LECTURE 8:
DATABASE DESIGN

COMP2004J: Databases and Information Systems

Dr. Ruihai Dong (ruihai.dong@ucd.ie)
UCD School of Computer Science
Beijing-Dublin International College

Designing a database

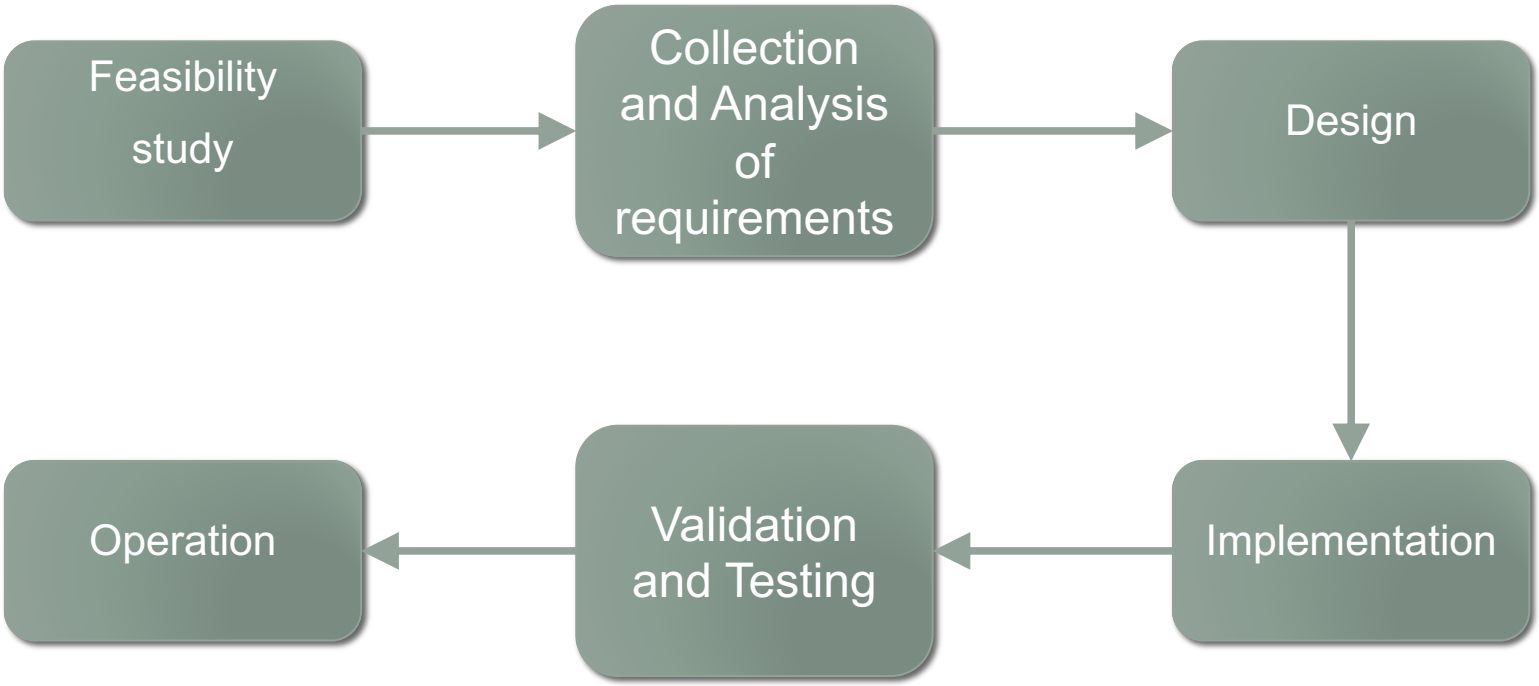
- An information system is a component of an organisation that **manages** (gets, processes, stores, communicates) the **information** of interest
  - usually, the information system operates in support to other components of the organisation.

- Database design is just one of the many activities in the development of an **information system** within an organisation.

- It should therefore be presented within the wider context of the **information system life cycle**.

INFORMATION SYSTEM LIFE CYCLE

Information System Life Cycle

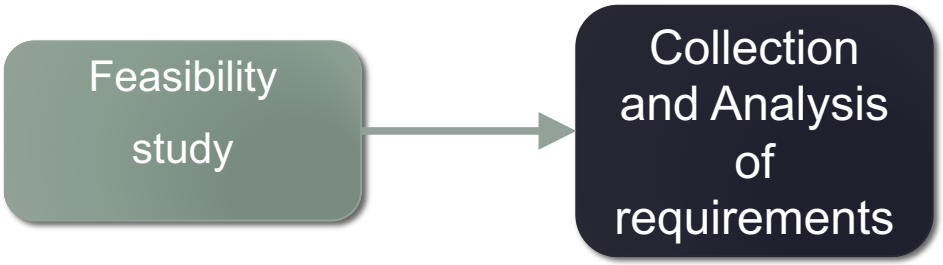


Information System Life Cycle

Feasibility study

Technical
Can hardware/software be found to implement the solution?
Time
Is the time scale acceptable?
Law
Can the problem be solved within the law?
Economic
Is the proposed solution possible within budget to run?
Cost Benefit
Will the solution improve efficiency and therefore be worth building
…

Information System Life Cycle



Collection and analysis of requirements is the definition and study of the properties and functionality of the information system.

