**Batch: D1 Roll No.: 16010123217**

**Experiment No. 1**

|  |
| --- |
| **Title:**Database application description and Design of Entity-Relationship diagram |

**Objective:** To comprehend the data requirements of the application and design the Entity-Relationship (ER) diagram for the database

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Expected Outcome of Experiment:**

CO1: Comprehend the Characteristics of Relational Database Management Systems.

CO2: Create Relational Database Designs Based on Entity-Relationship Models. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Books/ Journals/ Websites referred:**

1. G. K. Gupta :”Database Management Systems”, McGraw – Hill
2. Korth, Slberchatz, Sudarshan : “Database Systems Concept”, 6th Edition , McGraw Hill
3. Elmasri and Navathe, “Fundamentals of Database Systems”, 5thEdition, PEARSON Education.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Pre Lab/ Prior Concepts:**

**ER Model:**

The ER data model was developed to facilitate the database design by allowing specification of an enterprise schema that represents the overall logical structure of the database. The ER model is one of the several data models. The semantic aspect of the model lies in its representation of the meaning of the data. The ER model is very useful many database design tools drawn on concepts from the ER model. The ER model employs 3 basic notations: entity set, relationship set and attributes**.**

**Example Case Study**: List the data requirements for the database of the company Case Study : A company which keeps track of employee, department and projects.

The database designers provide the following description

* 1. The company is organized into departments. Each department has unique name, unique number, and particular employee to manage the department. We keep track of the start date and the employee begins managing the department. The department has several locations.
  2. The department controls a number of projects each of which has a unique name, unique number and a single location.
  3. We store each employee's name, social security number, address, salary, gender and dob. An employee is assigned one department but may work on several projects which are not necessarily controlled by the same department. We keep track of the department each employee works on each project and for insurance purpose. We keep each dependents first name, Gender, dob and relation.

# Procedure for doing the ER diagram experiment

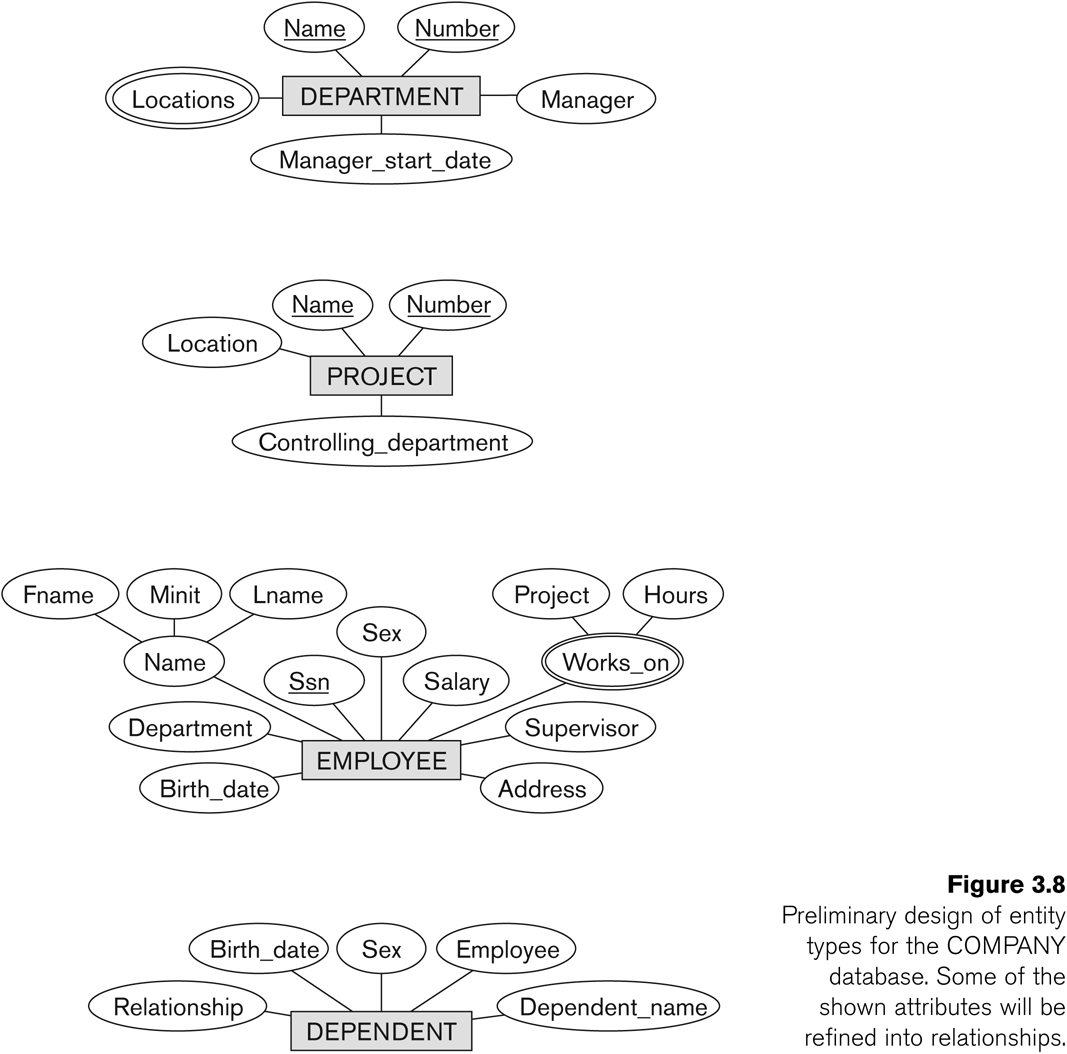
1. Identifying the Entities-Strong and weak entities ( Nouns from the problem definition )

