FACULTY NAME : DR.K.GEETHA

DESIGNATION : GUEST LECTURER

DEPARTMENT : SCHOOL OF COMPUTER SCIENCE,ENGINEERING AND APPLICATIONS

CLASS : M.S.C (DS)

SEMESTER : I

SUBJECT : DATABASE SYSTEMS

SUBJECT CODE : MDS24014

1. Database Basics

- **Database**: A collection of related data organized in a way that supports efficient retrieval, insertion, and deletion of data.
- **DBMS (Database Management System)**: Software that enables users to define, create, maintain, and control access to the database.
- **Data Models**: Frameworks for organizing data, including Hierarchical, Network, Relational, and Object-oriented models.
- **Schema**: The structure that defines the organization of data, including tables, fields, and relationships in the database.

2. Relational Model

- **Table (Relation)**: A set of rows and columns, where each row represents a record and each column represents a field.
- **Primary Key**: A unique identifier for each record in a table.
- **Foreign Key**: A field in one table that references the primary key of another table, establishing a relationship.
- **Normalization**: The process of structuring a relational database to reduce redundancy and improve data integrity, typically achieved through Normal Forms (1NF, 2NF, 3NF, BCNF).

3. SQL (Structured Query Language)

- **Data Definition Language (DDL)**: Commands to define database structure (e.g., `CREATE`, `ALTER`, `DROP`).
- **Data Manipulation Language (DML)**: Commands for data handling (e.g., `SELECT`, `INSERT`, `UPDATE`, `DELETE`).
- **Data Control Language (DCL)**: Commands to control access (e.g., `GRANT`, `REVOKE`).

4. Transactions and Concurrency Control

- **Transaction**: A sequence of database operations that must be completed as a single unit.
- **ACID Properties**: Principles for reliable transactions:
  - **Atomicity**: Ensures all operations in a transaction complete successfully.
  - **Consistency**: Ensures transactions lead from one valid state to another.
  - **Isolation**: Transactions operate independently.
  - **Durability**: Once completed, transactions persist even in case of a system failure.
- **Concurrency Control**: Techniques to handle multiple transactions, such as locking and timestamping.

5. Indexes

- **Index**: A database structure that improves the speed of data retrieval.

- **Types**: Primary indexes (based on primary key), Secondary indexes (non-primary keys), and Unique indexes.

6. Database Security

- **Access Control**: Defining who can access the data and what operations they can perform.
- **Encryption**: Securing data by transforming it into an unreadable format unless decrypted.
- **Authentication**: Verifying the identity of users who access the database.

7. Backup and Recovery

- **Backup**: Creating copies of database files to protect against data loss.
- **Recovery**: Techniques to restore the database to its correct state in case of failure, such as point-in-time recovery and full database restore.

8. Data Warehousing and Data Mining

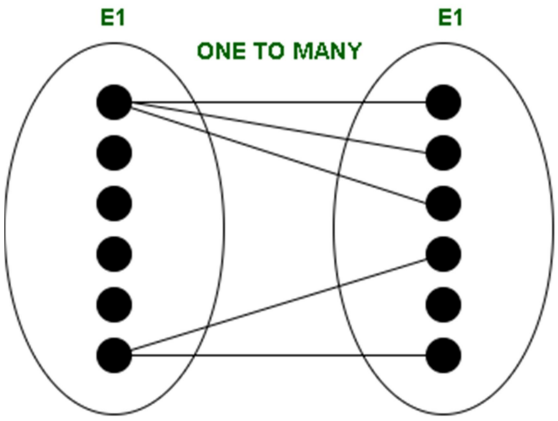
- **Data Warehouse**: A system for reporting and data analysis, integrating data from multiple sources.
- **Data Mining**: The process of discovering patterns and insights from large sets of data using statistical and machine-learning techniques.

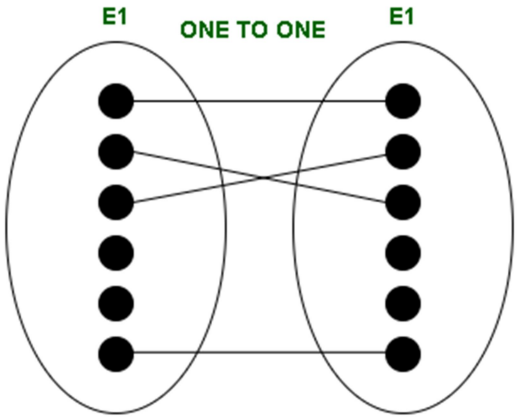
These notes give a high-level overview of essential database system concepts; each topic can be explored in more detail for comprehensive understanding.

