

# ENTITY RELATIONSHIP DIAGRAMS

Week 5

# OBJECTIVES

- How to use Entity–Relationship (ER) modeling in database design.
- Basic concepts associated with ER model.
- Introduce the diagrammatic technique for displaying an ER schema
- Show how a schema design is refined to include relationships.

# DATABASE DESIGN

Previously we looked at how to create and query a database...

## ➤ **BUT HOW DO WE DESIGN ONE?**

- Need to consider - What tables, keys, and constraints are needed?
- What is the database going to be used for?

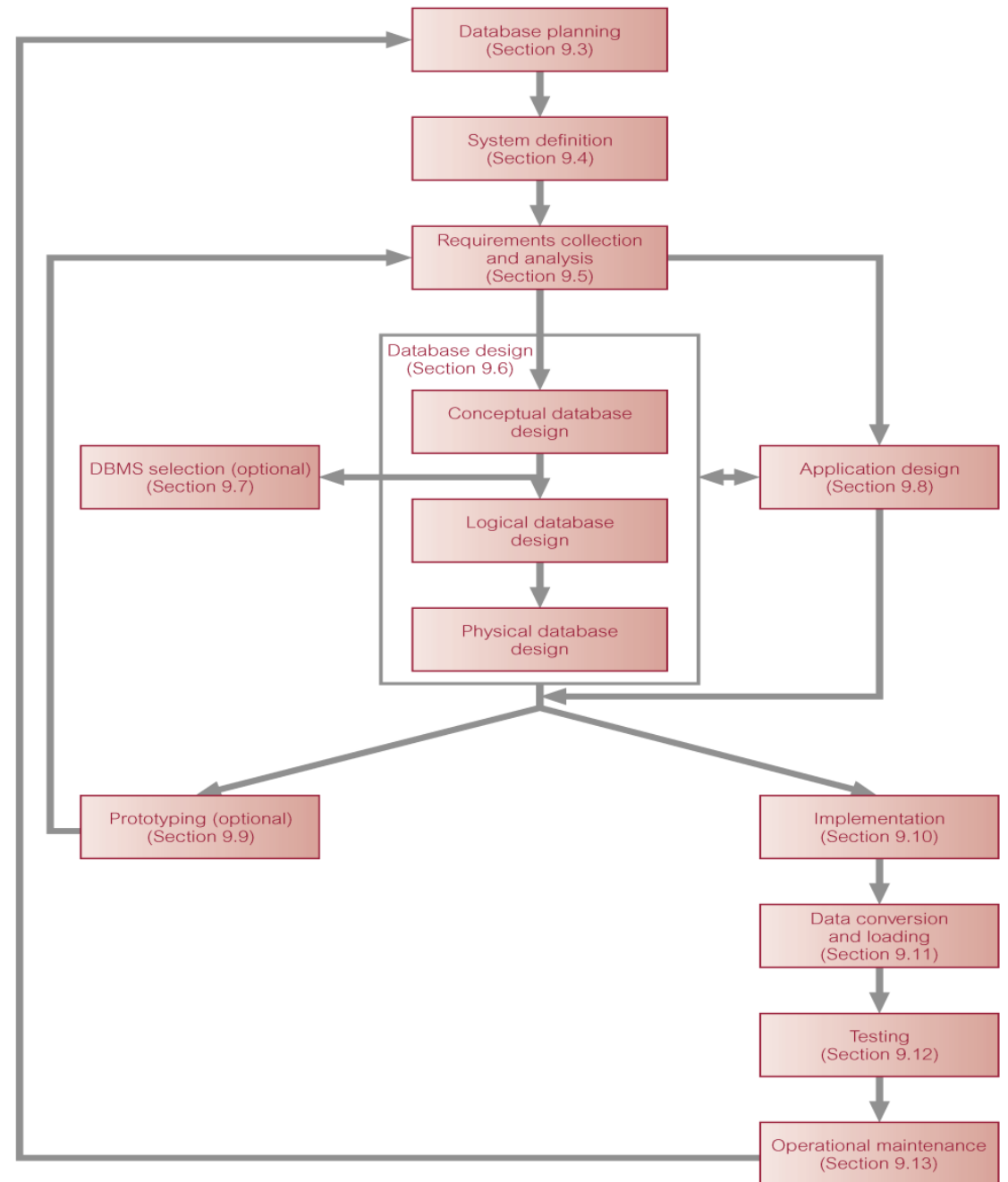
## **Methodology**

- Gather Requirements
- **Conceptual design** - Build a model independent of the choice of DBMS
- **Logical design** - Create the database in a given DBMS
- **Physical design** - How the database is stored in hardware

# OVERVIEW: INFORMATION SYSTEM (IS)

- Resources that enable collection, management, control, and dissemination of information throughout an organization.
- Database is fundamental component of IS, and its development/usage should be viewed from perspective of the wider requirements of the organization.

# STAGES OF THE DATABASE SYSTEM DEVELOPMENT LIFECYCLE



# DATABASE PLANNING AND DESIGN

- Conceptual modeling is a very important phase in designing a successful database application.
- **Entity–relationship (ER) model is a popular high-level conceptual data model.**
- Here we present the diagrammatic notation associated with the ER model, known as ER diagrams.