- We work out what the system should be able to do.

Information System Life Cycle

Collection and Analysis of requirements

User Registration
- Username input field
- Password field
- Checkbox – Accept Terms and Conditions
- Submit Button
- Save User in Database

Login
- Username input field
- Password field
- Submit Button
- Read User from Database

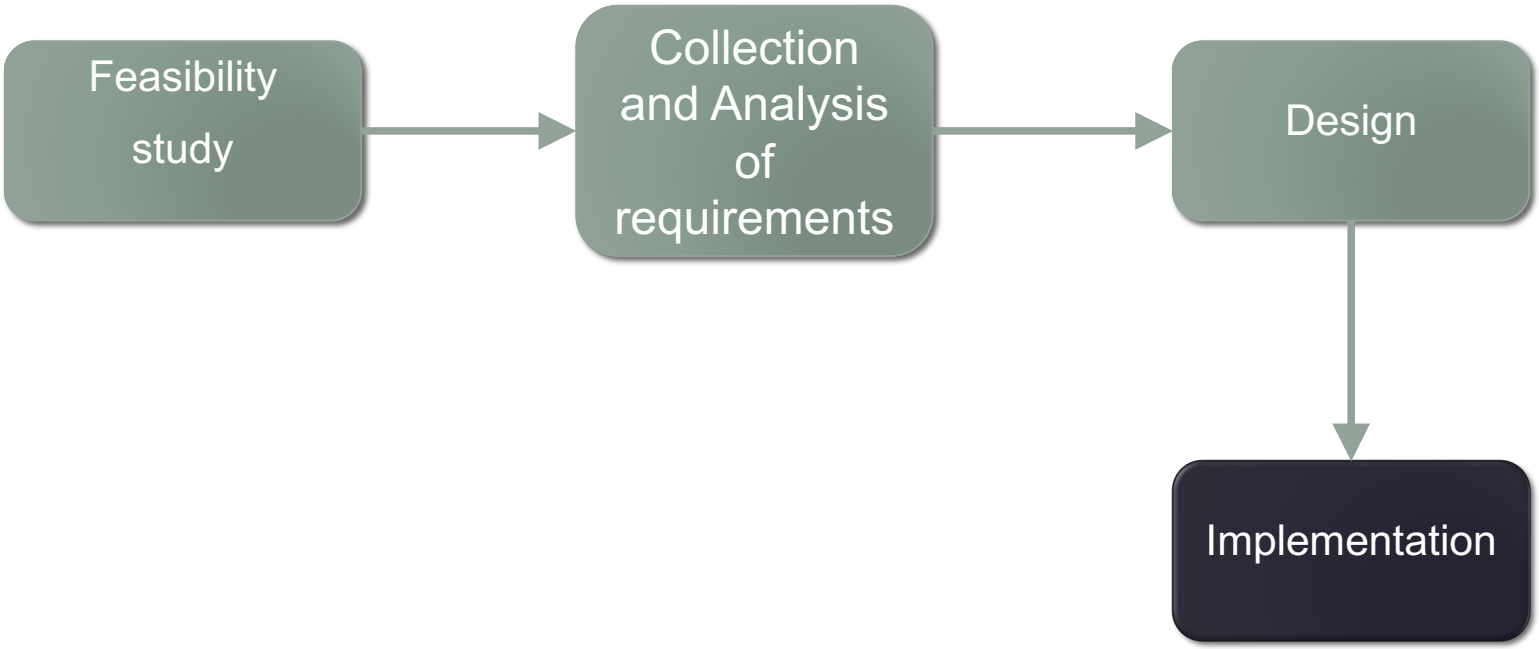
…

Information System Life Cycle

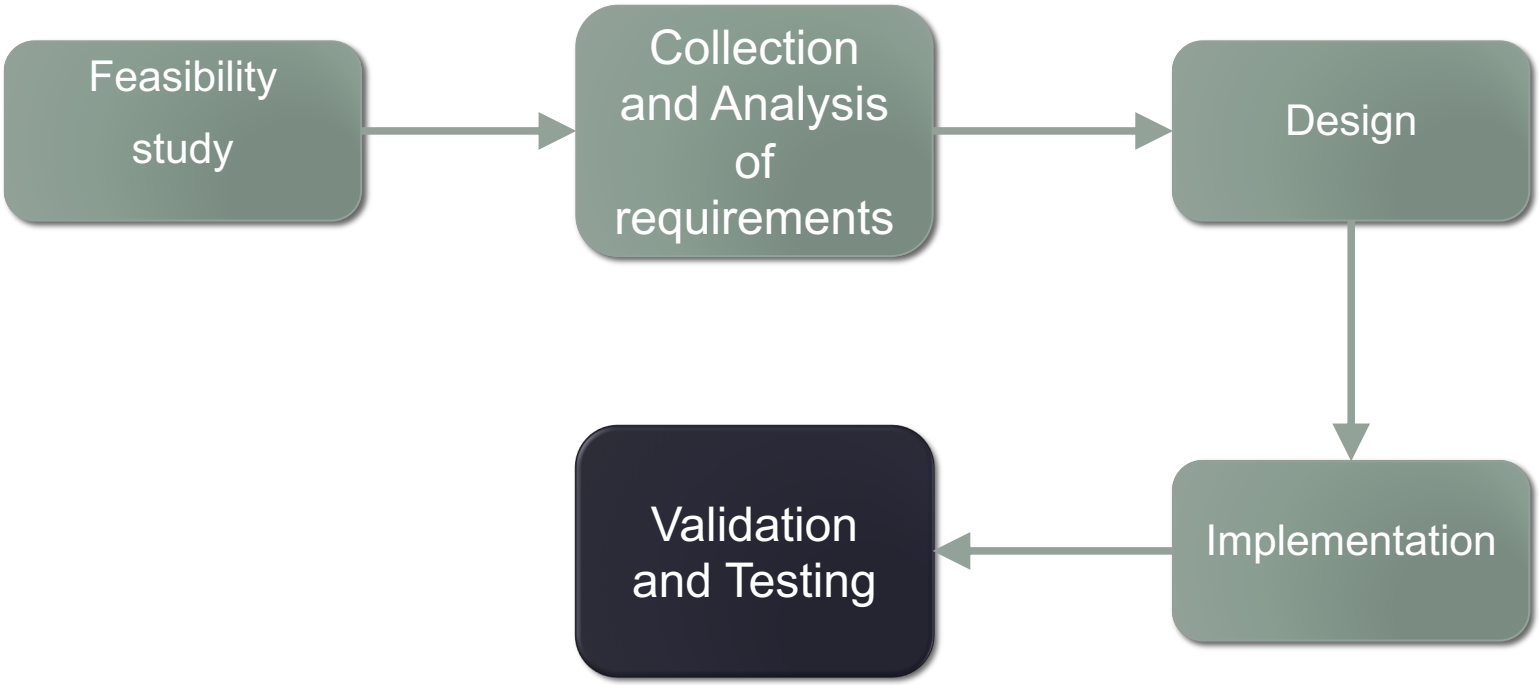
Design is generally divided into two tasks:

1. operational design and
2. database design.

