CHANDIGARH ENGINEERING COLLEGE

B.TECH.(CSE 5th Sem.)

Notes Subject: Database Management System

Subject Code: ((BTCS 501-18))

Unit 1

Introduction of DBMS (Database Management System)

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

A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner. It allows users to create, modify, and query a database, as well as manage the security and access controls for that database. DBMS provides an environment to store and retrieve data in convenient and efficient manner.

Key Features of DBMS

- **Data modelling:** A DBMS provides tools for creating and modifying data models, which define the structure and relationships of the data in a database.
- **Data storage and retrieval:** A DBMS is responsible for storing and retrieving data from the database, and can provide various methods for searching and querying the data.
- **Concurrency control:** A DBMS provides mechanisms for controlling concurrent access to the database, to ensure that multiple users can access the data without conflicting with each other.
- **Data integrity and security:** A DBMS provides tools for enforcing data integrity and security constraints, such as constraints on the values of data and access controls that restrict who can access the data.
- **Backup and recovery:** A DBMS provides mechanisms for backing up and recovering the data in the event of a system failure.
- **DBMS can be classified into two types:** Relational Database Management System (RDBMS) and Non-Relational Database Management System (NoSQL or Non-SQL)

DBMS Architecture 1-level, 2-Level, 3-Level

There are two types of database models that we generally use, logical model and physical model. Several types of architecture are there in the database which we will deal with in the next section.

Types of DBMS Architecture

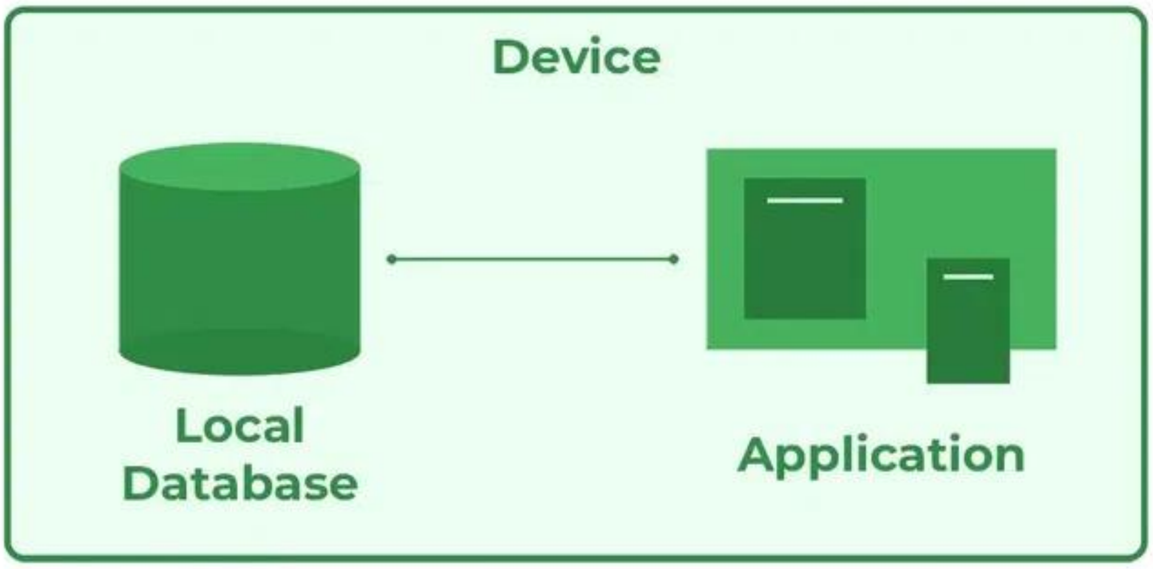
There are several types of DBMS Architecture that we use according to the usage requirements. Types of DBMS Architecture are discussed here.

- 1-Tier Architecture
- 2-Tier Architecture
- 3-Tier Architecture

The **3-level DBMS architecture** provides logical and physical data independence. For more insights, the **GATE CS Self-Paced Course** covers DBMS comprehensively

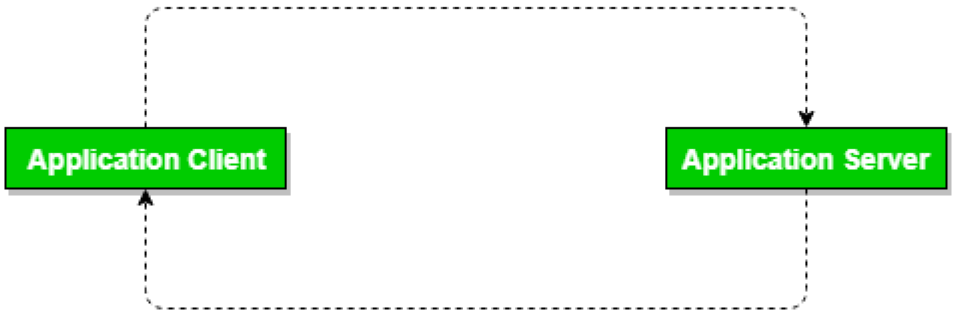
1-Tier Architecture

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. For Example: to learn SQL we set up an SQL server and the database on the local system. This enables us to directly interact with the relational database and execute operations. The industry won't use this architecture they logically go for 2-tier and 3-tier Architecture.



