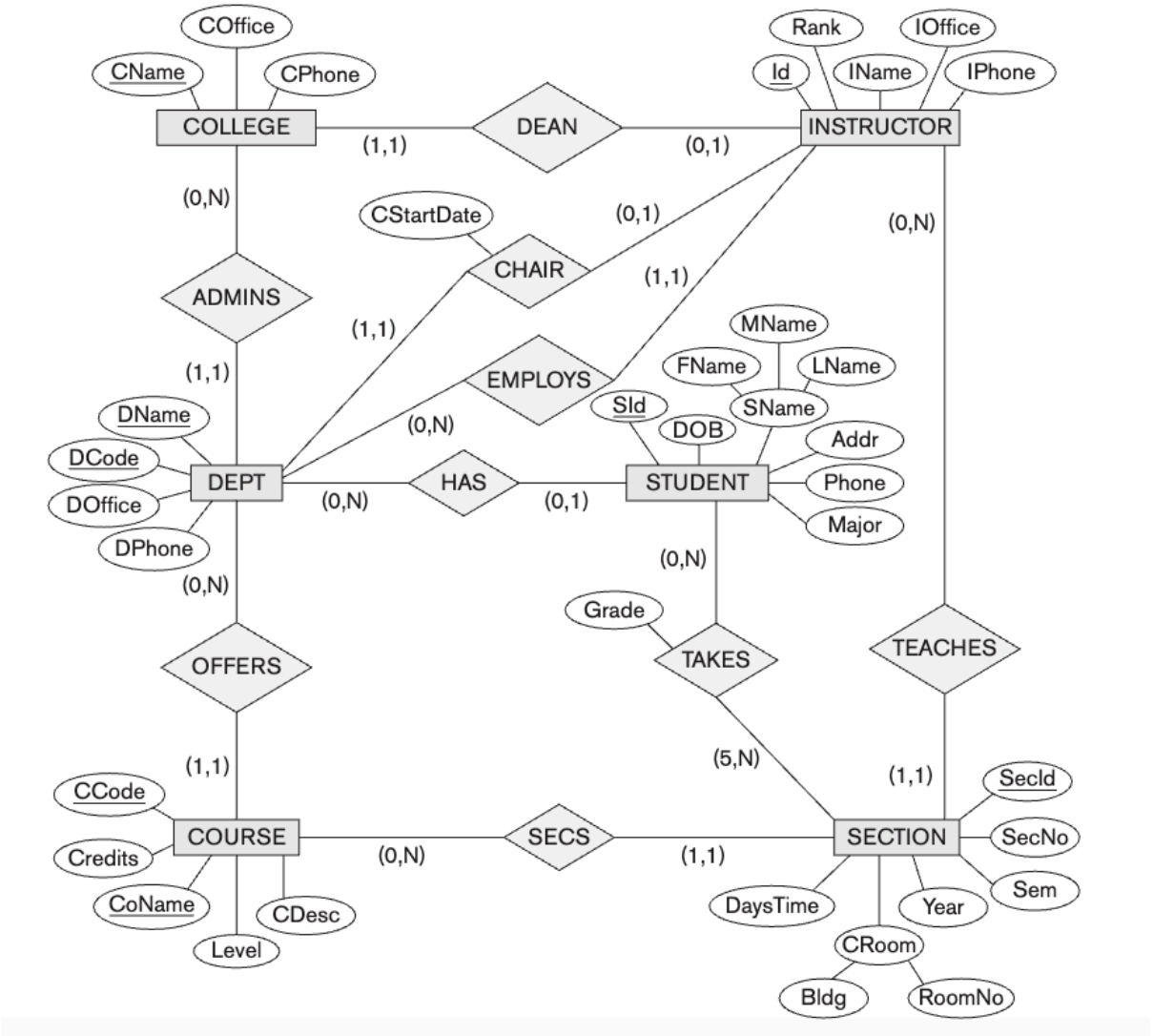
Homework assignment 2
Chapter 3

Problem 1

Which combinations of attributes have to be unique for each individual SECTION entity in the UNIVERSITY database shown in Figure 3.20 to enforce each of the following mini-world constraints:



a. During a particular semester and year, only one section can use a particular classroom at a particular DaysTime value
=> Answer:-

Unique Combination of Attributes:

- Year, Semester, CRoom (Classroom), DaysTime.

Explanation:

- To ensure that a classroom is not double-booked, the combination of Year, Semester, Classroom (CRoom), and DaysTime must be unique. No two sections should use the same room at the same time during a semester and year.

b. During a particular semester and year, an instructor can teach only one section at a particular DaysTime value.
=> Answer:-

Unique Combination of Attributes:

- Year, Semester, Instructor Id (Id), DaysTime.

