**Introduction of DBMS :**

<https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/>

**Explain about database system architecture in detail.**

A Database stores a lot of critical information to access data quickly and securely. Hence it is important to select the correct architecture for efficient data management. DBMS Architecture helps users to get their requests done while connecting to the database.

Types of DBMS Architecture

There are several types of DBMS Architecture that we use according to the usage requirements. Types of DBMS Architecture are discussed here.

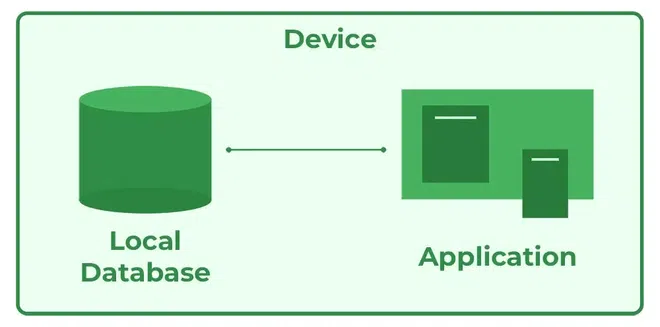
1-Tier Architecture

2-Tier Architecture

[3-Tier Architecture](https://www.geeksforgeeks.org/introduction-of-3-tier-architecture-in-dbms-set-2/)

**1-Tier Architecture**

In 1-Tier Architecture the database is directly available to the user, the user can directly sit on the DBMS and use it that is, the client, server, and Database are all present on the same machine.



**Advantages of 1-Tier Architecture**

**Simple Architecture:** 1-Tier Architecture is the most simple architecture to set up, as only a single machine is required to maintain it.

**Cost-Effective:** No additional hardware is required for implementing 1-Tier Architecture, which makes it cost-effective.

**Easy to Implement:**1-Tier Architecture can be easily deployed, and hence it is mostly used in small projects.

**2-Tier Architecture**

The 2-tier architecture is similar to a basic[client-server model](https://www.geeksforgeeks.org/client-server-model/). The application at the client end directly communicates with the database on the server side. APIs like ODBC and JDBC are used for this interaction.



**Advantages of 2-Tier Architecture**

**Easy to Access:** 2-Tier Architecture makes easy access to the database, which makes fast retrieval.

**Scalable:** We can scale the database easily, by adding clients or upgrading hardware.

**Low Cost:** 2-Tier Architecture is cheaper than 3-Tier Architecture and [Multi-Tier Architecture](https://www.geeksforgeeks.org/multi-tier-architecture-of-data-warehouse/).

**Easy Deployment:** 2-Tier Architecture is easier to deploy than 3-Tier Architecture.

**Simple:** 2-Tier Architecture is easily understandable as well as simple because of only two components.

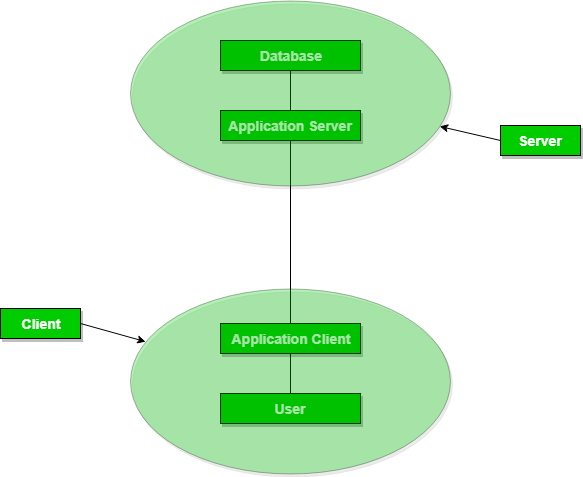
**3-Tier Architecture**

In [3-Tier Architecture](https://www.geeksforgeeks.org/introduction-of-3-tier-architecture-in-dbms-set-2/), there is another layer between the client and the server. The client does not directly communicate with the server. Instead, it interacts with an application server which further communicates with the database system and then the query processing and transaction management takes place. This intermediate layer acts as a medium for the exchange of partially processed data between the server and the client. This type of architecture is used in the case of large web applications.

**Advantages of 3-Tier Architecture**

**Data Integrity:** 3-Tier Architecture maintains Data Integrity. Since there is a middle layer between the client and the server, data corruption can be avoided/removed.