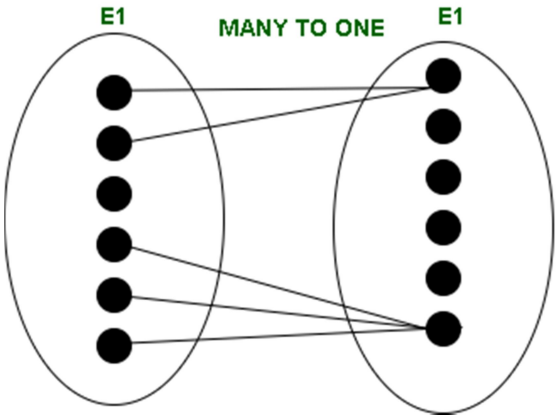
Structural Constraints of Relationships in ER Model

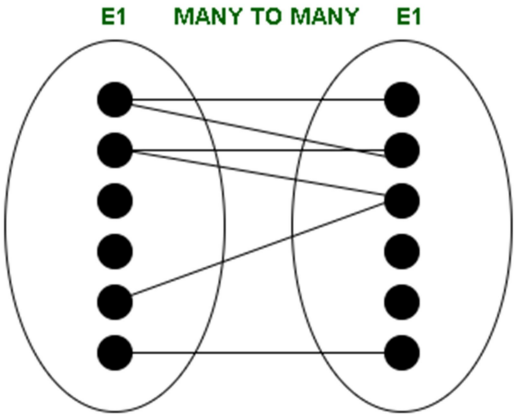
- Prerequisite – ER Model To understand Structural Constraints, we must take a look at Cardinality Ratios and Participation Constraints. **Cardinality Ratios of relationships :** The entities are denoted by rectangle and relationships by diamond.

  There are numbers (represented by M and N) written above the lines which connect relationships and entities. These are called cardinality ratios. These represent the maximum number of entities that can be associated with each other through relationship, R. **Types of Cardinality :** There can be 4 types of cardinality –









1. **One-to-one (1:1) –** When one entity in each entity set takes part at most once in the relationship, the cardinality is one-to-one.

2. **One-to-many (1: N) –** If entities in the first entity set take part in the relationship set at most once and entities in the second entity set take part many times (at least twice), the cardinality is said to be one-to-many.
3. **Many-to-one (N:1) –** If entities in the first entity set take part in the relationship set many times (at least twice), while entities in the second entity set take part at most once, the cardinality is said to be many-to-one.
4. **Many-to-many (N: N) –** The cardinality is said to be many to many if entities in both the entity sets take part many times (at least twice) in the relationship set

SQL CREATE INDEX Statement

The `CREATE INDEX` statement is used to create indexes in tables.

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

CREATE INDEX Syntax

Creates an index on a table. Duplicate values are allowed:

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

CREATE UNIQUE INDEX Syntax

Creates a unique index on a table. Duplicate values are not allowed:

```
CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

Note: The syntax for creating indexes varies among different databases. The refore: Check the syntax for creating indexes in your database.

CREATE INDEX Example

The SQL statement below creates an index named "idx_lastname" on the "LastName" column in the "Persons" table:

```
CREATE INDEX idx_lastname
ON Persons (LastName);
```

If you want to create an index on a combination of columns, you can list the column names within the parentheses, separated by commas:

