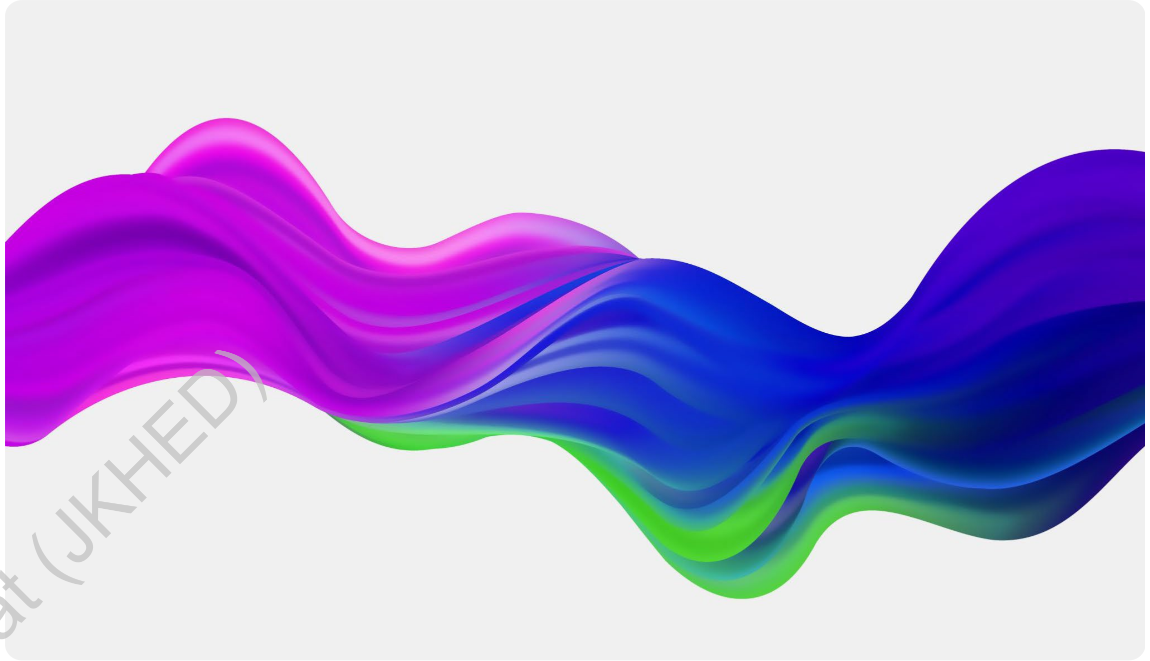
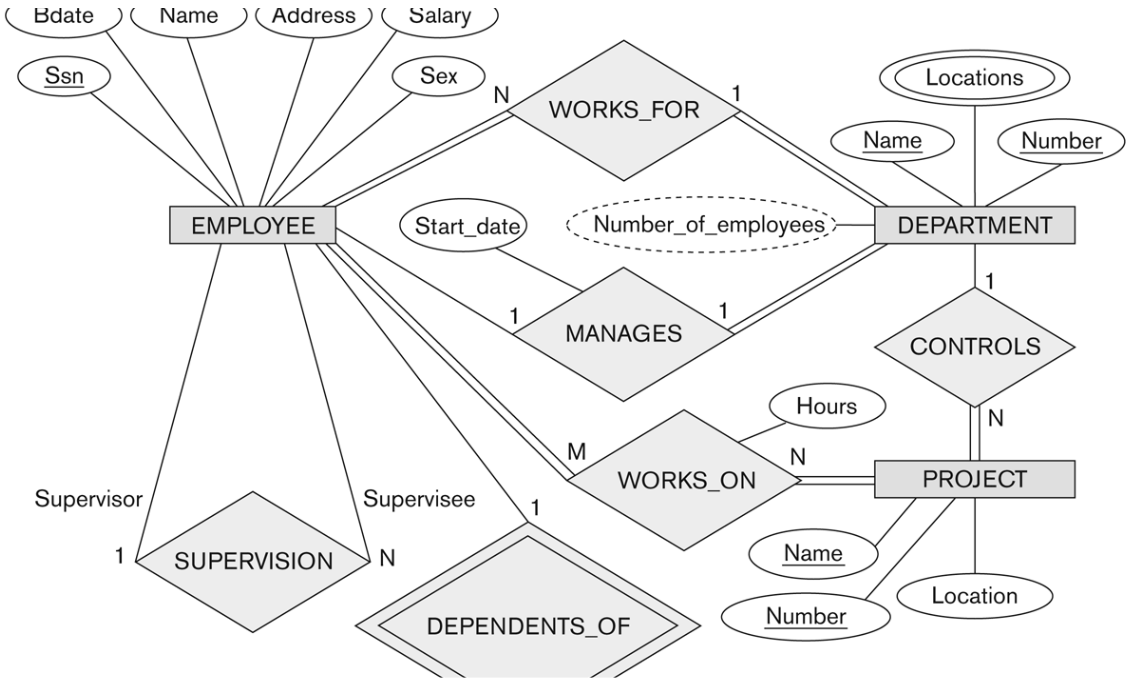
ER Model

By

Prof. Muhammad Iqbal Bhat

Government Degree College Beerwah





Topics

- ER Model
- ER Model Concepts
- ER Diagram Notations

Introduction:

The ER model is a conceptual model used in database design to represent entities, their attributes, and the relationships between them.

It was first introduced by Peter Chen in 1976 and has become one of the most widely used models in database design.

The ER model provides a visual representation of the data model, making it easier for designers to understand and communicate the database structure.

