Lecture 3.1

Entity Relationship Modeling

Dr. Vandana Kushwaha

Department of Computer Science
Institute of Science, BHU, Varanasi

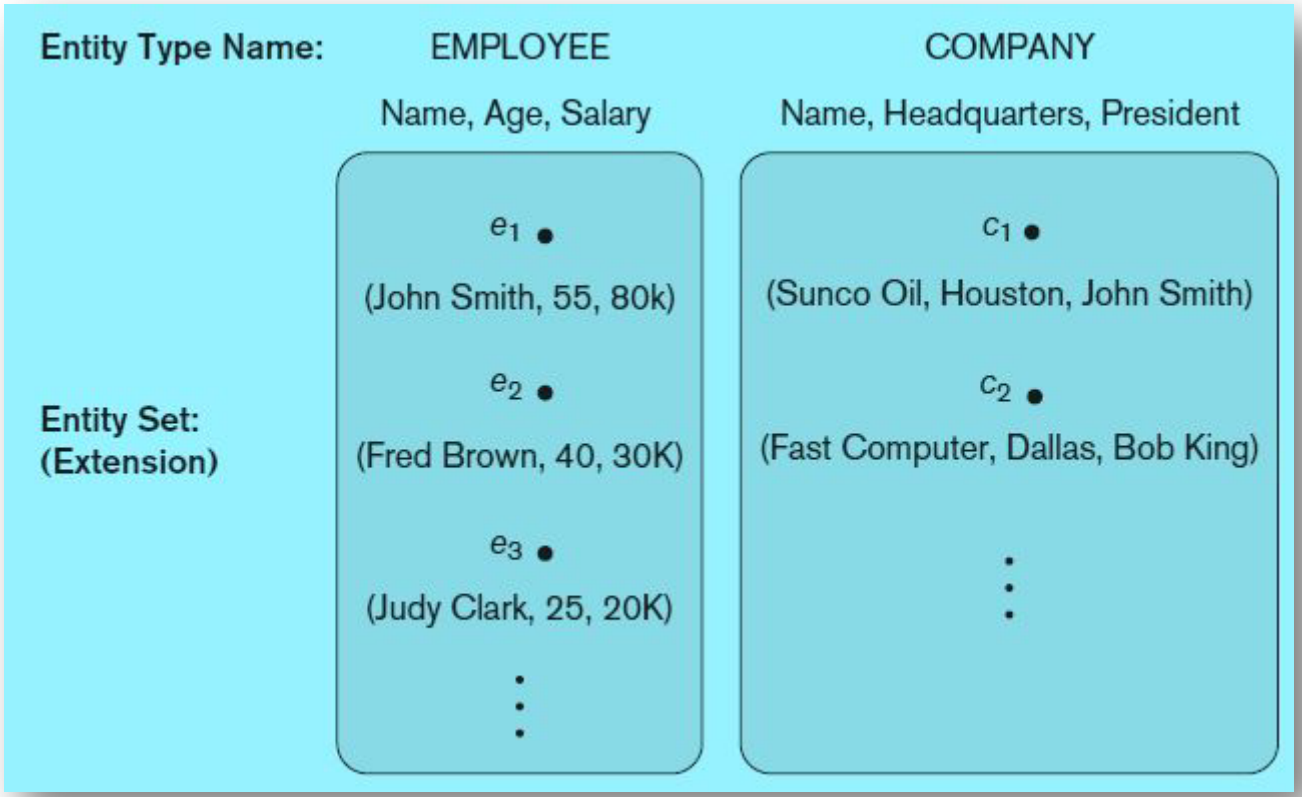
Introduction

- **Conceptual modeling** is a very important phase in designing a successful database application.
- The **ER(Entity-Relationship) model** describes data as ***entities***, ***relationships***, and ***attributes*****.**
- An **entity** is a "**thing" or "object**" in the **real world** that is distinguishable from all other objects.
- For **example**, each **person** in a **university** is an **entity.**
- An **entity** has a **set of properties**, and the **values** for some set of properties may uniquely identify an entity.
- For instance, a **person** may have a ***person id*** property whose value **uniquely identifies** that person.
- An **entity** may be **concrete**, such as a **person** or a **book,** or it may be **abstract,** such as a **course**, a **project**, or a **flight reservation.**

Entity Type and Entity Set

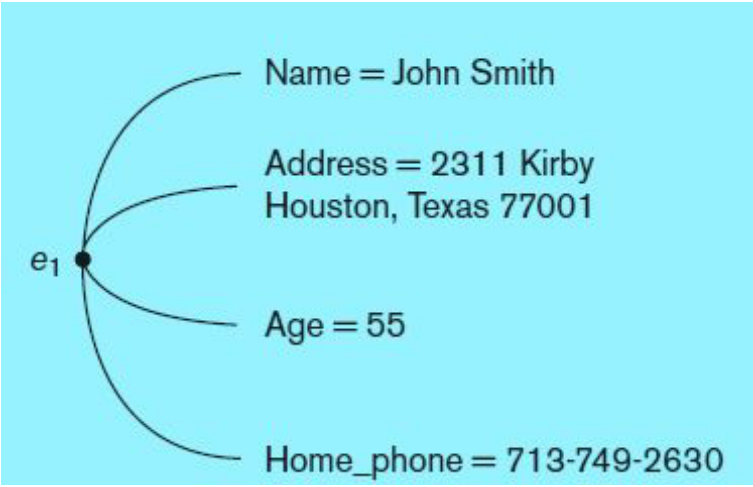
- An **entity type** defines a ***collection* (or *set*) of entities** that have the **same attributes.**
- Each **entity type** in the **database** is described by its **name** and **attributes**.
- **Example:** two entity types are **EMPLOYEE** and **COMPANY.**
- The **collection** of **all entities** 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 current set *of all employee entities* in the database.
- An **entity type** is represented in **ER diagrams** as a **rectangular box** enclosing the entity type name.

