Entity Relationship Diagram

Lecture 3

By: Dr. Marwa Hussien

Types of Keys in DBMS

- Keys in DBMS can take several forms:
  - Primary Keys: An attribute that is present in the table and uniquely identify each record in the table. It <u>should not</u> be repeated or have a NULL value.
    - Example: "Roll_no" for students, as no student can have the same number as another student.
  - Foreign Key: allow you to define and establish relationships between a pair of tables. If Table A needs to refer to the primary key in Table B, you'll use a foreign key in Table A so you have values in that table to match those in Table B.
  - Unique Key: very similar to primary keys in that both contain unique identifiers for the records in a table. The only difference is that a unique key can contain a null value, whereas a primary key can't.
    - Example: An attribute for an employee bank account "Bank_acc_num" uniquely identify each employee, but it could have a NULL value if the employee has no band account.

Types of Keys in DBMS

- Candidate Key: Though you may have picked a unique attribute to serve as your primary key, there may be other candidates within a table.
  - Example: "Student_ID", "Email", "Phone" are considered as candidate keys as that can be as unique as the student "Roll_no".
- Composite Key: If you have attributes that wouldn't be unique when taken alone but can be combined to form a unique identifier for a record, you have a composite key.
  - Example: if we have employees working for projects for a number of hours in each project. In this table we need "Emp_ID" and "Proj_No" both to uniquely identify each employee and his working project.
- Super Key: This term refers to the sets of attributes that uniquely identify a record, meaning it's a combination of candidate keys.
  - Example: In the Student table the sets of super keys are: {"Roll_no"}, {"Student_ID"}, {"Email"}, {"Phone"}, {"Roll_no", "Student_ID"}, {"Roll_no", "Student_ID", "Email"}, {"Student_ID", "Email", "Phone"}, {"Email","Phone"}, etc.
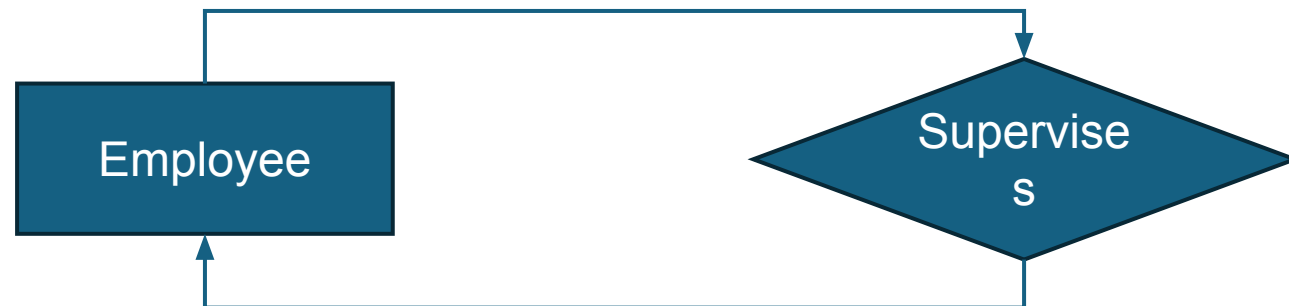
3. Relationships - ER Model

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

Student —— Enrolled in —— Course

3. Relationships - ER Model

- A recursive relationship: is a relationship between the same participating entity in distinct roles.
    - Example: the SUPERVISION relationship where EMPLOYEE participates twice in two distinct roles: supervisor (or boss) role and supervisee (or subordinate) role
    - Each relationship instance relates two distinct EMPLOYEE entities: One employee in supervisor role and One employee in supervisee role
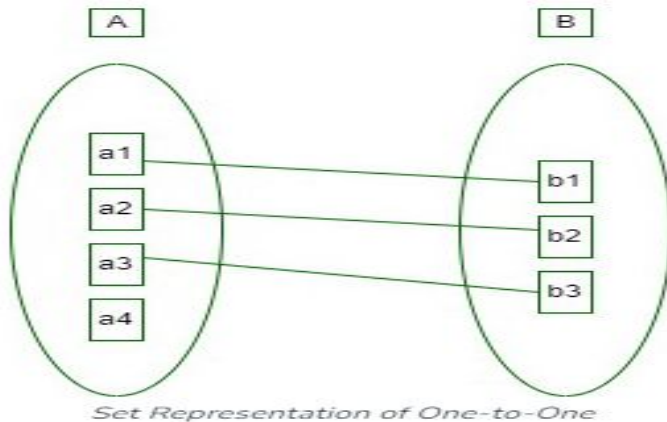
3. Relationships - ER Model

- Relationship Degree: the number of entities participating in a relationship.