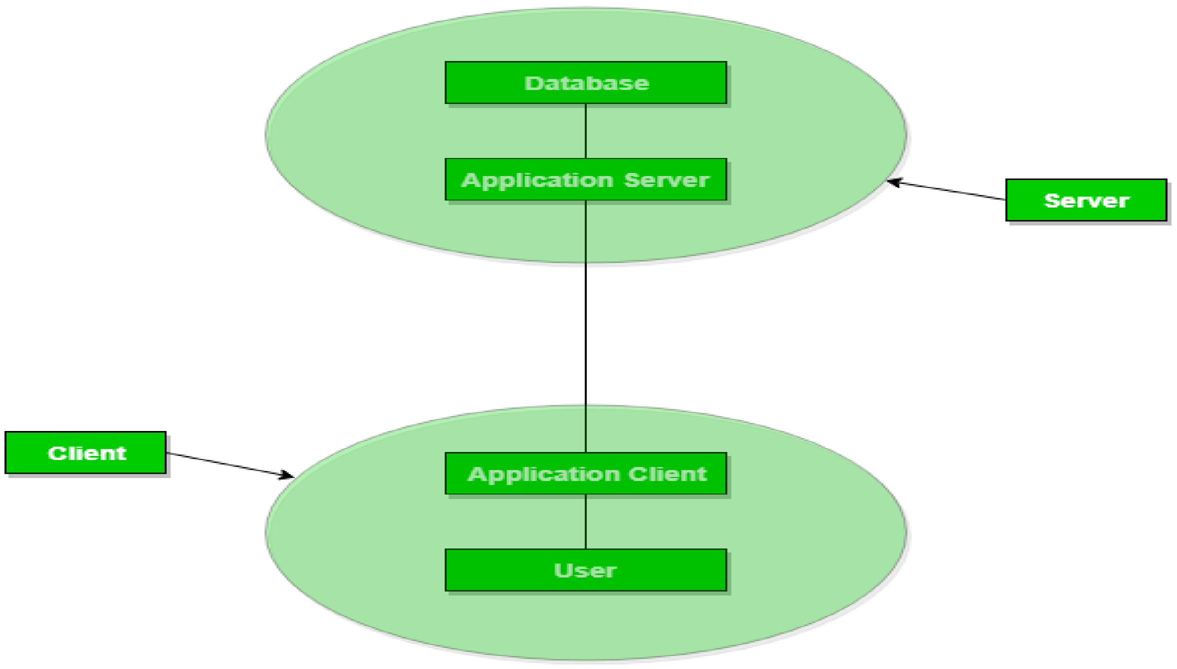
2-Tier Architecture

The 2-tier architecture is similar to a basic 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. The server side is responsible for providing query processing and transaction management functionalities. On the client side, the user interfaces and application programs are run. The application on the client side establishes a connection with the server side to communicate with the DBMS.
An advantage of this type is that maintenance and understanding are easier, and compatible with existing systems. However, this model gives poor performance when there are a large number of users.



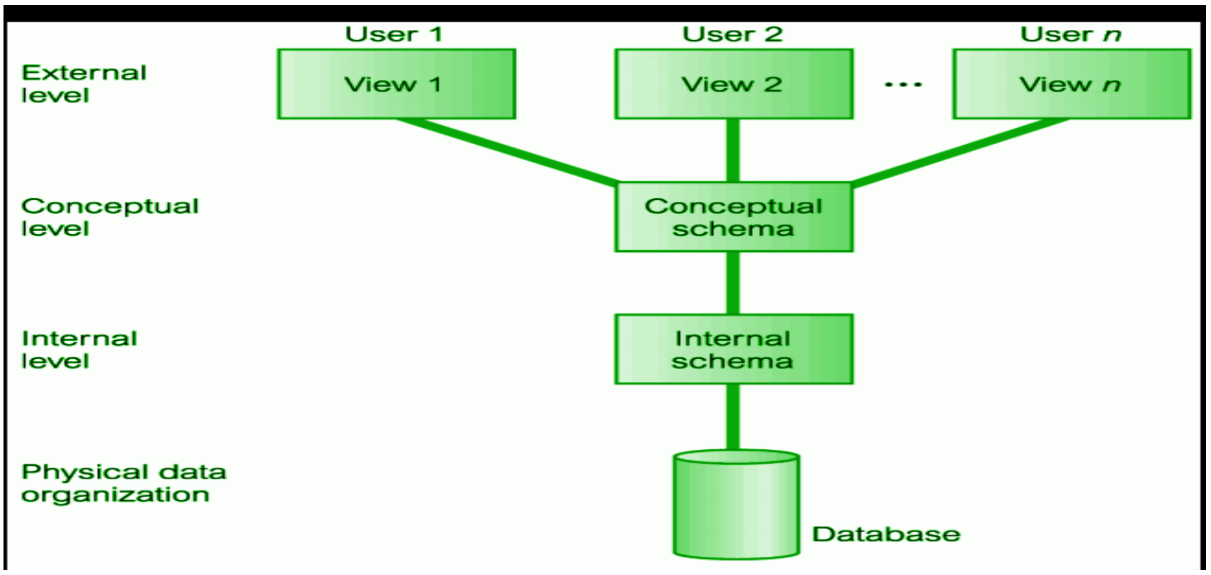
3-Tier Architecture

In 3-Tier Architecture , 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.



What is Data Independence in DBMS?

Data independence is a property of a database management system by which we can change the database schema at one level of the database system without changing the database schema at the next higher level. In this article, we will learn in full detail about data independence and will also see its types. If you read it completely, you will understand it easily.