It helps to organize the data and define relationships between entities, ensuring that the data is stored in a normalized form, which eliminates redundancy and ensures data integrity.

The ER model is a crucial part of the database design process and helps to ensure that the database meets the needs of the organization and its users.

Database Design Process:

The database design process is a systematic approach to creating a database that meets the needs of the organization and its users.

Steps:

- Requirement gathering: identifying the data requirements of the organization and its users.
- Conceptual design: creating a high-level view of the data model using the ER model.
- Logical design: creating a detailed view of the data model, including tables, columns, relationships, and constraints.
- Physical design: defining the storage structures and access methods for the database.
- Implementation: building and populating the database.
- Testing and maintenance: ensuring that the database performs as expected and making necessary changes to improve performance or meet changing requirements.

Two Main Activities in Database Design Process

Database Design

- Database design: creating the schema for the database and ensuring that it is normalized and meets performance and scalability requirements.

Applications Design

- Application design: designing the user interface and application architecture that will interact with the database.

Overview of Database Design

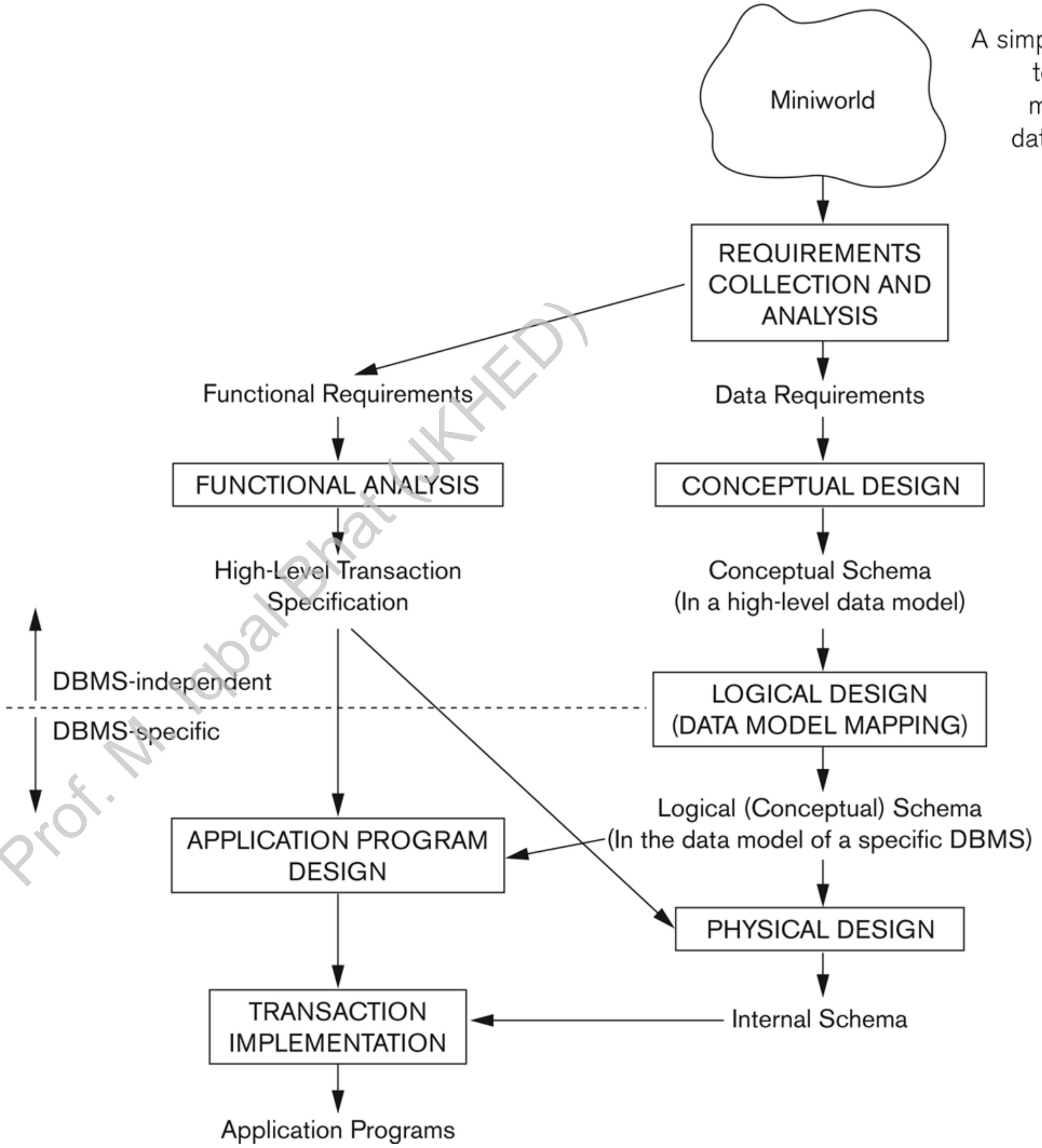

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Methodologies for Conceptional Design:

Methodologies for conceptual design provide a structured approach for creating a high-level view of the data model.