    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.

    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.

    3. **n-ary Relationship:** when there are n entities set participating in a relation, the relationship is called an n-ary relationship.
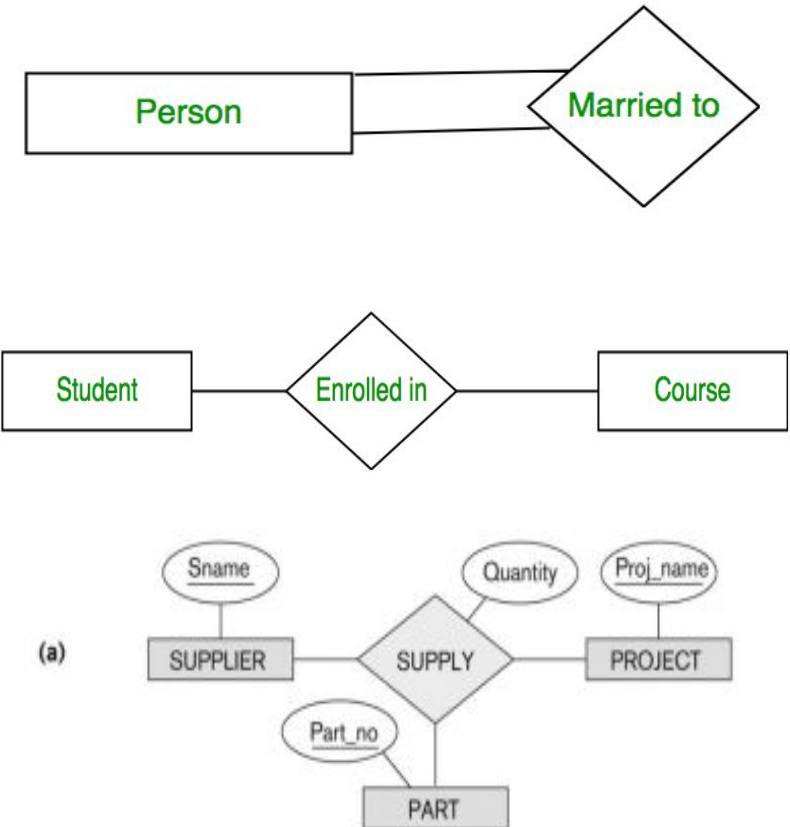
3. Relationships - ER Model

- **Relationship Cardinality:** The number of times an entity of an entity set participates in a relationship.

- Cardinality can be of different types:

  1. **One-to-One:** When each entity in each entity set can take part only once in the relationship.

     1. Assume that a male can marry one female and a female can marry one male.
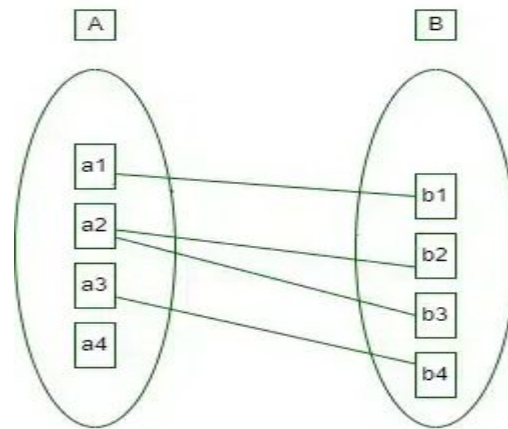

Set Representation of One-to-One
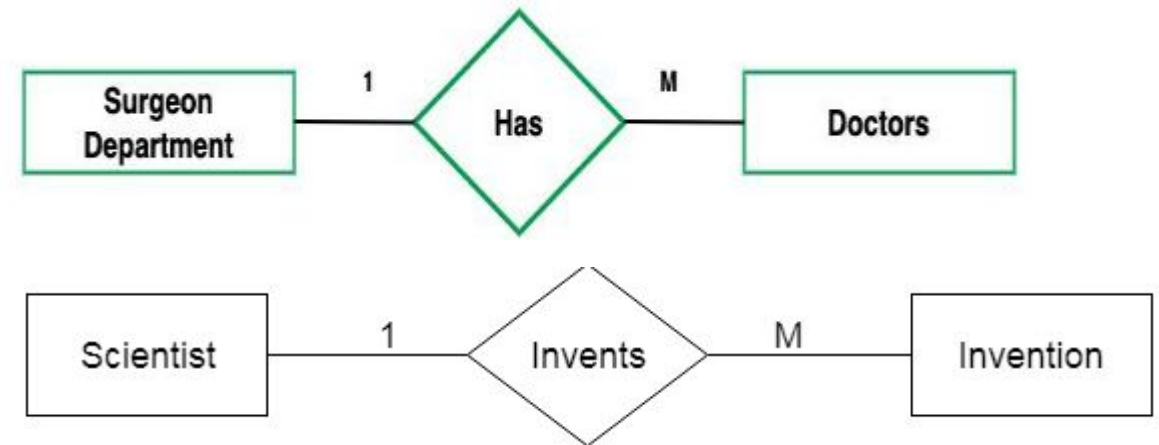
3. Relationships - ER Model

