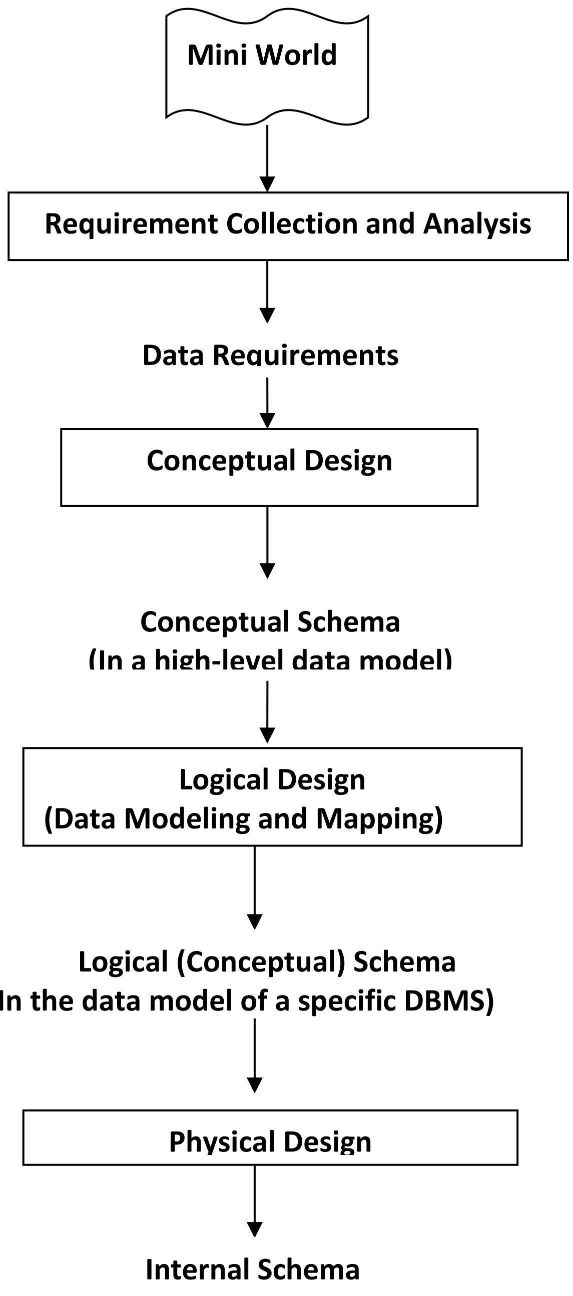
Module-2

ENTITY RELATIONSHIP MODEL

The Modeling Sequences


Mini World
Requirement Collection and Analysis
Data Requirements
Conceptual Design
Conceptual Schema
(In a high-level data model)
Logical Design
(Data Modeling and Mapping)
Logical (Conceptual) Schema
(In the data model of a specific DBMS)
Physical Design
Internal Schema

- **Entity Relationship Model or ER Model** is a high-level conceptual data model.
- It is used for conceptual design of database applications.
- This model describes the basic data structuring concepts, relationships and constraints.
- The diagrammatic notions associated with ER model is known as ER diagram.

The components of ER model are as follows.

Entities and Their Attributes:

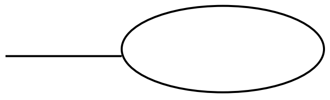
An **entity**, is a real-world *thing* or *object* in the real world with an independent existence.

An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course).

Attributes:

The properties which describe an entity is known as attributes. For example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job.

Attributes are represented by ellipses or ovals in an ER Diagram.



Types of attributes:

Simple (or Atomic) Attributes:

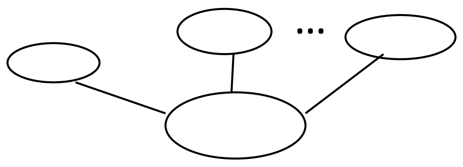
Attributes that are not divisible are called **simple** or **atomic attributes**. e.g., Aadhar Number or UID, Roll Number of a student, Employee ID, Vehicle Registration Number etc.

Composite Attributes:

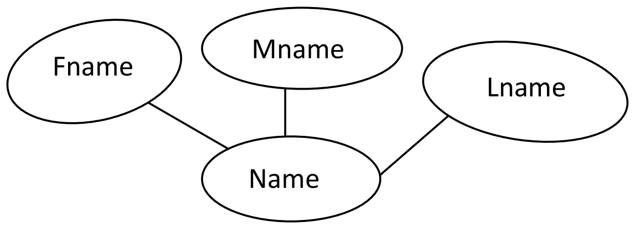
The attributes which can be divided into smaller subparts are known as **Composite attributes**

For example, the Address attribute of the EMPLOYEE entity can be subdivided into Street address, City, State, and Zip,

The name attribute can be further subdivided by First name, Middle name and Last Name.



Example:



The value of a composite attribute is the concatenation of the values of its component simple attributes.

Single-Valued Attribute:

The attributes have a single value for a particular entity are called **single-valued attributes**.

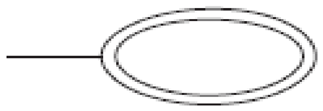
For example, Age is a single-valued attribute of an employee or a person.

Multivalued Attributes:

The attributes which have a set of values for the same entity is known as Multivalued attributes.

For example qualification of a person or an employee is a multivalued attribute.

Multivalued attributes are represented by double line ellipse or ovals

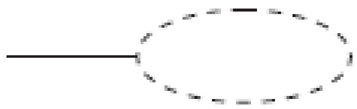


Stored attribute and Derived Attributes:

In some cases, two (or more) attribute values are related, for example, the Age and Birthdate attributes of a person. For a particular person entity, the value of Age can be determined from the current (today's) date and the value of that person's Birthdate.

The Age attribute is hence called a **derived attribute** and is said to be **derivable from** the Birth_date attribute, which is called a **stored attribute**.

The Derived attributes can be represented by dashed ellipse or dashed oval symbol



Entity Type:

An **entity type** defines a *collection* (or *set*) of entities that have the same attributes.

Each entity type in the database is described by its name and attributes.

An entity type describes the **schema** for a *set of entities* that share the same structure.