Types of Data Independence

There are two types of data independence.

- logical data independence
- Physical data independence

Logical Data Independence

- Changing the logical schema (conceptual level) without changing the external schema (view level) is called logical data independence.
- It is used to keep the external schema separate from the logical schema.
- If we make any changes at the conceptual level of data, it does not affect the view level.
- This happens at the user interface level.
- For example, it is possible to add or delete new entities, attributes to the conceptual schema without making any changes to the external schema.

Physical Data Independence

- Making changes to the physical schema without changing the logical schema is called physical data independence.
- If we change the storage size of the database system server, it will not affect the conceptual structure of the database.
- It is used to keep the conceptual level separate from the internal level.
- This happens at the logical interface level.
- Example – Changing the location of the database from C drive to D drive.Data Definition Language (DDL) in SQL

Data Definition Language (DDL) in SQL

DDL or Data Definition Language actually consists of the SQL commands that can be used to **defining**, **altering**, and **deleting** database structures such as **tables**, **indexes**, and **schemas**. It simply deals with descriptions of the database schema and is used to **create** and **modify** the structure of database objects in the database

- **Common DDL Commands**

Command	**Description**	**Syntax**
CREATE	Create database or its objects (table, index, function, views, store procedure, and triggers)	**CREATE** TABLE table_name (column1 data_type, column2 data_type, ...);
DROP	Delete objects from the database	**DROP** TABLE table_name;
ALTER	Alter the structure of the database	**ALTER** TABLE table_name ADD COLUMN column_name data_type;
TRUNCATE	Remove all records from a table, including all spaces allocated for the records are removed	**TRUNCATE** TABLE table_name;
COMMENT	Add comments to the data dictionary	**COMMENT** 'comment_text' ON TABLE table_name;
RENAME	Rename an object existing in the database	**RENAME** TABLE old_table_name TO new_table_name;

Data Manipulation Language (DML) in SQL
The **SQL commands** that deal with the **manipulation of data** present in the database belong to **DML** or Data Manipulation Language and this includes most of the **SQL statements.** It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

Common DML Commands

Command	**Description**	**Syntax**
INSERT	Insert data into a table	**INSERT** INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);
UPDATE	Update existing data within a table	**UPDATE** table_name SET column1 = value1, column2 = value2 WHERE condition;
DELETE	Delete records from a database table	**DELETE** FROM table_name WHERE condition;
LOCK	Table control concurrency	**LOCK** TABLE table_name IN lock_mode;
CALL	Call a PL/SQL or JAVA subprogram	**CALL** procedure_name(arguments);
EXPLAIN PLAN	Describe the access path to data	**EXPLAIN PLAN** FOR SELECT * FROM table_name;

Data Models in DBMS

A Data Model in Database Management System (DBMS) is the concept of tools that are developed to summarize the description of the database. Data Models provide us with a transparent picture of data which helps us in creating an actual database. It shows us from the design of the data to its proper implementation of data.

1. **ER Data Model**

The Entity Relationship Model is a model for identifying entities to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.

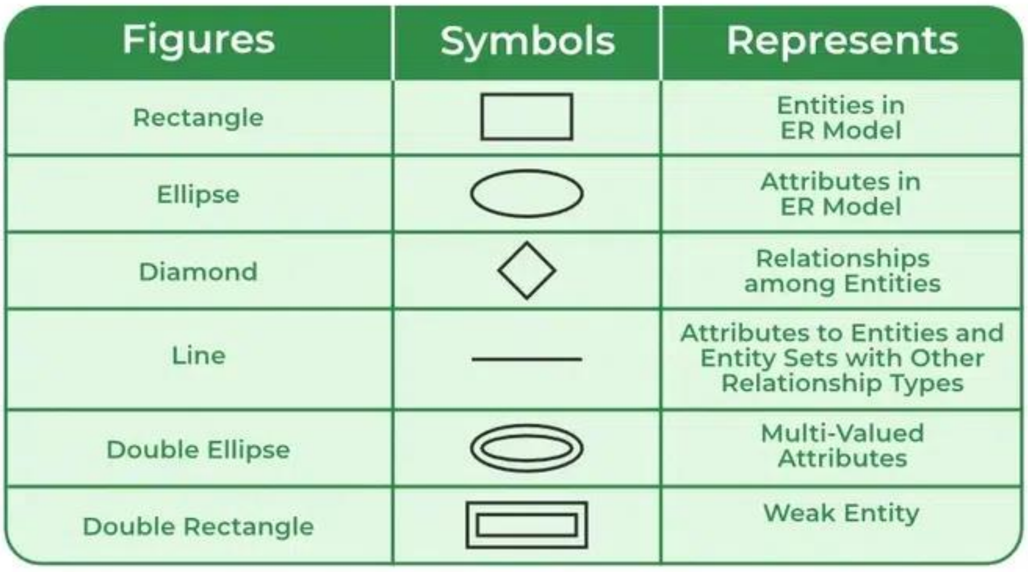
- ER diagrams represent the E-R model in a database, making them easy to convert into relations (tables).
- ER diagrams provide the purpose of real-world modelling of objects which makes them intently useful.

- ER diagrams require no technical knowledge and no hardware support.
- These diagrams are very easy to understand and easy to create even for a naive user.
- It gives a standard solution for visualizing the data logically.