Information System Life Cycle

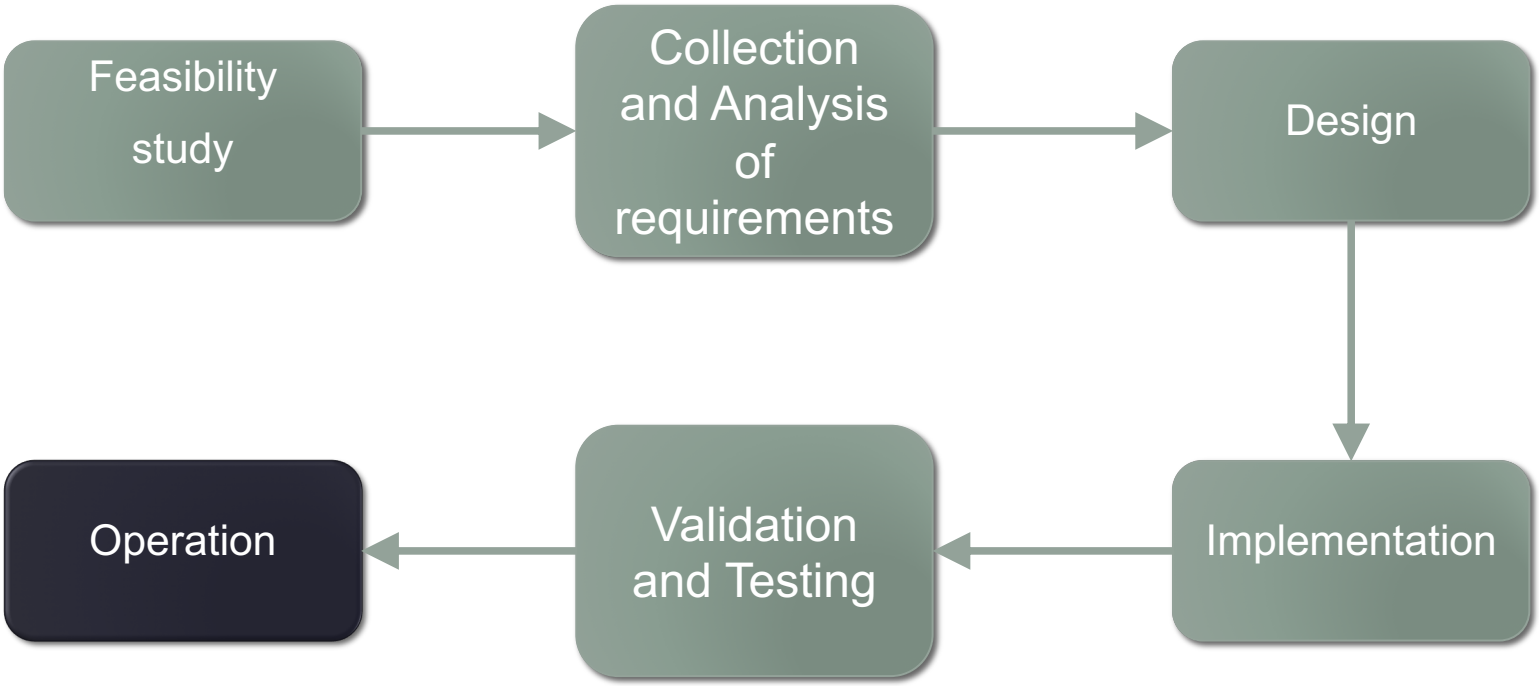


Information System Life Cycle



Validation and testing is done to check the correct functioning and quality of the information system.

Information System Life Cycle



Finally in the **operation** phase, the information system becomes live and performs the tasks for which it was originally designed.

Components of Computer-Based Information Systems (IS)

- Database (DB)
- Database software.
- Application software.
- Computer hardware (e.g., storage media).
- Personnel using and developing the system.

NOTE: the DB is a **fundamental** component

Information systems and databases

- The database constitutes **only one** of the components of an information system, which also includes **application programs**, **user interfaces** and other **service programs**.

- However, the **central role** that the data itself plays in an information system justifies an independent study of database design.

- For this reason, we deal with only those aspects of information system development that are closely related to databases, focusing on **data design** and on the related activities of **requirements collection and analysis**.

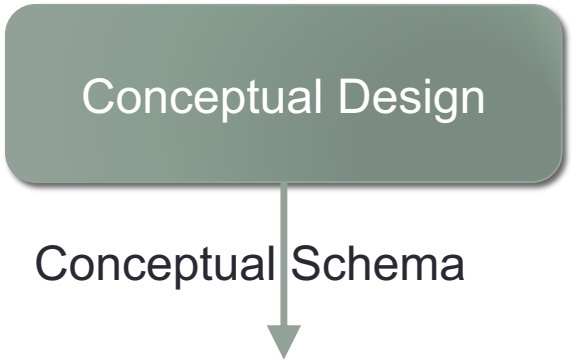
A DATABASE DESIGN METHODOLOGY

A database design methodology

- The most common database design methodology is based on a simple but highly efficient engineering principle: separate the decisions relating to '**what**' to represent in the database, from those relating to '**how**' to do it.

- This methodology is divided into three phases to be followed consecutively.

The Phases of Database Design



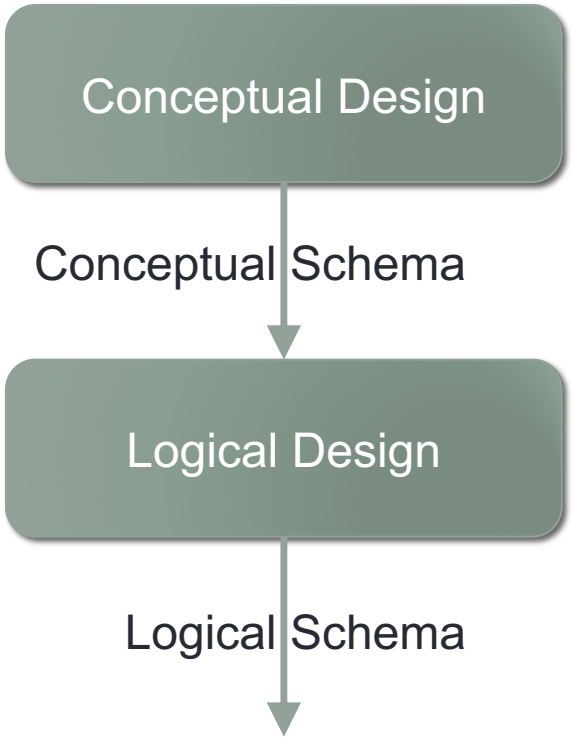
1. Conceptual Design

The purpose of this is to represent the informal requirements of an application.

Focus on the concepts that should be modelled, NOT the way they will be stored in a database: this will come later.