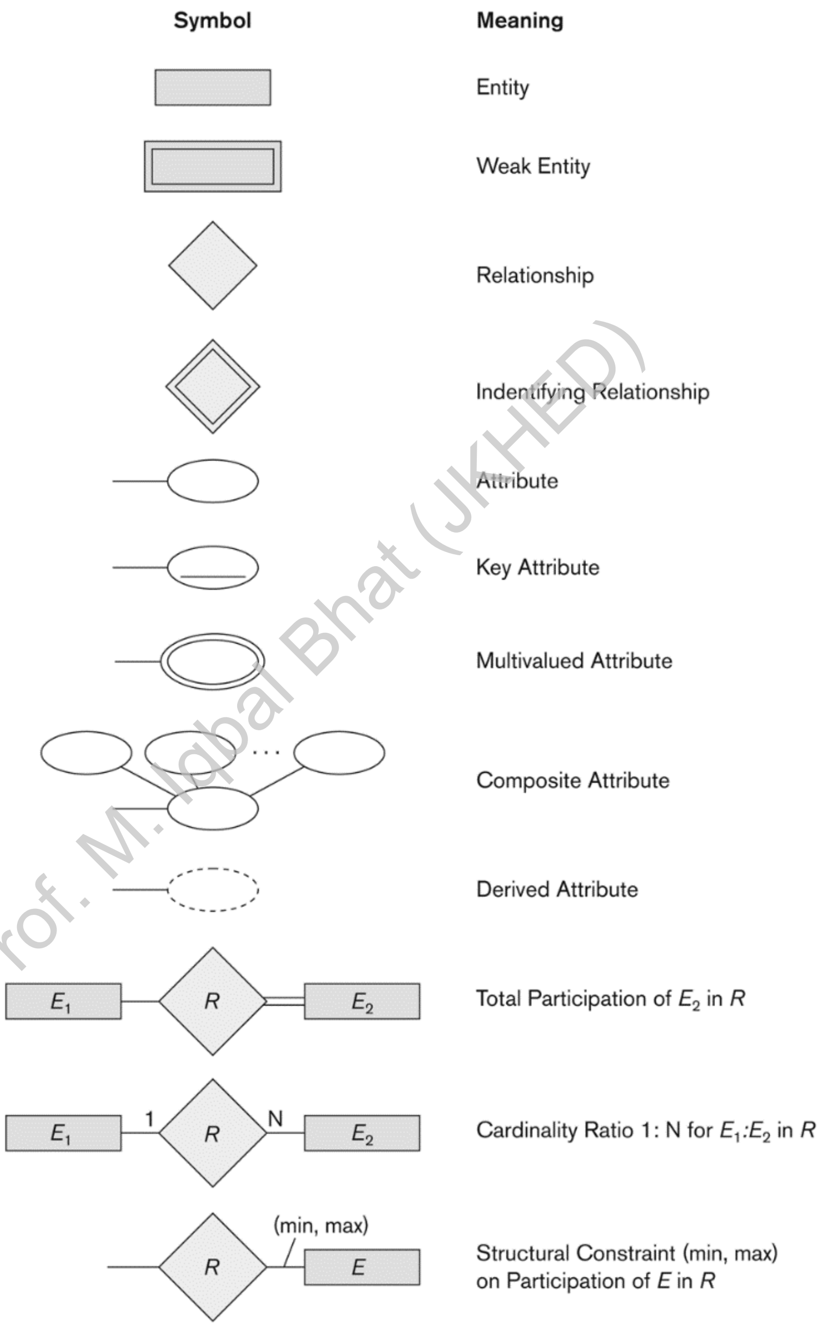
They help to ensure that the requirements of the organization and its users are captured accurately and efficiently.

Some common methodologies for conceptual design include:

- Object-Oriented Analysis (OOA)
- Information Engineering (IE)
- Entity-Relationship Approach (ERA)
- Unified Modeling Language (UML)

ER Model concepts

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

Total Participation of E_2 in R

Cardinality Ratio 1: N for E_1:E_2 in R

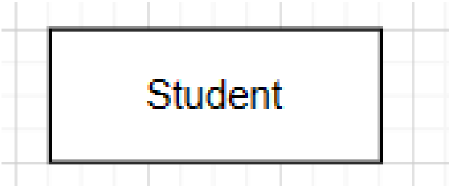
Structural Constraint (min, max) on Participation of E in R

Entity

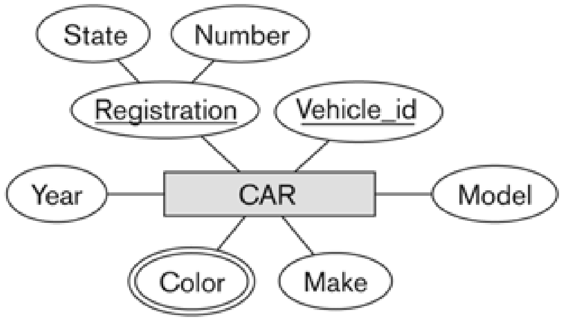
Entities are objects or concepts that exist independently and can be identified uniquely.

Examples of entities include customers, employees, products, and orders.

In the ER model, entities are represented by rectangles.



Attributes



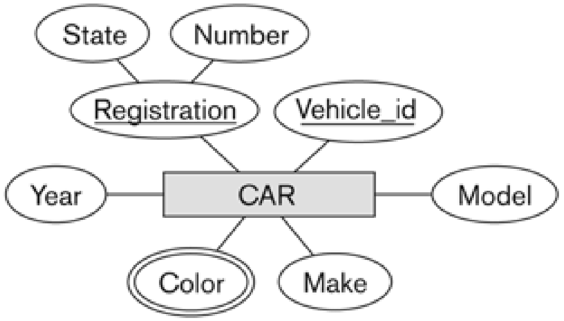
Attributes are characteristics or properties that describe an entity.

For example, a customer entity may have attributes such as name, address, and phone number.

In the ER model, attributes are represented by ovals connected to the entity rectangle.