Based on the requirements, we can identify four initial entity types in the COMPANY database:

* + DEPARTMENT
  + PROJECT
  + EMPLOYEE
  + DEPENDENT

1. Identify attributes of the Entity- keys, partial key, simple, composite, multivalued, derived ( characteristics of the entities)

The initial attributes shown are derived from the requirements description



1. **Identify relationships ( verbs )**

A **relationship** relates two or more distinct entities with a specific meaning.

* + For example, EMPLOYEE John Smith *works on* the ProductX PROJECT, or EMPLOYEE Franklin Wong *manages* the Research DEPARTMENT.

Relationships of the same type are grouped or typed into a **relationship type**.

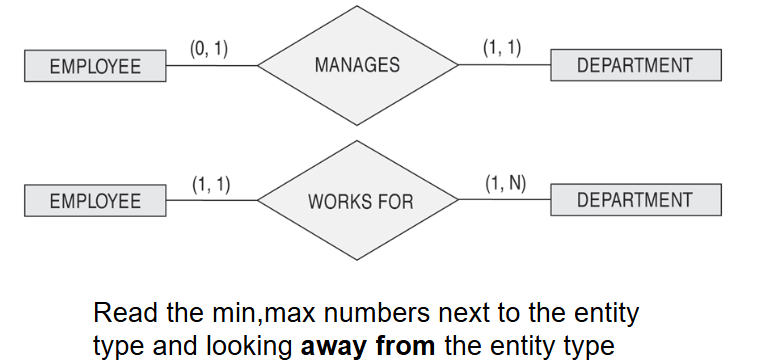
* + For example, the WORKS\_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.

The degree of a relationship type is the number of participating entity types.

* + Both MANAGES and WORKS\_ON are *binary* relationships.