Explanation:

- To prevent an instructor from being assigned to teach two sections at the same time, the combination of Year, Semester, Instructor Id, and DaysTime must be unique. This ensures that an instructor is only assigned to one section at a specific time.

c. During a particular semester and year, the section numbers for sections offered for the same course must all be different.
=> Answer:-

Unique Combination of Attributes:

- Course Code (CCode), Year, Semester, Section Number (SecNo).

Explanation:

- Each course can have multiple sections, but the Section Number should be unique within a course during a specific Year and Semester. So the combination of CCode, Year, Semester, and SecNo must be unique to ensure there is no duplication of section numbers within a course.

d. Can you think of any other similar constraints?
=> Answer:-

i) **Constraint:** A student can register for only one section of a course during a particular semester and year.

- Unique Combination of Attributes:
    - Student ID (SId), Course Code (CCode), Year, Semester.

Explanation: To ensure that a student doesn't register for multiple sections of the same course during a semester, the combination of SId, CCode, Year, and Semester must be unique. This ensures each student takes only one section of a course in a particular semester.

ii) **Constraint:** A section should not exceed its maximum capacity in class during a semester.

- Combination of Attributes for Validation:
    - Section Id (SecId), Number of Students Registered.

Explanation: For each section, there should be a maximum limit for the number of students. While this may not require a unique combination, it's a necessary constraint to track and enforce the section capacity.

Problem 2

Show an alternative design for the attribute described in Exercise 3.17 that uses only entity types (including weak entity types, if needed) and relationship types. (This is a continuation from problem 3 from the homework assignment 1)