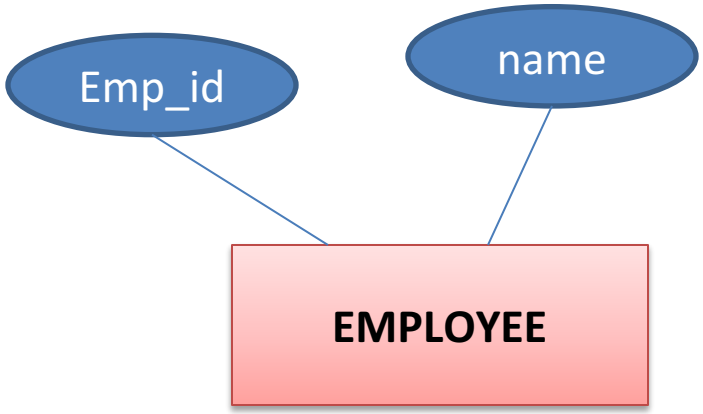
Entity Type and Entity Set



Attributes

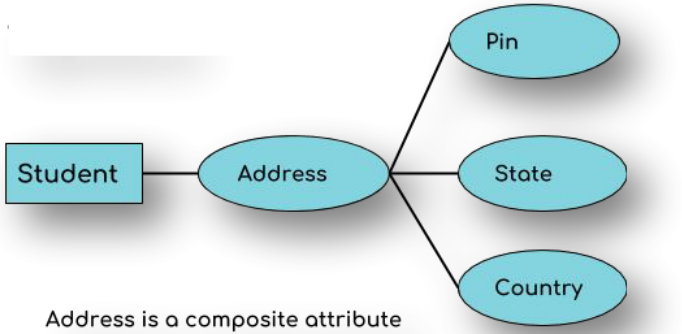
- An **entity** is represented by a set of **attributes**.
- **Attributes** are **descriptive properties** possessed by each member of an entity set.
- For **example**, an **EMPLOYEE entity** may be described by the **employee_id, employee name, address, age** and **phone_no.**
- Each **entity** has a **value** for each of its **attributes.**
- In **ER diagram attribute names** are enclosed in **ovals** and are attached to their entity type by straight lines.





Types of Attributes

- **Simple(Atomic) Attribute**
- Attributes that are **not divisible** are called simple or atomic attributes.
- **Example: age** attribute of entity employee.
- **Composite Attribute**
- **Composite attributes** can be **divided into smaller subparts**, which represent more basic attributes with independent meanings.
- For **example,** the Address attribute of the EMPLOYEE entity can be subdivided into Street_address, City, State, and Zip.
- Composite attributes are attached to their component attributes by straight lines.



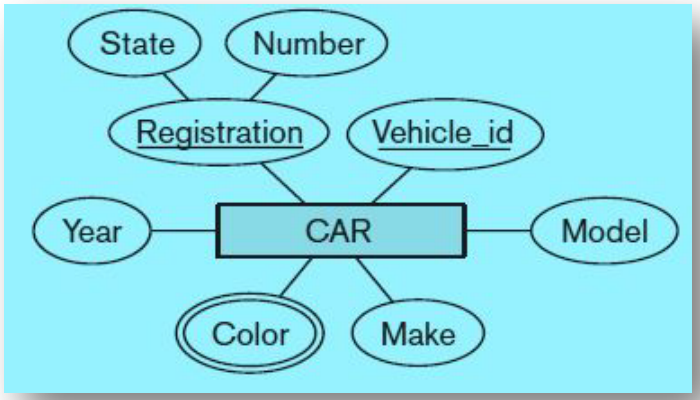
Address is a composite attribute

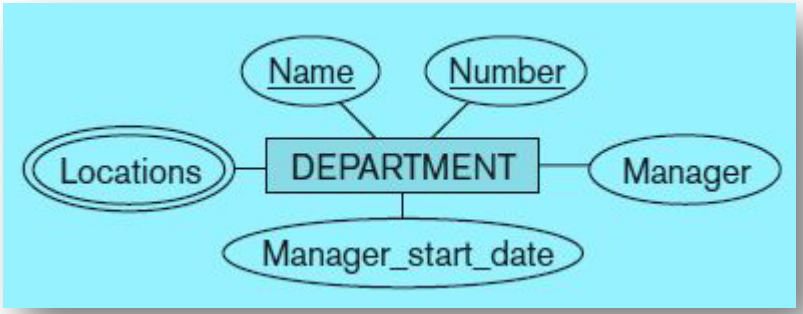
Types of Attributes

- **Single-valued attributes**
- Attributes have a **single value** for a particular entity; such attributes are called **single-valued**.
- For **example**, Age is a single-valued attribute of a person.
- **Multivalued Attributes**
- An attribute can have a **set of values** for the same entity called **multivalued**.
- **Example:** a Colors attribute for a car, or a College_degrees attribute for a person.
- Cars with one color have a single value, whereas two-tone cars have two color values.
- Similarly, one person may not have a college degree, another person may have one, and a third person may have two or more degrees.
- Therefore, different people can have different *numbers* of *values* for the **College_degrees attribute**.

Types of Attributes

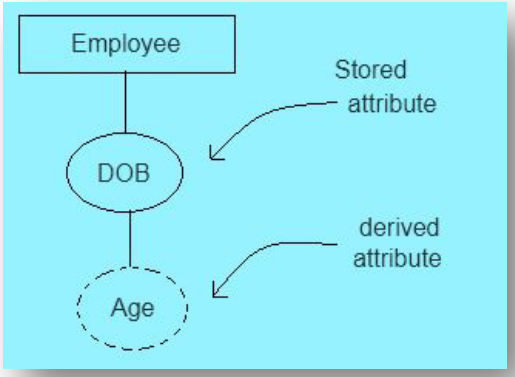
- **Multivalued attribute** may have lower and upper bounds to constrain the *number of values* allowed for each individual entity.
- For **example,** the Colors attribute of a car may be restricted to have between one and three values, if we assume that a car can have three colors at most.
- Multivalued attributes are displayed in **double ovals**.





Types of Attributes

- **Stored versus Derived Attributes**
- In some cases, two (or more) attribute values are related—for example, the Age and Birth_date 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 Birth_date.**
- 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**.
- **Derived Attributes** are represented as **dashed oval shape**.