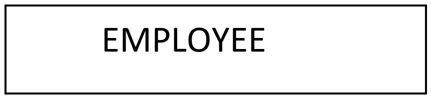
Entity Set

The collection of all entities of a particular entity type in the database at any point in time is called an **entity set .**

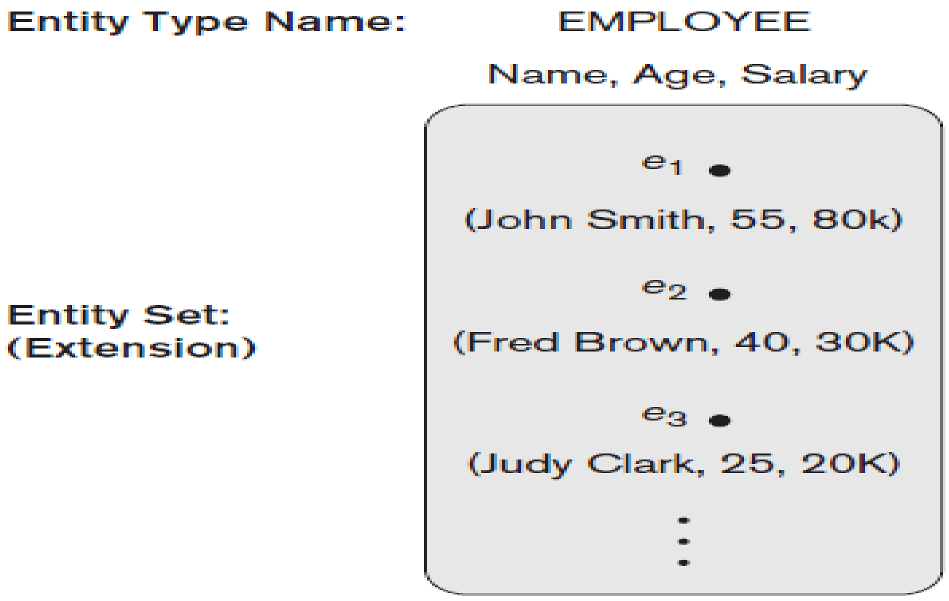
The entity set is usually referred to using the same name as the entity type.

For example, EMPLOYEE refers to both a *type of entity* as well as the entity set (i.e. the current collection *of all employee entities* in the database)

In an ER diagram an entity type can be represented by a rectangle enclosing the name of the entity type.



Note: The name of an entity is always a noun.



Key Attributes:

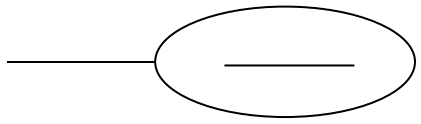
The concept of key is a constraint on the entities of an entity type. Which is also known as **uniqueness constraint** on attributes.

This constraint specifies that no two entities of a particular entity type can have the same value for key attributes.

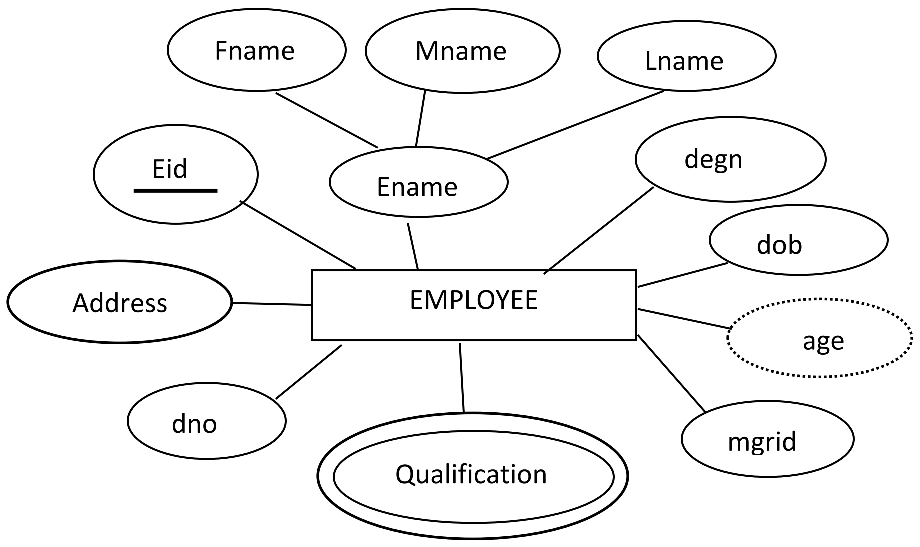
key attribute is one attribute or more than one attributes whose values can be used to identify each entity in an entity type uniquely.

For example, *Employee-id* can be a **key** for *EMPLOYEE* entity type *as no two employee entities can have the same value for their ids.*

In ER diagram each key attribute has its name underlined inside the oval.



Following is an example of representing an EMPLOYEE entity type with different attributes.



Value Set or Domain of an Attribute:

Each attribute is associated with a set of values for the entities in an entity type. These set of values is known as value set or domain of that attribute.

Relationship Type (or Relationship):

- A relationship type connects two or more entity types
- A **relationship type** among *entity* types defines a set of associations among entities from these entity types. **The set of association is known as relationship set**

For example,
consider a relationship type **WORKS_FOR** between the two entity types **EMPLOYEE** and **DEPARTMENT**. This **WORKS_FOR** relationship type associates each employee with the department for which the employee works.

Formal Definition of Relation Type, Relationship Set and Relationship instances:

A **relationship type** *R* among *n* entity types $E_1, E_2, \ldots, E_n$ defines a set of associations or a **relationship set** among entities from these entity types.

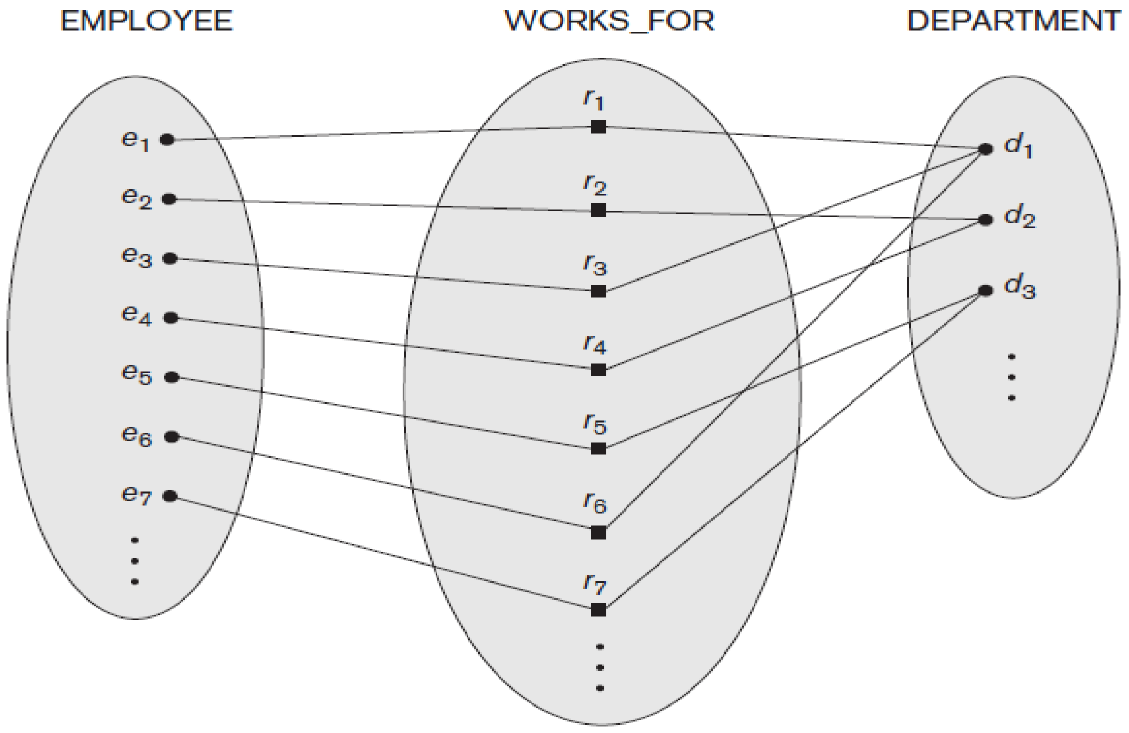
A relationship type and its corresponding relationship set are generally referred to by the *same name*, *R*.

