***1.Difference between Multitasking and Multiprocessing :***

***Multi*-*tasking:***Multi-tasking is the logical extension of multiprogramming. In this system, the CPU executes multiple jobs by switching among them typically using a small time quantum, and these switches occur so frequently that the users can interact with each program while it is running.

Multitasking is further classified into two categories: Single User & Multiuser.

***Multiprocessing* :**   
Multiprocessing is a system that has two or more than two processors. In this, CPUs are added for increasing computing speed of the system. Because of Multiprocessing, there are many processes that are executed simultaneously.

Multiprocessing is further classified into two categories: Symmetric Multiprocessing and Asymmetric Multiprocessing.

**Difference between Multitasking and Multiprocessing :**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Multi-tasking** | **Multiprocessing** |
| **1.** | **The execution of more than one task simultaneously is known as multitasking.** | **The availability of more than one processor per system, that can execute several set of instructions in parallel is known as multiprocessing.** |
| **2.** | **The number of CPU is one.** | **The number of CPUs is more than one.** |
| **3.** | **It takes moderate amount of time.** | **It takes less time for job processing.** |
| **4.** | **In this, one by one job is being executed at a time.** | **In this, more than one process can be executed at a time.** |
| **5.** | **It is economical.** | **It is less economical.** |
| **6.** | **The number of users is more than one.** | **The number of users is can be one or more than one.** |
| **7.** | **Throughput is moderate.** | **Throughput is maximum.** |
| **8.** | **Its efficiency is moderate.** | **Its efficiency is maximum.** |
| **9.** | **It is of two types: Single user multitasking and Multiple user multitasking.** | **It is of two types: Symmetric Multiprocessing and Asymmetric Multiprocessing.** |

***2. Structured Data vs Unstructured Data:***

* Structured data is standardized, clearly defined, and searchable data, while unstructured data is usually stored in its native format.
* Structured data is quantitative, while unstructured data is qualitative.
* Structured data is often stored in data warehouses, while unstructured data is stored in data lakes.
* Structured data is easy to search and analyze, while unstructured data requires more work to process and understand.
* Structured data exists in predefined formats, while unstructured data is in a variety of formats.

Structured data is fairly straightforward to deal with, whereas unstructured data is more complex and harder to organize and extract.

The programming language used for structured data is [SQL](https://www.integrate.io/blog/the-sql-vs-nosql-difference/) (Structured Query Language). Developed by IBM in 1974, SQL handles relational databases and doesn’t require advanced coding skills. Typical examples of structured data are names, addresses, credit card numbers, numerical data, Microsoft Excel files, text files, and so on.

Typical examples of unstructured data are rich media, text, social media activity, video files, audio files, surveillance imagery, and various other file formats.

***3.Difference between E-R Model and Relational Model in DBMS:***

|  |  |
| --- | --- |
| **ER model** | **Relational model** |
| Developed by Peter Chen in 1976. | Developed by E.F. Codd in 1970. |
| ER model is the high level or conceptual model. | It is the representational or implementation model. |
| It is used by people who don’t know how database is implemented. | It is used by programmers. |
| It represents collection of entities and describes relationship between them. | It represent data in the form of tables and describes relationship between them. |
| It consists of components like Entity, Entity Type, Entity Set. | It consists of components like domain, attributes, tuples. |
| It is easy to understand the relationship between entities. | It is less easy to derive the relationship between different tables. |
| It describes cardinality. | It does not describe cardinality. |
| E-R model does not define data dependencies. | Relational model defines dependencies in tables. |
| E-R model represents relationships as associations. | Relational model represents relationships as join tables. |
| E-R model is more granular in terms of data representation. | Relational model is less granular. |
| E-R model is more flexible than the relational model. | Relational model is less flexible than E-R model. |
| E-R model does not involve normalization. | Relational model involves normalization. |
| E-R model use case is useful for initial planning and design. | Relational model use case  is useful for implementation and maintenance |
| **Some of the popular Language and Notations used-**   * Chen * UML * Crow’s foot * Bachman and others. | **Some of the popular Language and Notations used-**   * SQL * MySQL |