Types of Attributes

- **NULL Values**
- **Case 1: NULL as *not applicable***
- In some cases, a particular entity **may not** have an **applicable value** for an **attribute.**
- For **example**, the **Apartment_number** attribute of an **address** applies only to addresses that are in apartment buildings and not to other types of residences, such as single-family homes.
- Similarly, a **College_degrees** attribute applies only to people with college degrees.
- For such situations, a special value called **NULL** is created.
- An **address** of a single-family home would have **NULL** for its **Apartment_number attribute**, and a **person** with **no college degree** would have **NULL** for **College_degrees attribute.**

Types of Attributes

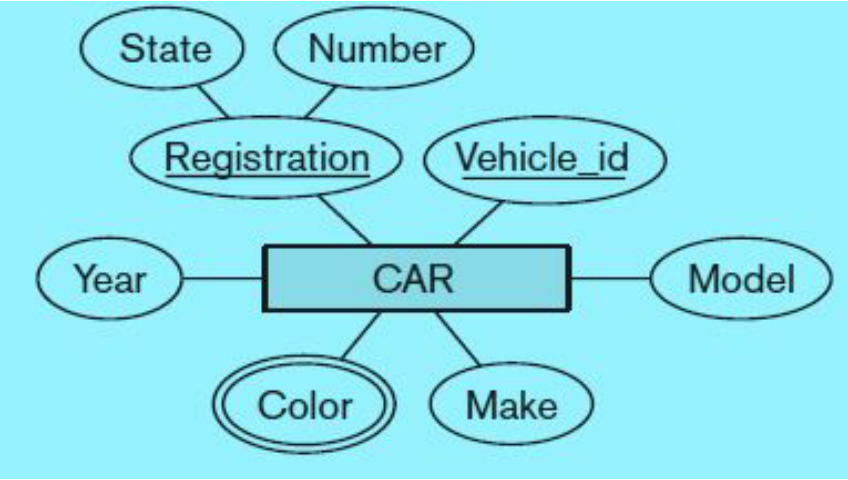
- **Case 2: NULL as *unknown***
- **NULL** can also be used if we **do not know** the **value** of an **attribute** for a **particular entity.**
- For **example**, if we do not know the **home phone number** of 'John Smith'.
- The ***unknown* category** of **NULL** can be further classified into **two cases**.
  - The **first case** arises when it is **known** that the **attribute value exists** but is ***missing***—for instance, if the **Height attribute** of a **person** is listed as **NULL.**
  - The **second case** arises when it is ***not known*** whether the attribute value exists—for example, if the **Home_phone attribute** of a person is **NULL.**

Key Attributes of an Entity Type

- An important **constraint** on the **entities** of an **entity type** is the **key** or **uniqueness constraint** on **attributes.**
- An **entity type** usually has **one or more attributes** whose values are **distinct** for each individual entity in the entity set.
- Such an **attribute** is called a **key attribute**, and its values can be used to **identify** each entity **uniquely.**
- **Example**, the **Name attribute** is a **key** of the **COMPANY entity** type because no two companies are allowed to have the same name.
- For the **PERSON** entity type, a typical key attribute is **Ssn** (Social Security number).
- In **ER diagrammatic notation**, each **key attribute** has its name **underlined** inside the **oval.**

Key Attributes of an Entity Type

- Sometimes **several attributes** together form a **key,** meaning that the ***combination of the attribute*** values must be distinct for each entity referred as **composite key .**
- The **Registration attribute** is an example of a **composite key** formed from **two simple component attributes**, **State** and **Number**, neither of which is a **key** on its own.



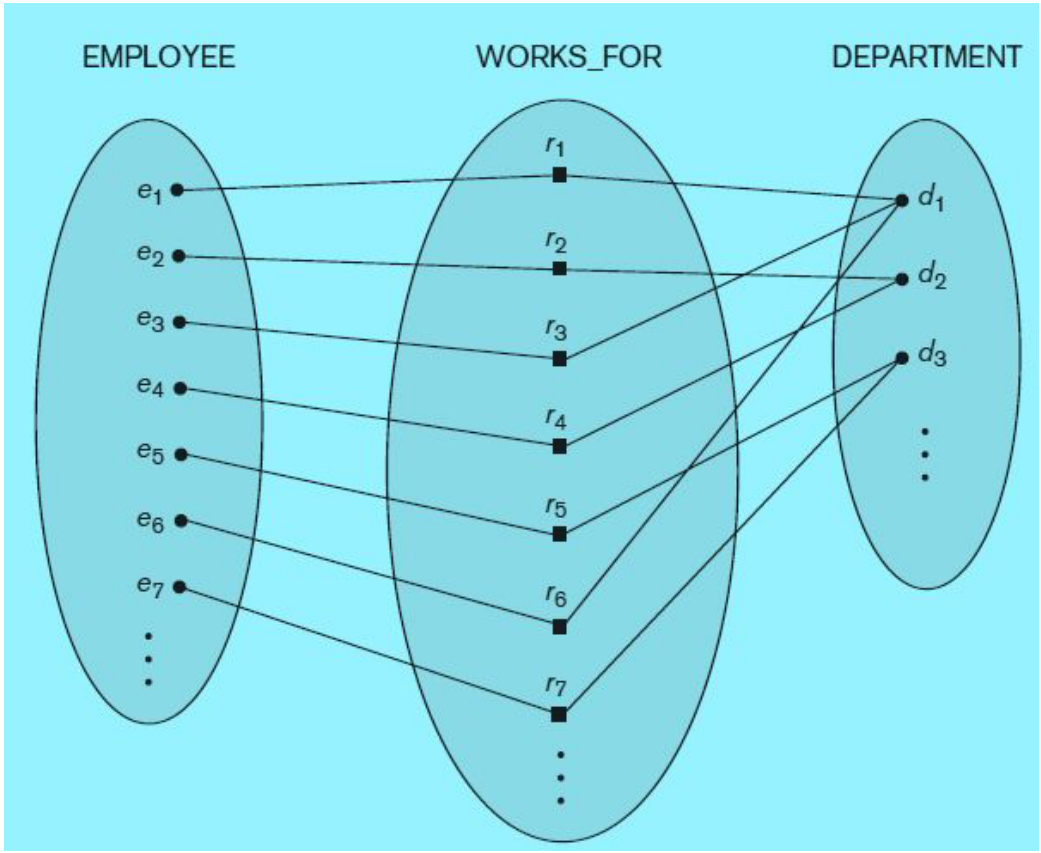
Key Attributes of an Entity Type

- Some entity types have ***more than one*** **key** attribute.
- For example, each of the **Vehicle_id** and **Registration** attributes of the entity type **CAR** , is a **key** in its own right.
- An **entity type** may also have ***no key***, in which case it is called a ***weak entity type.***

