

Experiment No. 3

Aim:- To study & implement the ER model for the database.

Software: ER Diagram Online Tool , DBDiagram

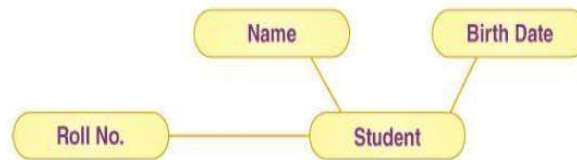
Theory:

An **Entity-Relationship (ER) Diagram** is a visual representation of a database structure that illustrates entities, their attributes, and the relationships between them. It helps in designing a structured database by clearly defining how data elements interact. ER diagrams use standard symbols like rectangles for entities, ellipses for attributes, and diamonds for relationships. They are widely used in database modeling to ensure efficient data organization and integrity.

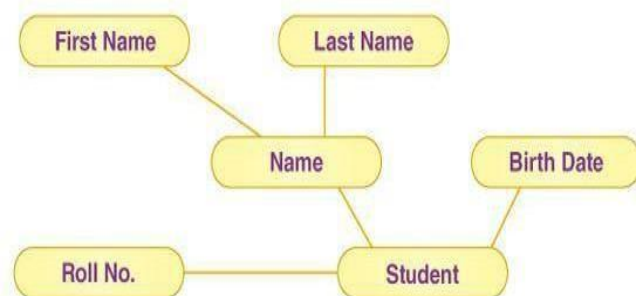
Entities: Represented as **rectangles**, these signify real-world objects (e.g., Employees, Departments, Projects).



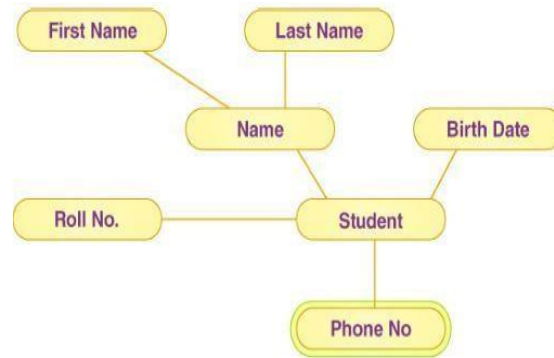
□ **Attributes:** Represented as **ellipses**, describing properties of entities (e.g., Employee Name, Salary).
□



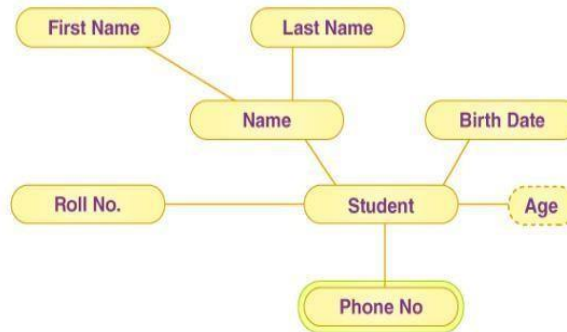
In case the attributes are composite, then these are separated further into a tree-like structure. The attribute of each node is then attached to it. To put it another way, composite attributes are basically represented by ellipses joined by an ellipse.



A double ellipse is used to represent multivalued attributes.



Dashed ellipses represent derived attributes.



Relationship

Different shaped boxes signify relationships. Inside the diamond box is written the name of the relationship. A line connects all the entities or rectangles that are involved in a relationship. For example, Tiffany works in the Biology department. It would be represented as follows:

Cardinality and Binary Relationship

The term “binary relationship” refers to a relationship in which two entities are involved. The number of instances of an entity from any relationship that can be connected or associated with the relation is known as cardinality.

One-to-One (1:1) Relationship

- Each entity in **Set A** is associated with exactly one entity in **Set B**, and vice versa.
- This type of relationship is often used when data should be split for security, performance, or logical organization.

One-to-Many (1:N) Relationship

- One entity in **Set A** can relate to multiple entities in **Set B**, but each entity in **Set B** is related to only one entity in **Set A**.

Many-to-One (N:1) Relationship

- This is essentially the inverse of a One-to-Many relationship.
- Multiple entities in **Set A** are associated with a single entity in **Set B**.

Many-to-Many (M:N) Relationship

- Multiple entities in **Set A** relate to multiple entities in **Set B**.
- This typically requires a **junction table** (or associative table) to break it down into two One-to-Many relationships.

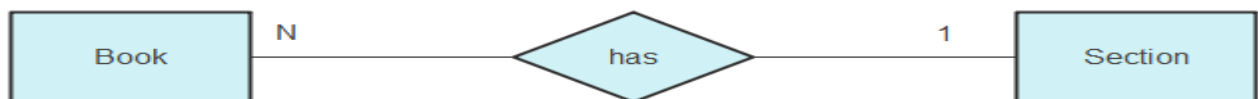
one-to-one (1:1)



one-to-many (1:N)



many-to-one (N:1)



many-to-many (M:N)



Participation Constraints

1. **Partial Participation** – □ Not all instances of an entity are required to participate in the relationship.