- Cardinality can be of different types:
  2. **One-to-Many:** when more than one instance of an entity is associated with a relationship, it is marked as '1:N'.
     - Assume that one department can accommodate many doctors. So the Cardinality will be 1 to M. It means one department has many Doctors.

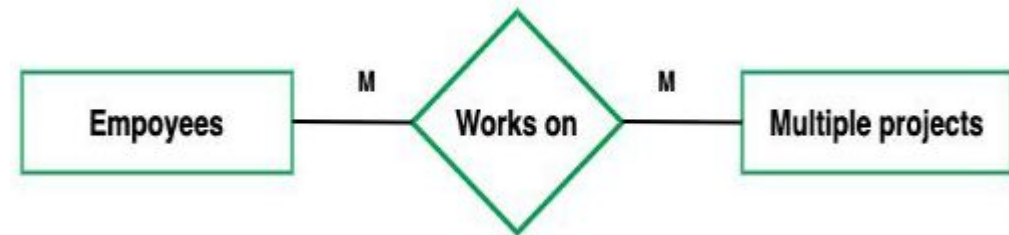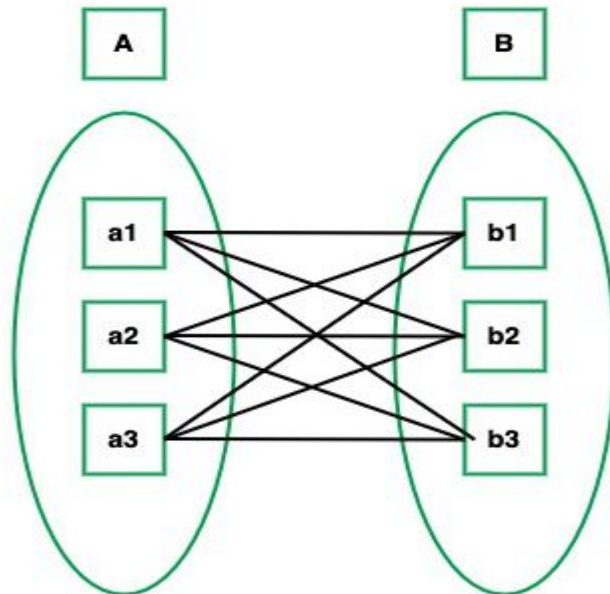

Set Representation of One-to-Many

3. Relationships - ER Model

- Cardinality can be of different types:
  3. **Many-to-Many:** when entities in all entity sets can take part more than once in the relationship.
     - Assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.

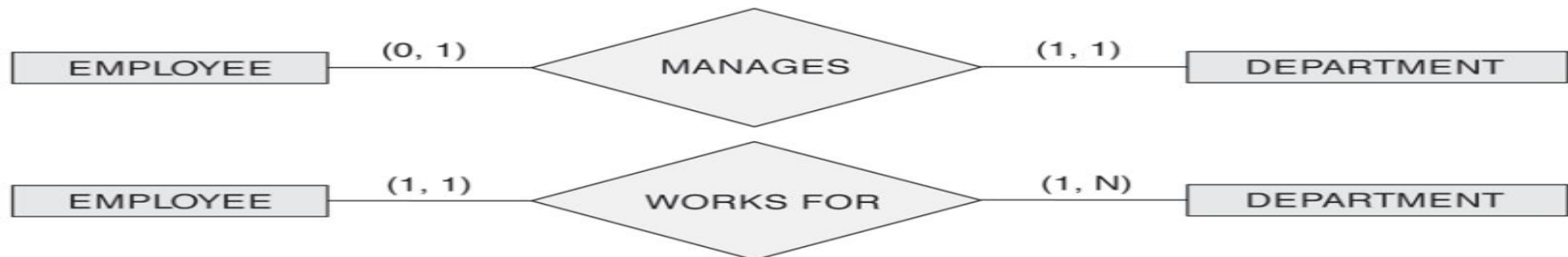# 3. Relationships - ER Model

- Participation Constraint:

  1. **Total Participation:** each entity in the entity set must participate in the relationship.