Output is a **conceptual schema**, often using an **Entity-Relationship Model.**

The Phases of Database Design

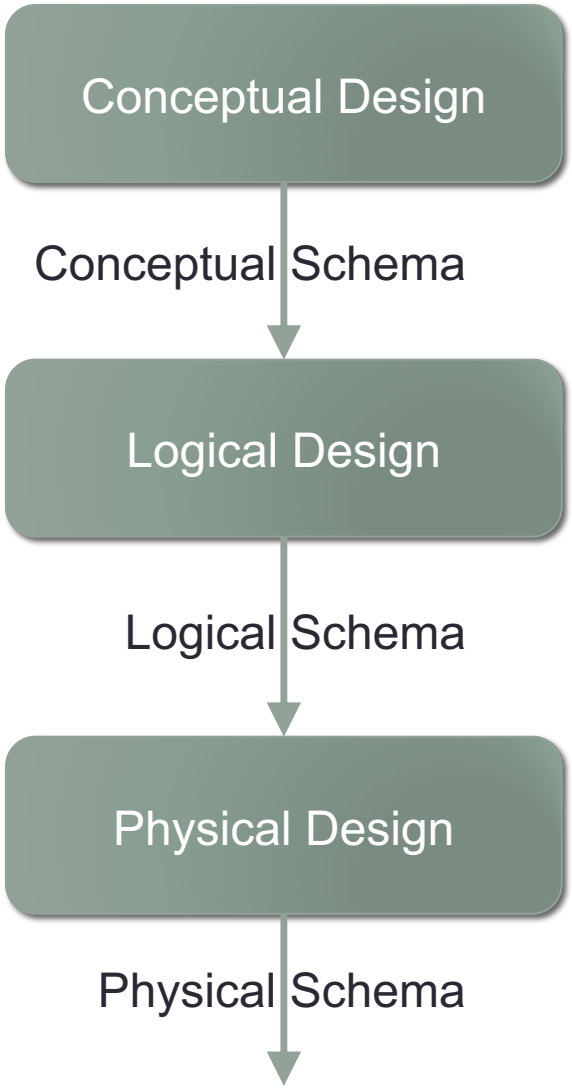


2. Logical Design

This consists of the translation of the conceptual schema into the **logical schema** of the database that refers to a logical data model.

This is the definition of the tables and attributes that will be created to store the data in our database.

The Phases of Database Design



3. Physical design

In this phase, the logical schema is completed with the details of the physical implementation (file organisation and indexes) on a given DBMS.

The product is called the physical schema and refers to a physical data model.

CONCEPTUAL DESIGN: THE ENTITY RELATIONSHIP (E-R) MODEL

The Entity Relationship model

- The Entity-Relationship (E-R) model is a **conceptual data model**.
- It is capable of describing the data requirements of an application in a way that is **easy to understand** and is **independent** of the criteria for the management and organisation of data on a database system.
- An E-R model is normally shown as a graphical representation.
- (See Chapter 7 of Elmasri and Navathe Book, 6th Edition)

Entities

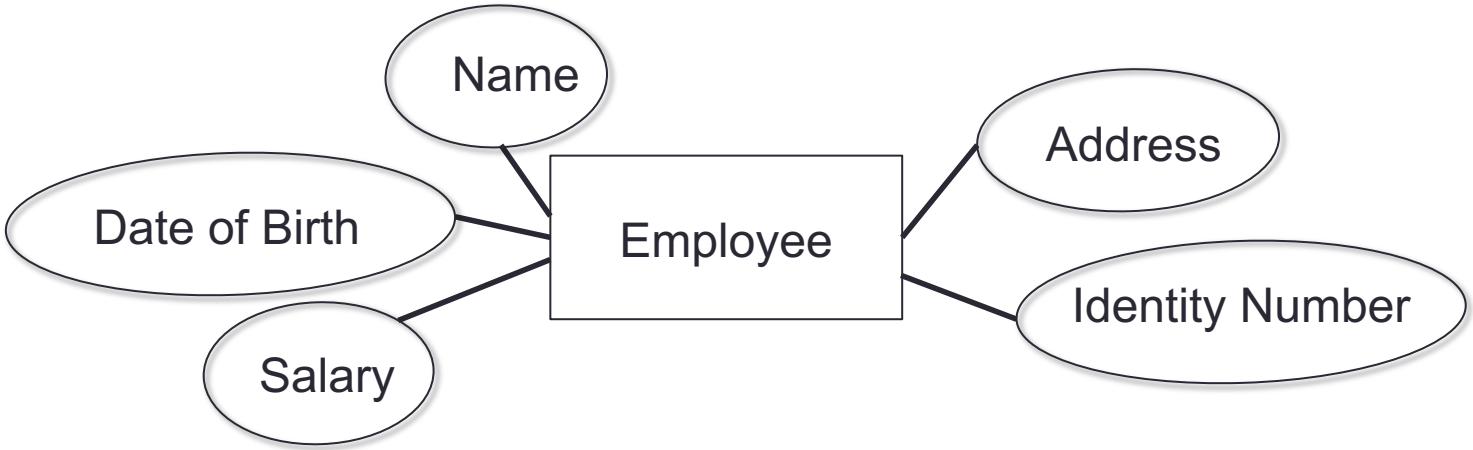
- **Entity:** 

- An “entity” is some thing that we want to store data about.
- It can be something that has;
  - **physical existence** (e.g. a person, building, car, etc.) or
  - **conceptual existence** (e.g. a university module, a job, etc.)
- In the E-R model we define **types** of entities.
  - An “occurrence” of an entity is similar to an object that is an instance of a class.