Each attribute has a *value set* (or data type) associated with it – e.g. integer, string, date, enumerated type, …

Types of Attributes



Simple

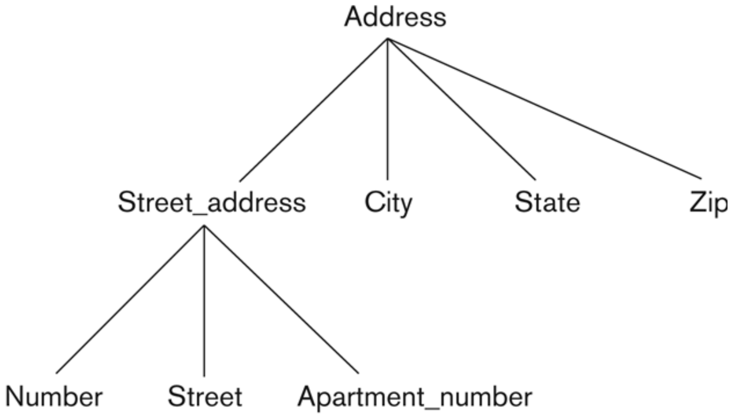
- Each entity has a single atomic value for the attribute. For example, SSN or Sex.

Composite

- The attribute may be composed of several components. For example:
  - Address(Apt#, House#, Street, City, State, ZipCode, Country), or
  - Name(FirstName, MiddleName, LastName).
  - Composition may form a hierarchy where some components are themselves composite.

Multi-valued

- An entity may have multiple values for that attribute. For example, Color of a CAR or Previous Degrees of a STUDENT.
  - Denoted as {Color} or {PreviousDegrees}.



Entity types and Key Attributes:

Entities with the same basic attributes are grouped or typed into an entity type.

- For example, the entity type EMPLOYEE and PROJECT.

An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type.

- For example, SSN of EMPLOYEE.

A key attribute may be composite.

- VehicleTagNumber is a key of the CAR entity type with components (Number, State).

An entity type may have more than one key.

- The CAR entity type may have two keys:
  - VehicleIdentificationNumber (popularly called VIN)
  - VehicleTagNumber (Number, State), aka license plate number.

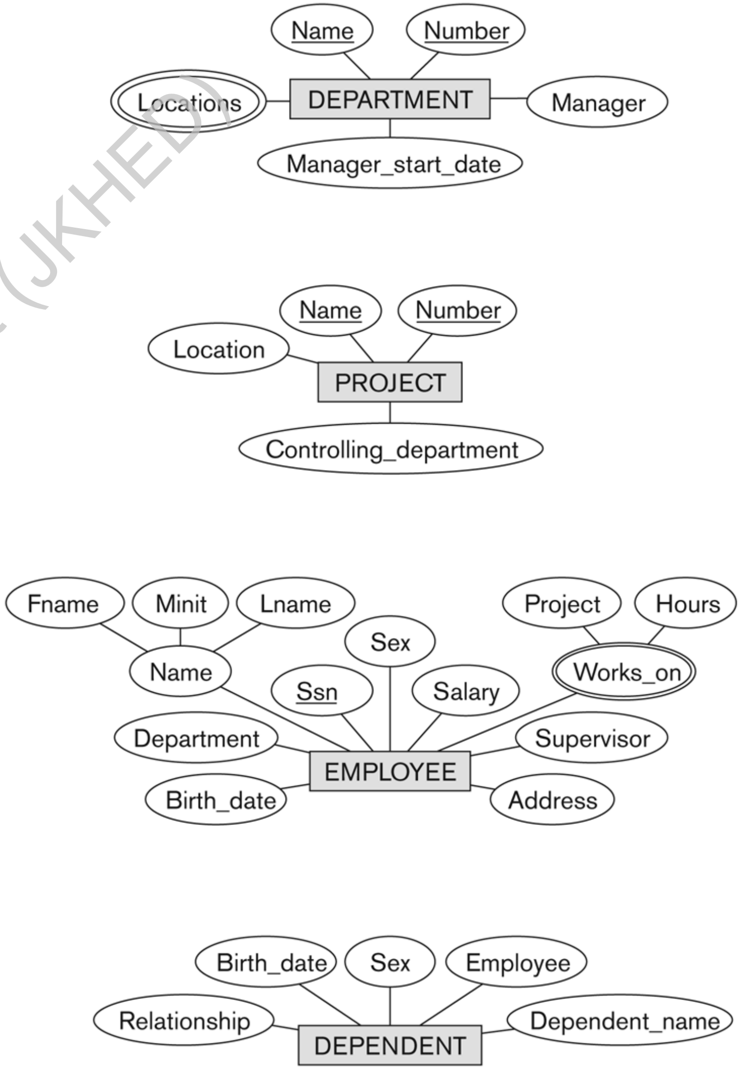
Each key is underlined (Note: this is different from the relational schema where only one “primary key is underlined).

Example COMPANY Database

- We need to create a database schema design based on the following (simplified) **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.

Example COMPANY Database

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
    - DEPARTMENT
    - PROJECT
    - EMPLOYEE
    - DEPENDENT



Refining the initial design by introducing **relationships**

The initial design is typically not complete

Some aspects in the requirements will be represented as **relationships**

ER model has three main concepts:

- Entities (and their entity types and entity sets)
- Attributes (simple, composite, multivalued)
- Relationships (and their relationship types and relationship sets)

Relationships and Relationship Types

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

Relationship instances of the WORKS_FOR N:1 relationship between EMPLOYEE and DEPARTMENT