# ENTITY RELATIONSHIP MODELLING

- An E-R model is a particular type of data model suited to designing relational databases.
- The main component of the model is the Entity-Relationship Diagram.
- The E-R diagram is a simple way of representing the data entities being modelled and the relationships between these data entities.
- **It is easy to transform E-R diagrams to the Relational Model (data entities correspond to relations and relationships correspond to the implied associations created by keys and foreign keys of relations).**

# ENTITY RELATIONSHIP DIAGRAMS

- **A graphical representation of the entire database system**
- Used as a **design and planning tool**
- Provides a high-level conceptual data model supporting the user's perception of the data
- DBMS and hardware independent
- Composed of entities, attributes and relationships



# ER MODELS

## ➤ **Entity**

- An entity is any object in the system that we want to model and store information about
- They are objects or items of interest

## ➤ **Attributes**

- An attribute is a property of an entity

## ➤ **Relationship**

- Links between entities

➤ In a University Database we might have:

➤ **Entity** for Students, Modules and Lecturers

➤ Students **attributes** such as: StudentID, Name, Course

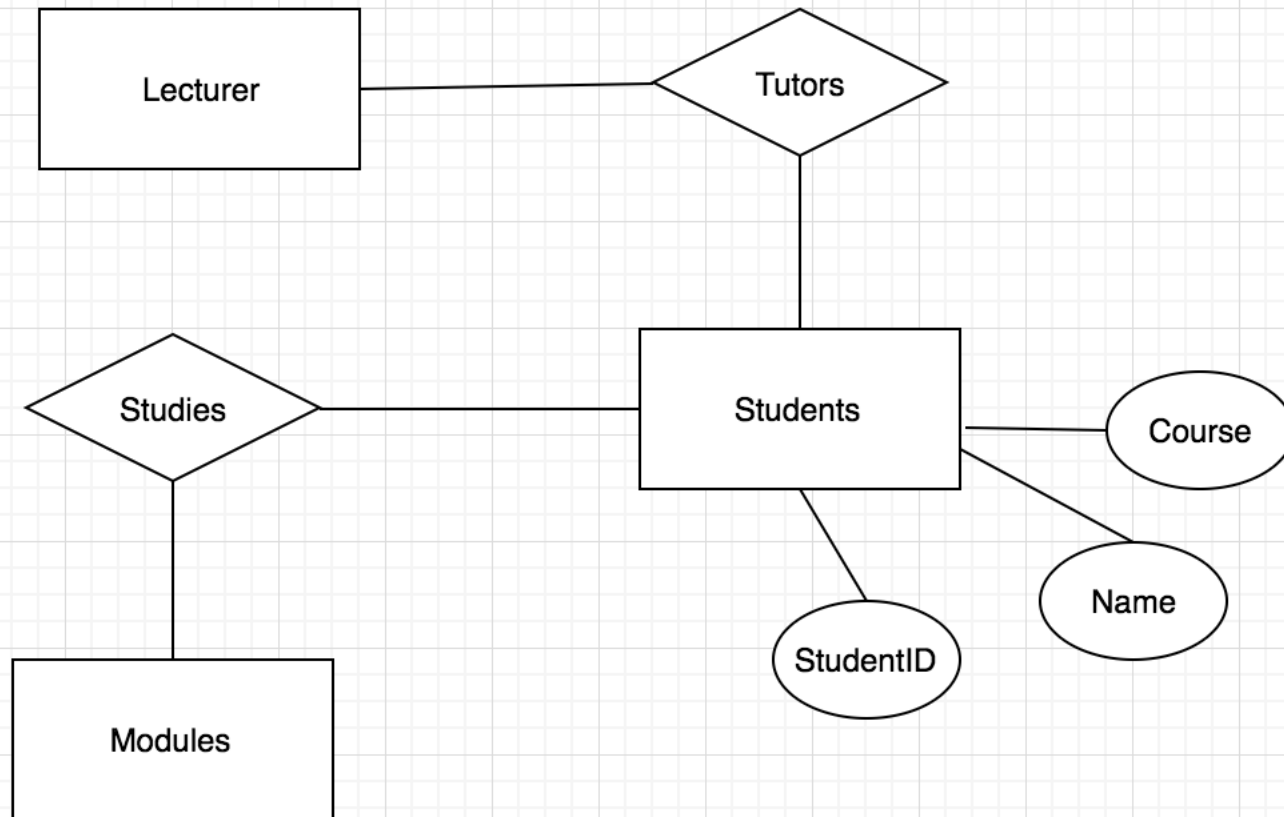
➤ **Relationships** with Modules (enrolment) and Lecturers (tutor)

# TYPES OF ERD (ENTITY RELATIONSHIP DIAGRAMS)

## ➤ **Numerous styles exist:**

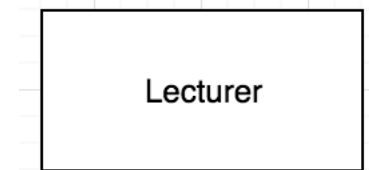
- Chen model (named after the originator of ER modeling, Dr. Peter P.S. Chen)
  - Information Engineering (IE, or “crows feet”)
  - Unified Modeling Language (UML)
  - Etc..
- The differences between the styles are only on how the entities and their relationships are illustrated
  - When you’re familiar with one style, it’s straight-forward to learn another
  - This module focuses on Chen’s ERD style

# ERD EXAMPLE



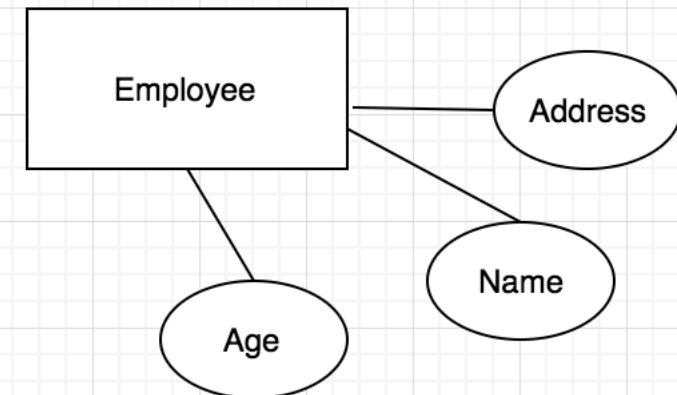
# ENTITIES

- **Entity is a thing or object in the real world** with an independent existence. An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course)
- Entities are generally **nouns** (a word that functions as the name of some specific thing or set of things, such as living creatures, objects, places, actions, qualities, states of existence, or ideas.)
- **Entities** are represented by rectangles in Chen's ERD's