Attributes

- **Attribute:** 

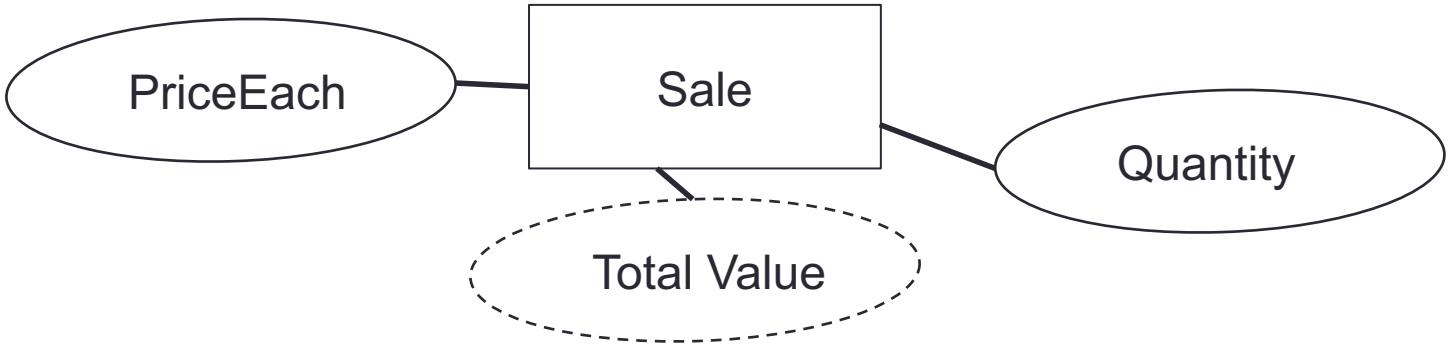
- Entities are described by their “attributes”, which are the properties that describe an entity.
- Each attribute has a “domain” (a set of allowed values)
- For example, an “employee” entity might be described by the following attributes: identity number, name, date of birth, salary, address, etc.



Attributes

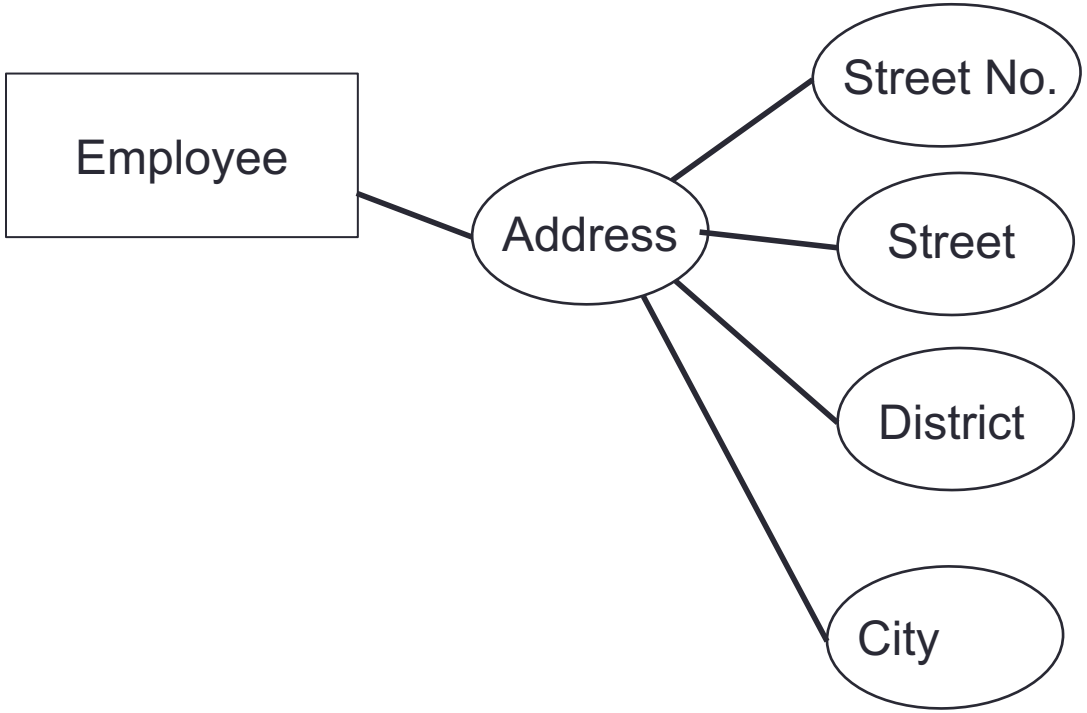
- A double oval indicates that an entity can have **multiple values** for the same attribute.