***4.Cardinality in DBMS:***

In database management, cardinality plays an important role. Here cardinality represents the number of times an entity of an entity set participates in a relationship set. Or we can say that the cardinality of a relationship is the number of tuples (rows) in a relationship. Types of cardinality in between tables are:

* one-to-one
* one-to-many
* many-to-one
* many-to-many

***5.What is Cursor in SQL ?***

**Cursor** is a Temporary Memory or Temporary Work Station. It is Allocated by [Database](https://www.geeksforgeeks.org/what-is-database/)Server at the Time of Performing [DML](https://www.geeksforgeeks.org/dml-full-form/)(Data Manipulation Language) operations on the Table by the User. Cursors are used to store Database Tables.

There are 2 types of Cursors: Implicit Cursors, and Explicit Cursors. These are explained as following below.

* **Implicit Cursors:** Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.
* **Explicit Cursors:** Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manner.

***6.Difference between Physical and Logical Data Independence :***

**1. Physical Data Independence :**   
The physical data independence is basically used to separate conceptual levels from the internal/physical levels. It is easy to achieve physical data independence. With this type of independence, user is able to change the physical storage structures or the devices which have an effect on the conceptual schema.

**Examples of changes under Physical Data Independence :**

* It is by the use of new storage device like Hard Drive or Magnetic Tapes
* Modifying the file organization technique in the Database
* Switching to different data structures.
* Changing the access method.
* Modifying indexes.
* To change the compression techniques or hashing algorithms.
* To change the Location of Database from say C drive to D Drive.

**2. Logical Data Independence :**   
Logical Data Independence is used to change the conceptual scheme without changing the following things :

* External views
* External API or programs

**Examples of changes under Logical Data Independence :**

* To Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs
* Merging two records into one
* To break an existing record i.e to divide the record into two or more records

|  |  |
| --- | --- |
| **Physical Data Independence** | **Logical Data Independence** |
| It mainly concern about how the data is stored into the system. | It mainly concerned about the structure or the changing data definition. |
| It is easy to retrieve. | It is difficult to retrieve because the data is mainly dependent on the logical structure of data. |
| As compared to the logical independence it is easy to achieve physical data independence. | As compared to the physical independence it is not easy to achieve logical data independence. |
| Any change at the physical level, does not require to change at the application level. | The change in the logical level requires a change at the application level. |
| The modifications made at the internal level may or may not be needed to improve the performance of the structure. | The modifications made at the logical level is significant whenever the logical structure of the database is to be changed. |
| It is concerned with the internal schema. | It is concerned with the conceptual schema. |
| Example: Change in compression techniques, Hashing algorithms and storage devices etc. | Example: Add/Modify or Delete a new attribute. |

***7.A Data Dictionary Definition***

A Data Dictionary is a collection of names, definitions, and attributes about data elements that are being used or captured in a database, information system, or part of a research project. It describes the meanings and purposes of data elements within the context of a project, and provides guidance on interpretation, accepted meanings and representation. A Data Dictionary also provides metadata about data elements. The metadata included in a Data Dictionary can assist in defining the scope and characteristics of data elements, as well the rules for their usage and application.

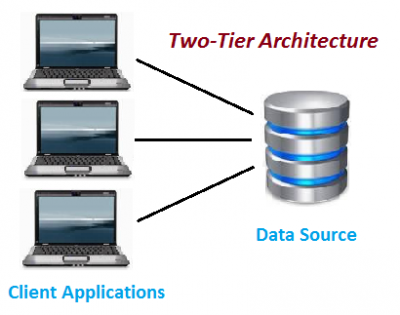
Referecing vs Referenced Relation :

The relation that is referencing to the other relation is known as REFERENCING RELATION , while that relation to which the other relations refer is known as REFERENCED RELATION.