  Symbols Used in ER Model

  ER Model is used to model the logical view of the system from a data perspective which consists of these symbols:

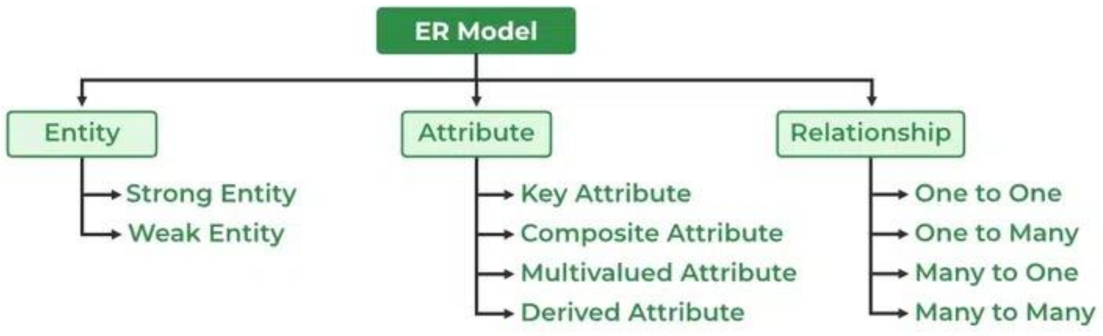
- **Rectangles:** Rectangles represent Entities in the ER Model.
- **Ellipses:** Ellipses represent Attributes in the ER Model.
- **Diamond:** Diamonds represent Relationships among Entities.
- **Lines:** Lines represent attributes to entities and entity sets with other relationship types.
- **Double Ellipse:** Double Ellipses represent Multi-Valued Attributes.
- **Double Rectangle:** Double Rectangle represents a Weak Entity.



Figures	Symbols	Represents
Rectangle	▭	Entities in ER Model
Ellipse	⬭	Attributes in ER Model
Diamond	◇	Relationships among Entities
Line	—	Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse	◎	Multi-Valued Attributes
Double Rectangle	▣	Weak Entity

Components of ER Diagram

ER Model consists of Entities, Attributes, and Relationships among Entities in a Database System.

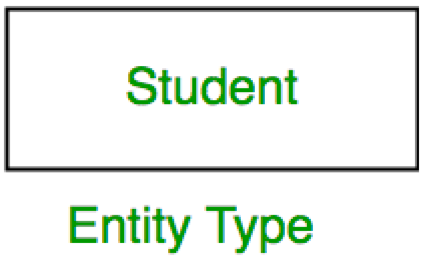


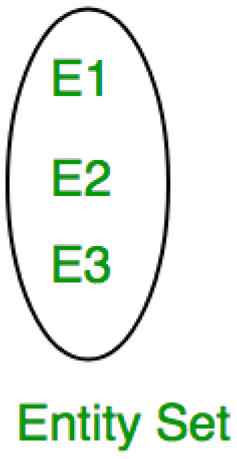
What is Entity?

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

What is Entity Set?

An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as:





Entity Set

We can represent the entity set in ER Diagram but can't represent entity in ER Diagram because entity is row and column in the relation and ER Diagram is graphical representation of data.

Types of Entity

There are two types of entity:

1. Strong Entity

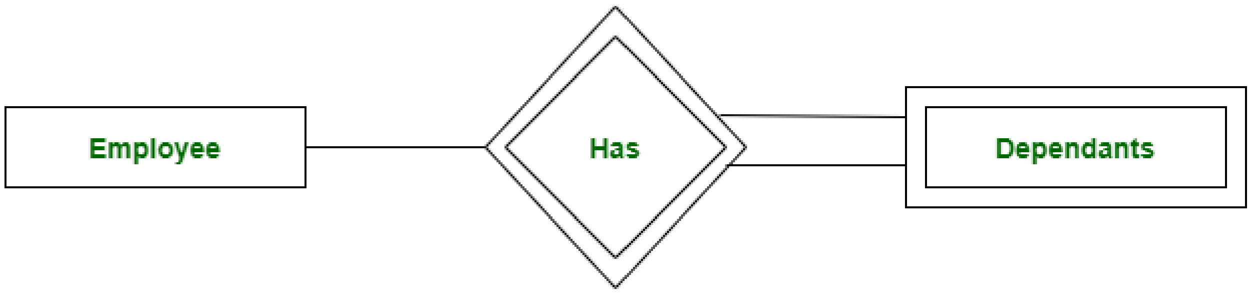
A Strong Entity is a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has a primary key, that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

2. Weak Entity

An Entity type has a key attribute that uniquely identifies each entity in the entity set. But some entity type exists for which key attributes can't be defined. These are called Weak Entity types .

For Example, A company may store the information of dependents (Parents, Children, Spouse) of an Employee. But the dependents can't exist without the employee. So Dependent will be a **Weak Entity Type** and Employee will be Identifying Entity type for Dependent, which means it is **Strong Entity Type** .

A weak entity type is represented by a Double Rectangle. The participation of weak entity types is always total. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.



Strong Entity and Weak Entity

What is Attributes?

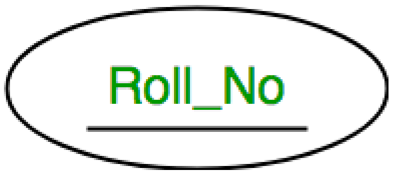
Attributes are the properties that define the entity type. For example, Roll_No, Name, DOB, Age, Address, and Mobile_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.