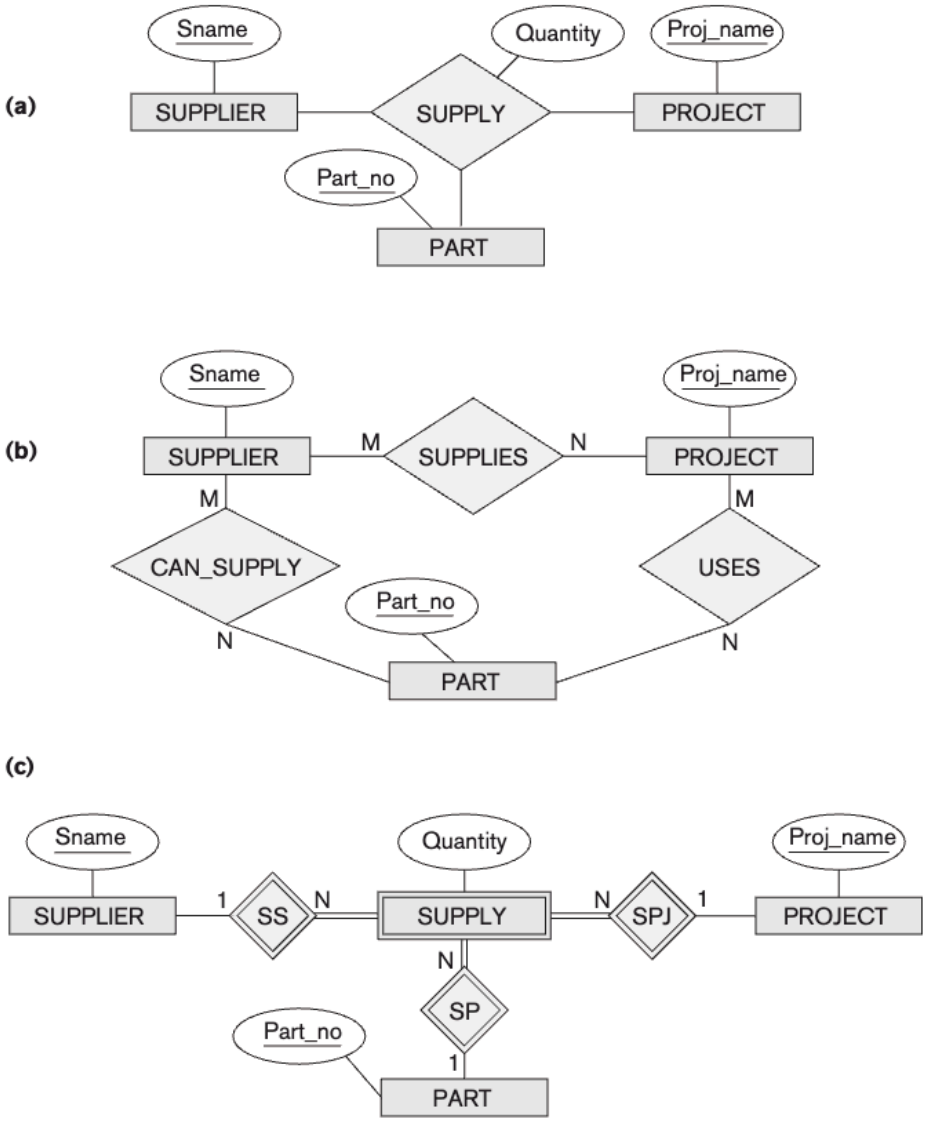


Figure 3.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

=> Answer:-

To redesign the relationships and attributes in the diagram using only “entity types” (including weak entity types, if needed) and “relationship types” we can break down the complex relationships and attributes into more granular entity and relationship structures.

➔ Step-by-Step Breakdown

- **Remove attributes from relationships:** In the given diagrams, attributes such as `Quantity` are attached to relationships (e.g., in the **SUPPLY** relationship). Instead, we will create new entity types to replace these attributes.

- **Replace multivalued attributes and dependent entities with weak entities:** For multivalued attributes or attributes that depend on a specific entity (like the combination of parts and projects), we can model these as weak entities.

➔ Alternative Design Based on Diagram (c)

- ***Entities:***
  i) **SUPPLIER:**
    - `Sname` (supplier name).

  ii) **PART:**
    - `Part_no` (part number).

  iii) **PROJECT:**
    - `Proj_name` (project name).

  iv) **SUPPLY** (New Entity):
    - Represents the supply transaction (replaces the **SUPPLY** relationship from the original design).
    - Attributes: `Quantity` (tracks the number of parts supplied).
    - **Weak entity** because its existence depends on both the **SUPPLIER** and the **PART.**

- ***Relationships:***

  i) **SUPPLIES** (between **SUPPLIER** and **SUPPLY**):
  - Each **SUPPLIER** can participate in multiple **SUPPLY** entities.
  - Relationship is one-to-many (1:N), since one supplier can supply multiple parts.

  ii) **CONTAINS** (between **SUPPLY** and **PART**):
  - Each **SUPPLY** entity is associated with exactly one **PART**.
  - The relationship is many-to-one (N:1), since each supply transaction contains a specific part.

  iii) **PROJECT_SUPPLY** (between **SUPPLY** and **PROJECT**):
  - Each **SUPPLY** entity is associated with exactly one **PROJECT**.
  - The relationship is many-to-one (N:1), as each supply goes to a particular project.

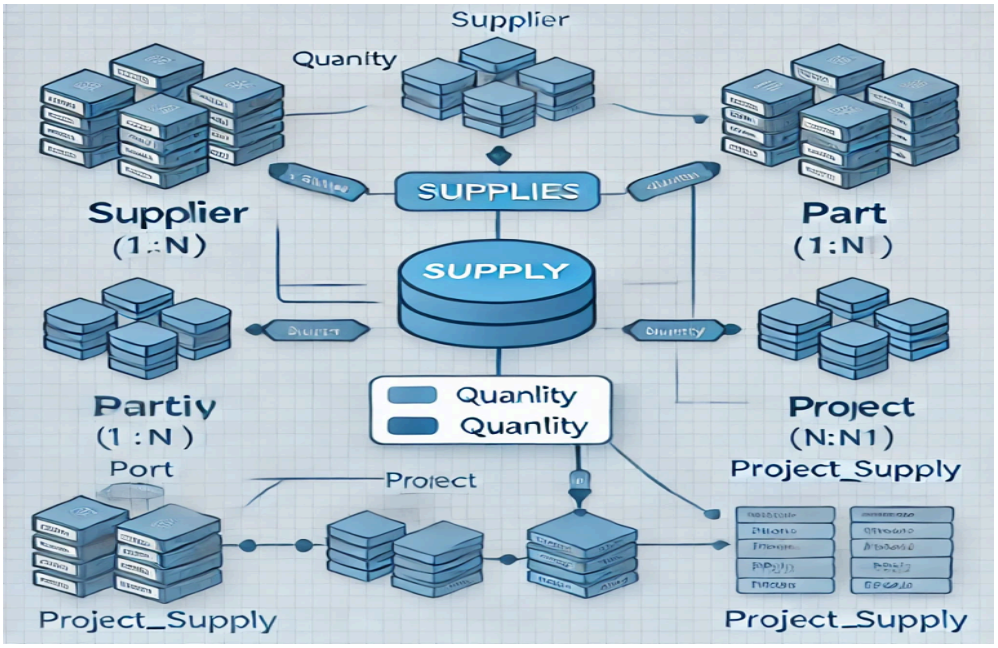
- ***Redesigned ER Diagram:***

  - **Entities:**
    a. SUPPLIER (Sname)
    b. PART (Part_no)
    c. PROJECT (Proj_name)
    d. **SUPPLY** (Weak entity):
       → Attributes: Quantity