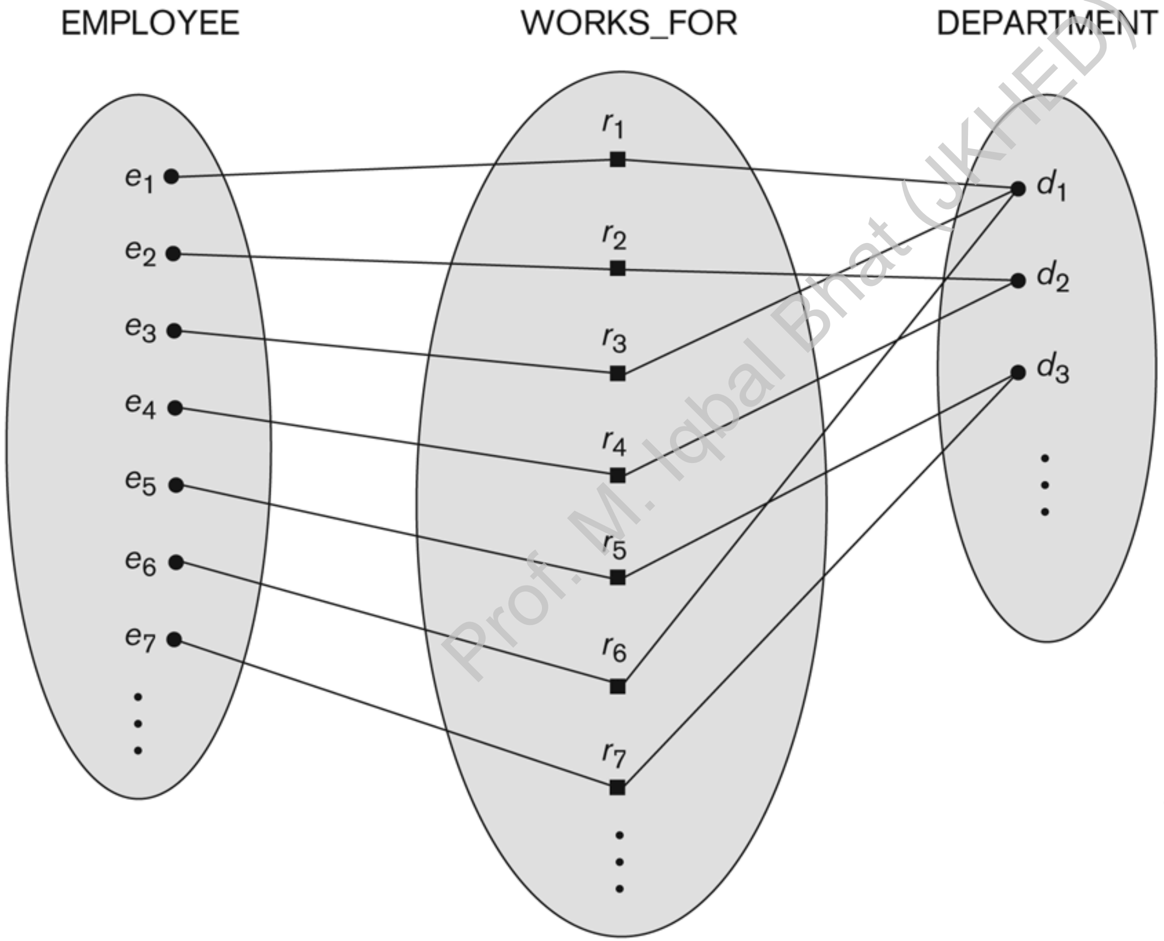


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship instances of the M:N WORKS_ON relationship between EMPLOYEE and PROJECT