Mathematically, the relationship set *R* is a set of **relationship instances** r_i, where each r_i associates *n* individual entities $(e_1, e_2, \ldots, e_n)$, and each entity e_j in r_i is a member of entity set E_j, $1 \leq j \leq n$.

Hence, a relationship set is a mathematical relation on $E_1, E_2, \ldots, E_n$;

i.e., $R \subseteq E_1 \times E_2 \times \ldots \times E_n$.

Each of the entity types $E_1, E_2, \ldots, E_n$ is said to **participate** in the relationship type R; similarly, each of the individual entities $e_1, e_2, \ldots, e_n$ is said to **participate** in the relationship instance $r_i = (e_1, e_2, \ldots, e_n)$.

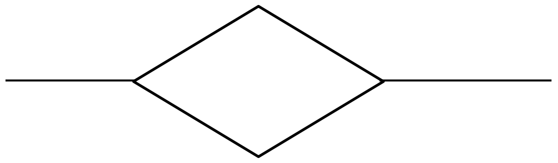


The above figure shows Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

In the above figure, each relationship instance r_i is shown connected to the EMPLOYEE and DEPARTMENT entities that participate in r_i. The employees e_1, e_3, and e_6 work for department d_1 the employees e_2 and

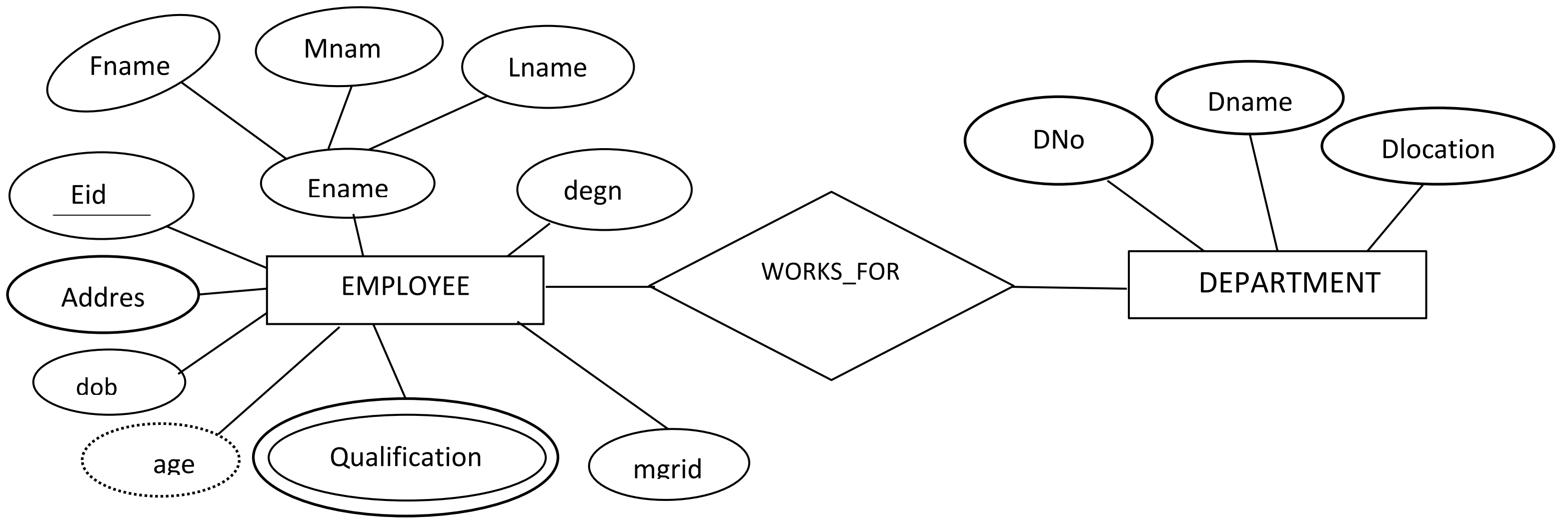
e_4 work for department d_2 and the employees e_5 and e_7 work for department d_3.

In ER diagrams, relationship types are displayed as diamond-shaped boxes, which are connected by straight lines to the rectangular boxes representing the participating entity types.



The relationship name is always a verb which is written inside the diamond shaped box.

The following is the diagrammatic notations of association of ***EMPLOYEE*** entity type with ***DEPARTMENT*** entity type through the relationship type ***WORKS-FOR***