# ATTRIBUTES

- Each entity has **attributes—the particular properties that describe it.**
- For example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job.
- The name of the **attribute is written in the oval**
- Attributes can be simple or composite (next slide for more on this)
- In Chen's model, they appear inside ovals and are attached to their entity
- Note that entity types can have a large number of attributes



# ATTRIBUTE TYPES

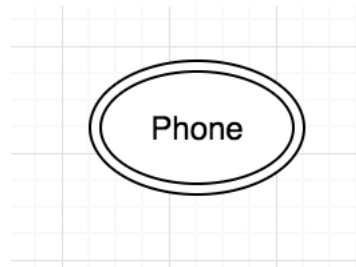
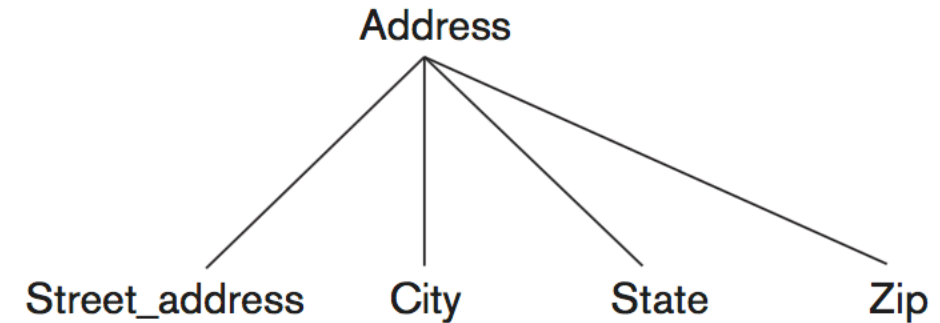
➤ Properties of entities that describe their characteristics

➤ Types:

➤ **Simple:** Attribute that is not divisible e.g. age

➤ **Composite:** Attribute composed of several simple attributes e.g. **address (as shown above)**

➤ **Multiple** (or multi-value attribute) have a set of values for the same entity—for instance, 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 any college degrees, 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. Another example is phone numbers (home phone, mobile)



# ATTRIBUTE TYPES (CONTINUED...)

➤ Properties of entities that describe their characteristics

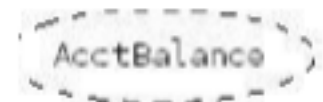
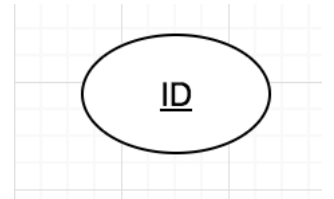
➤ Types:

➤ **Key:** Uniquely identifies the entity. The name of each primary key attribute is underlined. e.g. ID, PPSN, Chassis number

➤ **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 attribute example: Attribute whose values are generated from other attribute:

➤ E.g.  $\text{AcctBalance} = \text{TotalCredit} - \text{TotalDebit}$



# RELATIONSHIPS

➤ Relationships are an association between two or more entities

➤ Examples:

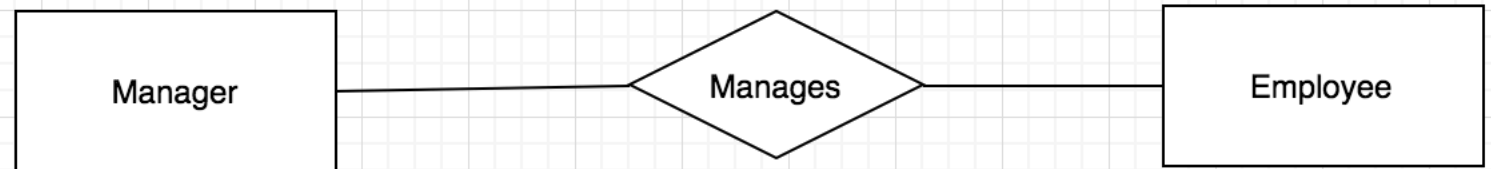
- Each student takes several modules
- Each Module is taught by a lecturer
- Each employee works for a single department
- Each manager manages a employee

➤ Relationships have:

- A name
- A set of entities that participate in them
- A degree (the number of entities that participate (most have a degree of 2))
- A **cardinality ratio** e.g. DEPARTMENT:EMPLOYEE is of cardinality ratio **1:N**, meaning that each department can be related to (that is, employs) **any number of employees (N)**, but an employee can be related to (work for) at most one department
  - N stands for any number of related entities (zero or more).

➤ **Relationships** are represented by diamonds

➤ A relationship captures how entities are related to one another. Relationships can be thought of as **verbs**, linking two or more nouns. Examples: an owns relationship between a company and a computer, a supervises relationship between an employee and a department, a performs relationship between an artist and a song,