     - If each student must enroll in a course, the participation of students will be total.
     - Total participation is shown by a double line in the ER diagram.

  2. **Partial Participation:** the entity in the entity set may or may NOT participate in the relationship.

     - If some courses are not enrolled by any of the students, the participation in the course will be partial.

Total Participation and Partial Participation
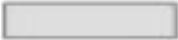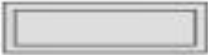
3. Relationships - ER Model

- (min, max) notation for relationship structural constraints:
  - Specified on each participation of an entity E in a relationship R.
  - Specifies that each entity E participates in at least min and at most max relationship instances in R
  - Examples:
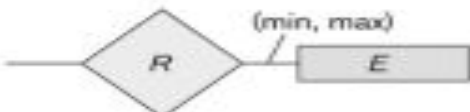   - A department has exactly one manager and an employee can manage at most one department.
      - Specify (0,1) for participation of EMPLOYEE in MANAGES
      - Specify (1,1) for participation of DEPARTMENT in MANAGES
    - An employee can work for exactly one department but a department can have any number of employees.
      - Specify (1,1) for participation of EMPLOYEE in WORKS_FOR

EMPLOYEE	(0, 1)	MANAGES	(1, 1)	DEPARTMENT

EMPLOYEE	(1, 1)	WORKS FOR	(1, N)	DEPARTMENT

3. Relationships - ER Model

- A relationship can have attributes:
  - Its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.
    - A value of HoursPerWeek depends on a particular (employee, project) combination
  - Most relationship attributes are used with M:N relationships
    - In 1:N relationships, they can be transferred to the entity type on the N-side of the relationship

Figure 3.14
Summary of the notation for ER diagrams.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
E_1 — R — E_2	Total Participation of E_2 in R
E_1 —1— R —N— E_2	Cardinality Ratio 1 : N for $E_1:E_2$ in R
R (min, max) E	Structural Constraint (min, max) on Participation of E in R

13

ERD Notations

- **Entity-Relationship Diagram (ERD) Notations** are symbols used to visually represent the structure of a database, including entities, relationships, and attributes.

- The most common notations are:
  1. Crow's Foot Notation
  2. Chen Notation

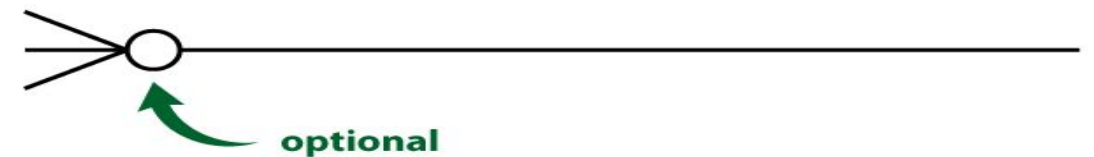# Symbols in Crow's Foot Notation

- Entities and Attributes:
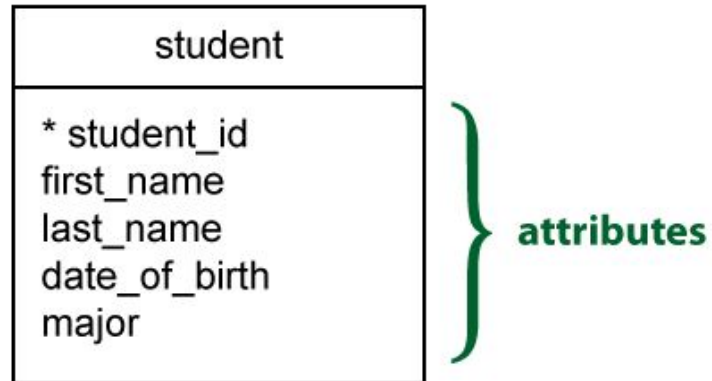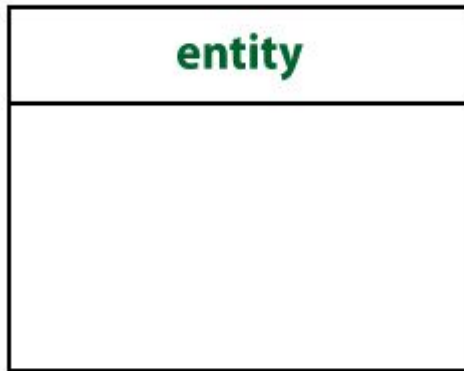
Symbols in Crow's Foot Notation