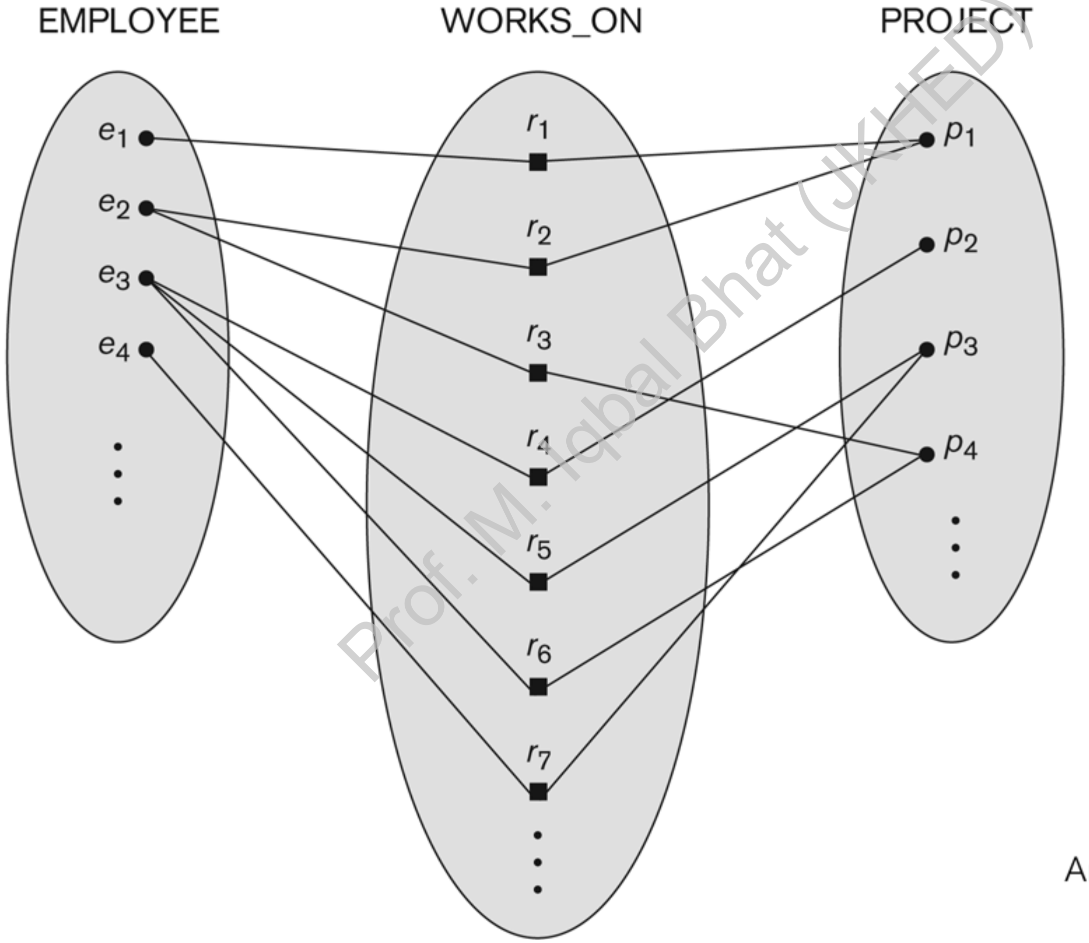
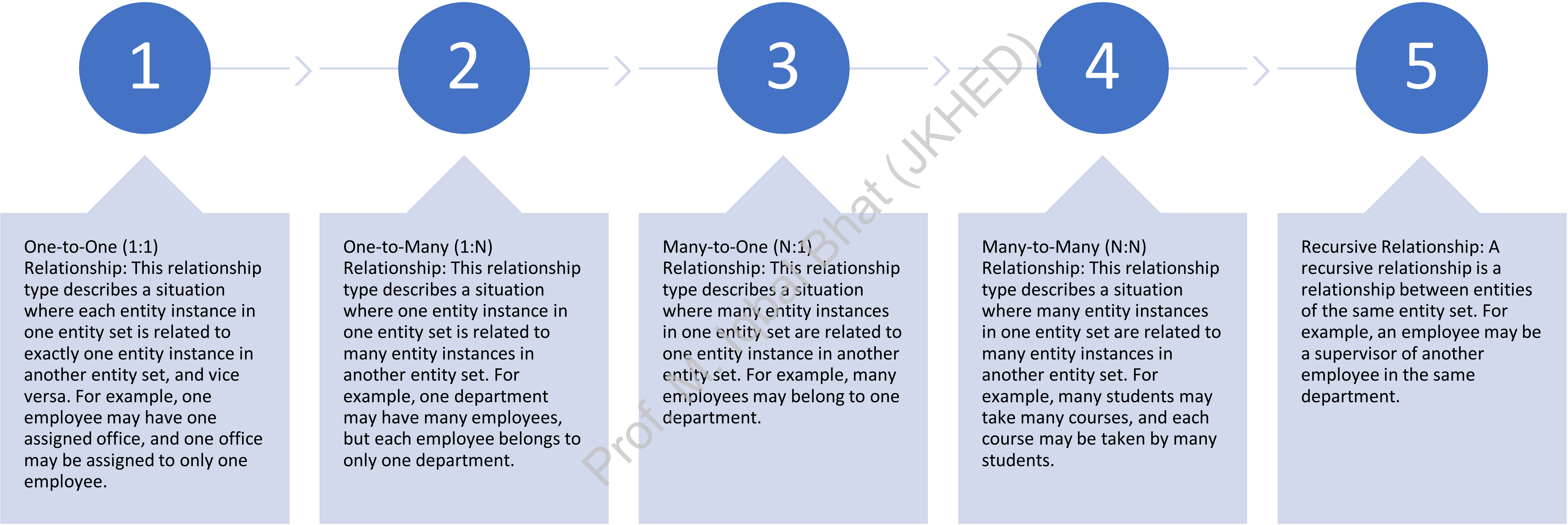


Figure 3.13
An M:N relationship, WORKS_ON.

Relationship Types:



One-to-One (1:1) Relationship: This relationship type describes a situation where each entity instance in one entity set is related to exactly one entity instance in another entity set, and vice versa. For example, one employee may have one assigned office, and one office may be assigned to only one employee.

One-to-Many (1:N) Relationship: This relationship type describes a situation where one entity instance in one entity set is related to many entity instances in another entity set. For example, one department may have many employees, but each employee belongs to only one department.