# DEGREE OF A RELATIONSHIP

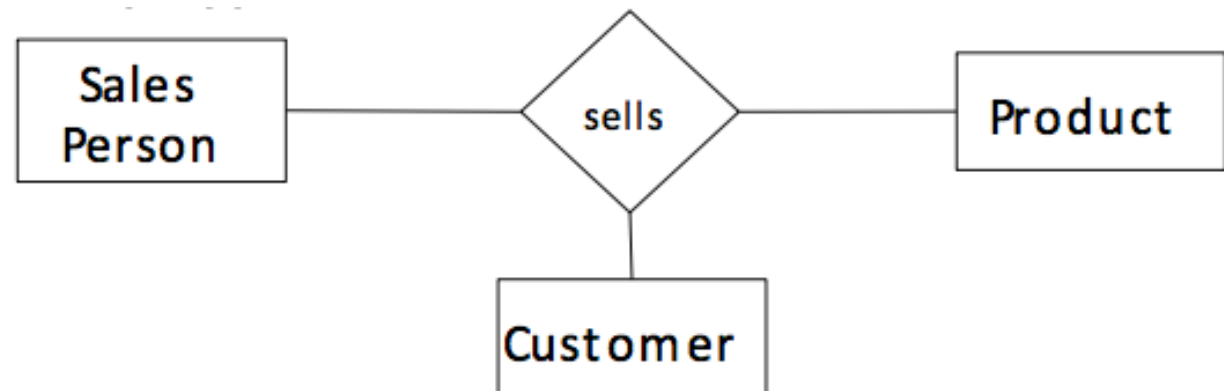
- The number of participating entities in a relationship is known as the degree of the relationship
- If there are two entity types involved, it's a binary relationship type

If there are three entity types involved, it's a ternary relationship type

Binary - involve 2 entities



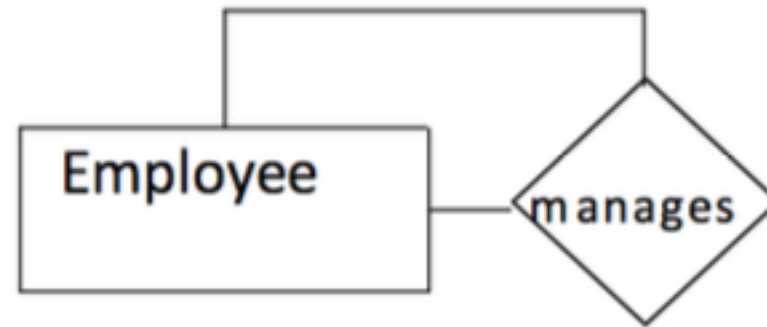
Ternary - involve 3 entities



# DEGREE OF A RELATIONSHIP CONTINUED...

- Unary relationships are also known as **recursive relationships**. It's a relationships where the same entity participates more than once in different roles
- In this example, we are saying that employees are managed by other employees

Unary- involve 1 entity



# CARDINALITY RATIOS

- Typically relationships are an association between two or more entities
- Each entity in a relationship can participate in zero, one, or more than one instances of that relationship
- There are three basic types of relationships that you may encounter:
  - one-to-one (1:1),
  - one-to-many (1:M), and
  - many-to-many (M:N or M:M).

# CARDINALITY RATIOS CONTINUED...

## ➤ one-to-one (1:1)

- E.g. Each lecturer has a unique office
- E.g. an employee can manage at most one department and a department can have at most one manager.
- E.g. The airport is located in one and only one town, and the town has one and only one airport.”

## ➤ one-to-many (1:M)

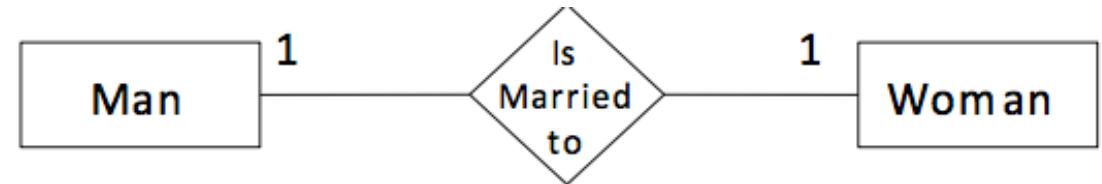
- E.g. A lecturer may tutor many students, but each student has just one tutor
- E.g. A painter can paint many paintings, each painting is painted by one painter
- E.g. A customer places many orders and an order is placed by one customer. A “1” indicates that an order comes from one customer. **The “M” (or an “N”) indicates that a customer can place many orders.**

## ➤ many-to-many (M:N or M:M)

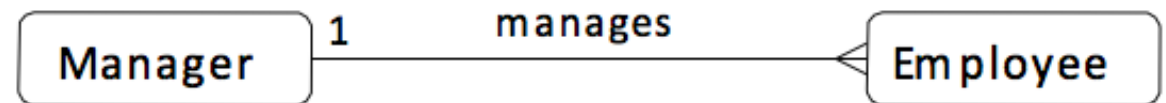
- Each student takes several modules and each module is taken by several students

# CARDINALITY EXAMPLES (WITH DIAGRAMS)

➤ A one to one relationship- a man can only marry one woman, and a woman can only marry one man, so it is a one to one (1:1) relationship



➤ A one to many relationship - one manager manages many employees, but each employee only has one manager, so it is a one to many (1:m) relationship- (Crows foot notation)

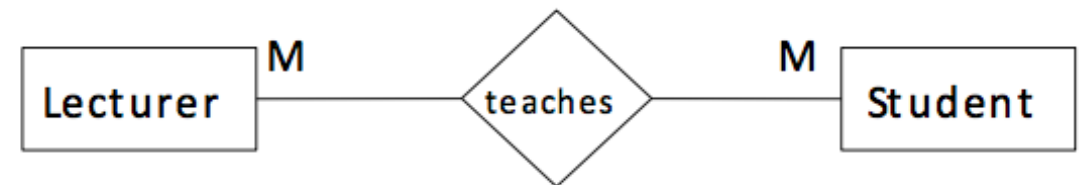


# CARDINALITY EXAMPLES (WITH DIAGRAMS)

- A many to one relationship - many students study one course. They do not study more than one course, so it is a many to one (m:1) relationship



