**Database management System**

**Department of Computer Science**

**Milan Kumar Nayak**

**6370158954**

**Basudev Godabari Degree College**

At /- Kesaibahal , Dist/- Sambalpur , Pin – 768228

bgdegreecollege@gmail.com **,Mob – 9438000807**

**UNIT- 1**

**Database:**

Database is a collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc. For Example, university database organizes the data about students, faculty, and admin staff etc. which helps in efficient retrieval, insertion and deletion of data from it.

**Database management system :**

 A database management system (DBMS) is a software package designed to define, manipulate, retrieve and manage data in a database. Or we can say that -

 A database management system (DBMS) refers to the technology for creating and managing databases. DBMS is a software tool to organize (create, retrieve, update, and manage) data in a database.

The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Usually, people use software such as DBASE IV or V, Microsoft ACCESS, or EXCEL to store data in the form of a database. A datum is a unit of data. Meaningful data combined to form information. Hence, information is interpreted data - data provided with semantics. MS. ACCESS is one of the most common examples of database management software.

* A **database management system** stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

Characteristics

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics −

* **Real-world entity** − A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.
* **Relation-based tables** − DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.
* **Isolation of data and application** − A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.
* **Less redundancy** − DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.
* **Consistency** − Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.
* **Query Language** − DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.
* **ACID Properties** − DBMS follows the concepts of **A**tomicity, **C**onsistency, **I**solation, and **D**urability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.
* **Multiuser and Concurrent Access** − DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.
* **Multiple views** − DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.
* **Security** − Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

**Why we Use DBMS**

To develop software applications In less time.

* Data independence and efficient use of data.
* For uniform data administration.
* For data integrity and security.
* For concurrent access to data, and data recovery from crashes.
* To use user-friendly declarative query language.

**Where is a Database Management System (DBMS) being Used?**

* Airlines: reservations, schedules, etc
* Telecom: calls made, customer details, network usage, etc
* Universities: registration, results, grades, etc
* Sales: products, purchases, customers, etc
* Banking: all transactions etc

**Advantages of DBMS :-**

A DBMS manages data and has many benefits. These are:

* **Data independence**: Application programs should be as free or independent as possible from details of data representation and storage. DBMS can supply an abstract view of the data for insulating application code from such facts.
* **Efficient data access:** DBMS utilizes a mixture of sophisticated concepts and techniques for storing and retrieving data competently. This feature becomes important in cases where the data is stored on external storage devices.
* **Data integrity and security:** If data is accessed through the DBMS, the DBMS can enforce integrity constraints on the data.
* **Data administration:** When several users share the data, integrating the administration of data can offer significant improvements. Experienced professionals understand the nature of the data being managed and can be responsible for organizing the data representation to reduce redundancy and make the data to retrieve efficiently.

**Component of DBMS**

* Users: Users may be of any kind such as DB administrator, System developer, or database users.
* Database application: Database application may be Departmental, Personal, organization's and / or Internal.
* DBMS: Software that allows users to create and manipulate database access,
* Database: Collection of logical data as a single unit.

Users :-

A typical DBMS has users with different rights and permissions who use it for different purposes. Some users retrieve data and some back it up. The users of a DBMS can be broadly categorized as follows −

* **Administrators** − Administrators maintain the DBMS and are responsible for administrating the database. They are responsible to look after its usage and by whom it should be used. They create access profiles for users and apply limitations to maintain isolation and force security. Administrators also look after DBMS resources like system license, required tools, and other software and hardware related maintenance.
* **Designers** − Designers are the group of people who actually work on the designing part of the database. They keep a close watch on what data should be kept and in what format. They identify and design the whole set of entities, relations, constraints, and views.
* **End Users** − End users are those who actually reap the benefits of having a DBMS. End users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.

**-:Database system concept and Architecture :-**

**Data models :-**

Data models define how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

The very first data model could be flat data-models, where all the data used are to be kept in the same plane. Earlier data models were not so scientific, hence there are many model but among them the **Relational Model** is the most widely used database model.

 there are other models too -

* Hierarchical Model
* Network Model
* Entity-relationship Model
* Relational Model

**Hierarchical Model**

 This database model organises data into a tree-like-structure, with a single root, to which all the other data is linked. The heirarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node.

This model efficiently describes many real-world relationships like index of a book, recipes etc.

In hierarchical model, data is organised into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.



**Network Model :**

This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node.

In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.

This was the most widely used database model, before Relational Model was introduced.

****

**Database Schema:-**

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It’s the database designers who design the schema to help programmers understand the database and make it useful.

A database schema can be divided broadly into two categories −

* **Physical Database Schema** − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
* **Logical Database Schema** − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Database Instance :-

 It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

If a database system is not multi-layered, then it becomes difficult to make any changes in the database system. Database systems are designed in multi-layers as we learnt earlier.

Data Independence

A database system normally contains a lot of data in addition to users’ data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.

Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

Physical Data Independence

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

#

# -:Entity Relationship Model:-

* ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
* It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
* In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

**For example,** Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.

##

## C:\Users\Jagannath\Desktop\dbms-er-model-concept-diagram.pngComponent of ER model:-

## Entity

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

##

Example of Entity in DBMS :

Entity in DBMS can be a real-world object with an existence, For example, in a **College**database, the entities can be Professor, Students, Courses, etc.

Entities has attributes, which can be considered as properties describing it, for example, for Professor entity, the attributes are  **Professor\_Name, Professor\_Address, Professor\_Salary,** etc. The attribute value gets stored in the database.

**<Professor>**

|  |  |  |  |
| --- | --- | --- | --- |
| **Professor\_ID** | **Professor\_Name** | **Professor\_City** | **Professor\_Salary** |
| P01 | Tom | Sydney | $7000 |
| P02 | David | Brisbane | $4500 |
| P03 | Mark | Perth | $5000 |

Here, **Professor\_Name, Professor \_Address and Professor \_Salary** are attributes.
          **Professor\_ID** is the primary key

## Types of DBMS Entities

The following are the types of entities in DBMS −

**A ) Strong Entity**

The strong entity has a primary key. Weak entities are dependent on strong entity. Its existence is not dependent on any other entity.

Strong Entity is represented by a single rectangle –

**Strong entity**

Continuing our previous example, **Professor**is a strong entity here, and the primary key is  **Professor\_ID.**

**B ) Weak Entity**

The weak entity in DBMS do not have a primary key and are dependent on the parent entity. It mainly depends on other entities.

Weak Entity is represented by double rectangle −

**Weak entity**

Continuing our previous example, **Professor**is a strong entity, and the primary key is  **Professor\_ID**. However, another entity is**Professor\_Dependents**, which is our Weak Entity.

**<Professor\_Dependents>**

|  |  |  |
| --- | --- | --- |
| **Name** | **DOB** | **Relation** |

This is a weak entity since its existence is dependent on another entity **Professor**, which we saw above. A Professor has Dependents.

## Attributes :-

Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).

**Composite**

If the attributes are **composite**, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.

****

**Multivalued**

Multivalued attributes are depicted by double ellipse.



**Derived**

Derived attributes are depicted by dashed ellipse.



## ER - Relationship

The association among entities is called a relationship. For example, an employee **works\_at** a department, a student **enrolls** in a course. Here, Works\_at and Enrolls are called relationships.

### Relationship Set :

A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.

### Degree of Relationship :

The number of participating entities in a relationship defines the degree of the relationship.

* Binary = degree 2
* Ternary = degree 3
* n-ary = degree
* Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

### Binary Relationship and Cardinality

A relationship where two entities are participating is called a **binary relationship**. Cardinality is the number of instance of an entity from a relation that can be associated with the relation.

* **One-to-one** − When only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.

 ****

* **One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship.
* **Many-to-one** − When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship.
* **Many-to-many** − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.



### Participation Constraints

* **Total Participation** − Each entity is involved in the relationship. Total participation is represented by double lines.
* **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.



### Entity-Set and Keys :-

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, the roll\_number of a student makes him/her identifiable among students.

* **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
* **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.
* **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

# Enhanced Entity Relationship Model

**Prerequisite –**
 Today’s time the complexity of the data is increasing so it becomes more and more difficult to use the traditional ER model for database modeling. To reduce this complexity of modeling we have to make improvements or enhancements were made to the existing ER model to make it able to handle the complex application in a better way.
Enhanced entity-relationship diagrams are advanced database diagrams very similar to regular ER diagrams which represents requirements and complexities of complex databases.
It is a diagrammatic technique for displaying the Sub Class and Super Class; Specialization and Generalization; Union or Category; Aggregation etc.

**Generalization and Specialization –**
 These are very common relationship found in real entities. However this kind of relationships was added later as enhanced extension to classical ER model. **Specialized class** are often called as **subclass** while **generalized class** are called superclass, probably inspired by object oriented programming. A sub-class is best understood by **“IS-A analysis”**. Following statements hopefully makes some sense to your mind “Technician IS-A Employee”, “Laptop IS-A Computer”.

An entity is specialized type/class of other entity. For example, Technician is special Employee in a university system Faculty is special class of Employee. We call this phenomenon as generalization/specialization. In the example here Employee is generalized entity class while Technician and Faculty are specialized class of Employee.

**Example –** This example instance of **“sub-class”** relationships. Here we have four sets employee: Secretary, Technician, and Engineer. Employee is super-class of rest three set of individual sub-class is subset of Employee set.

****

* An entity belonging to a sub-class is related with some super-class entity. For instance emp no 1001 is a secretary, and his typing speed is 68. Emp no 1009 is engineer (sub-class) and her trade is “Electrical”, so forth.
* Sub-class entity “inherits” all attributes of super-class; for example employee 1001 will have attributes eno, name, salary, and typing speed.

**Enhanced ER model of above example –**