Attribute

Types of Attributes

1. Key Attribute

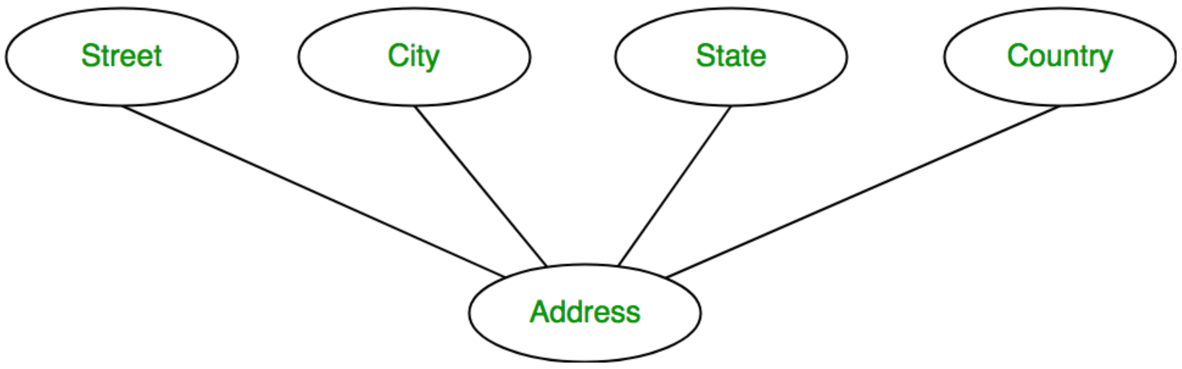
The attribute which **uniquely identifies each entity** in the entity set is called the key attribute. For example, Roll_No will be unique for each student. In ER diagram, the key attribute is represented by an oval with underlying lines.



Key Attribute

2. Composite Attribute

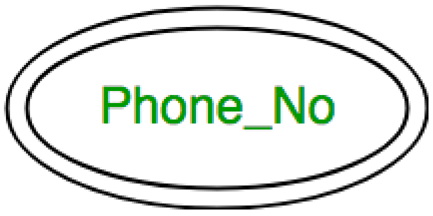
An attribute **composed of many other attributes** is called a composite attribute. For example, the Address attribute of the student Entity type consists of Street, City, State, and Country. In ER diagram, the composite attribute is represented by an oval comprising of ovals.



Composite Attribute

3. Multivalued Attribute

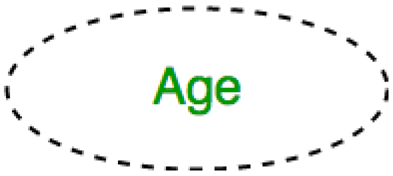
An attribute consisting of more than one value for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.



Multivalued Attribute

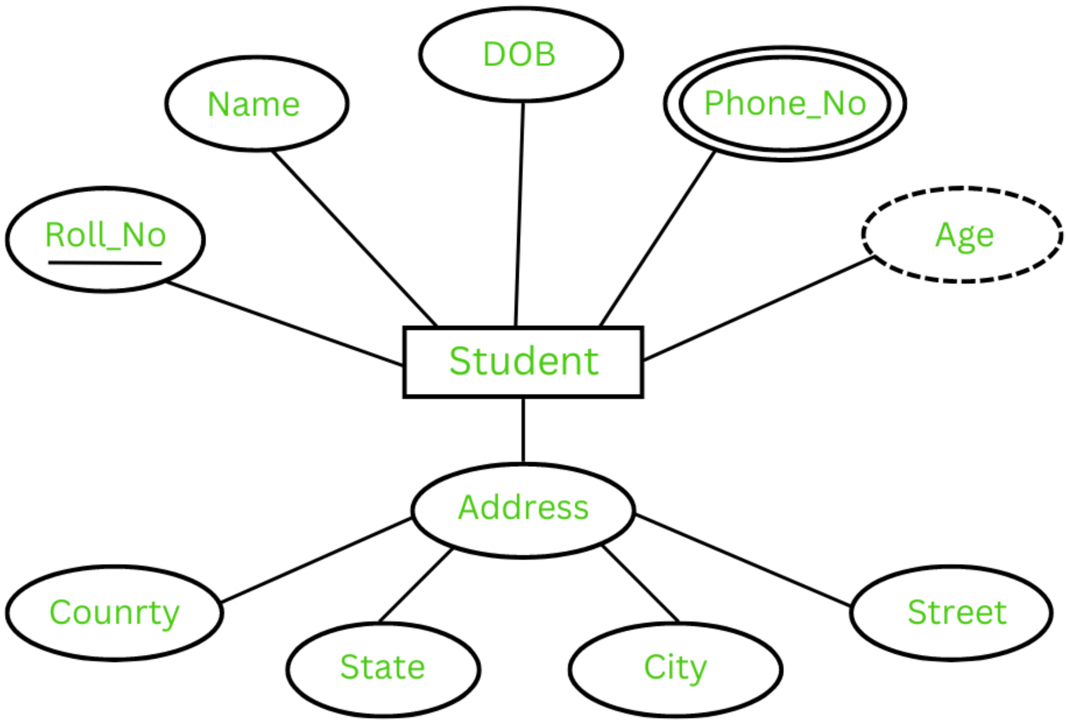
4. Derived Attribute

An attribute that can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval.