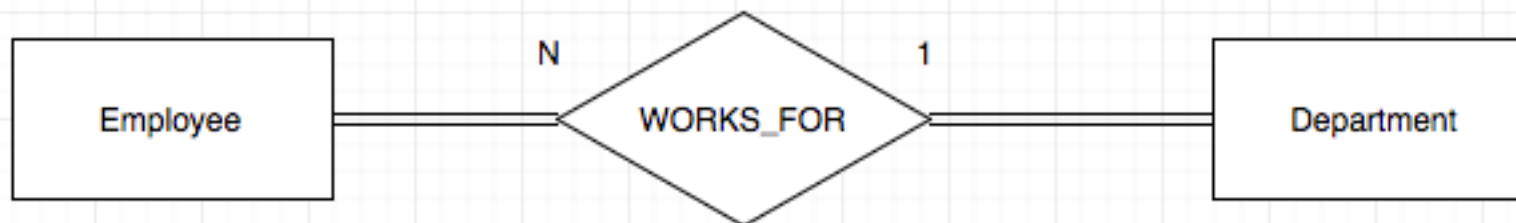
- A many to many relationship - One lecturer teaches many students and a student is taught by many lecturers, so it is a many to many (m:n) relationship



- Note: Remember that, when we are specifying data relationships, we are indicating possible relationships, and not necessarily requiring that all instances of all entities participate in every documented relationship.

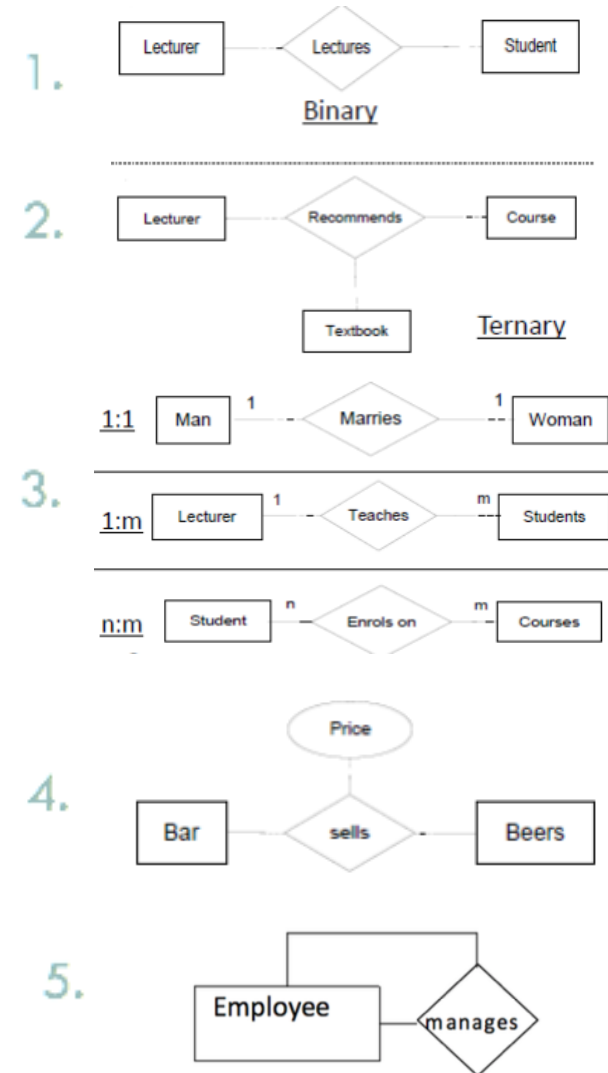
# CARDINALITY RATIOS FOR BINARY RELATIONSHIPS

- The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in.
- For example, **DEPARTMENT:EMPLOYEE** is typically cardinality ratio **1:N**, meaning that each department can be related to (that is, employs) any number of employees (**N**), (**N stands for any number of related entities (zero or more)**) but an employee can be related to (work for) at most one department (**1**). This means that for this particular relationship type **WORKS\_FOR**, a particular department entity can be related to any number of employees (**N indicates there is no maximum number**).



# RELATIONSHIPS SUMMARY

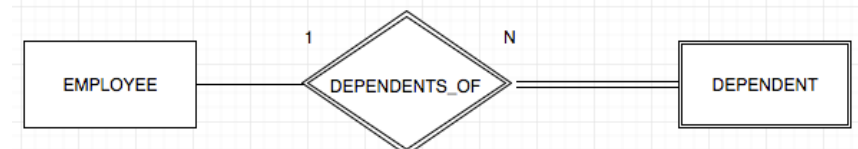
1. Are **bi-directional** (ie can be put 2 ways)
2. **Degree:** binary (i.e. involve only two entities), ternary (i.e. involve three participating entities).
3. **Cardinality:** Entity types may be linked in more than one way.
4. May have **properties** (attributes)
5. Can be **Recursive**.





# MORE ER DEFINITIONS — WEAK 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.
- Another example: A weak entity is one that can only exist when owned by another one. For example: a ROOM can only exist in a BUILDING.
- In ER diagrams a weak entity set is indicated by a bold (or double-lined) rectangle (the entity) connected by a bold (or double-lined) type arrow to a bold (or double-lined) diamond (the relationship).
- Another Example: entity type DEPENDENT, related to EMPLOYEE, which is used to keep track of the dependents of each employee via a 1:N relationship