- Relationships:
  - Cardinality:

  - Participation Constraint:

multiplicity of one

multiplicity of many

mandatory

optional

Symbols in Crow's Foot Notation

- **Note:**
  - Cardinality symbol is put next to the entity on the relationship.
  - Participation Constraint symbol is put beside the other entity on the relationship.
  - Min-max constraint symbol is put next to the entity on the relationship.
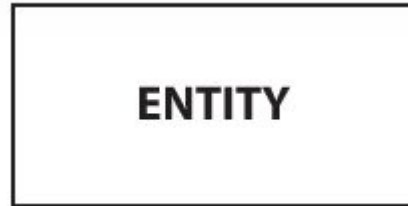
- **Example:**
  - A lecturer teaches many courses or may not teach courses at all, a course is thought by one Lecturer.

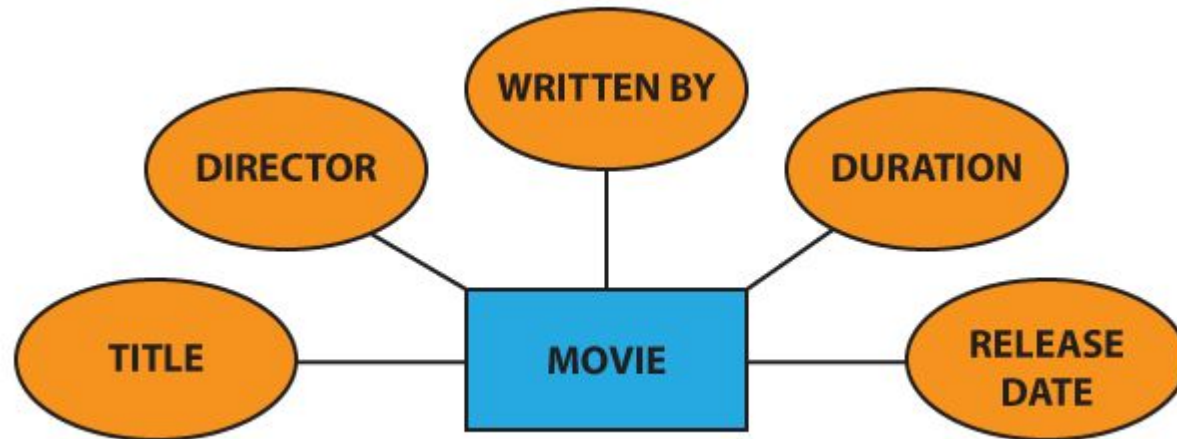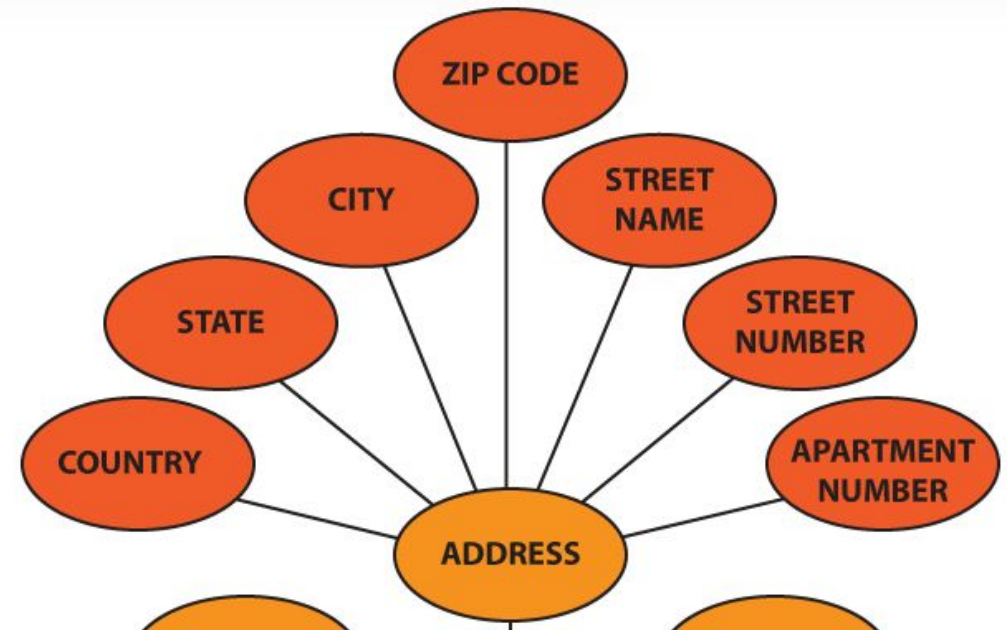# Symbols in Chen Notation