**8.Advantages and Disadvantages of Three-Tier Architecture in DBMS:**

Two-Tier Architecture:

The two-tier is based on Client Server architecture. The two-tier architecture is like client server application. The direct communication takes place between client and server. There is no intermediate between client and server. Because of tight coupling a 2 tiered application will run faster.



The above figure shows the architecture of two-tier. Here the direct communication between client and server, there is no intermediate between client and server.

The Two-tier architecture is divided into two parts:

1) Client Application (Client Tier)

2) Database (Data Tier)

On client application side the code is written for saving the data in the SQL server database. Client sends the request to server and it process the request & send back with data. The main problem of two tier architecture is the server cannot respond multiple request same time, as a result it cause a data integrity issue.

**Advantages:**

->Easy to maintain and modification is bit easy

->Communication is faster

**Disadvantages:**

->In two tier architecture application performance will be degrade upon increasing the users.

->Cost-ineffective

**Three-Tier Architecture:**

Three-tier architecture typically comprise a presentation tier, a business or data access tier, and a data tier. Three layers in the three tier architecture are as follows:

1) Client layer

2) Business layer

3) Data layer

**1) Client layer:**

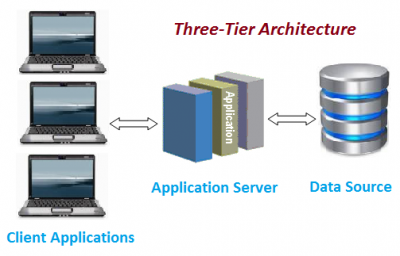
It is also called as Presentation layer which contains UI part of our application. This layer is used for the design purpose where data is presented to the user or input is taken from the user. For example designing registration form which contains text box, label, button etc.

**2) Business layer:**

In this layer all business logic written like validation of data, calculations, data insertion etc. This acts as a interface between Client layer and Data Access Layer. This layer is also called the intermediary layer helps to make communication faster between client and data layer.

**3) Data layer:**

In this layer actual database is comes in the picture. Data Access Layer contains methods to connect with database and to perform insert, update, delete, get data from database based on our input data.



**Advantages :**

->High performance, lightweight persistent objects

->Scalability – Each tier can scale horizontally

->Performance – Because the Presentation tier can cache requests, network utilization is minimized, and the load is reduced on the Application and Data tiers.

->High degree of flexibility in deployment platform and configuration

->Better Re-use

->Improve Data Integrity

->Improved Security – Client is not direct access to database.

->Easy to maintain and modification is bit easy, won’t affect other modules

->In three tier architecture application performance is good.

**Disadvantages :**

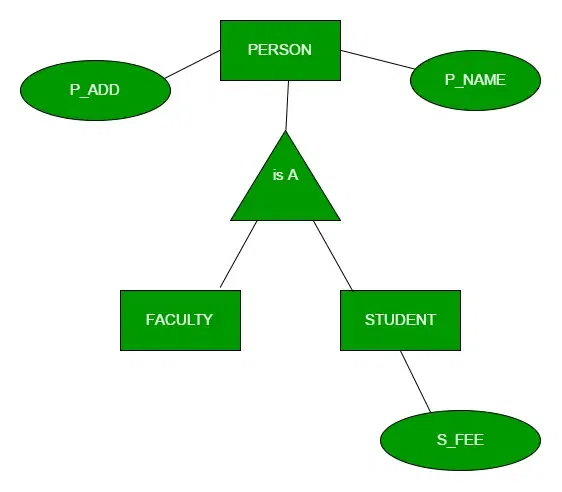
->Increase Complexity/Effort

**9.Generalization, Specialization and Aggregation in ER Model :**

**Generalization**

Generalization is the process of extracting common properties from a set of entities and creating a generalized entity from it. It is a bottom-up approach in which two or more entities can be generalized to a higher-level entity if they have some attributes in common. For Example, STUDENT and FACULTY can be generalized to a higher-level entity called PERSON as shown in Figure 1. In this case, common attributes like P\_NAME, and P\_ADD become part of a higher entity (PERSON), and specialized attributes like S\_FEE become part of a specialized entity (STUDENT).