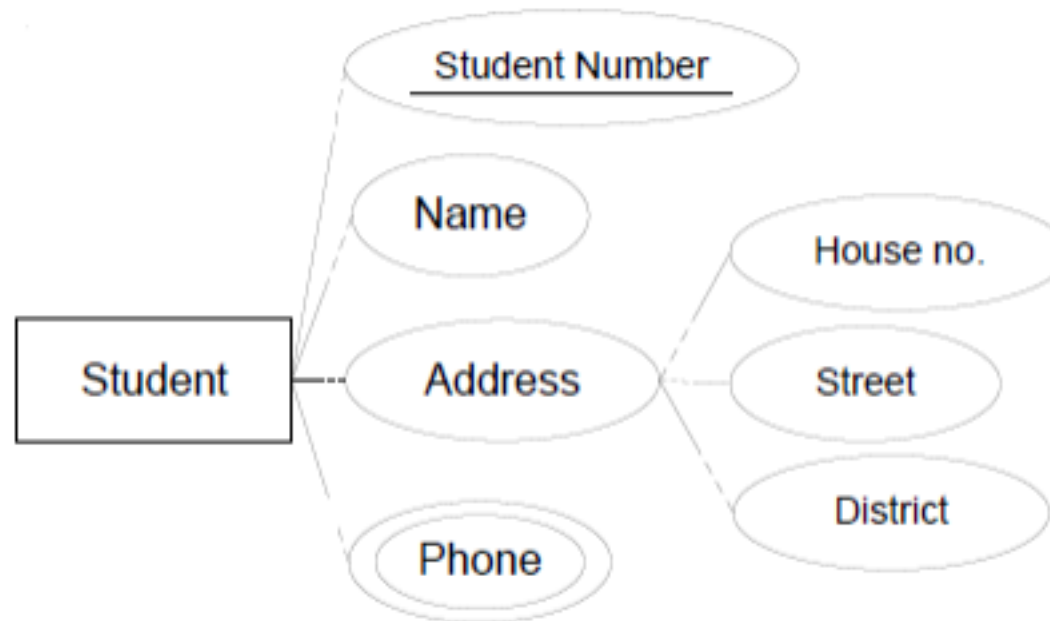


# WEAK ENTITY EXAMPLE


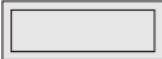
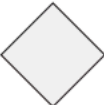




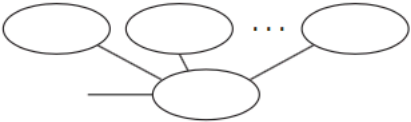

- In a typical database, customers and orders entities we can store data about a customer in its database before the customer places an order. Therefore, an instance of the customer entity does not have to be related to any instances of the order entity.
- However, the reverse is not true in this database. An order must be related to a customer. Without a customer, an order cannot exist. An order is, therefore, an example of a weak entity, one that cannot exist in the database, unless a related instance of another entity is present and related to it.
- An instance of the customer entity can be related to zero, one, or more orders. However, an instance of the order entity must be related to one and only one customer.
- 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. **The partial key attribute is underlined with a dashed or dotted line.**

# ER EXAMPLE 1: ENTITY WITH ATTRIBUTES

- “A student has a unique student number (identifying), a name, an address (with street number, street and district) and **several** phone numbers”



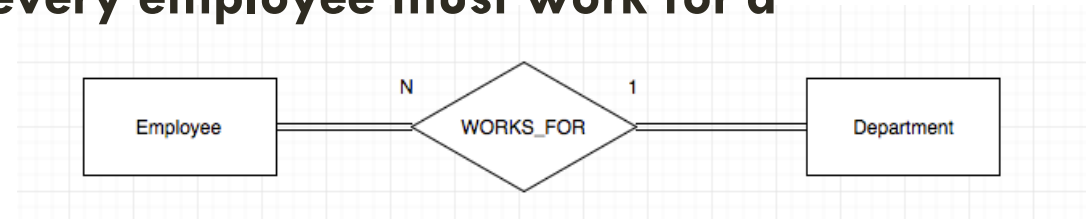
# SUMMARY OF THE NOTATION FOR ER DIAGRAMS

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute

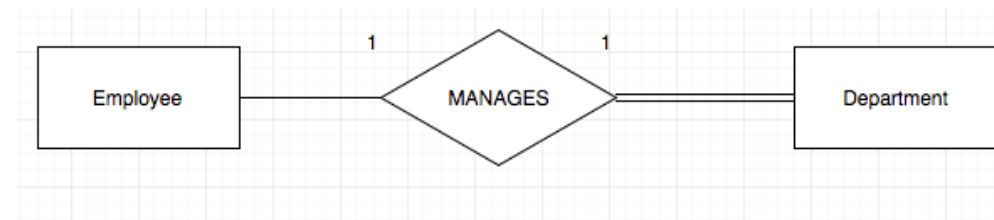
# TOTAL/PARTIAL PARTICIPATION

Two types of participation constraints—total and partial—that we illustrate by example.

**Total Participation:** If a company policy states that **every employee must work for a department**



**Partial Participation:** If 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.



Note: In ER diagrams, total participation is displayed as a double line connecting the participating entity type to the relationship, whereas partial participation is represented by a single line

# PROPER NAMING AND REFINING ER DESIGN

- When designing a database schema, the choice of names for entity types, attributes, relationship types, and (particularly) roles is **not** always straightforward.
- One should choose names that convey, as much as possible, the meanings attached to the different constructs in the schema.
- **The cardinality ratio and participation constraint of each relationship type are determined from the requirements. If some cardinality ratio or dependency cannot be determined from the requirements, the users must be questioned further to determine these structural constraints.**
- We choose to use **singular names for entity types**, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type.
- In our ER diagrams, we will use the convention that entity type and relationship type names are in uppercase letters, attribute names have their initial letter capitalized

# DESIGN CHOICES FOR ER CONCEPTUAL DESIGN

- It is occasionally difficult to decide whether a particular concept should be modeled as an entity type, an attribute, or a relationship type.
- In general, the schema design process should be considered **an iterative refinement process**, where an initial design is created and then iteratively refined until the most suitable design is reached.

# DRAW THE DATABASE ER DIAGRAM WITH THE FOLLOWING INFORMATION:

**The company you work for wants to digitize their time cards.**

You have been asked to design the database for submitting and approving time cards.

