single-tier architecture, all the components of the DBMS reside on a single server. It is the most straightforward approach.
* In the two-tier architecture, the client-server approach is implemented.
* The three-tier DBMS architecture is an extension of the two-tier architecture where an intermediate layer, i.e., the Application layer is added between the client and the server. It is the most widely used architecture.

## Achieving Data Independence in DBMS Through Data Abstraction

To achieve Data Independence, the first step is to ensure **Data Abstraction**. Data Abstraction can be defined as extracting the necessary data by ignoring the remaining irrelevant details.

If we take the example of a real-world entity, ATM is one of the best examples of data abstraction. We all use an **ATM machine for cash withdrawals**, money transfers, etc. in our daily life.

The main purpose of data abstraction is to achieve data independence. There are three levels of abstraction.

* **Physical or Internal Level -**Physical level is the lowest level of data abstraction and It indicates how the data will be stored and describes the complex data structures and access methods to be used by the database. The internal level is used to describe the entire database architecture.
* **Conceptual or Logical Level -**The Conceptual database schema is additionally called the logical structure because it defines the logical relations between the data. The separation of the conceptual view from the internal view enables us to provide a logical description of the database concepts without the need to specify physical structures. The conceptual level comes between the physical level and the view level. It provides the link between the external schema and the internal schema of the database.
* **External or View Level -**It is the highest level of data abstraction. The external level describes the user interaction with the centralized database management system. This level is used to provide a [Graphical User Interface](https://unstop.com/blog/difference-between-gui-and-cui) to the user, and the user does not know about the file structure, access method, and other internal details of the database.

Based on the data abstraction, there are two levels of data independence in DBMS:

* **Physical level data independence**
* **Logical level data independence**

Let’s discuss the properties of these two levels of data independence.

### 1. Physical Level Data Independence

**Physical Data Independence** can be defined as the ability to change the physical level without affecting the logical or Conceptual level. Physical data independence gives us the freedom to modify the - Storage device, File structure, location of the database, etc. without changing the definition of conceptual or view level.

**Example:**For example, if we take the database of the banking system and we want to scale up the database by changing the storage size and also want to change the file structure, we can do it without affecting any functionality of logical schema.

**Below changes can be done at the physical layer without affecting the conceptual layer -**

* Changing the storage devices like SSD, hard disk and magnetic tapes, etc.
* Changing the access technique and modifying indexes.
* Changing the compression techniques or hashing algorithms.

### 2. Logical Level Data Independence

**Logical Data Independence** is a property of a database that can be used to change the logic behind the logical level without affecting the other layers of the database. Logical data independence is usually required for changing the conceptual schema without having to change the external schema or application programs. It allows us to make changes in a conceptual structure like adding, modifying, or deleting an attribute in the database.

**Example:** If there is a database of a banking system and we want to add the details of a new customer or we want to update or delete the data of a customer at the logical level data will be changed but it will not affect the Physical level or structure of the database.

**These changes can be done at a logical level without affecting the application program or external layer.**

* Adding, deleting, or modifying the entity or relationship.
* Merging or breaking the record present in the database.

## Advantages of Data Independence

Data independence in Database Management Systems (DBMS) offers several significant advantages, which contribute to the efficient management and maintenance of database infrastructure. Here are some of the key advantages:

* **Flexibility:** Data independence allows for changes to be made in the database schema (structure) without affecting the way data is accessed or presented to users. This flexibility makes it easier to adapt the database to evolving requirements and business needs.
* **Application Compatibility:** Changes to the logical schema do not impact the application programs or queries that rely on the database. This means that existing applications can continue to function correctly even when the database structure changes, reducing the risk of disruptions.
* **Easier Maintenance:** Database administrators can perform routine maintenance tasks, such as reorganizing data for performance optimization or implementing security updates, without disrupting user access or application functionality.
* **Enhanced Security:** Data independence allows for security measures and access controls to be implemented at the logical level, protecting sensitive data from unauthorized access or modification. Security policies can be enforced without exposing the underlying physical storage details.
* **Data Continuity:** When migrating data to new storage technologies or platforms, data independence ensures that the logical schema remains consistent, preserving data continuity and application functionality.
* **Scalability:** As the database grows, data independence facilitates the addition of new data elements or tables without affecting existing queries or applications. This scalability is crucial for accommodating increasing data volumes.
* **Reduced Development Time:** Developers can focus on designing and building applications without needing to worry about changes in the underlying database structure. This separation of concerns can lead to faster development cycles.
* **Ease of Integration:** Data independence simplifies the integration of data from multiple sources into a single database storage system. External schemas can be defined to provide unified views of the data, regardless of its source or format.
* **Data Integrity:** Changes to the logical schema can be managed carefully to ensure data integrity and consistency. Referential integrity constraints and validation rules can be applied at the logical level to maintain data quality.
* **Adaptation to Technology Changes:** As technology evolves, the physical storage and organization of data may need to change to take advantage of new hardware or software capabilities. Data independence allows these changes to be made without affecting the logical schema.
* **Reduced Risk:** By minimizing the impact of schema changes, data independence reduces the risk of errors and data corruption that can occur when modifying a database's structure hence helping in the improvement in database security.

## Disadvantages of Data Independence

While data independence offers many advantages in database management systems, it's essential to consider its potential disadvantages and limitations:

* **Complexity:** Maintaining multiple levels of schema (external, conceptual, and internal) to achieve data independence can introduce complexity into the database system. This complexity can make database design and management more challenging.
* **Performance Overhead:** Implementing data independence can sometimes result in performance overhead. The added layers of abstraction between the logical and physical data can impact query performance and data retrieval efficiency.
* **Resource Consumption:** Managing data independence may require additional system resources, such as storage space and processing power, to handle the various schema layers and translations between them.
* **Potential for Redundancy:** In some cases, data independence may lead to data redundancy. Different external schemas might require the same data to be stored in multiple formats or physical locations, which can increase storage requirements and synchronization challenges.
* **Migration Complexity:** While data independence simplifies schema changes, it may not eliminate all complexities associated with data migration. Migrating data between different versions of the database organization or across different DBMS platforms can still be complex and time-consuming.
* **Compatibility Challenges:** Changes made to the logical or conceptual schema may not always be compatible with existing application programs or queries. This can require additional effort to ensure backward compatibility and may involve rewriting or updating applications.
* **Data Integrity Risk:** Changes in the logical schema, if not managed carefully, can lead to data integrity issues. Ensuring that data remains consistent and that referential integrity constraints are maintained can be challenging when altering the logical schema.
* **Development and Maintenance Effort:** Implementing data independence often requires thorough planning, documentation, and adherence to best practices. It can involve additional development and maintenance effort to create and manage various schema layers and ensure that changes do not introduce errors.
* **Training and Expertise:** Database administrators and developers may require specific training and expertise to effectively manage data independence in a DBMS. Understanding how changes at one level of the schema affect other levels is crucial for maintaining data integrity.
* **Risk of Over-Abstraction:** In an attempt to achieve data independence, designers may over-abstract the logical schema, leading to a lack of transparency in the database structure. This can make it more challenging for developers and users to understand and work with the data.
* **Potential for Suboptimal Physical Design:** Complete physical data independence can make it challenging to optimize the physical storage of data effectively. Performance optimizations that require tight integration between logical and physical structures may not be feasible.

Data at one level can be changed without affecting the other level, which helps to improve the performance of the system. The bottom line is that data independence is one of the important factors for designing a bigger database to handle a huge amount of data, helping us improve the quality of the data and ensuring its security.