  - **Relationships:**
    a. **SUPPLIES** (1:N relationship between **SUPPLIER** and **SUPPLY**)
    b. **CONTAINS** (N:1 relationship between **SUPPLY** and **PART**)
    c. **PROJECT_SUPPLY** (N:1 relationship between **SUPPLY** and **PROJECT**)

- ***Key Points:***
  - I have replaced the original **SUPPLY** relationship with a new entity **SUPPLY**, which has `Quantity` as an attribute and acts as a bridge entity between **SUPPLIER**, **PART**, and **PROJECT**.
  - The structure removes attributes from the relationships, converts them into entities, and uses weak entities where necessary (such as **SUPPLY**).

Below is the ER diagram based on the redesign of the original supply system. It shows the relationships between **SUPPLIER**, **PART**, **PROJECT**, and **SUPPLY** (as a weak entity) with the appropriate relationships.



Problem 3

Consider the ER diagram in Figure 3.23. Assume that an employee may work in up to two departments or may not be assigned to any department.

Assume that each department must have one and may have up to three phone numbers. Supply (min, max) constraints on this diagram. State clearly any additional assumptions you make. Under what conditions would the relationship HAS_PHONE be redundant in this example?

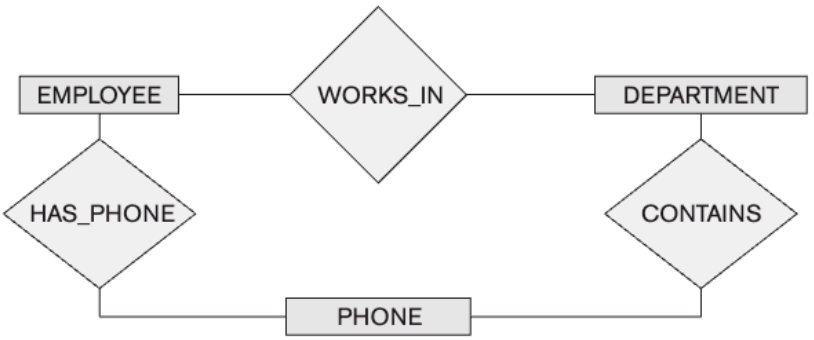


Figure 3.23
Part of an ER diagram for a COMPANY database.

=> Answer:-

The ER diagram illustrates the relationships between **EMPLOYEE**, **DEPARTMENT**, and **PHONE** in a company database. The key relationships and entities involved are:

- **EMPLOYEE** works in a **DEPARTMENT** (relationship: **WORKS_IN**).
- **DEPARTMENT** contains one or more **PHONE** numbers (relationship: CONTAINS).
- **EMPLOYEE** has a phone number through the relationship **HAS_PHONE**.

- **Assumptions for (min, max) constraints:**
    a. **WORKS_IN** (Employee and Department relationship):
        - An employee may work in **up to two departments** or may not work in any department.
        - So the cardinality for **EMPLOYEE** to **DEPARTMENT** would be (0, 2).
        - A department may have multiple employees or none, so the cardinality for **DEPARTMENT** to **EMPLOYEE** is (0, N).

b. **CONTAINS** (Department and Phone relationship):
   - Each department must have at least one and at most three phone numbers.
   - The cardinality from **DEPARTMENT** to **PHONE**is (1, 3).
   - One phone number can only be associated with one department, so the cardinality from **PHONE** to **DEPARTMENT** is (1, 1).

c. **HAS_PHONE** (Employee and Phone relationship):
   - We assume an employee may have **one or more** phone numbers, hence cardinality (0, N) from **EMPLOYEE** to **PHONE**.
   - A phone number can only belong to one employee, so the cardinality from **PHONE** to **EMPLOYEE** is (1, 1).

- Conditions where the **HAS_PHONE** relationship might be redundant:- The **HAS_PHONE** relationship would be redundant if:

a. Phone numbers are only assigned to **departments** and not to **individual employees**. In this case, any phone number associated with an employee would be indirectly derived through their association with the department they work in.

b. If an organization policy states that **employees only use department phones**, making personal phone numbers for employees are made irrelevant.

In these cases, the **HAS_PHONE** relationship could be removed, as phones would be linked to departments, and employees would have access to the department's phone through their work desk.