- A timecard should have hours worked and date submitted
- Each timecard is associated with exactly one employee
- Each timecard should have a unique id
- Each timecard has a status: it is either approved, not approved, or pending
- Each employee has a unique id
- Each employee has a name and address.
- Each employee submits a time card every pay period. i.e. In 1 year, they will submit multiple time cards
- Each employee either has direct deposit or physical cheque as their method of payment
- Each employee is associated with exactly one manager
- Each manager has a unique id and a name
- Each manager is in charge of multiple employees
- Each manager approves time cards for multiple employees



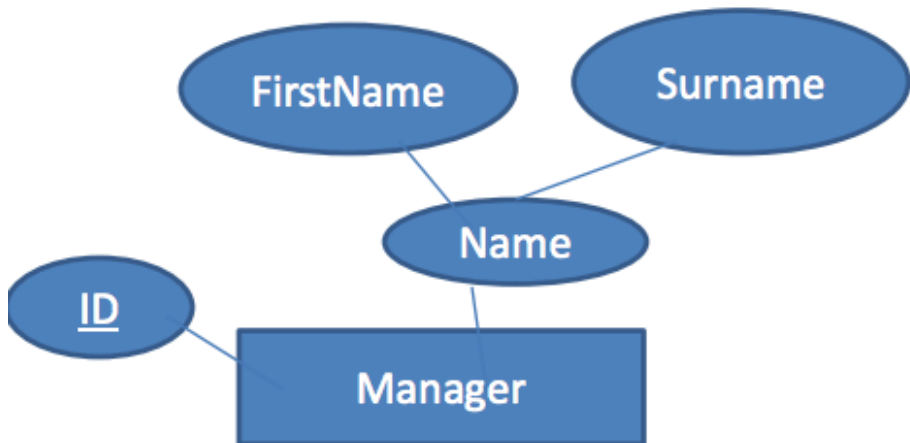
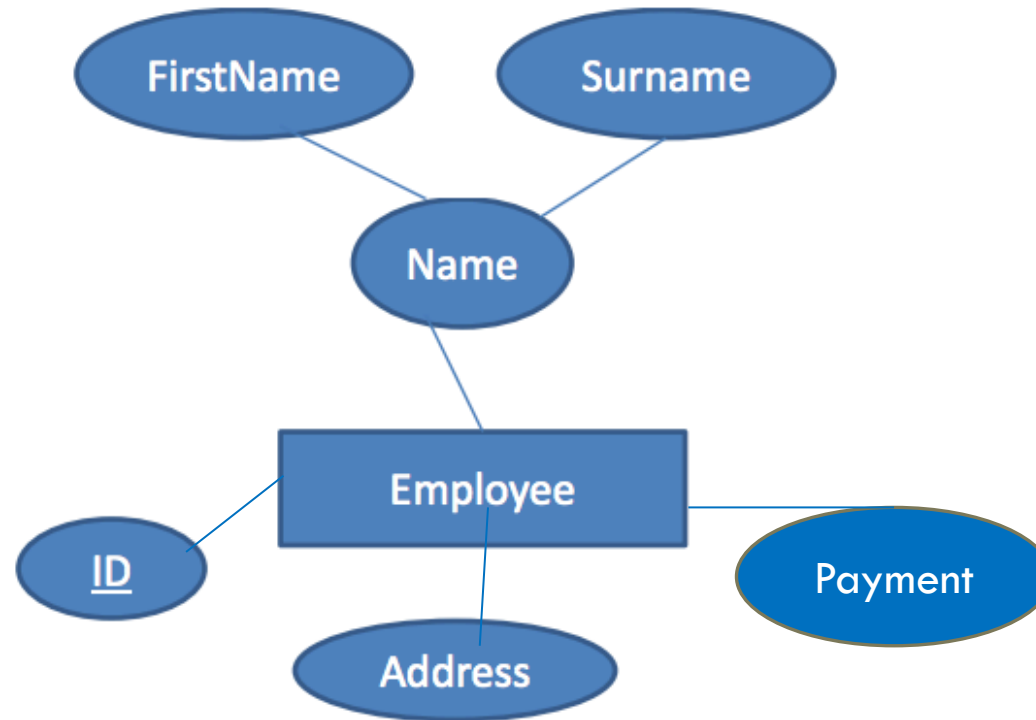
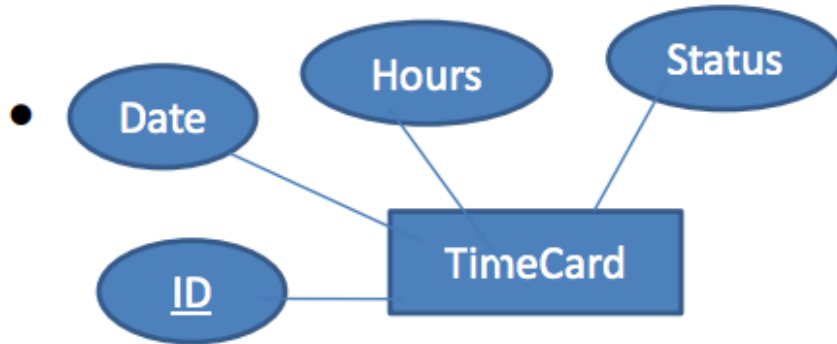
# WHAT ARE THE ENTITIES?