Notation: Represented by a **single line** between the entity and the relationship in an ER diagram.

2. **Total Participation** – □ Every instance of an entity must be associated with at least one instance of the related entity.

Notation: Represented by a **double line** between the entity and the relationship in an ER diagram.



Participation in Enrolled relationship set: **Partial** Course
Total Student

Pros and Cons of ER Diagram

Pros

Clear Representation of Data

- ER diagrams provide a visual structure of the database, making it easy to understand relationships between entities.

Helps in Database Design

- They assist developers in designing efficient and well-structured databases, reducing redundancy and ensuring normalization.

Improves Communication

- ER diagrams serve as a common language between stakeholders (developers, database administrators, and business analysts).

Cons

Complexity in Large Databases

- For large-scale systems, ER diagrams can become too complex, making them difficult to read and manage.

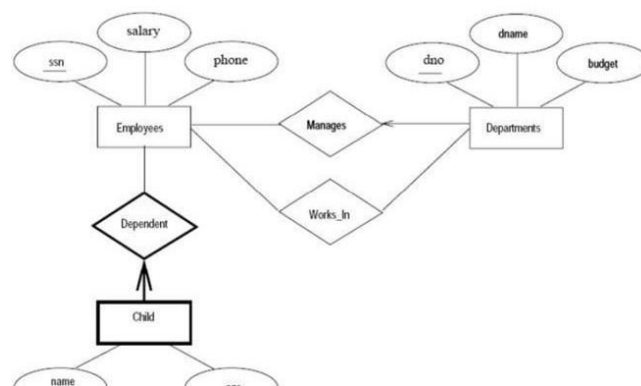
Lack of Implementation Details

- ER diagrams focus on the conceptual model and do not cover technical aspects like indexes, constraints, and performance optimizations.

No Standard Notation

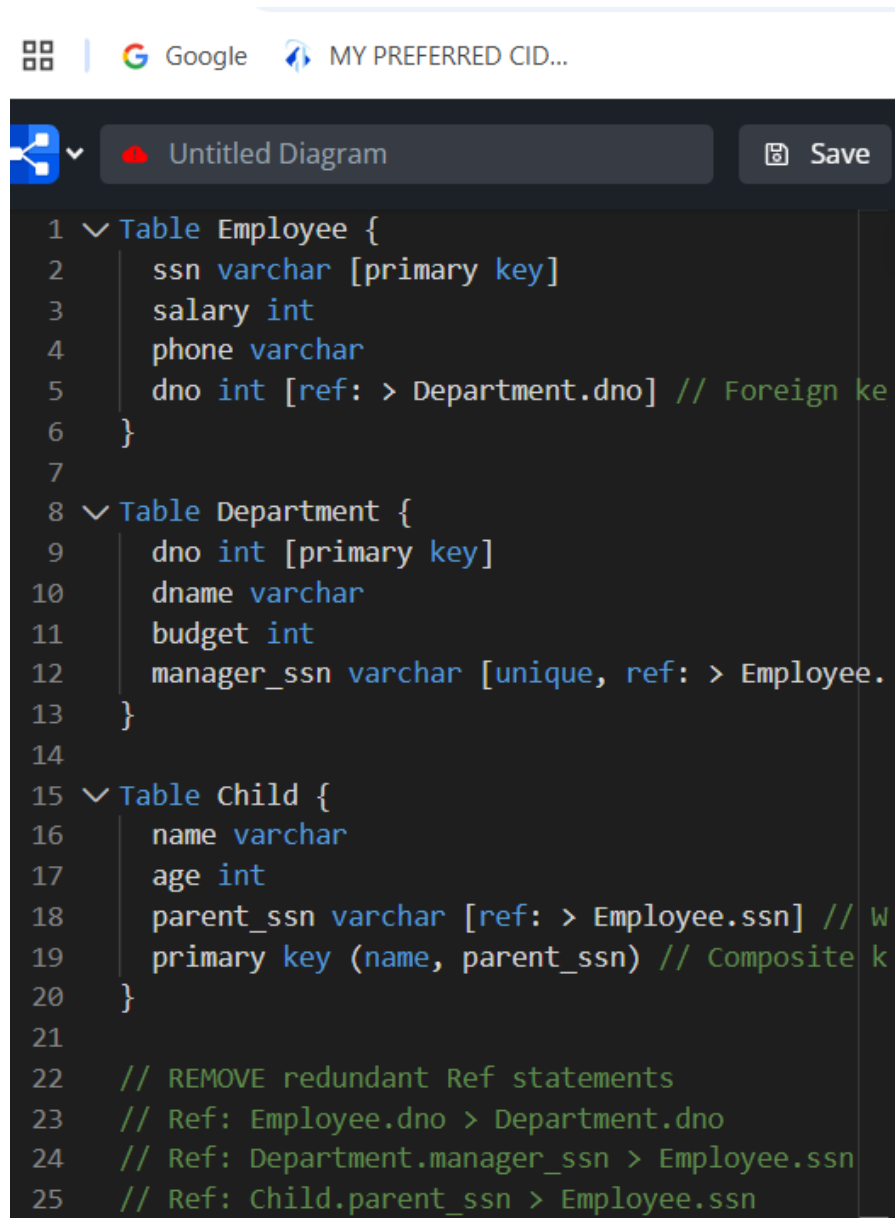
- Different notations exist (Chen, Crow's Foot, UML), leading to inconsistencies in interpretation.