- A dashed oval indicates that an attribute is a **derived attribute** (i.e. it can be calculated from the entity's other attributes).



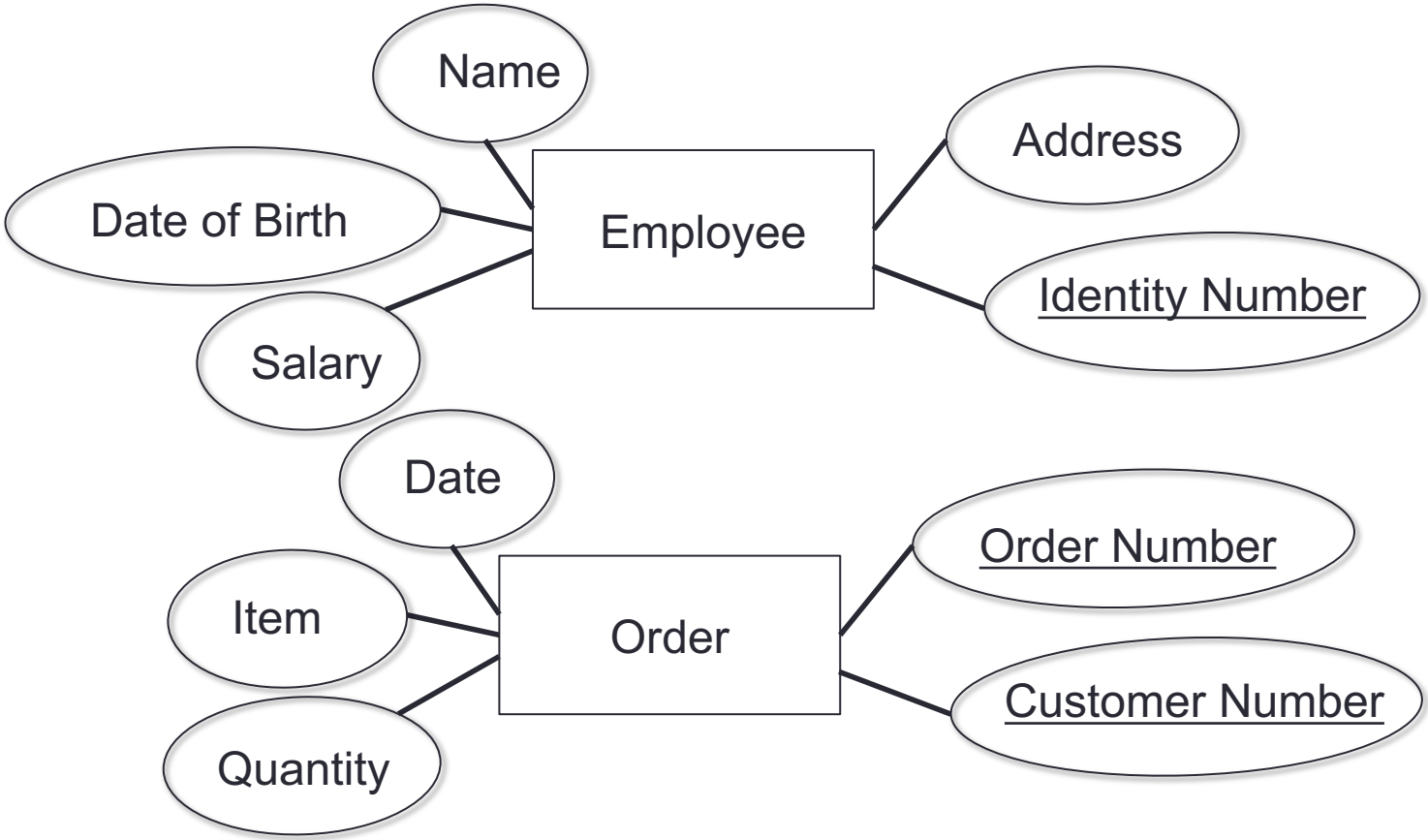
Attributes

- Sometimes, an attribute can be a **composite attribute**, made up of a number of component attributes.