TimeCard

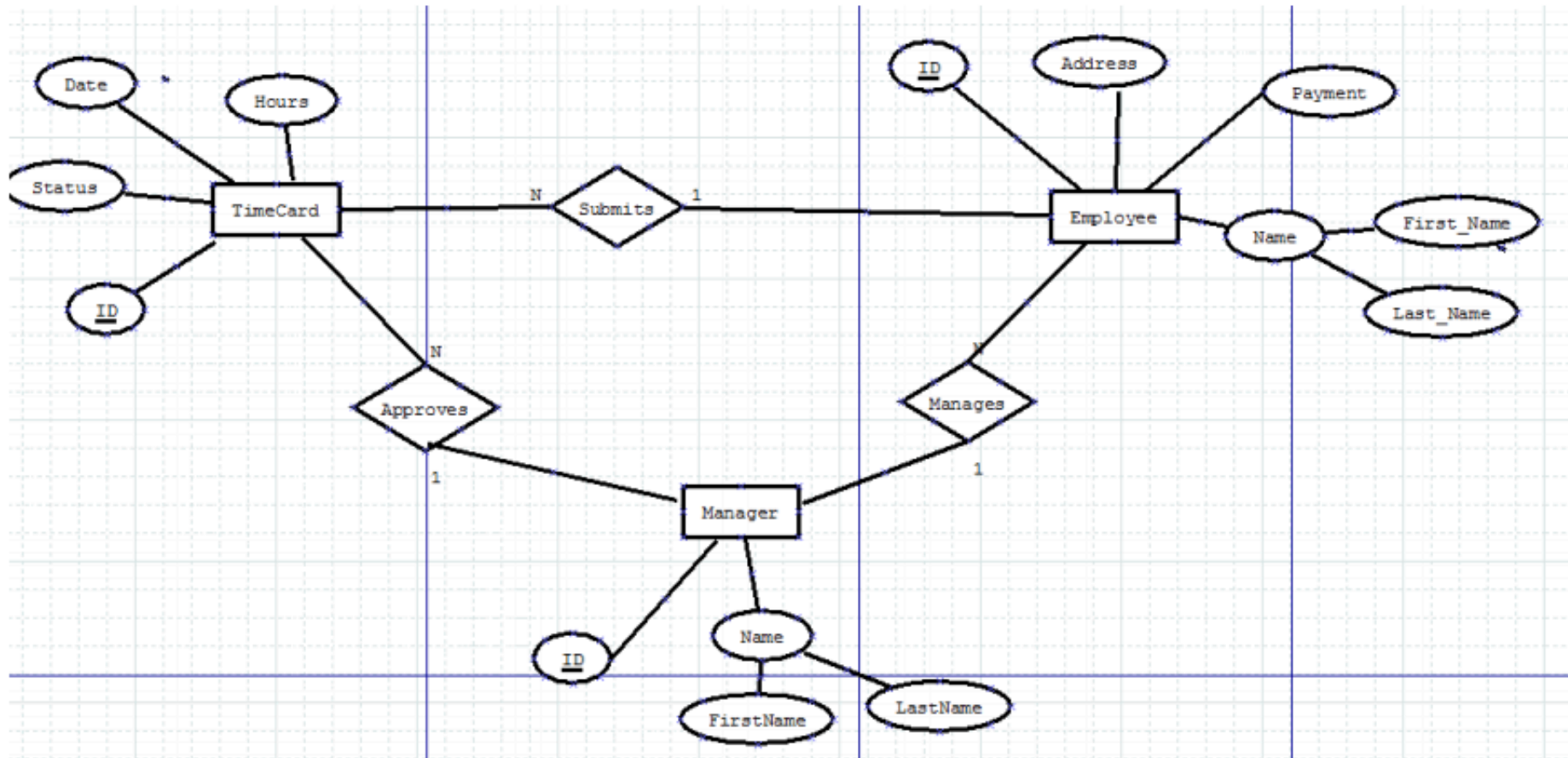
Manager

Employee

# WHAT ARE THE ATTRIBUTES?



# WHAT IS THE MAXIMUM CARDINALITY OF THE RELATIONSHIPS BETWEEN THE ENTITIES



# EXERCISE 1 - JANUARY PAPER 2013-2014

## QUESTION

### ➤ Requirements

- The database must store book, author, publisher and warehouse information.
- For every book you must capture the title, isbn, year and price information. The isbn value is unique for a book.
- For every author you must store an id, name, address and the URL of their homepage. Each author can write many books, and each book can have many authors, for example.
- For every publisher you must store an id, name, address, phone number and an URL of their website.
- Books are stored at several warehouses, each of which has a code, address and phone number.
- A book has only one publisher.
- The warehouse stocks many different books. A book may be stocked at multiple warehouses.
- The database records the number of copies of a book stocked at various warehouses.
- **Design an ER diagram for such a bookstore.** Your ER diagram must show entities, attributes and the relationships between entities. [Document any assumptions that you make] (16 marks)

# PREPARING YOUR SOLUTION

- Sketch out your initial ER diagram first out on paper (use a pencil and eraser if you need to change your diagram)
- Then use your browser to go to <https://draw.io>
- You will find most of the icons you are looking for in the **General** tab and also the **Misc** tabs to find the correct icons

# EXERCISE 2 - AUGUST 2014: EXAM QUESTION

- United Direct Artists (UDA) is an insurance broker that specialise in insuring paintings for galleries. You are required to design a database for this company.
- The database must store painters, paintings, and galleries information.
- Painters have a unique number, Name, and phone number
- Paintings have unique number, title and price
- Galleries have unique number, owner, phone number, commission rate and address
- A painting is painted by a particular artist, and that painting is exhibited in a particular gallery. A gallery can exhibit many paintings, but each painting can be exhibited in only one gallery. Similarly, a painting is painted by a single painter, but each painter can paint many paintings.

# EXERCISE 3 — MORE DETAILED EXAMPLE...

- Book Example: Fundamentals of Database Systems (Elmasri & Navathe)
- Database Application Question (called Company Database)

## Overview

- The company wishes to create a database to keep track of a company's employees, departments and projects.
- Suppose that after the requirements collection and analysis phase, the database designers provide the following description of the miniworld – the part of the company that will be represented in the database.

# COMPANY DATABASE-QUESTION

- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- We store each employee's name, Social Security number,<sup>2</sup> address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. We keep track of the current number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee (who is another employee).
- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, sex, birth date, and relationship to the employee.

REPRESENT THE ABOVE IN A ER DIAGRAM.