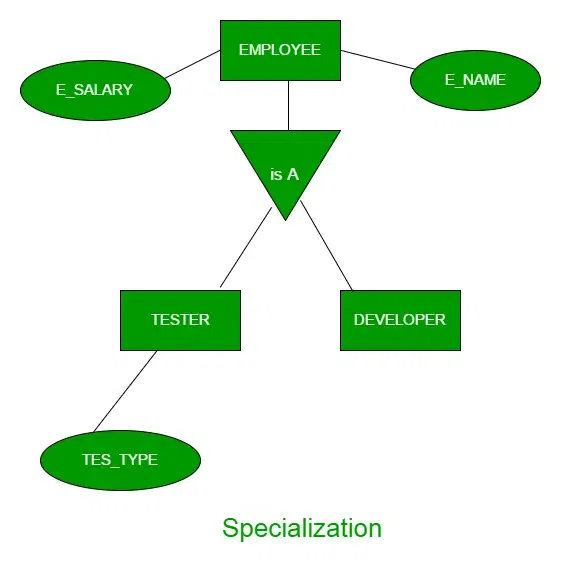
Generalization is also called as ‘ Bottom-up approach”.



**Specialization :**

In specialization, an entity is divided into sub-entities based on its characteristics. It is a top-down approach where the higher-level entity is specialized into two or more lower-level entities. For Example, an EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER, etc. as shown in Figure 2. In this case, common attributes like E\_NAME, E\_SAL, etc. become part of a higher entity (EMPLOYEE), and specialized attributes like TES\_TYPE become part of a specialized entity (TESTER).

Specialization is also called as ” Top-Down approch”.



**Inheritance:** It is an important feature of generalization and specialization

Attribute inheritance: allows lower level entities to inherit the attributes of higher level entities and vice versa.

in diagram: Car entity is an inheritance of Vehicle entity ,So Car can acquire attributes of Vehicle example:car can acquire Model attribute of Vehicle.

attribute

**example of Attribute Inheritance :**

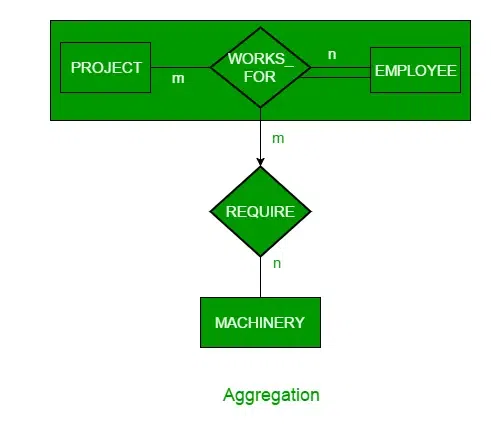
**Participation inheritance:** In participation inheritance, relationships involving higher level entity set also inherited by lower level entity and vice versa.

in diagram: Vehicle entity has an relationship with Cycle entity ,So Cycle entity can acquire attributes of lower level entities i.e Car and Bus since it is inheritance of Vehicle.

**Aggregation :**

An ER diagram is not capable of representing the relationship between an entity and a relationship which may be required in some scenarios. In those cases, a relationship with its corresponding entities is aggregated into a higher-level entity. Aggregation is an abstraction through which we can represent relationships as higher-level entity sets.

For Example, an Employee working on a project may require some machinery. So, REQUIRE relationship is needed between the relationship WORKS\_FOR and entity MACHINERY. Using aggregation, WORKS\_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into a single entity and relationship REQUIRE is created between the aggregated entity and MACHINERY.



**Representing Aggregation Via Schema**

To represent aggregation, create a schema containing the following things.

->the primary key to the aggregated relationship

->the primary key to the associated entity set

->descriptive attribute, if exists