**Security:**3-Tier Architecture Improves Security. This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.



**Three-schema Architecture**

<https://www.geeksforgeeks.org/introduction-of-3-tier-architecture-in-dbms-set-2/>

**Difference between File System and DBMS :**

<https://www.geeksforgeeks.org/difference-between-file-system-and-dbms/>

**Data models :**

<https://www.geeksforgeeks.org/data-models-in-dbms/>

**instances:**

<https://www.geeksforgeeks.org/instance-in-database/>

**Schemas :**

<https://www.geeksforgeeks.org/database-schemas/>

**Data independence :**

<https://www.geeksforgeeks.org/what-is-data-independence-in-dbms/>

**Data Base Language and interfaces :**

<https://www.geeksforgeeks.org/database-languages-in-dbms/>

**Database Structure :**

<https://www.geeksforgeeks.org/structure-of-database-management-system/>

**ER data model:**

<https://www.geeksforgeeks.org/introduction-of-er-model/>

**attributes of ER model :**

<https://www.geeksforgeeks.org/types-of-attributes-in-er-model/>

**transforming ER diagram into the tables**

<https://www.geeksforgeeks.org/how-to-convert-er-diagrams-to-tables-in-dbms/>

**Comparison between the three types of models.**

<https://www.geeksforgeeks.org/difference-between-hierarchical-network-and-relational-data-model/>

**object oriented data Model :**

<https://www.geeksforgeeks.org/basic-object-oriented-data-model/>

**Network data model :**

<https://www.geeksforgeeks.org/network-model-in-dbms/>

**Relational data model :**

<https://www.geeksforgeeks.org/relational-model-in-dbms/>

**What are the functions of DBA?**

**Database Installation and Configuration**: DBAs are responsible for installing database software and configuring it to meet the organization's requirements

**Database Design and Schema Management:** DBAs work with application developers and data architects to design and implement database schemas that efficiently store and organize data.

**Data Security and Access Control**: DBAs implement security measures to protect sensitive data from unauthorized access, manipulation, or disclosure.

**Backup and Recovery Planning**: DBAs develop and maintain backup and recovery strategies to minimize the risk of data loss and ensure business continuity in the event of hardware failures, disasters, or human errors.

**Capacity Planning and Scalability**: DBAs forecast future growth in database usage and capacity requirements, based on business projections and historical data trends.

**Aggregation, specialization and generalization**

Aggregation, specialization, and generalization are concepts commonly used in database design and modeling to represent relationships between entities and to organize data in a hierarchical structure.

1. **Aggregation:**

Aggregation is a modeling technique where one or more entities are grouped together to form a higher-level entity. It represents a "whole-part" relationship, where the higher-level entity encompasses the lower-level entities.

Example:

In a university database, a "Department" entity can be aggregated from multiple "Course" entities. Each department contains multiple courses, and the department entity represents the aggregation of these courses.

1. **Specialization:**

Specialization is a process of defining subclasses or specialized entities based on a common superclass or generalized entity. It represents an "is-a" relationship, where the specialized entities inherit attributes and relationships from the generalized entity but may also have their own unique attributes and relationships.

Example**:**

In a vehicle database, "Car" and "Truck" can be specialized from a common superclass "Vehicle." Both cars and trucks share common attributes like "make," "model," and "year," but trucks may have additional attributes such as "cargo capacity."

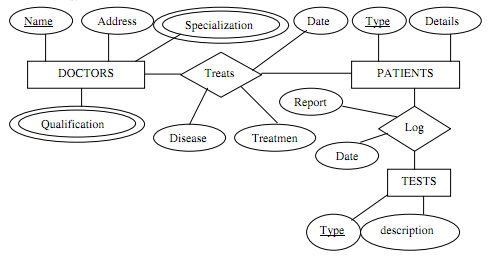
1. **Generalization:**

Generalization is the opposite of specialization. It involves combining multiple subclasses or specialized entities into a more generalized superclass. It represents an abstraction of common attributes and relationships shared by the specialized entities.

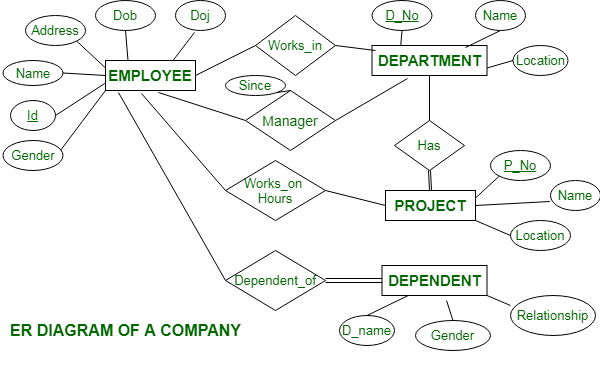
Example:

In an employee database, "Manager" and "Staff" entities can be generalized into a common superclass "Employee." Both managers and staff share common attributes and relationships such as "name," "hire date," and "salary."