ER Diagram:



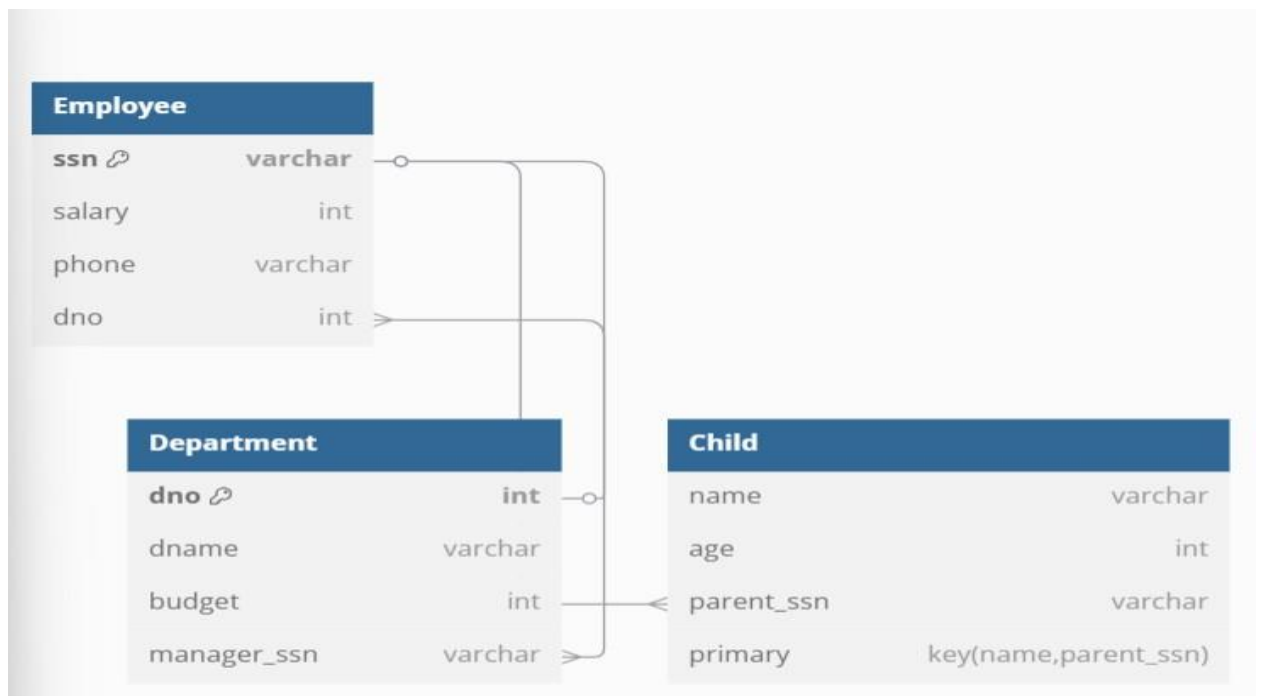
Example:

A company database needs to store information about employees (identified by ssn, with salary and phone as attributes), departments (identified by dno, with dname and budget as attributes), and children of employees (with name and age as attributes). Employees work in departments; each department is managed by an employee; a child must be identified uniquely by name when the parent (who is an employee; assume that only one parent works for the company) is known.

Code:-

```
1  Table Employee {
2    ssn varchar [primary key]
3    salary int
4    phone varchar
5    dno int [ref: > Department.dno] // Foreign key
6  }
7
8  Table Department {
9    dno int [primary key]
10   dname varchar
11   budget int
12   manager_ssn varchar [unique, ref: > Employee.ssn]
13 }
14
15 Table Child {
16   name varchar
17   age int
18   parent_ssn varchar [ref: > Employee.ssn] // Foreign key
19   primary key (name, parent_ssn) // Composite key
20 }
21
22 // REMOVE redundant Ref statements
23 // Ref: Employee.dno > Department.dno
24 // Ref: Department.manager_ssn > Employee.ssn
25 // Ref: Child.parent_ssn > Employee.ssn
```

ER Diagram:-



Conclusion:- Hence we have successfully learned to study & implement the ER model for the database.