Relationship Set

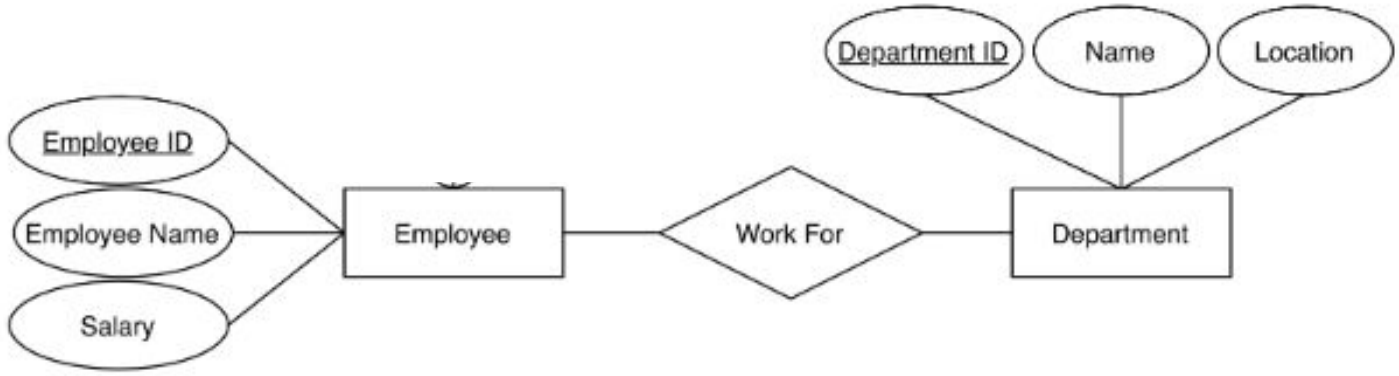
- A **Relationship** is an **association** among **several entities**.
- A **relationship set R** is a **set** of **relationships** of the **same type.**
- Mathematically **R** is a **set of relationship** instances **ri**, where each **ri** associates **n entities (e1,e2,……..en).**
- Each **relationship instance *ri*** in ***R*** is an **association of entities**, where the association includes exactly one entity from each participating entity type.
- **Example**
- Consider a **relationship type WORKS_FOR** between the **two entity types EMPLOYEE** and **DEPARTMENT.**
- Which associates each employee with the department for which the employee works in the corresponding entity set.
- Each **relationship instance** in the **relationship set WORKS_FOR** associates **one EMPLOYEE entity** and **one DEPARTMENT entity.**

Relationship Set



Relationship Set

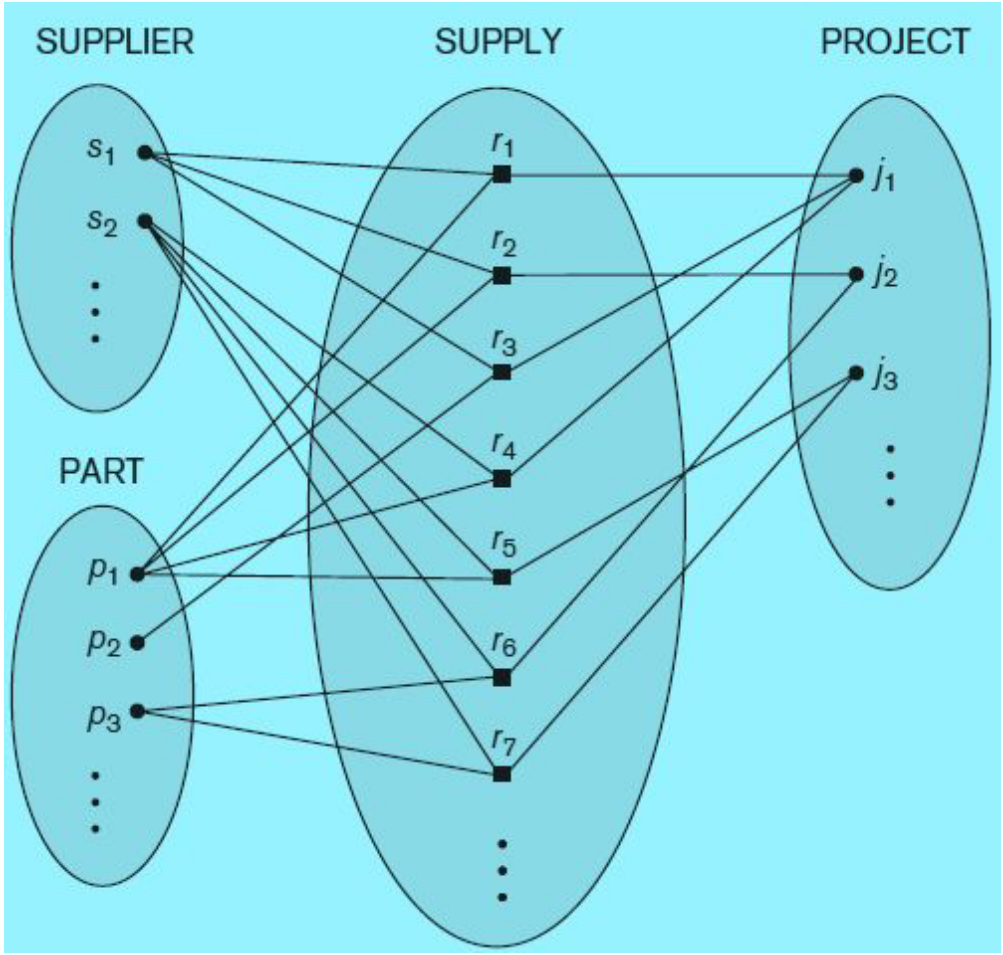
- In **ER diagrams**, **Relationship types** are displayed as **Diamond-shaped boxes**, which are connected by **straight lines** to the **rectangular boxes** representing the participating entity types.
- The **relationship name** is displayed in the **diamond-shaped box.**



Degree of a Relationship Type

- The **degree** of a relationship type is the **number of participating entity types**.
- Hence, the **WORKS_FOR relationship** is of **degree two.**
- A **relationship type** of **degree two** is called **binary**, and one of degree three is called **ternary**.
- An **example** of a **ternary relationship** is SUPPLY, shown in Figure(next slide).
- Where each relationship instance *ri* associates **three entities**—a **supplier** *s*, a **part** *p*, and **a project** *j*—whenever *s* supplies part *p* to project *j*.
- **Relationships** can generally be of **any degree**, but the ones **most common** are **binary relationships**.