```
CREATE INDEX idx_pname
ON Persons (LastName, FirstName);
```

Different Types of Database Users

Data StructureDatabaseDBMS

Database users interact with data to update, read and modify the given information on a daily basis. There are various types of database users and we will learn in detail about them.

Database users can be divided into the following types –

- End Users
  - Naive users / Parametric users
  - Sophisticated users
- Application Programmer or Specialized users or Back-End Developer
- System Analyst
- Database Administrator (DBA)
- Temporary Users or Casual Users

These users can access the database and recover the data using various applications.

Let's have a quick understanding of all the types in detail –

- **End Users/Parametric Users** − These users access the database from the front end with the help of a pre-developed application. They have little knowledge about the design and working of databases.

  There are two types of end users −
  - **Naive Users** − These naive users are those users who don't have any database knowledge. They depend on pre-developed applications like Bank Management

Systems, Library Management Systems, Hospital Management Systems, and Railway Ticket Booking Systems(IRCTC) and get the desired result.

- **Sophisticated Users** − These users interact with the system without writing a program and have separate databases for personal use. In the database, the user passes each query to the query processor.

  There are two ways to interact with a system −

  - They use the structure query language to run the query on the database.
  - They use the tool of data analysis software. For example, data engineers and data scientists are familiar with databases.