Derived Attribute

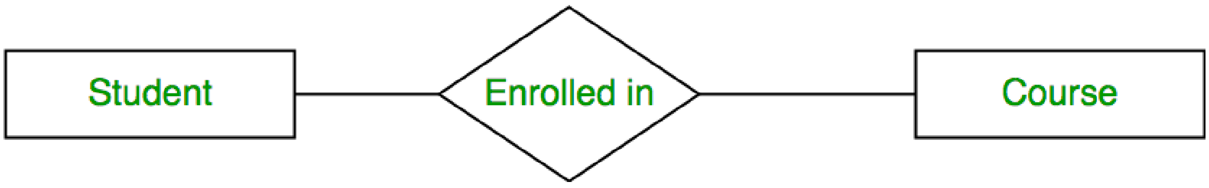
The Complete Entity Type Student with its Attributes can be represented as:



Entity and Attributes

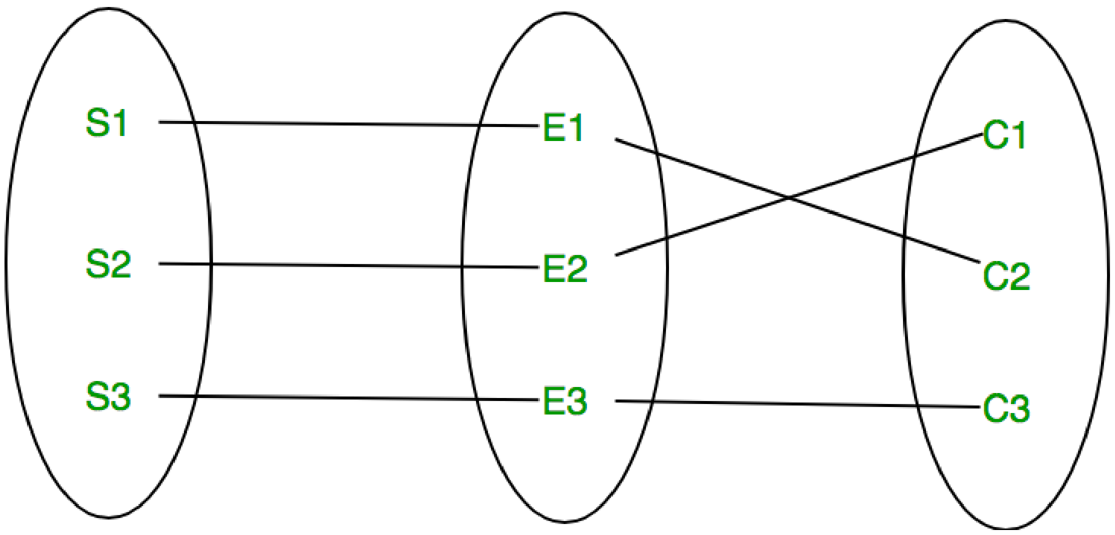
Relationship Type and Relationship Set

A Relationship Type represents the association between entity types. For example, ‘Enrolled in’ is a relationship type that exists between entity type Student and Course. In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.



Entity-Relationship Set

A set of relationships of the same type is known as a relationship set. The following relationship set depicts S1 as enrolled in C2, S2 as enrolled in C1, and S3 as registered in C3.

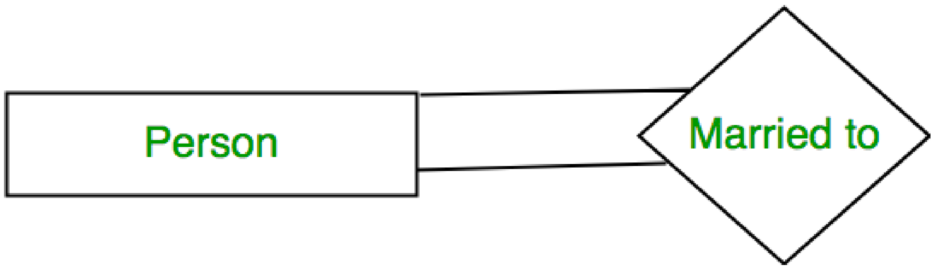


Relationship Set

Degree of a Relationship Set

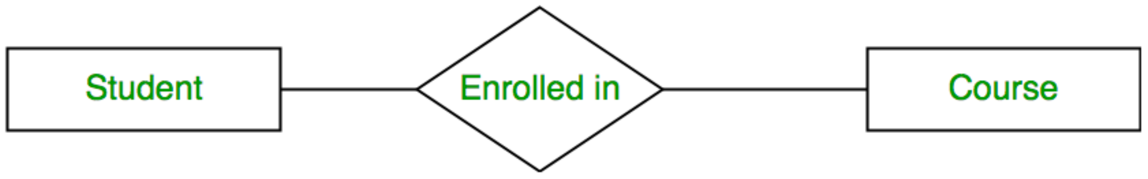
The number of different entity sets participating in a relationship set is called the degree of a relationship set.

1. Unary Relationship: When there is only ONE entity set participating in a relation, the relationship is called a unary relationship. For example, one person is married to only one person.