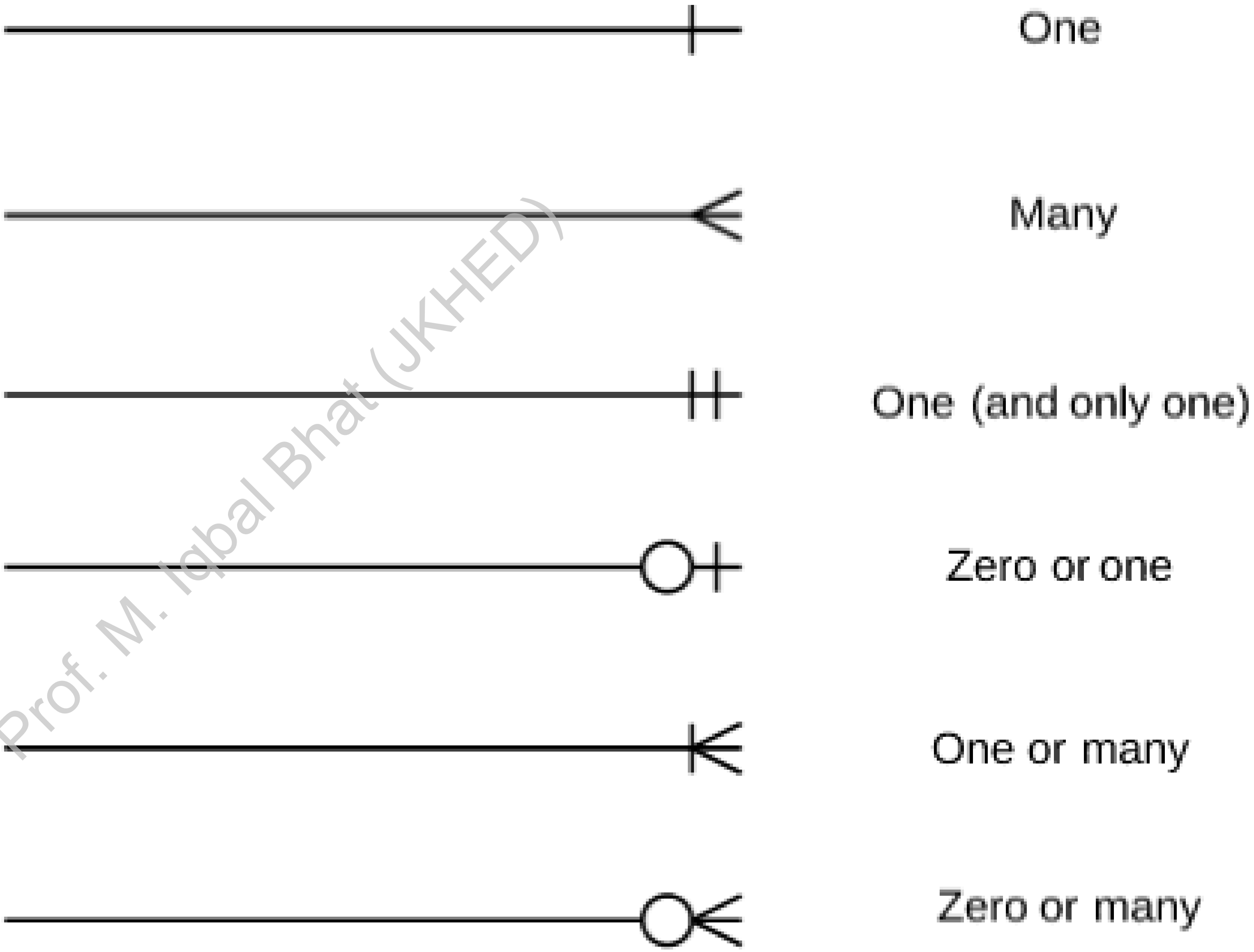
Many-to-One (N:1) Relationship: This relationship type describes a situation where many entity instances in one entity set are related to one entity instance in another entity set. For example, many employees may belong to one department.

Many-to-Many (N:N) Relationship: This relationship type describes a situation where many entity instances in one entity set are related to many entity instances in another entity set. For example, many students may take many courses, and each course may be taken by many students.

Recursive Relationship: A recursive relationship is a relationship between entities of the same entity set. For example, an employee may be a supervisor of another employee in the same department.

Cardinality: It describes how many instances of one entity are related to a certain number of instances of another entity. In other words, it specifies the maximum number of times an instance of one entity can be associated with instances of another entity.

Diagrams



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)