- Entity:

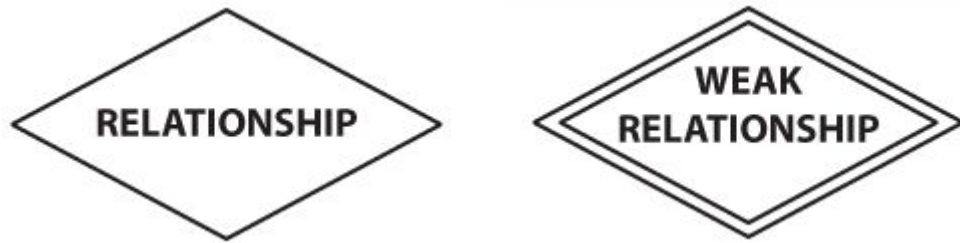|		
ENTITY	WEAK ENTITY	ASSOCIATIVE ENTITY

- Attributes

Symbols in Chen Notation

• Attribute types:

Symbols in Chen Notation

- Relationships:

- Participation:

- Cardinality:
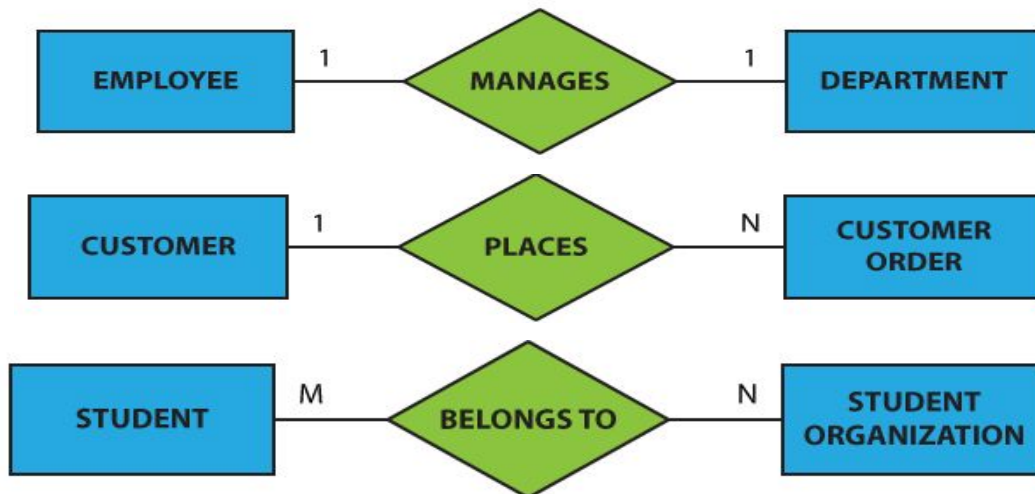
How to Draw ER Diagram?

1. Identifying all the Entities, and place them in a Rectangle, and labeling them accordingly.

2. Identify the relationship between them and pace them accordingly using the Diamond, and make sure that, Relationships are not connected to each other.

3. Attach attributes to the entities properly.

4. Remove redundant entities and relationships.

5. Add proper colors to highlight the data present in the database.

Example COMPANY Database

- We need to create a database schema design based on the following requirements of the COMPANY Database:
  - The company is organized into DEPARTMENTs. Each department has a name, number and an employee who manages the department. We keep track of the start date of the department manager. A department may have several locations.
  - Each department controls a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.
  - The database will store each EMPLOYEE's social security number, address, salary, sex, and birthdate. Each employee works for one department but may work on several projects. The DB will keep track of the number of hours per week that an employee currently works on each project. It is required to keep track of the direct supervisor of each employee.
  - Each employee may have a number of DEPENDENTs. For each dependent, the DB keeps a record of name, sex, birthdate, and relationship to the employee.