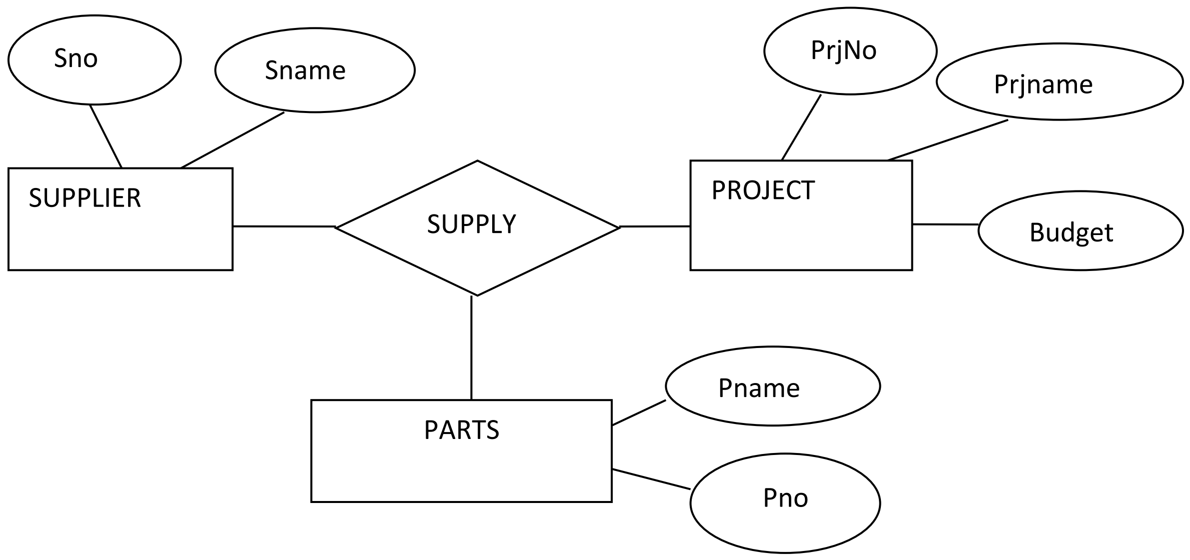
Degree of a Relationship Type

The number of participating entity types in a relationship type is known as the **degree** of a relationship type.

A relationship type of **degree two is** called **binary**, and one of **degree three** is called **ternary**.

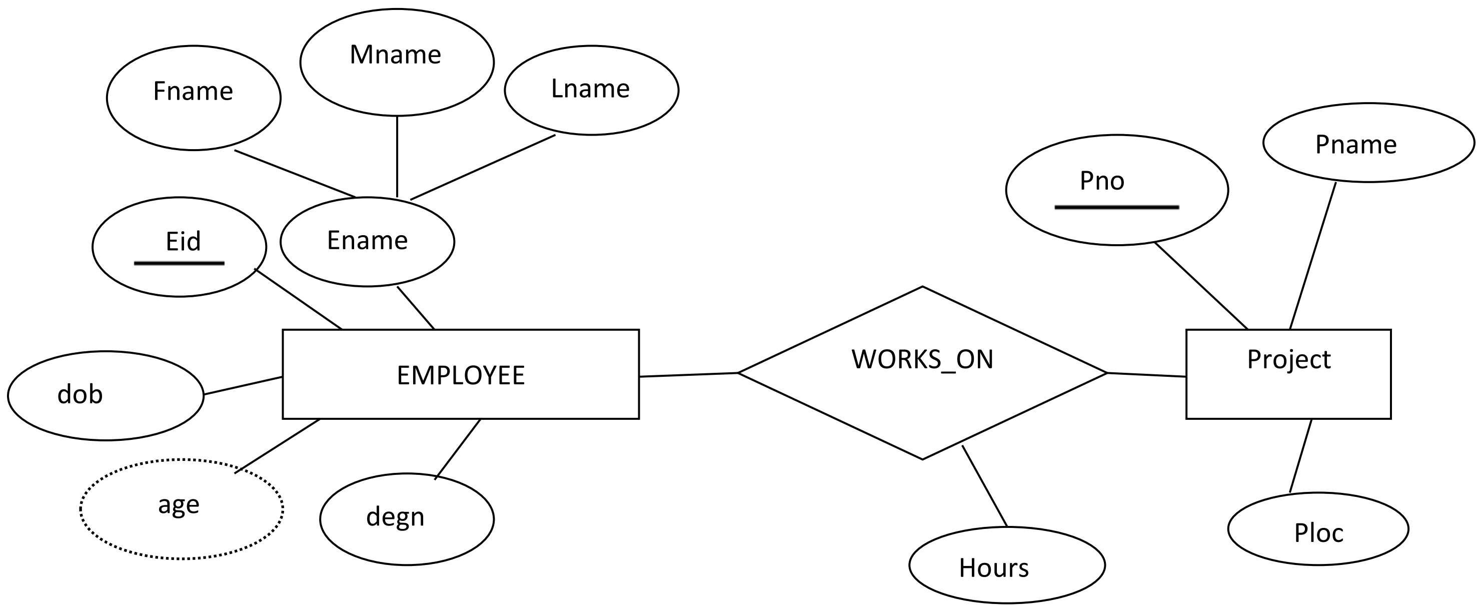
e.g., the **WORKS_FOR** relationship is of **degree two** or a **binary relationship** type.

e.g. the following is the example of a **ternary relationship**.



Attributes of Relationship Type:

- It is useful to attach attributes to relationship types.
- Such an attribute is a property of the participating entity types in the relationship types or the participating entities in the relationship set.

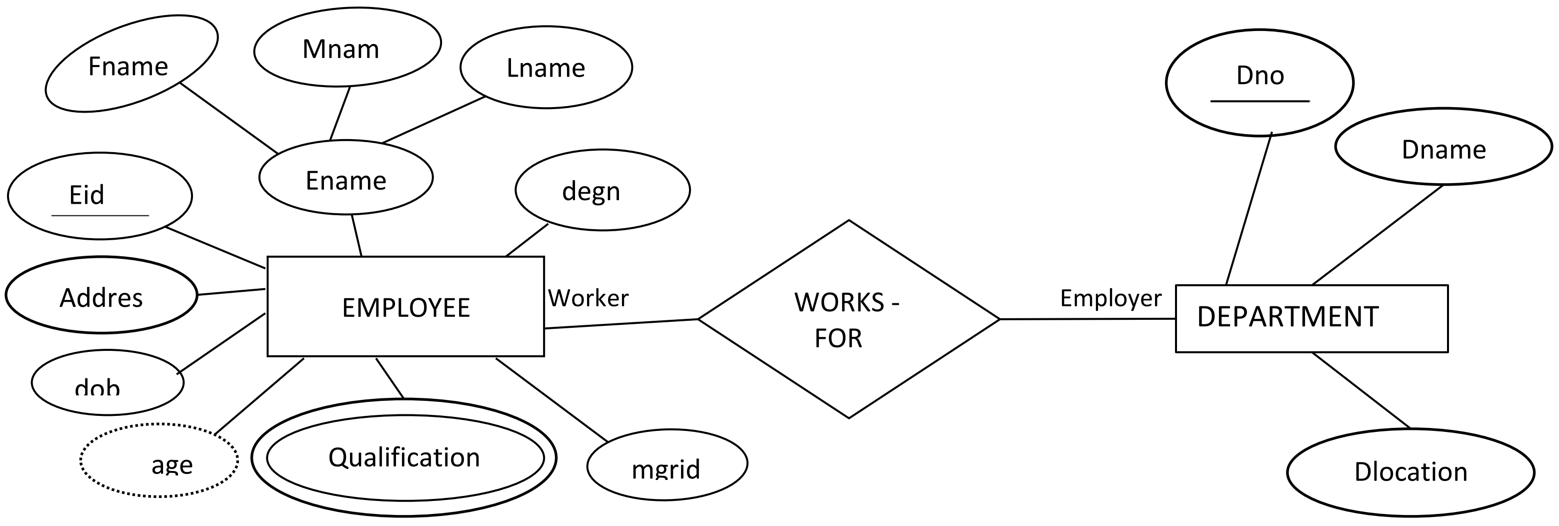


Here, Hour is a property of both EMPLOYEE and PROJECT, not of one alone.so, it is represented as attribute of the relationship type WORKS_ON.