Degree of a Relationship Type

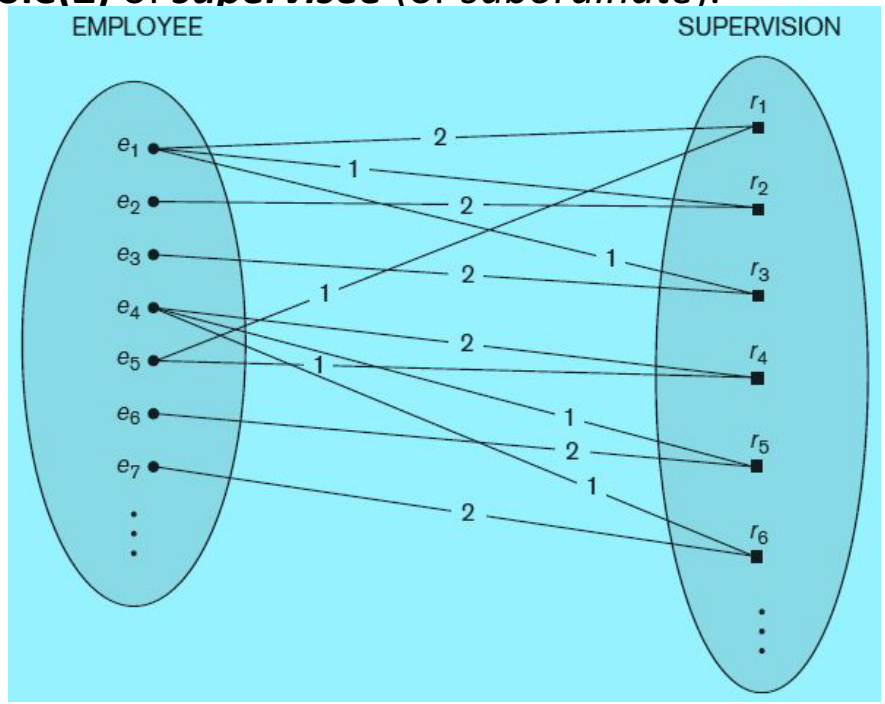


Role Names and Recursive Relationships

- Each **entity type** that **participates i**n 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 helps to explain what the relationship means.
- **Example,** in the **WORKS_FOR relationship type**, **EMPLOYEE** plays the **role of *employee*** or ***worker*** and **DEPARTMENT plays** the **role** of ***department* or *employer*.**
- **Role names** are **not technically necessary** in relationship types where all the participating entity types are distinct.
- However, in some cases the ***same* entity type** participates **more than once** in a relationship type in ***different roles*.**
- In such cases the **role name becomes essential** for distinguishing the meaning of the role that each participating entity plays.
- Such relationship types are called **Recursive relationships**.

Recursive Relationships

- The **SUPERVISION relationship** type relates an **employee** to a **supervisor**, where both **employee** and **supervisor** entities are **members** of the same **EMPLOYEE entity set**.
- Hence, the **EMPLOYEE entity** type ***participates twice*** in **SUPERVISION:**
  - once in the **role(1)** of ***supervisor*** (or *boss*), and
  - once in the **role(2)** of ***supervisee*** (or *subordinate*).



Constraints Relationship Types

- **Relationship types** usually have certain **constraints** that limit the possible combinations of entities that may participate in the corresponding relationship set.
- There are mainly **two types of constraints:**

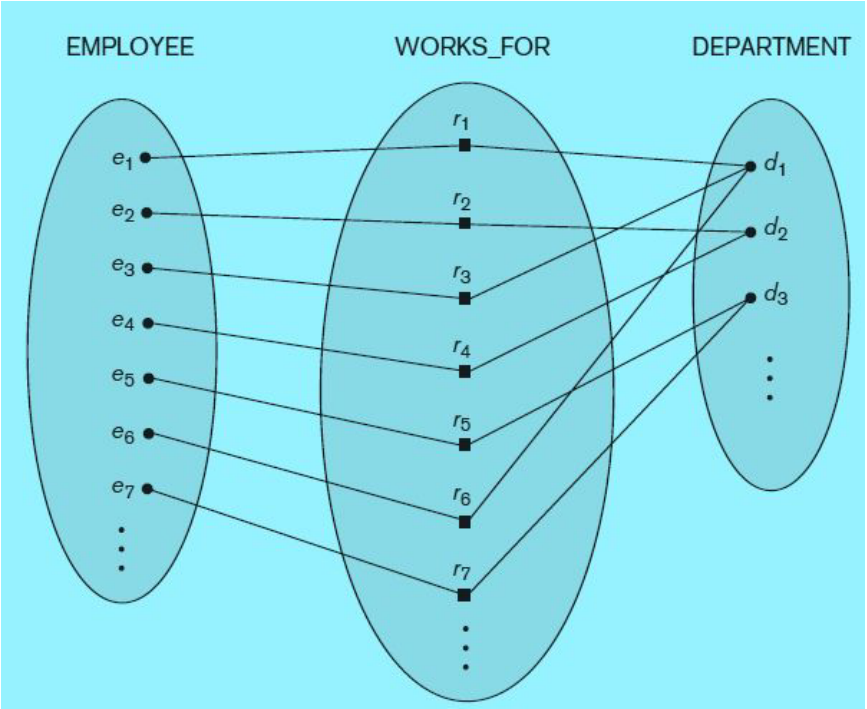
  1. **Cardinality Ratios**
  2. **Participation Constraints**

1. Cardinality Ratios

- 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 relationship types are 1:1, 1:N, N:1, and M:N.

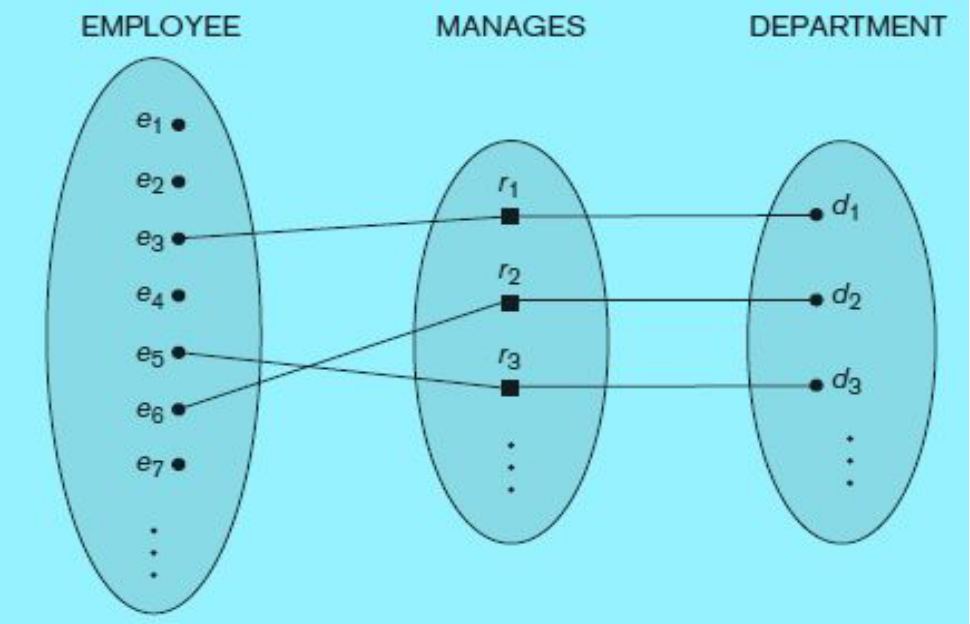
Cardinality Ratios

- **Cardinality ratio 1:N**
- In **WORKS_FOR** binary relationship type, **DEPARTMENT:EMPLOYEE** is of **cardinality ratio 1:N.**
- Meaning that each department can be related to (that is, employs) any number of employees, but an employee can be related to (work for) only one department.