In ER diagrams, we represent the *relationship type* as follows:

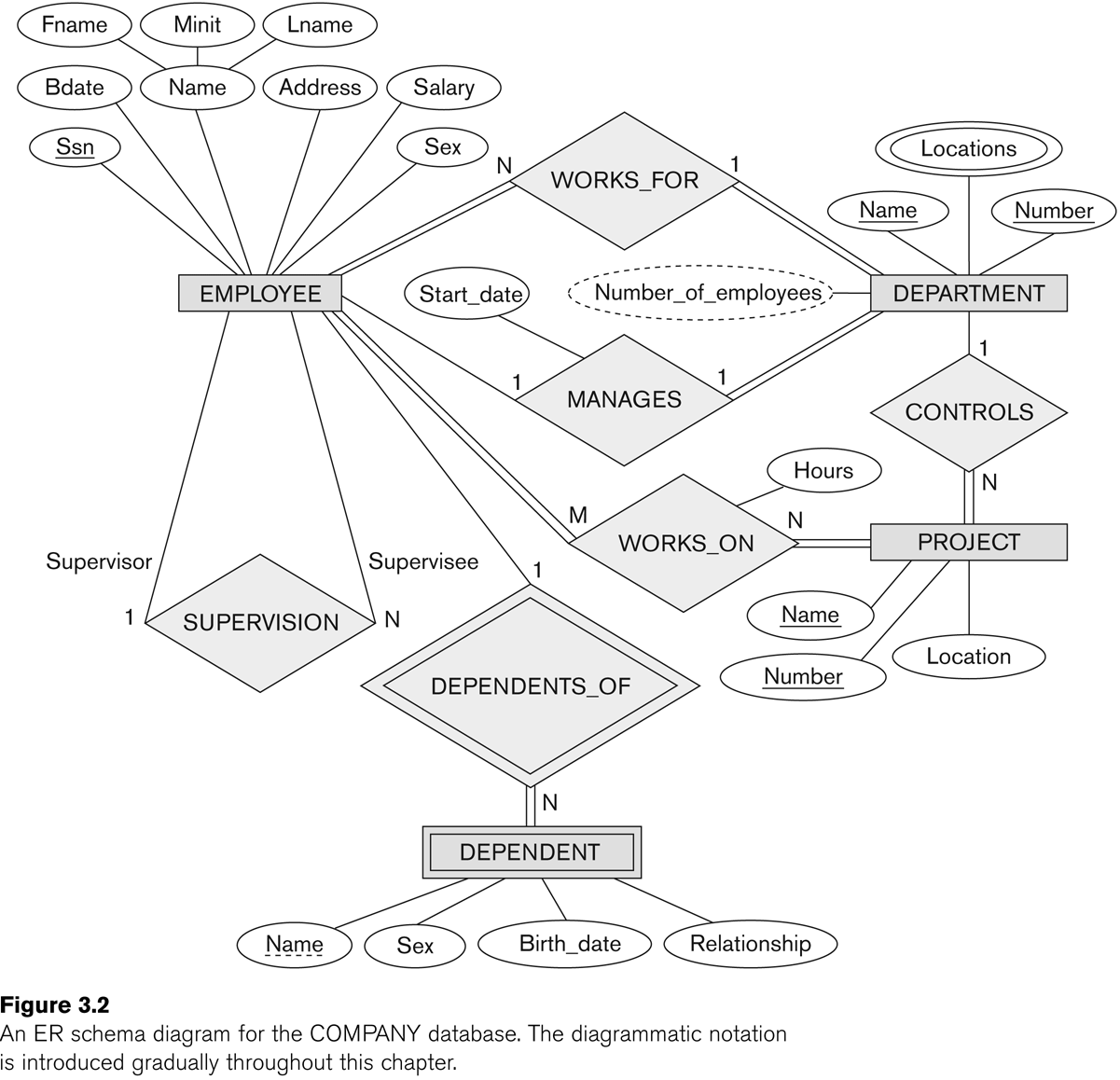
* + Diamond-shaped box is used to display a relationship type
  + Connected to the participating entity types via straight lines