**Q6. Construct an E-R diagram for hospital with a set of patients and a set of doctors. Associate with each patient a log of the various tests and examination conducted. Also show tables for various entities with attributes.**

****

**Q8. Draw an ER diagram for a small marketing company database. Assume suitable data.**

****

**Q9. Why the hierarchical data model is considered inflexible?**

The hierarchical data model is considered inflexible primarily due to its rigid structure and limited ability to represent complex relationships between data entities. Here are some reasons why the hierarchical data model is perceived as inflexible:

**Fixed Parent-Child Relationships**:

In a hierarchical data model, data is organized in a tree-like structure where each record has a single parent record and can have multiple child records.

**Limited Querying Capabilities**:

Querying hierarchical data can be cumbersome, especially when dealing with complex relationships or nested structures.

**Data Redundancy**:

The hierarchical model tends to lead to data redundancy, as the same data may need to be duplicated across multiple branches of the hierarchy to satisfy the parent-child relationship requirements.

**Lack of Flexibility in Schema Evolution**:

Making changes to the hierarchical data model, such as adding new data elements or modifying relationships, can be challenging and may disrupt existing applications or data access patterns.

**Difficulty in Representing Many-to-Many Relationships**:

Hierarchical data models struggle to represent many-to-many relationships between data entities, as they typically enforce a strict one-to-many relationship between parent and child records.

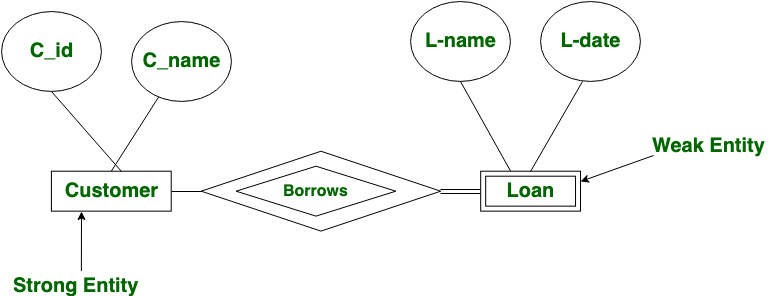
**Scalability Concerns**:

As the size and complexity of the data increase, the hierarchical model may become less scalable and more difficult to manage.

**Q10 . Explain strong and weak entities. How weak entity can be converted to a strong entity.**

**Strong entity :** A strong entity is not dependent on any other entity in the schema. A strong entity will always have a primary key. Strong entities are represented by a single rectangle. The relationship of two strong entities is represented by a single diamond. Various strong entities, when combined together, create a strong entity set.

**Week entity :** A weak entity is dependent on a strong entity to ensure its existence. Unlike a strong entity, a weak entity does not have any primary key. It instead has a partial discriminator key. A [weak entity](https://www.geeksforgeeks.org/weak-entity-set-in-er-diagrams/) is represented by a double rectangle. The relation between one strong and one weak entity is represented by a double diamond. This relationship is also known as identifying relationship.



**Conversion of Weak Entity to Strong Entity**:

To convert a weak entity into a strong entity, you would need to modify its structure so that it can exist independently with its own unique identifier. This can be achieved by adding attributes to the weak entity to create a composite key, making it a strong entity.

For example, consider a scenario where the "Loan" entity is currently a weak entity dependent on the "Customer" entity. To convert it into a strong entity, you could:

1. Add additional attributes to the "Account" entity to create a composite key that uniquely identifies each account, such as account number and customer ID.
2. Elevate an existing attribute of the "Account" entity, such as the account number, to serve as its unique identifier, making it independent of the "Customer" entity.

**11. Types of attributes :**

**Simple attribute :**

An attribute that cannot be further subdivided into components is a simple attribute.   
**Example:** The roll number of a student, the id number of an employee.

**Composite attribute :**

An attribute that can be split into components is a composite attribute.

**Example:** The address can be further split into house number, street number, city, state, country, and pin code, the name can also be split into first name middle name, and last name.

**Single-valued attribute :**

The attribute which takes up only a single value for each entity instance is a single-valued attribute.

**Example:** The age of a student.

**Multi-valued attribute :**

The attribute which takes up more than a single value for each entity instance is a multi-valued attribute.

**Example:**Phone number of a student: Landline and mobile.

**Derived attribute :**

An attribute that can be derived from other attributes is derived attributes.

**Example:** Total and average marks of a student.

**Complex attribute :**

Those attributes, which can be formed by the nesting of composite and multi-valued attributes, are called “*Complex                 Attributes*“. These attributes are rarely used in DBMS(DataBase Management System). That’s why they are not so popular.

**Stored attribute:**

 The stored attribute are those attribute which doesn’t require any type of further update since they are stored in the database.

**Example:**DOB(Date of birth) is the stored attribute.

**Key attribute:**

Key attributes are those attributes that can uniquely identify the entity in the entity set.

**Example:** Roll-No is the key attribute because it can uniquely identify the student.