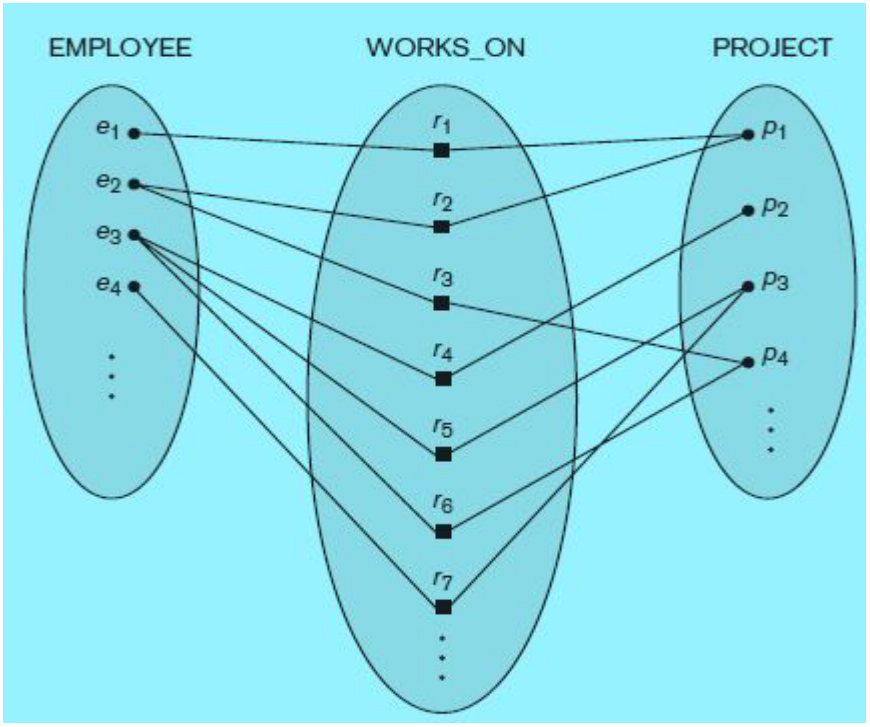
Cardinality Ratios

- **Cardinality ratio 1:1**
- An example of a **1:1 binary relationship** is **MANAGES** , which relates a department entity to the employee who manages that department.
- At any point in time—an employee can manage one department only and a department can have one manager only.



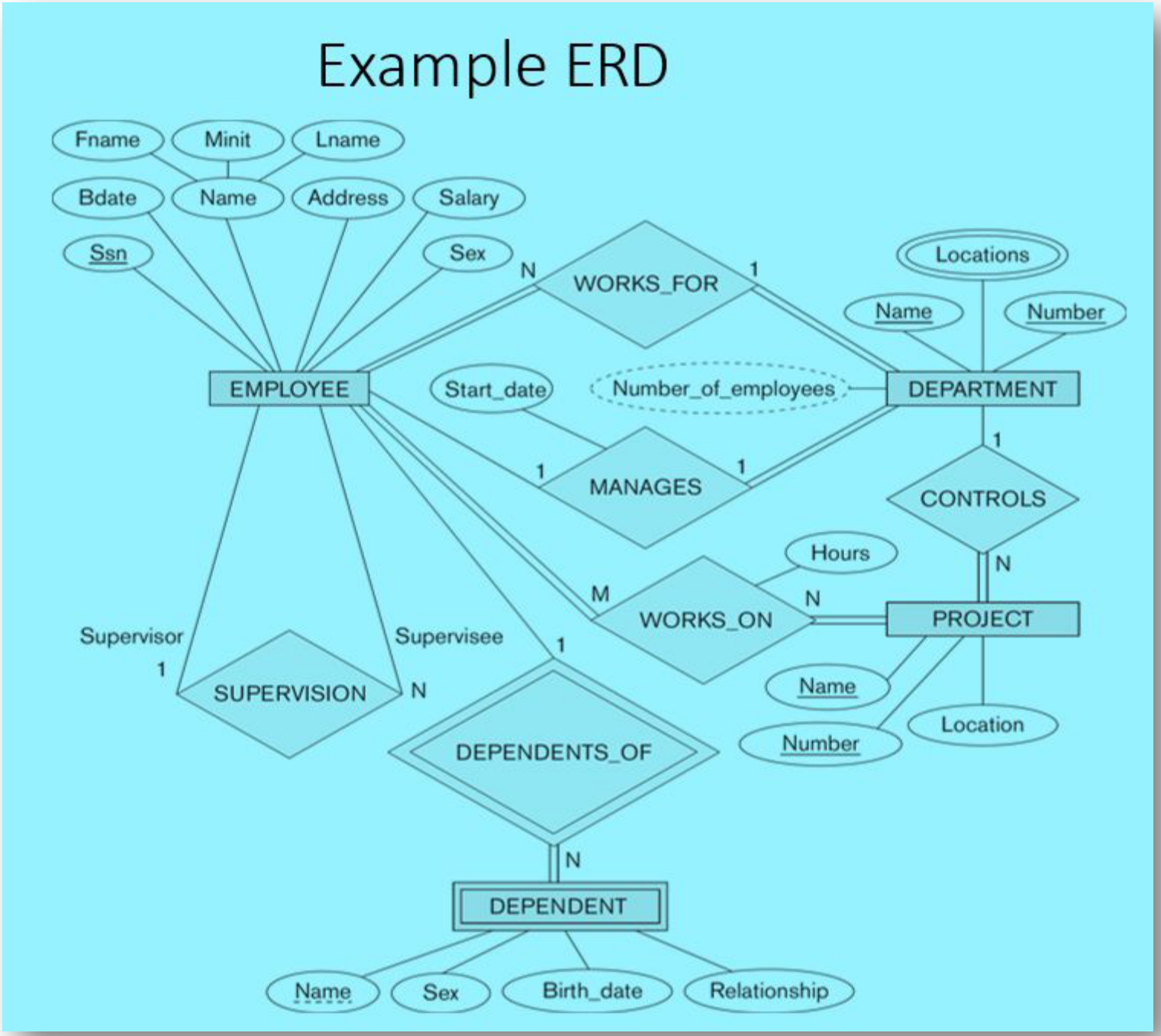
Cardinality Ratios

- **Cardinality ratio M:N**
- The relationship type **WORKS_ON** is of **cardinality ratio M:N**, because an employee can work on several projects and a project can have several employees.