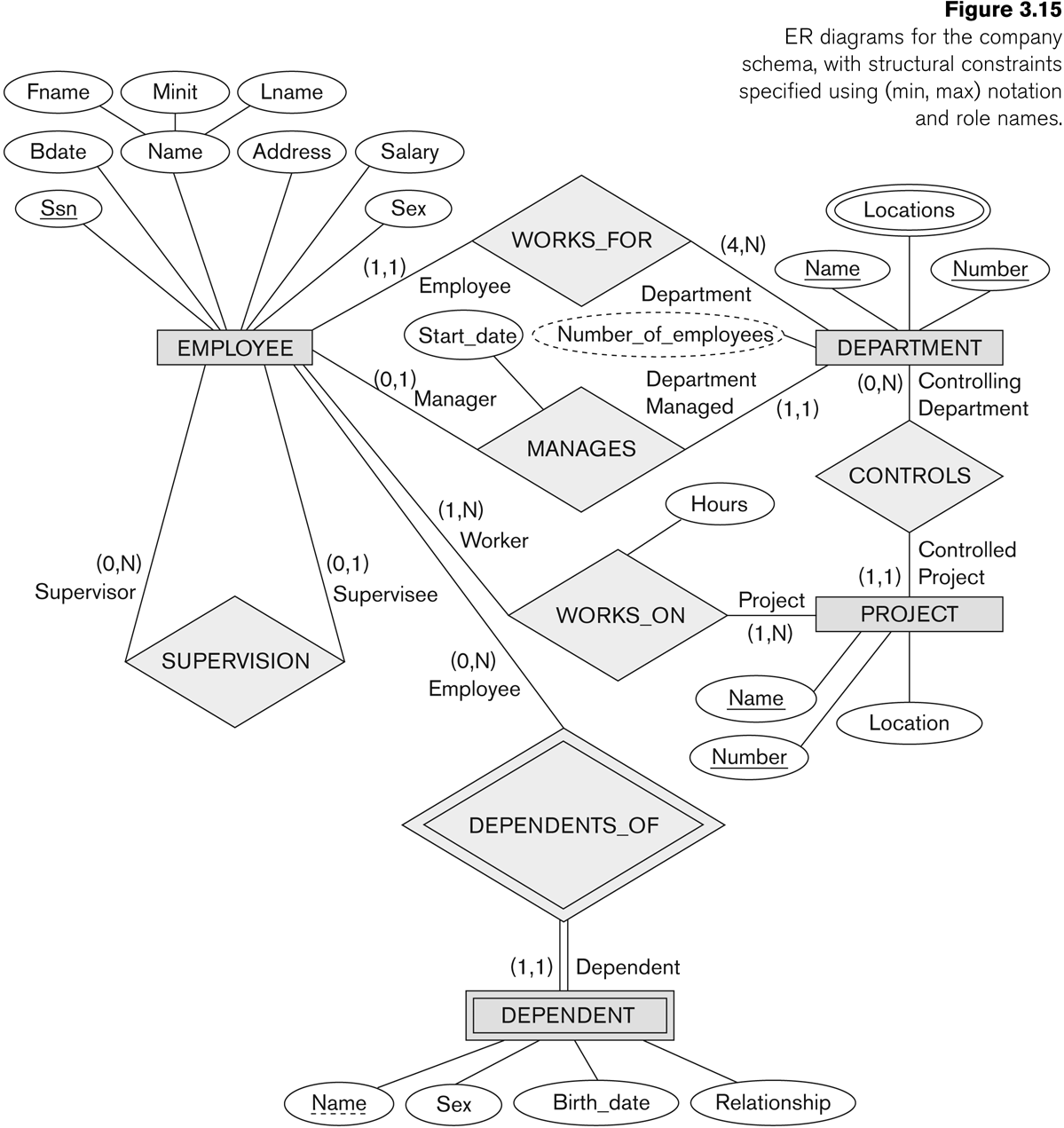
Listed below with their participating entity types:

* + 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 company Case Study Database:



Identify the structural constraints of the relationship (cardinality ratio, participation constraints**)**



**:**

**Problem Definition of student case study:**

CodeKeep is a platform designed to help competitive programmers manage and track their competitive programming profiles and upcoming contests. The platform enables users to store their achievements across multiple competitive programming sites, view upcoming contests in a unified calendar, and monitor them progress in various competitions. The system will store information related to users, contests, platforms, and user participation in contests. Additionally, users will be able to link multiple profiles from different platforms

to their account.