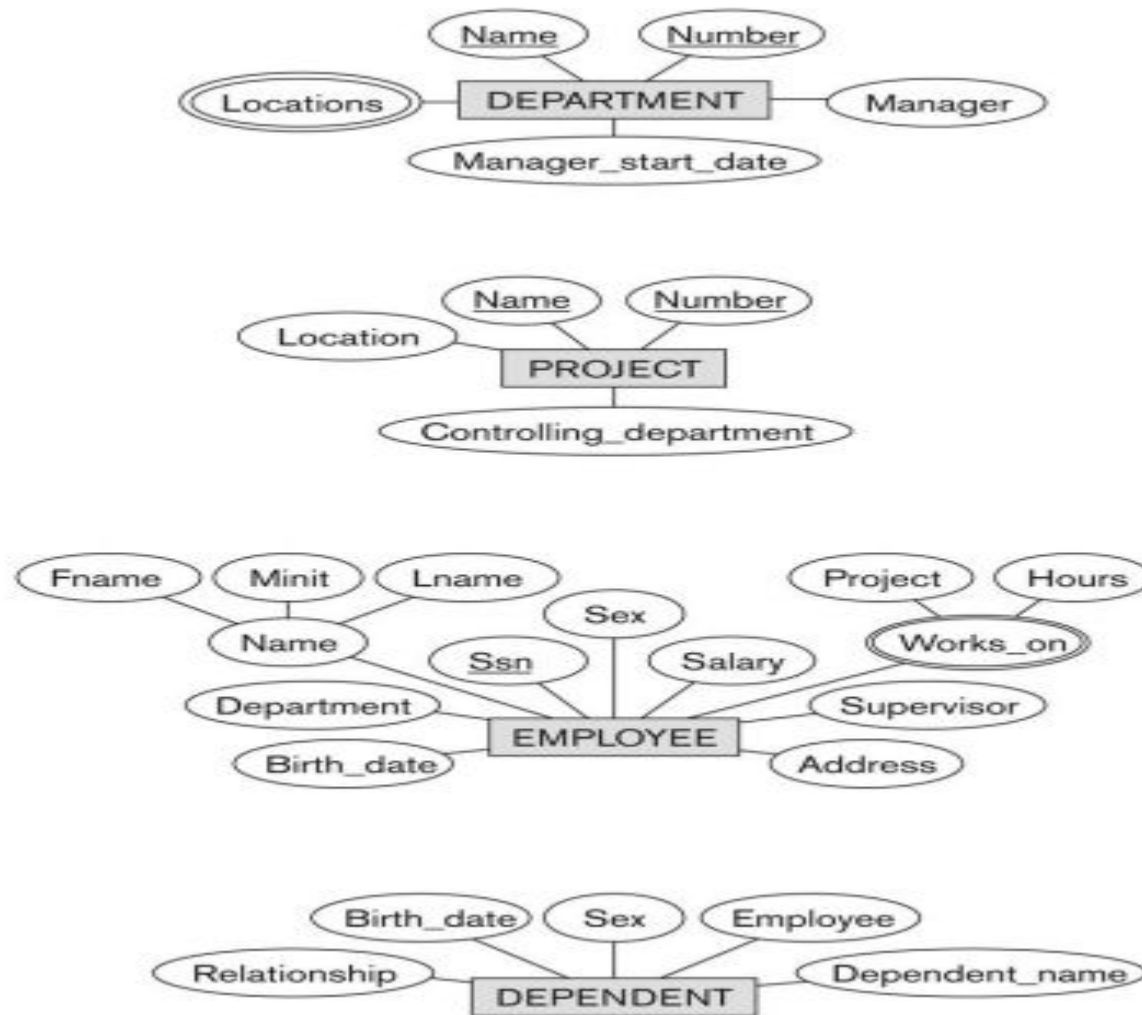# Initial Conceptual Design of Entity Types for the COMPANY Database Schema

Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

23

Refining the COMPANY database schema by introducing relationships

- By examining the requirements, six relationship types are identified
- All are binary relationships(degree 2)
- Listed below with their participating entity types:
  - WORKS_FOR (between EMPLOYEE, DEPARTMENT)
  - MANAGES (also between EMPLOYEE, DEPARTMENT)
  - CONTROLS (between DEPARTMENT, PROJECT)
  - WORKS_ON (between EMPLOYEE, PROJECT)
  - SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
  - DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)