Role names:

- Each entity type that participates in a relationship type plays a particular role in the relationship.
- The **role name** signifies the role that a participating entity from the entity type plays in each relationship instance, and it helps to explain what the relationship means.

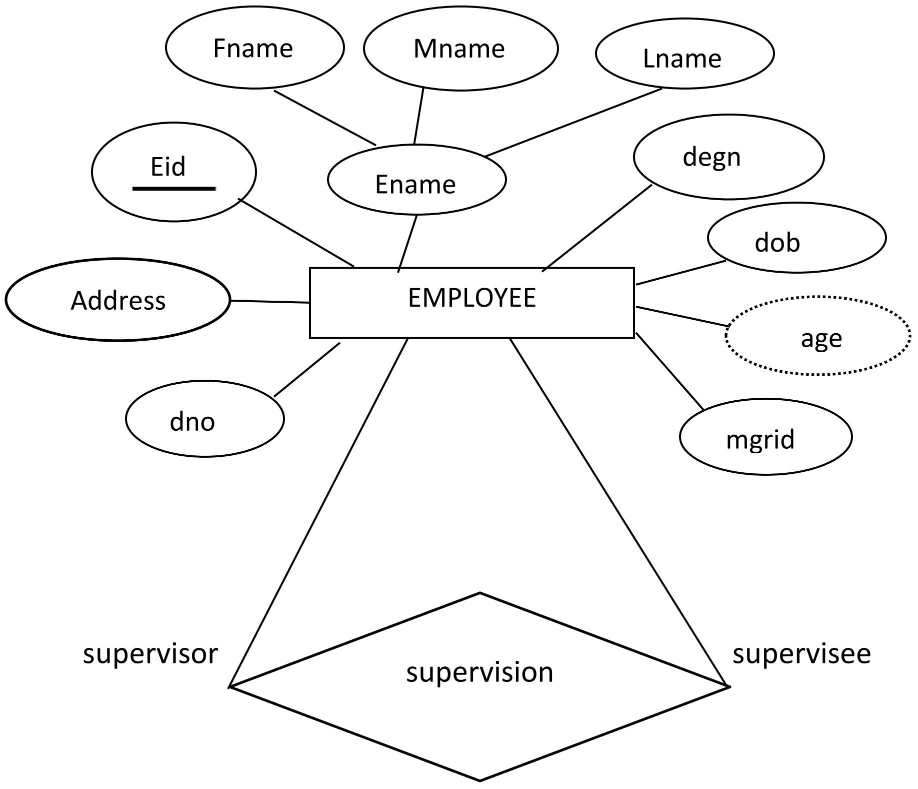
For example, in the WORKS_FOR relationship type, EMPLOYEE plays the role of *employee* or *worker* and DEPARTMENT plays the role of *department* or *employer*.



NOTE: ***Role names are optional in binary relationship types.***

Recursive Relationship:

The relationship type in which ***same* entity type participates more than once in *different roles*** is known as ***Recursive relationship*** or ***self-referencing relationship.***



The **SUPERVISION** relationship type is a **recursive relationship** that relates *an employee to a supervisor (or manager*), **where both *employee* and *supervisor* entities are members of the same EMPLOYEE entity set**.

NOTE: ***Role names are optional in binary relationships but it is mandatory in Recursive relationships i.e., role names must be specified in recursive relationships in ER Diagram.***

Constraints on Relationship Types

Relationship types usually have certain constraints that restricts the possible combinations of entities that may participate in the corresponding relationship set.

These constraints are

- Cardinality ratio constraints
- Participation constraints.

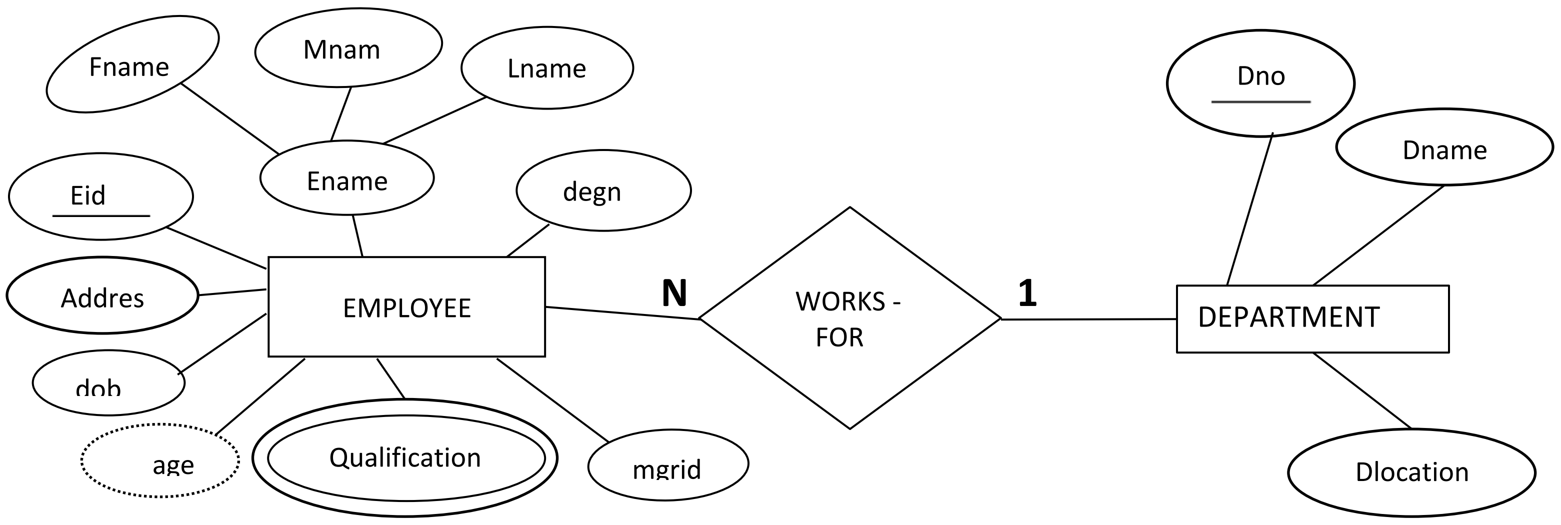
*cardinality ratio and Participation constraints are also called as **Structural Constraints** of a relationship type*