**Entity Relationship Model w.r.t Problem definition**

**1 ENTITIES**

**USER** – Represents an individual who participates in competitive programming contests. o Subgroup 1: **Personal Information**

▪ user\_id (Primary Key)

▪ name

▪ email

o Subgroup 2: **Competitive Profiles**

▪ user\_profiles (Multivalued Attribute) – Multiple links to competitive programming  profiles across different platforms.

2. **CONTEST** – Represents an upcoming programming contest. Each contest has a date, platform, and  other relevant details.

o Subgroup 1: **Contest Details**

▪ contest\_id (Primary Key)

▪ contest\_name

▪ contest\_date

▪ contest\_start

▪ duration

▪ derived: contest\_end

o Subgroup 2: **Platform Information**

▪ platform\_id (Foreign Key to PLATFORM)

3. **PLATFORM** – Represents a competitive programming platform where contests are hosted. o Subgroup 1: **Platform Information**

▪ platform\_id (Primary Key)

▪ platform\_name

▪ website\_url

4. **PROBLEMS** – Represents a user’s involvement in a contest. It tracks a user’s rank and score in each  contest they participate in.

o Subgroup 1: Problem Information

▪ problem\_id (Primary Key)

▪ difficult score

▪ platform\_key(fk)

**2. Multivalued Attribute:**

∙ **USER** has a multivalued attribute:

o user\_profiles – Stores multiple links to competitive programming profiles on various  platforms (e.g., Codeforces, LeetCode, AtCoder, etc)

**3. RELATIONSHIPS WITH PARTICIPATING ENTITIES**

**USER** can have many **PARTICIPATION** records, as they can participate in multiple contests. ∙ **CONTEST** can have many **PARTICIPATION** records, as multiple users can participate in the same  contest.

∙ Each **CONTEST** is hosted by a **PLATFORM**, so there is a one-to-many relationship between  **PLATFORM** and **CONTEST**.

∙ **USER** can link multiple profiles (as a multivalued attribute), so a user can have many associated  profiles from different platforms, which may be stored separately.

1. **RELATIONSHIPS WITH CONSTRAINTS AND ROLES**

 **USER-Participation-Contest**

* **Name**: Participation
* **Type**: Many-to-Many
* **Constraints**:
  + Each USER can participate in multiple CONTESTS.
  + Unique combination of user\_id and contest\_id.

 **CONTEST-HostedBy-PLATFORM**

* **Name**: HostedBy
* **Type**: One-to-Many
* **Constraints**:
  + Each CONTEST belongs to one PLATFORM.
  + A PLATFORM can host multiple CONTESTS.

 **USER-Has-CompetitiveProfiles**

* **Name**: HasProfiles
* **Type**: One-to-Many
* **Constraints**:
  + Each USER can have multiple user\_profiles.
  + profile\_url must be unique.

 **PROBLEM-AssociatedWith-PLATFORM**

* **Name**: AssociatedWith
* **Type**: One-to-Many
* **Constraints**:
  + Each PROBLEM belongs to one PLATFORM.
  + A PLATFORM can have multiple PROBLEMS.

**ER Model:**

A diagram of a flowchart

Description automatically generated

**Conclusion:**  
The CodeKeep system effectively organizes competitive programming data using entities like USER, CONTEST, PLATFORM, and PROBLEM, with well-defined relationships. It supports tracking user profiles, contest participation, and achievements across platforms, ensuring a unified and structured approach to managing competitive programming activities.

**Post lab Descriptive Questions**

1. Discuss the concept of aggregation. W.r.t your case study give example for aggregation

Aggregation is a relationship between two classes that represents a "has-a" relationship. It is a type of association where one class contains a reference to another class, and the lifecycle of the contained object is independent of the container class. Aggregation is often referred to as a weak association because the contained object can exist independently of the container.

**Example (CodeKeep):**  
In the USER-Participation-CONTEST relationship:

* Participation (between USER and CONTEST) can be aggregated to represent achievements like badges or certificates.
* Attributes for the aggregated entity:
  + badge\_earned
  + certificate\_link.