Full Diagram

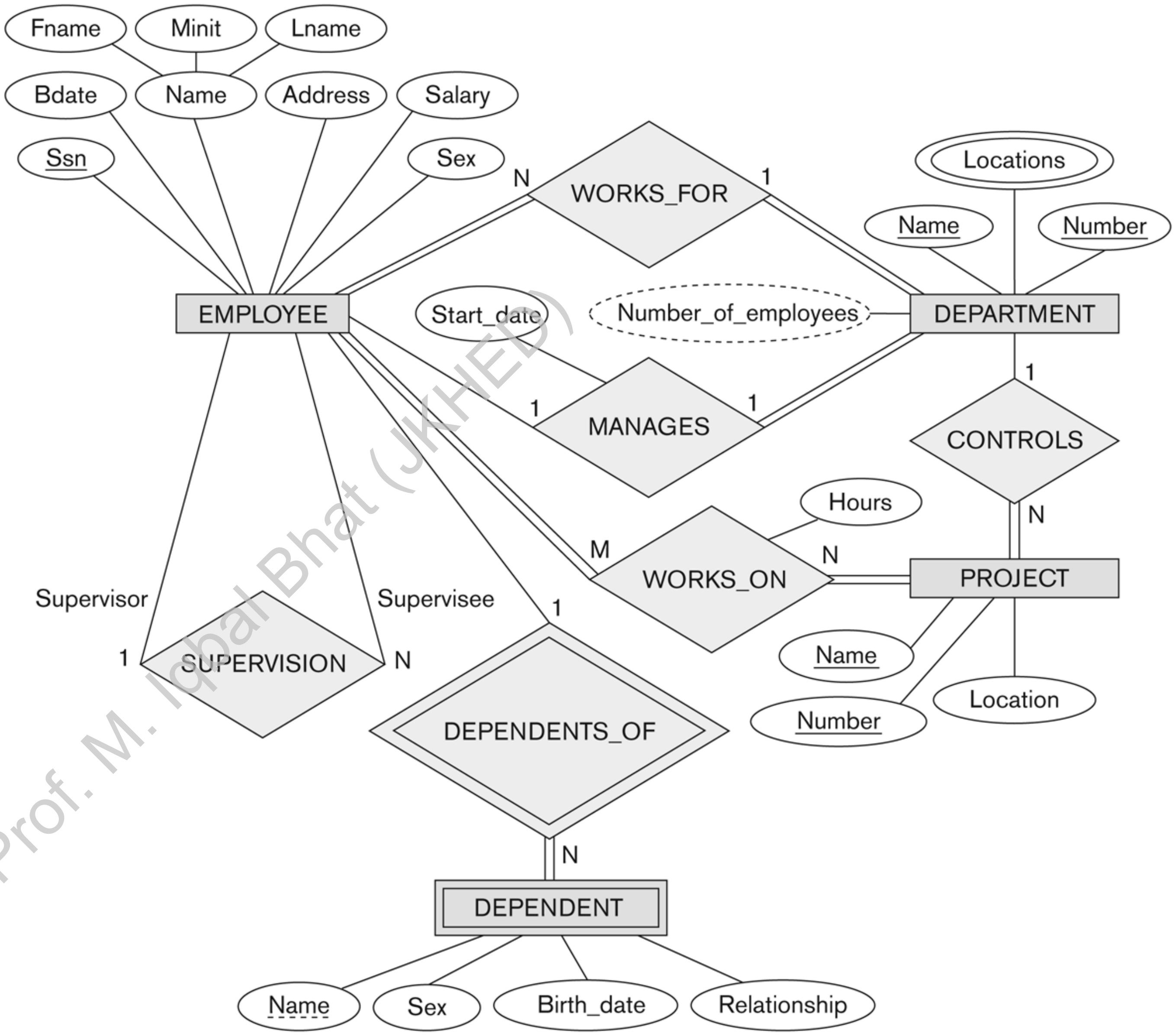


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Weak Entity Types

- An entity that does not have a key attribute and that is identification-dependent on another entity type.
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying relationship type
- **Example:**
  - A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
  - Name of DEPENDENT is the *partial key*
  - DEPENDENT is a *weak entity type*
  - EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

Full Diagram

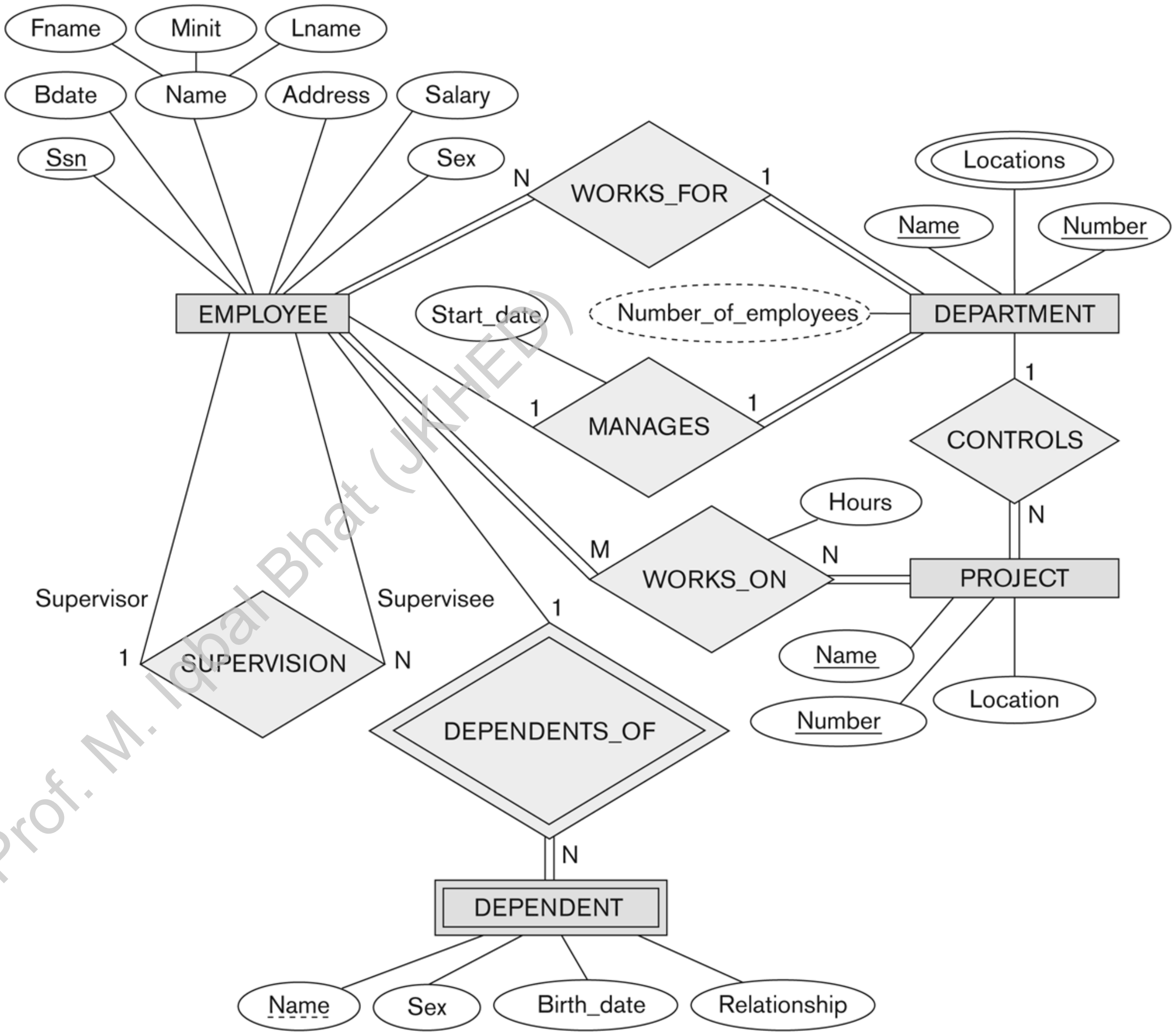


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.