Cardinality ratio constraints:

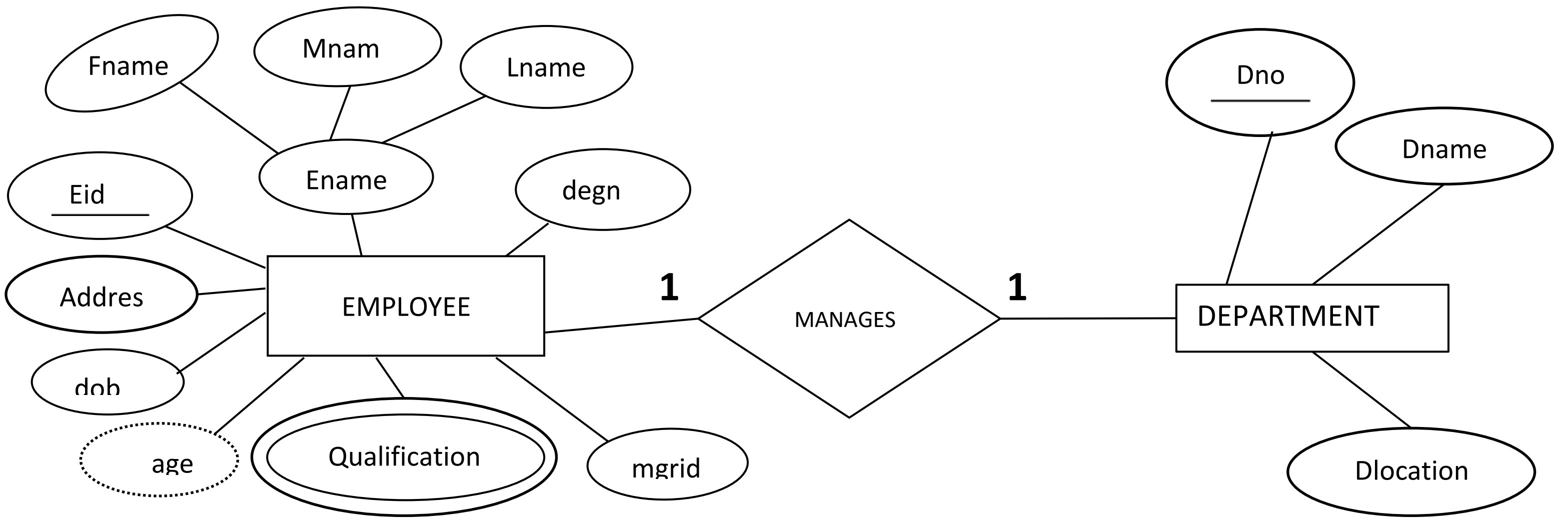
The **cardinality ratio** for a binary relationship specifies the *maximum* number of relationship instances that an entity can participate in.

The possible cardinality ratios for binary relationships are **1:1, 1: N, N:1, M: N**

For example, in the WORKS_FOR binary relationship type, DEPARTMENT:EMPLOYEE is of cardinality ratio 1:N, meaning that each department can be related to any number of employees (N),but an employee can be related to or work for at most one department (1).



An example of a 1:1 binary relationship is MANAGES ,which relates a department entity to the employee who manages that department. This represents constraints that—at any point in time—an employee can manage at most one department and a department can have at most one manager

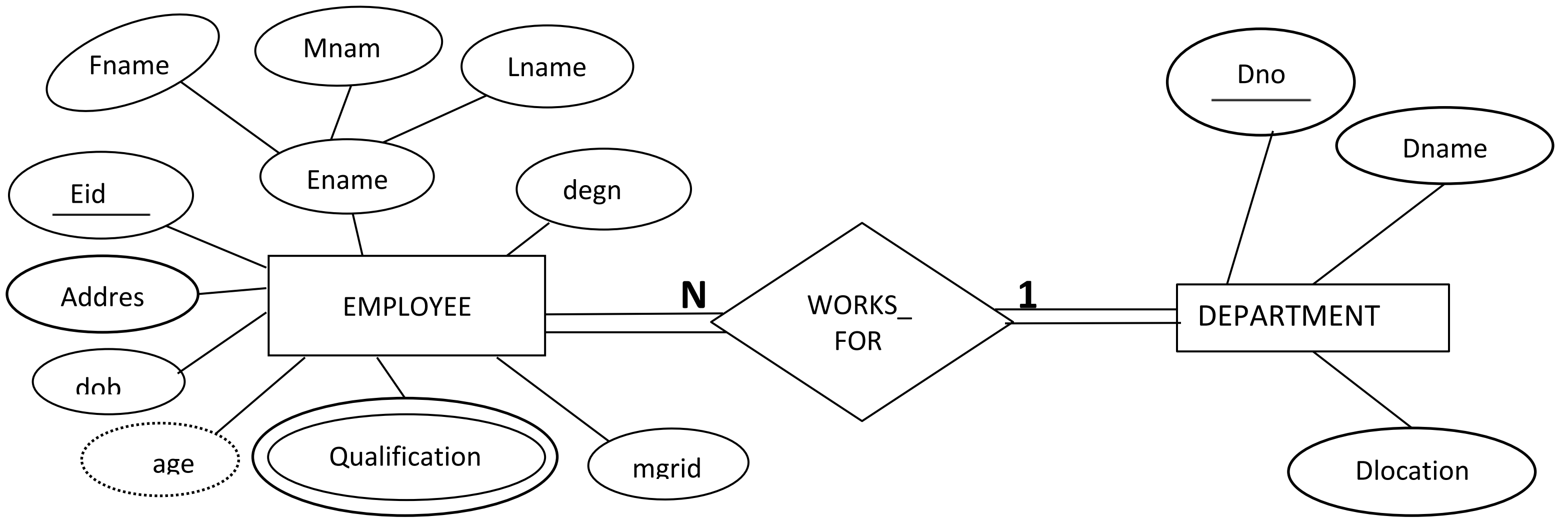


Participation Constraints:

- The **participation constraint** specifies whether the existence of an entity depends on its being related to another entity via the relationship type.
- This constraint specifies the *minimum* number of relationship instances that each entity can participate in and is sometimes called the **minimum cardinality constraint**.
- There are two types of participation constraints.
  - Total Participation Constraint
  - Parial Participation Constraint