- **Application programmers/Specialized programmers/Back-End Developer** − These programmers write the code for an application program that uses the database. The application programmer can make the application according to user requirements and control software that runs an entire computer system. The application program is written in any programming language like C#, .net, JAVA, etc., and focuses on business, engineering, and science program.

  Application Programmers are divided into four different types −

  - Web Developers
  - Computer Hardware Programmers
  - Database Developers
  - Software Developers

  Examples of Application programmers develop software like −

  - Content access software
  - Educational Software
  - Information Worker Software
  - Media Development Software
  - Product Engineering Software
  - Enterprise Software

- **System Analyst** − A System Analyst has also known as a business technology analyst. These professionals are responsible for the design, structure, and properties of databases. The application programmer uses the specifications provided by the system analyst to construct the software that is used by end users. The analyst will gather information from the shareholders as well as end users to understand their requirements and translate it into functional specifications for the new system.

  Examples of System Analysts −

  - They serve as team leaders.
  - They are responsible for managing projects.
  - They are the supervisor who manages the lower-level information Staff.

- **Database Administrator (DBA)** − The DBA is the group of people that includes everything required to manage and solve every complex. The DBA can easily use the database to find the information they need and to plan the goal of the database. To meet

future needs, they are ready for future scope and provide solutions for end users. Therefore, they are known for high-level management.

For example −

- To handle the data loss.
- To secure the privacy of data.