Unary Relationship

2. Binary Relationship: When there are TWO entities set participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.



Binary Relationship

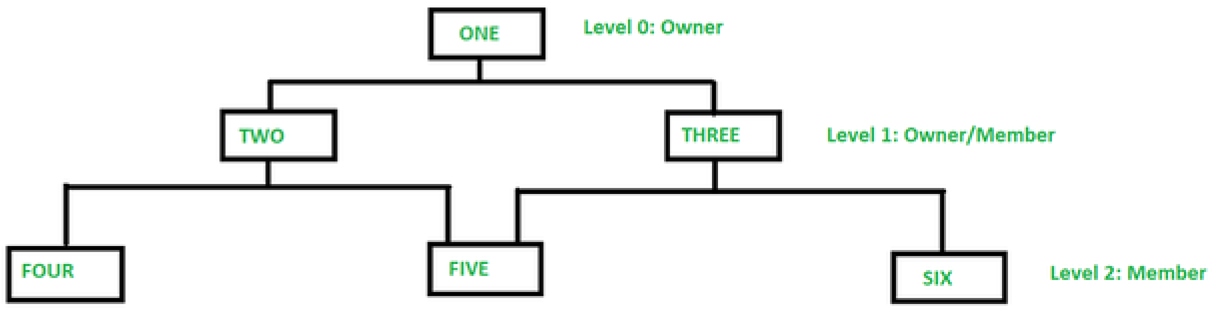
3. Ternary Relationship: When there are three entity sets participating in a relationship, the relationship is called a ternary relationship.

4. N-ary Relationship: When there are n entities set participating in a relationship, the relationship is called an n-ary relationship.

Network Model in DBMS

The Network Model in a Database Management System (DBMS) is a data model that allows the representation of many-to-many relationships in a more flexible and complex structure compared to the Hierarchical Model. It uses a graph structure consisting of nodes (entities) and edges (relationships) to organize data, enabling more efficient and direct access paths.

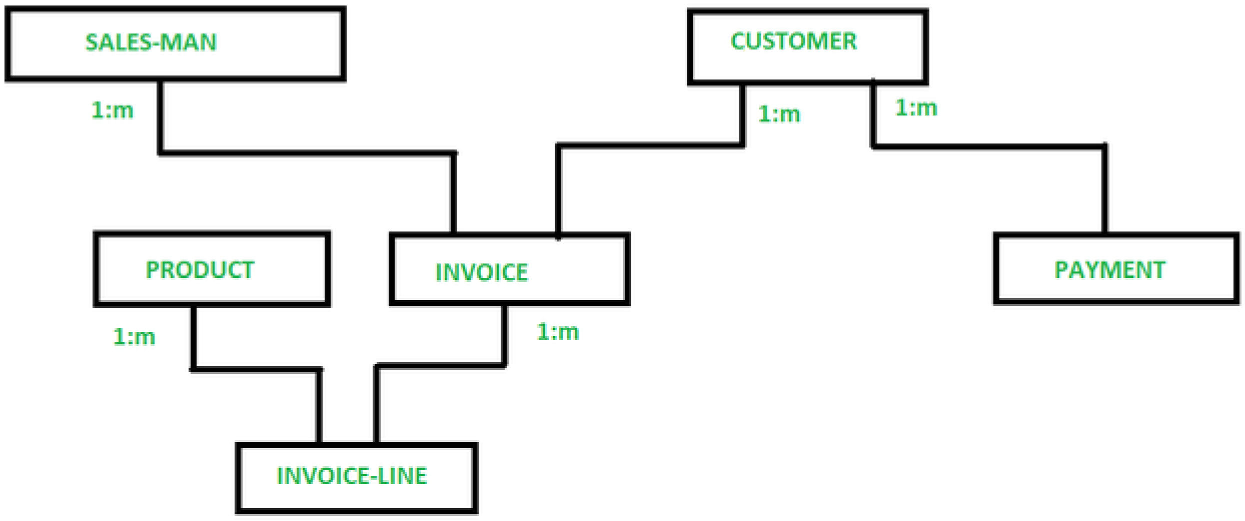
Structure of a Network Model



In the above figure, member TWO has only one owner 'ONE' whereas member FIVE has two owners i.e, TWO and THREE. Here, each link between the two record types represents 1 : M relationship between them. This model consists of both lateral and top-down connections between the nodes. Therefore, it allows 1: 1, 1 : M, M : N relationships among the given entities which helps in avoiding data redundancy problems as it supports multiple paths to the same record. There are various examples such as TOTAL by Cincom Systems Inc., EDMS by Xerox Corp., etc.

Example : Network model for a Finance Department.

Below we have designed the network model for a Finance Department:



Network model of Finance Department.

So, in a network model, a one-to-many (1: N) relationship has a link between two record types. Now, in the above figure, SALES-MAN, CUSTOMER, PRODUCT, INVOICE, PAYMENT, INVOICE-LINE are the types of records for the sales of a company. Now, as you can see in the given figure, INVOICE-LINE is owned by PRODUCT & INVOICE. INVOICE has also two owners SALES-MAN & CUSTOMER.

Relational Model in DBMS

The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations. The relational model is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.

The relational model represents how data is stored in Relational Databases. A relational database consists of a collection of tables, each of which is assigned a unique name. Consider a relation STUDENT with attributes ROLL_NO, NAME, ADDRESS, PHONE, and AGE shown in the table.