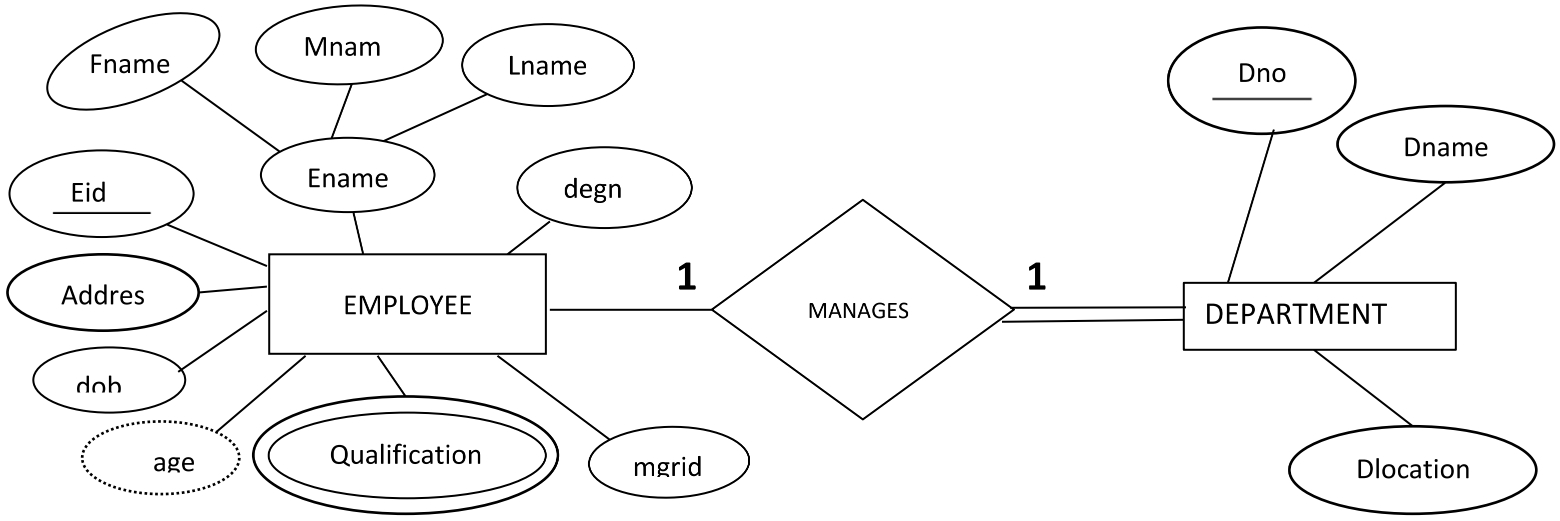
Total Participation Constraint:

- It specifies that every entities of an entity type must be related to other entities via a relationship type.
- Total participation is also called **existence dependency**.
- For example, If a company policy states that *every* employee must work for a department, then an employee entity can exist only if it participates in at least one WORKS_FOR relationship instance. Thus, the participation of EMPLOYEE in WORKS_FOR is called **total participation**
- In ER diagrams, total participation (or existence dependency) is displayed as a *double line* connecting the participating entity type to the relationship.



Parial Participation Constraint:

- It specifies that some of the entities of an entity type are related to some other entities via a relationship type, but not necessarily all.
- In ER diagrams, paertial participation is displayed as a single *line* connecting the participating entity type to the relationship.
- For example, it is not expected that every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is **partial**, meaning that *some* or *part of the set of* employee entities are related to some department entity via MANAGES, but not necessarily all. At the same time, the DEPARTMENT entity type has **total participation** in the MANAGES relationship.



Weak Entity Type:

Entity types that do not have key attributes of their own are called **weak entity types.**

The entity types that do have a key attribute are called **Regular entity types** or **strong entity types.** e.g. *EMPLOYEE, DEPARTMENT* etc.

Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values.

We call this other entity type the **identifying** or **owner entity type**.

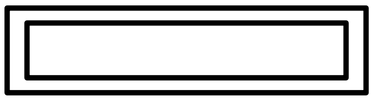
The relationship type that relates a weak entity type to its owner entity type is known as the **identifying relationship** of the weak entity type.

A weak entity type always has a ***total participation constraint*** (existence dependency) with respect to its identifying relationship.

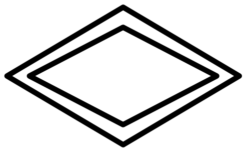
A weak entity type normally has a **partial key**, which is the attribute that can uniquely identify weak entities that are *related to the owner entity.*

ER diagram notation for Weak Entity Type:

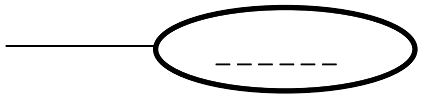
In ER diagrams, a **weak entity type** is represented by **double line rectangle**



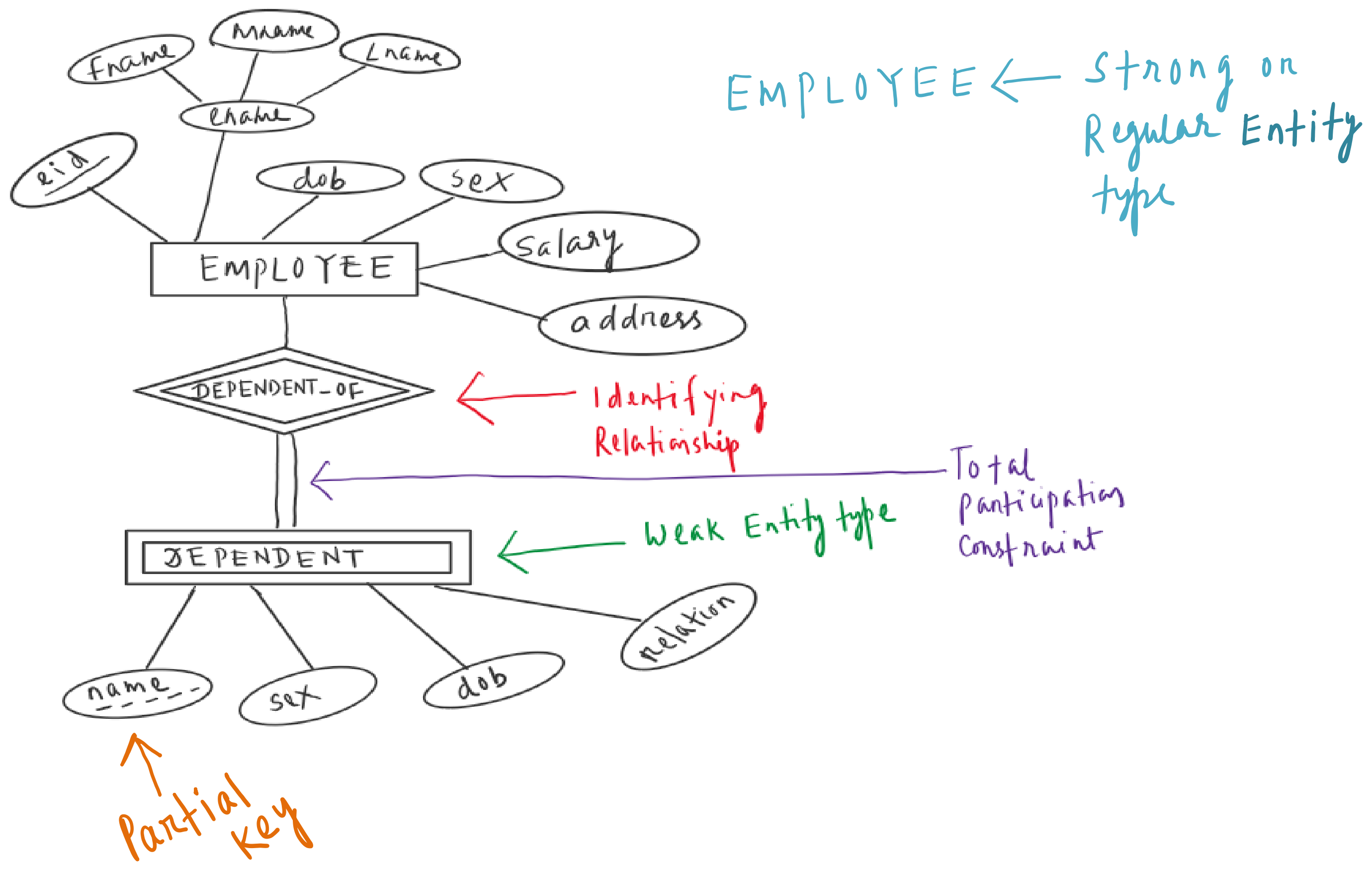
The **identifying relationship** is represented by **diamonds with double lines**