- Monitor the recovery and backup of the database.

- **Temporary Users/Casual Users** − These users utilize the database for testing and are only accessible for a limited time. According to business requirements, these users update a little or new information to the database with the help of a database administrator. It helps to maintain the security and integrity of data.

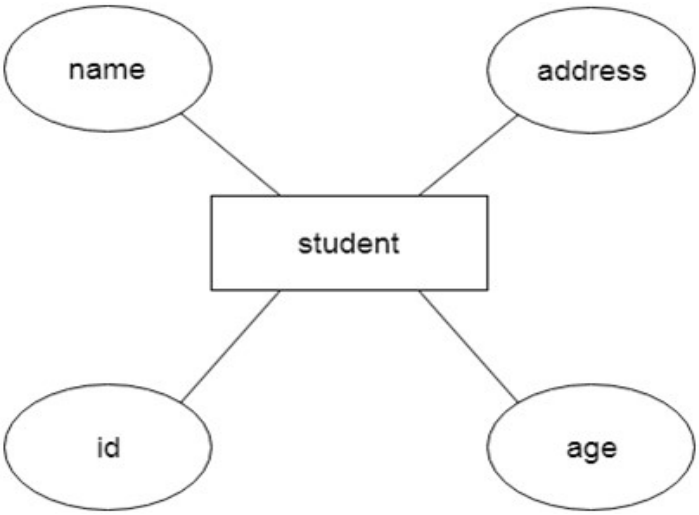
  For example

  High-level management people are temporary users with little knowledge of DBMS.

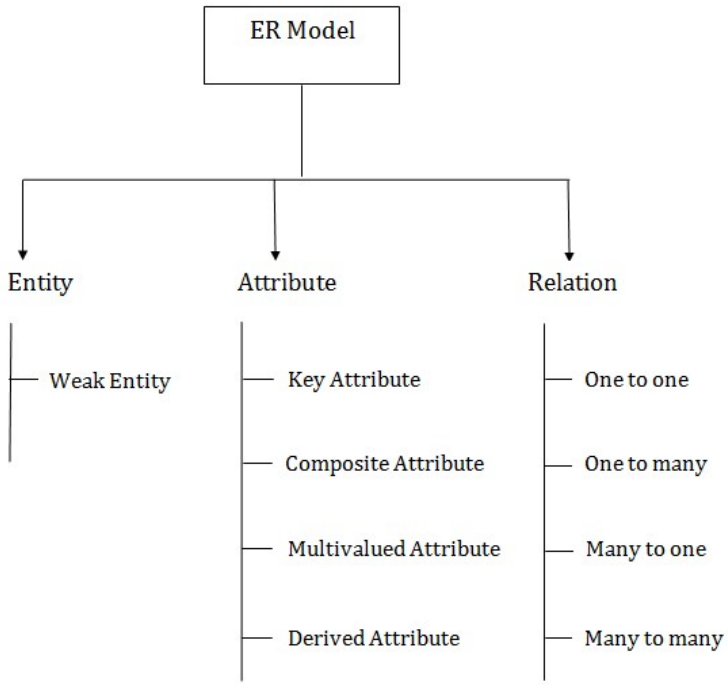
R (Entity Relationship) Diagram in DBMS

- 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-rel63366666
- 36.
- 36.ationship 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.



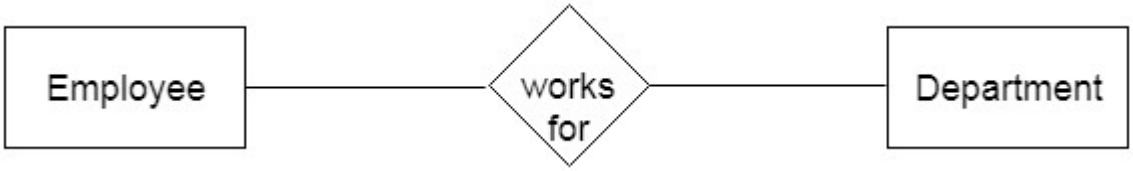
Component of ER Diagram



1. Entity:

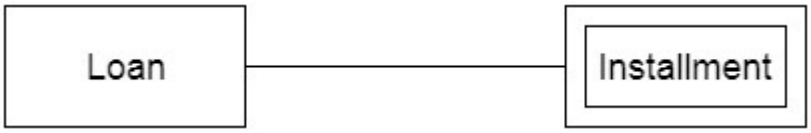
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



a. Weak Entity

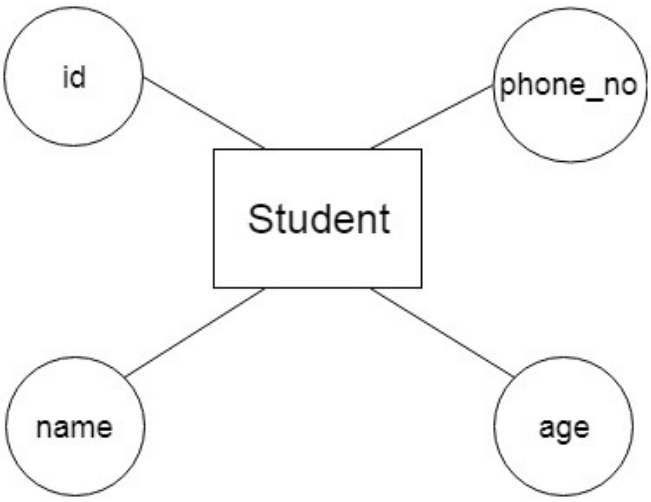
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

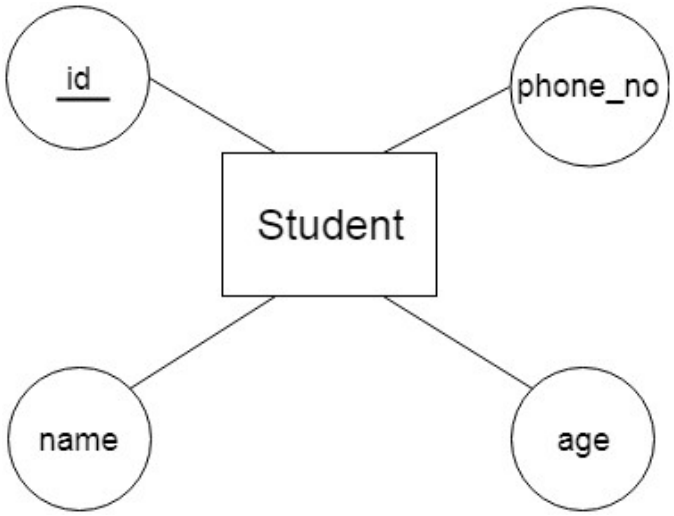
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



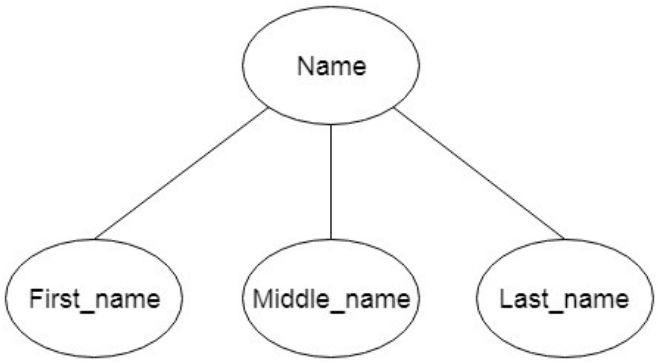
a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b. Composite Attribute

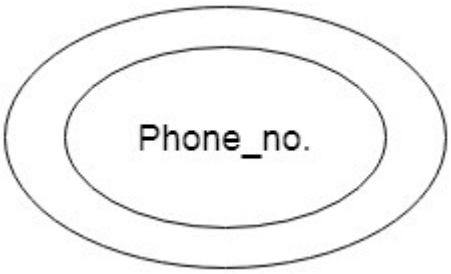
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

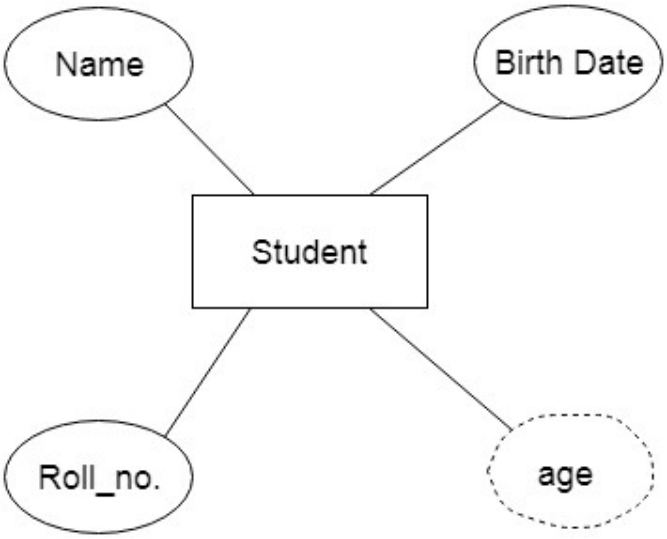
For example, a student can have more than one phone number.



d. Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

Types of relationship are as follows:

a. One-to-One Relationship

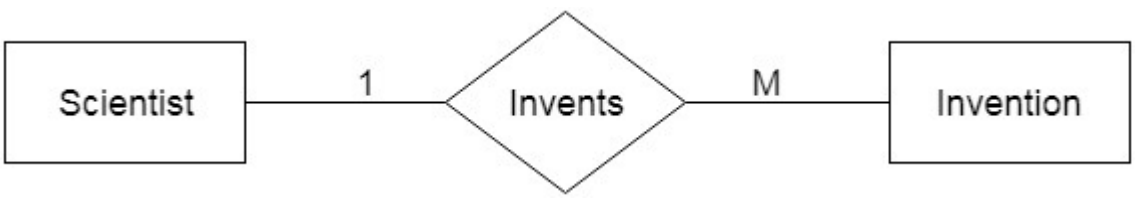
When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

b. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

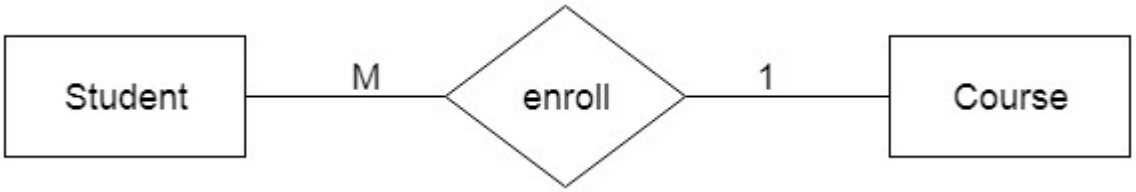
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.



d. Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.