- **Cardinality ratios** for binary relationships are represented on **ER diagrams** by **displaying 1, M, and N on the diamonds**.

Example ERD


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
WORKS_ON
N
PROJECT
Supervisor
1
SUPERVISION
N
Supervisee
1
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Participation Constraints

- The **participation constraint(structural constraints)** specifies the *minimum* number of relationship instances that each entity can participate in.
- There are two types of participation constraints—**total** and **partial.**
- **Total participation constraint**
- The participation of an entity set *E* in a relationship set *R* is said to be **total** if every entity in *E* participates in at least one relationship in *R*.
- 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**.
- Total participation is also called **existence dependency**.

Participation Constraints

- **Partial participation constraint**
- If only some entities in ***E*** participate in relationships in ***R***, the participation of entity set *E* in relationship *R* is said to be **partial**.
- **Example:**
- In **Company database** we do not expect 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.
- In **ER diagrams**, **total participation** (or existence dependency) is displayed as a ***double line*** connecting the participating entity type to the relationship,.
- Whereas **partial participation** is represented by a ***single line.***

Attributes of Relationship Types

- **Relationship types** can also have **attributes**, similar to those of entity types.
- **Example 1:** to record the **number of hours per week** that an **employee works on** a particular project, we can include an **attribute Hours** for the **WORKS_ON relationship type.**
- **Example 2:** to include the date on which a manager started managing a department via an attribute **Start_date** for the **MANAGES** relationship type.
- **Attribute Migration**
- **Attributes of 1:1 relationship** types can be migrated to one of the participating entity types.
- For example, the Start_date attribute for the MANAGES relationship can be an attribute of either EMPLOYEE or DEPARTMENT.

Attributes of Relationship Types

- For a **1:N relationship type**, a relationship attribute **can be migrated *only* to the entity type on the N-side** of the relationship.
- For example, if the WORKS_FOR relationship also has an attribute Start_date that indicates when an employee started working for a department, this attribute can be included as an attribute of EMPLOYEE.
- For **M:N relationship types**, some attributes may be determined by the *combination of participating entities* in a relationship instance, not by any single entity.
- Such **attributes *must be specified as relationship attributes***.
- An **example** is the **Hours attribute** of the **M:N relationship WORKS_ON**; the number of hours per week an employee currently works on a project is determined by an employee project combination and not separately by either entity.

Weak Entity Types

- Entity types that do not have key attributes of their own are called **weak entity types**.
- In contrast, **regular entity types** that do have a key attribute—which include all the examples discussed so far—are called **strong entity types**.
- 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**, and we call the relationship type that relates a weak entity type to its owner 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** because a weak entity cannot be identified without an owner entity.

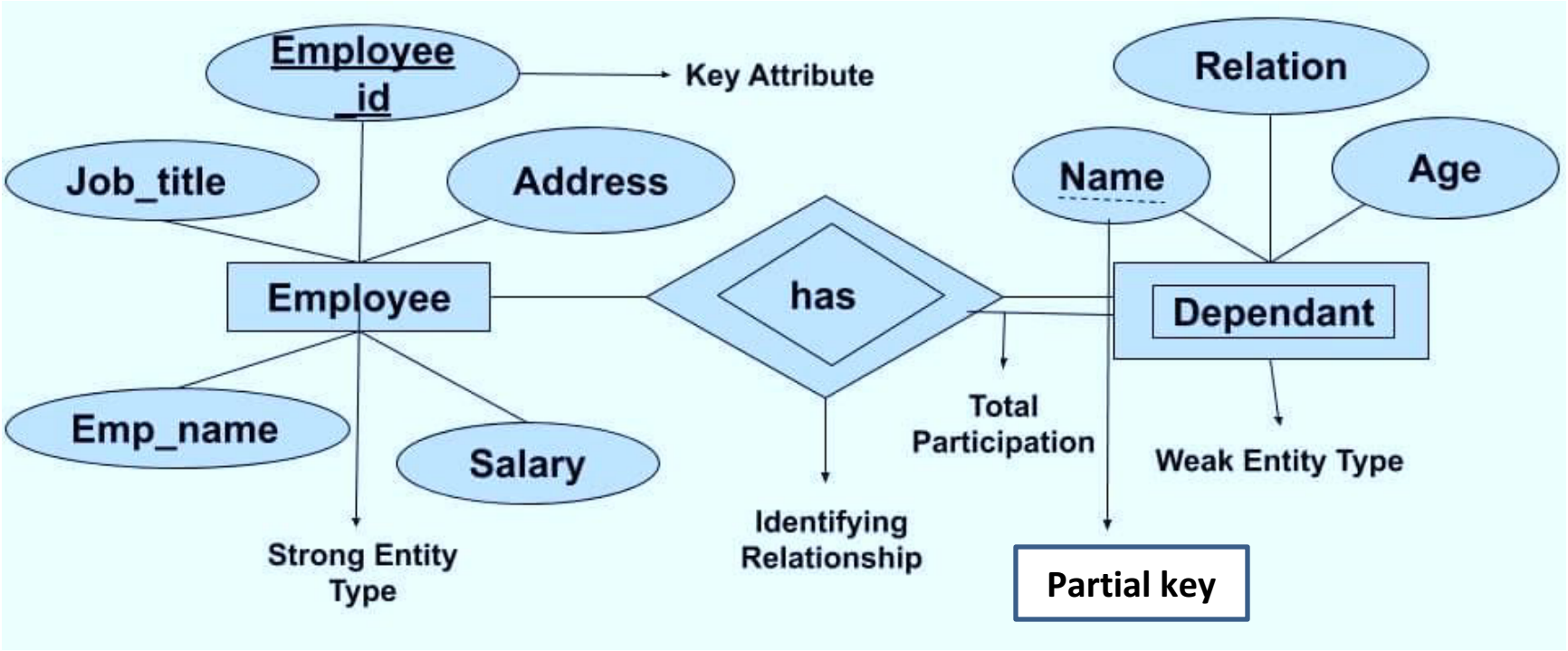
Weak Entity Types

- However, **not every existence dependency results in a weak entity type.**
- For **example,** a DRIVER_LICENSE entity cannot exist unless it is related to a PERSON entity, even though it has its own key (License_number) and hence is not a weak entity.
- Consider the entity type DEPENDENT, related to EMPLOYEE, which is used to keep track of the dependents of each employee via a 1:N relationship.
- The attributes of DEPENDENT are Name, Birth_date, Gender, and Relationship (to the employee).
- Two dependents of *two distinct employees* may, by chance, have the same values for Name, Birth_date, Gender, and Relationship, but they are still distinct entities.
- They are identified as distinct entities only after determining the *particular employee entity* to which each dependent is related.
- Each employee entity is said to *own* the dependent entities that are related to it.