The partial key attribute is underlined with a dashed or dotted line.



Given below is the example of regular entity type, weak entity type and identifying relationship



ER diagram example:

Consider the following aspects about a company database.

The COMPANY database keeps track of a company's employees, departments, and projects. Suppose that after the requirements collection and analysis phase, the database designers provide the following description that will be represented in the database.

The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. A department may have several locations. A department controls a number of projects, each of which has a unique name, a unique number, and a single location.

The database will store each employee's name, Social Security number or id, address, salary, sex (gender), and birth date. An employee works for a department, but may work on several projects, which are not necessarily controlled by the same department.

It is required to keep track of the direct supervisor of each employee (who is another employee).

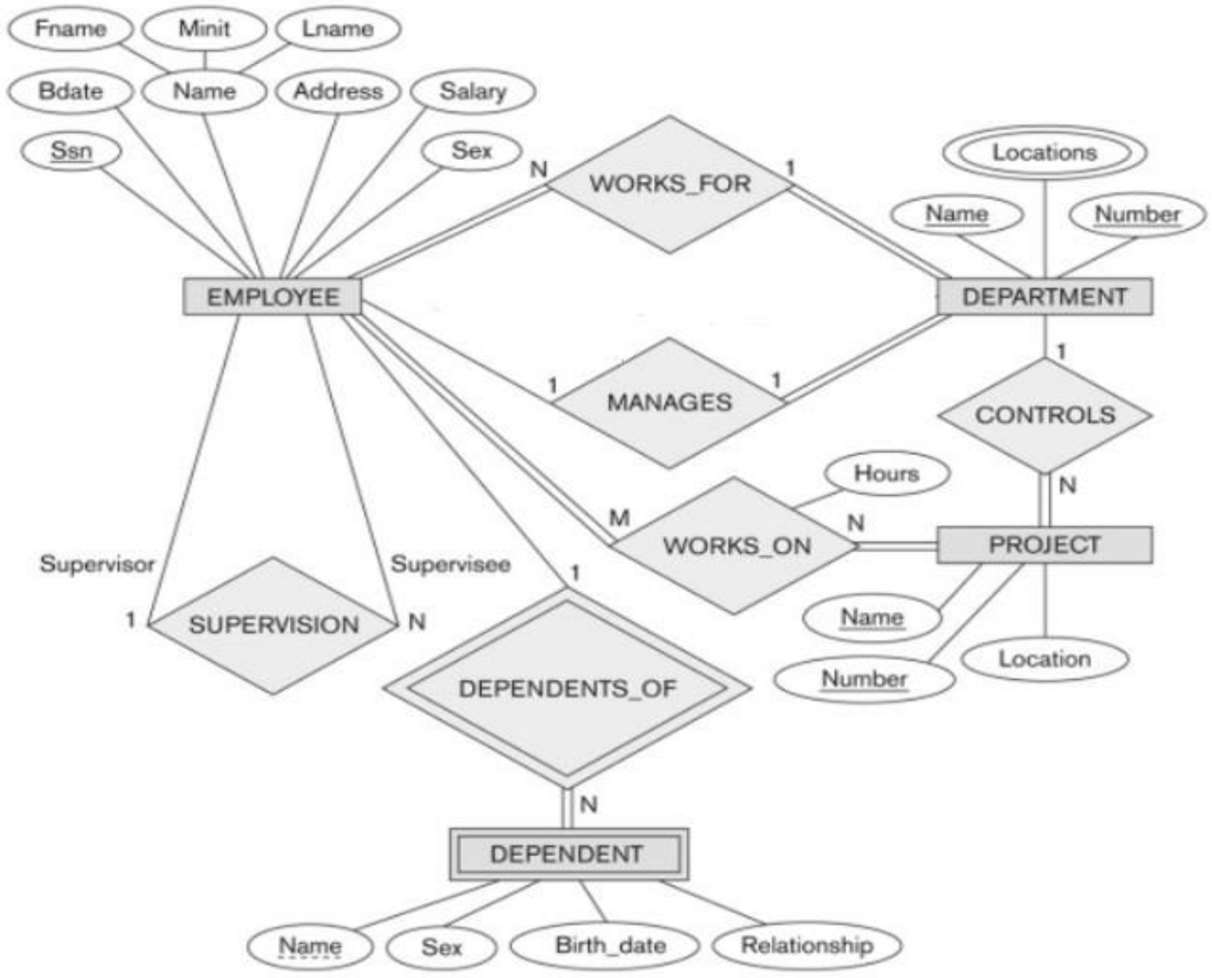
The database will keep track of the dependents of each employee for insurance purposes, including each dependent's first name, sex, birth date, and relationship to the employee.

Base on the above information, design an ER diagram for the **Company Database.**

Explanations:

The following colours are used for Entity types, attributes and relationships in the above example.

🟩	Entity Types
🟠	Attributes
🔶	Relationships



(ER Diagram of a company Database)