Table Student

ROLL_NO	**NAME**	**ADDRESS**	**PHONE**	**AGE**
1	RAM	DELHI	9455123451	18

ROLL_NO	NAME	ADDRESS	PHONE	AGE
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		18

Important Terminologies

- **Attribute:** Attributes are the properties that define an entity. e.g.; **ROLL_NO**, **NAME, ADDRESS**
- **Relation Schema:** A relation schema defines the structure of the relation and represents the name of the relation with its attributes. e.g.; STUDENT (ROLL_NO, NAME, ADDRESS, PHONE, and AGE) is the relation schema for STUDENT. If a schema has more than 1 relation, it is called Relational Schema.
- **Tuple:** Each row in the relation is known as a tuple. The above relation contains 4 tuples, one of which is shown as:

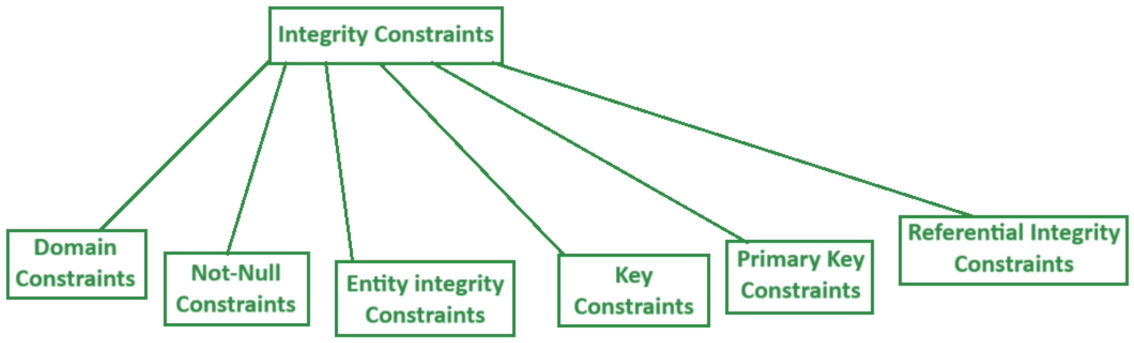
1	RAM	DELHI	9455123451	18

DBMS Integrity Constraints

Integrity constraints are the set of predefined rules that are used to maintain the quality of information. Integrity constraints ensure that the data insertion, data updating, data deleting and other processes have to be performed in such a way that the data integrity is not affected. They act as guidelines ensuring that data in the database remain accurate and consistent. So, integrity constraints are used to protect databases. The various types of integrity constraints are

Types of Integrity Constraints:

- Domain Constraints
- Not-Null Constraints
- Entity integrity Constraints
- Key Constraints
- Primary Key Constrains
- Referential integrity constraints



-

Domain Constraints

These are defined as the definition of valid set of values for an attribute. The data type of domain include string, char, time, integer, date, currency etc. The value of the attribute must be available in comparable domains.

Example:

Student_Id	**Name**	**Semester**	**Age**
21CSE100	Ramesh	5th	20
21CSE101	Kamlesh	5th	21
21CSE102	Aakash	5th	22
21CSE103	Mukesh	5th	20

Not-Null Constraints

It specifies that within a tuple, attributes overs which not-null constraint is specified must not contain any null value.

Example:

Let, the not-null constraint be specified on the "Semester" attribute in the relation/table given below, then the data entry of 4th tuple will violate this integrity constraint, because the "Semester" attribute in this tuple contains null value. To make this database instance a legal instance, its entry must not be allowed by database management system.

Student_id	**Name**	**Semester**	**Age**
21CSE100	Ramesh	5th	20
21CSE101	Kamlesh	5th	21
21CSE102	Akash	5th	22
21CSE103	Mukesh		20

Entity Integrity Constraints

Entity integrity constraints state that primary key can never contain null value because primary key is used to determine individual rows in a relation uniquely, if primary key contains null value then we cannot identify those rows. A table can contain null value in it except primary key field.

Example:

It is not allowed because it is containing primary key as NULL value.

Student_id	**Name**	**Semester**	**Age**
21CSE101	Ramesh	5th	20
21CSE102	Kamlesh	5th	21
21CSE103	Aakash	5th	22
	Mukesh	5th	20

Key Constraints

Keys are the entity set that are used to identify an entity within its entity set uniquely. An entity set can contain multiple keys, bit out of them one key will be primary key. A primary key is always unique, it does not contain any null value in table.

Example:

Student_id	Name	Semester	Age
21CSE101	Ramesh	5th	20
21CSE102	Kamlesh	5th	21
21CSE103	Aakash	5th	22
21CSE102	Mukesh	5th	20

It is now acceptable because all rows must be unique.

Primary Key Constraints

It states that the primary key attributes are required to be unique and not null. That is, primary key attributes of a relation must not have null values and primary key attributes of two tuples must never be same. This constraint is specified on database schema to the primary key attributes to ensure that no two tuples are same.

Example

Here, in the below example the Student_id is the primary key attribute. The data entry of 4th tuple violates the primary key constraint that is specifies on the database schema and therefore this instance of database is not a legal instance.

Student_id	Name	Semester	Age
21CSE101	Ramesh	5th	20
21CSE102	Kamlesh	5th	21
21CSE103	Akash	5th	22
21CSE103	Mukesh	5th	20

Referential integrity constraints

It can be specified between two tables. In case of referential integrity constraints, if a Foreign key in Table 1 refers to Primary key of Table 2 then every value of the Foreign key in Table 1 must be null or available in Table 2.

Example:

Here, in below example Block_No 22 entry is not allowed because it is not present in 2nd table.

Student_id	**Name**	**Semester**	**Block_No**
22CSE101	Ramesh	5th	20
21CSE105	Kamlesh	6th	21
22CSE102	Aakash	5th	20
23CSE106	Mukesh	2nd	22

Block_No	**Block Location**
20	Chandigarh
21	Punjab
25	Delhi