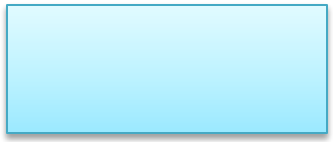
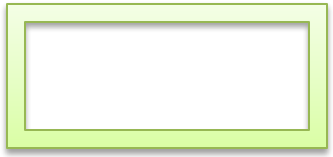
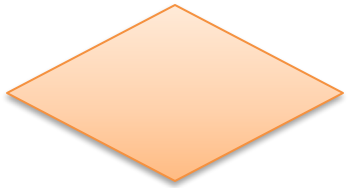
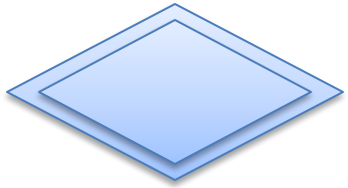
Weak Entity Types

- A **weak entity type** normally has a **partial key**, which is the attribute that can **uniquely identify weak entities** that are *related to the same **owner entity*****.**
- In our example, if we assume that no two dependents of the same employee ever have the same first name, the attribute Name of DEPENDENT is the **partial key.**
- In the worst case, a **composite attribute** of *all the weak entity's attributes* will be the partial key.
- In ER diagrams, both a weak entity type and its identifying relationship are distinguished by **surrounding their boxes** and **diamonds** with **double lines.**
- The **partial key attribute** is **underlined** with **a dashed** or **dotted line**.

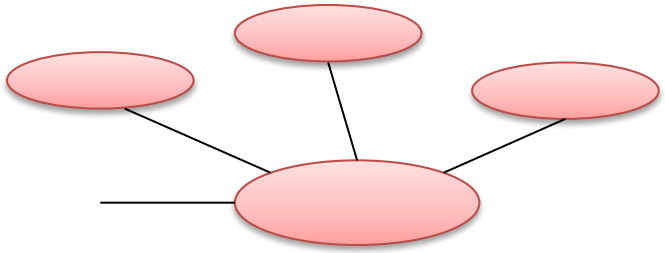
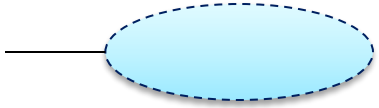
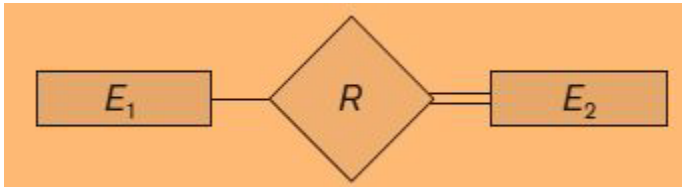
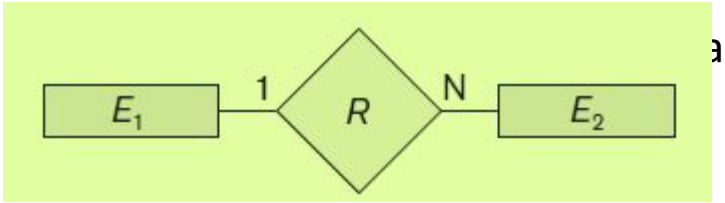
Weak Entity Types



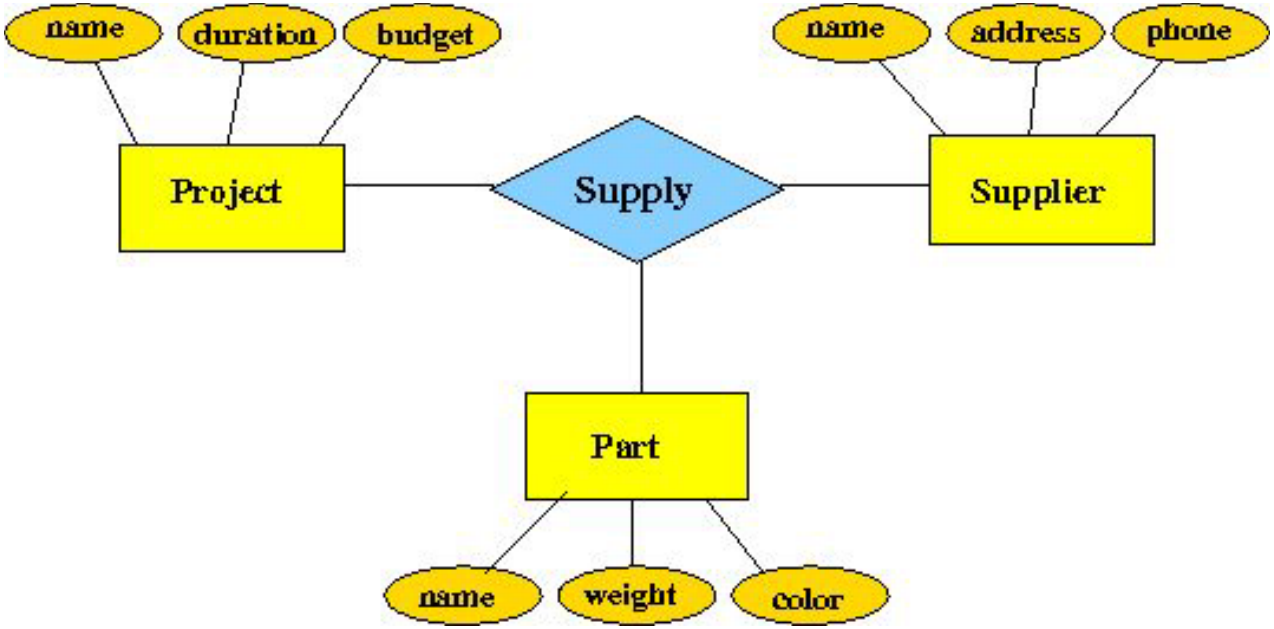
Summary

SYMBOL	MEANING
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute

Summary

SYMBOL	MEANING
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of *E*2 in *R*
	ardinality Ratio 1: N for *E*1:*E*2 in *R*

Ternary Relationship



Unary/Recursive Relationship