ER Diagram for the COMPANY Database

Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.
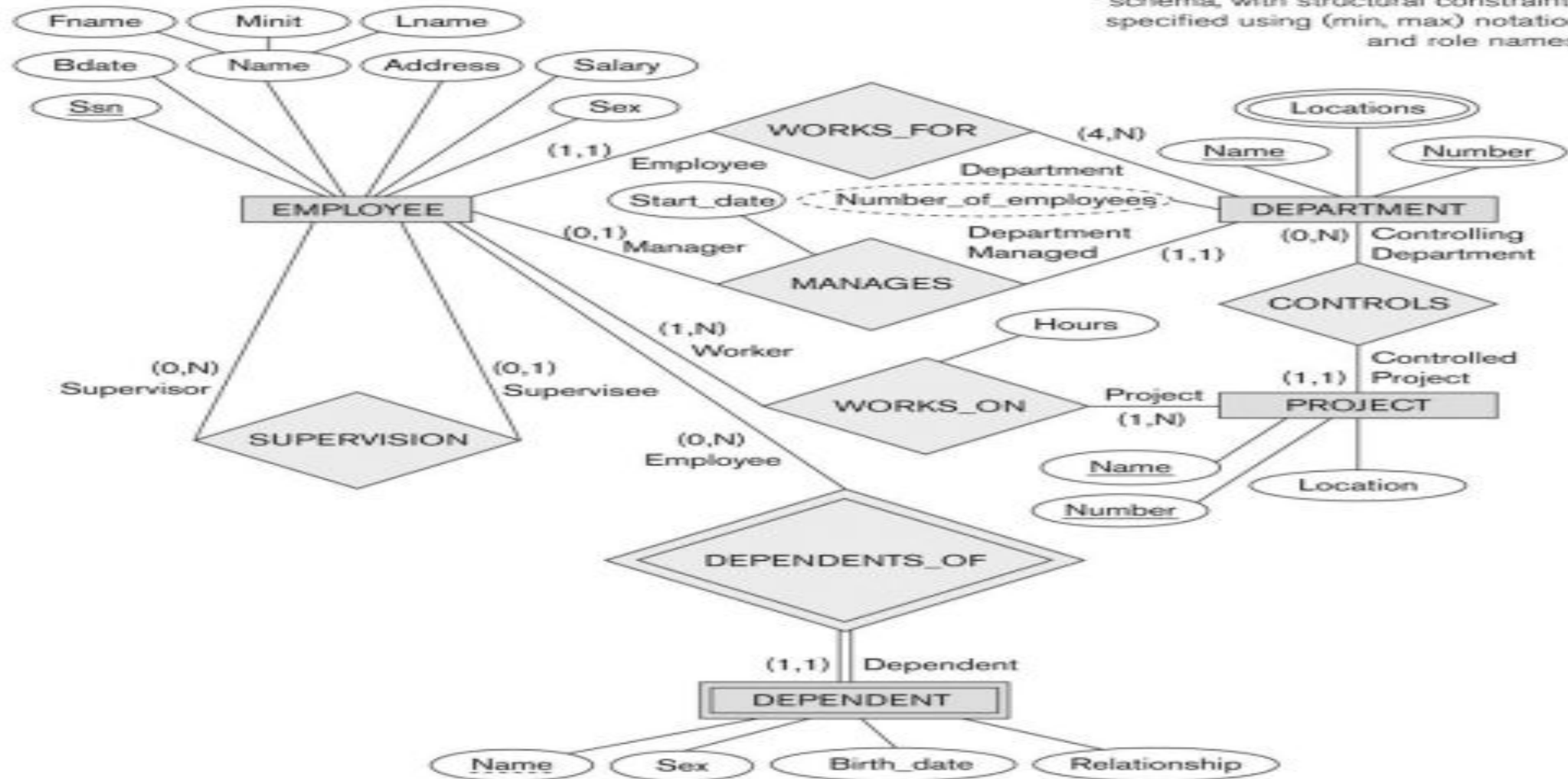
Assignment Problem- Banking Database

● Consider the following set of requirements for a Bank database that is used to keep track of Customer.

a) Each bank has a unique name.

b) Each branch has a number, name, address (number, street, city), and set of phones.

c) Customer includes their name, set of address (P.O. Box, city, zip code, country), set of phones, and social security number.

d) Accounts have numbers, types (e.g. saving, checking) and balance. Other branches might use the same designation for accounts. So, to name an account uniquely, we need to give both the branch number to which this account belongs to and the account number.

e) Not all bank customers must own accounts and a customer may have at most 5 accounts in the bank.

f) An account must have only one customer.

g) A customer may have many accounts in different branches.