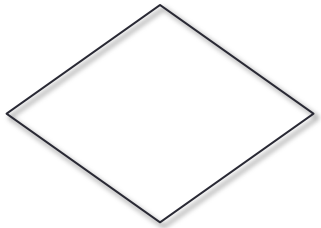


Attributes: Keys

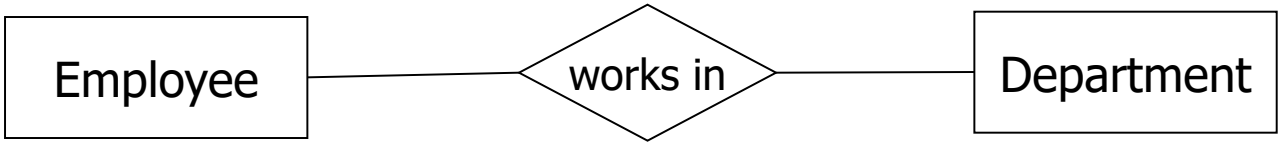
- If an attribute is the entity's primary key (or is part of a composite primary key), then its name is underlined.



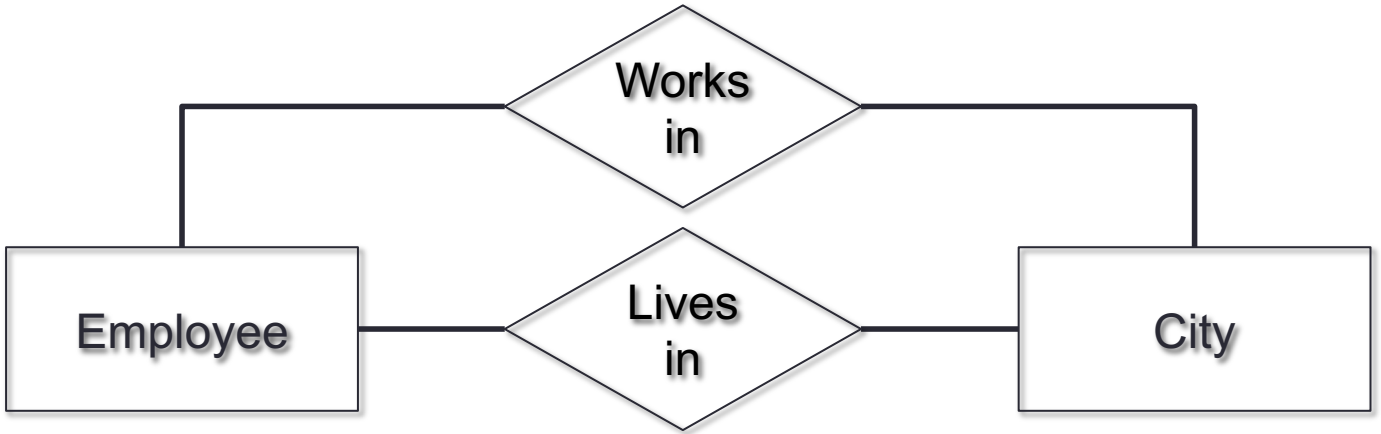
Relationships

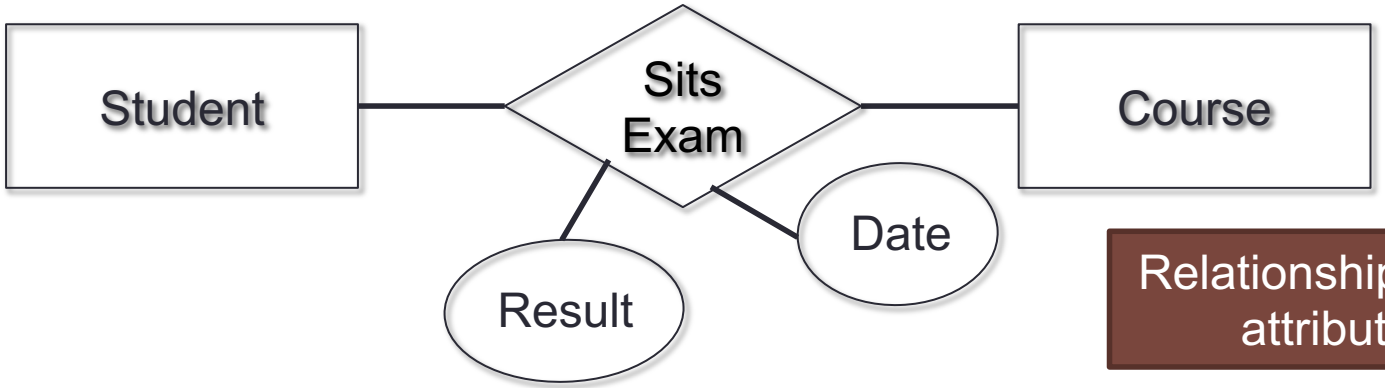
- **Relationship:** 

- A relationship is a **meaningful association** between two or more entity types. Some examples:
  - An **employee** WORKS IN a **department**.
  - A **student** is REGISTERED FOR a university **module**.
  - A **car** is OWNED BY a **person**.
  - A **salesperson** SELLS a **product** to a **customer**.



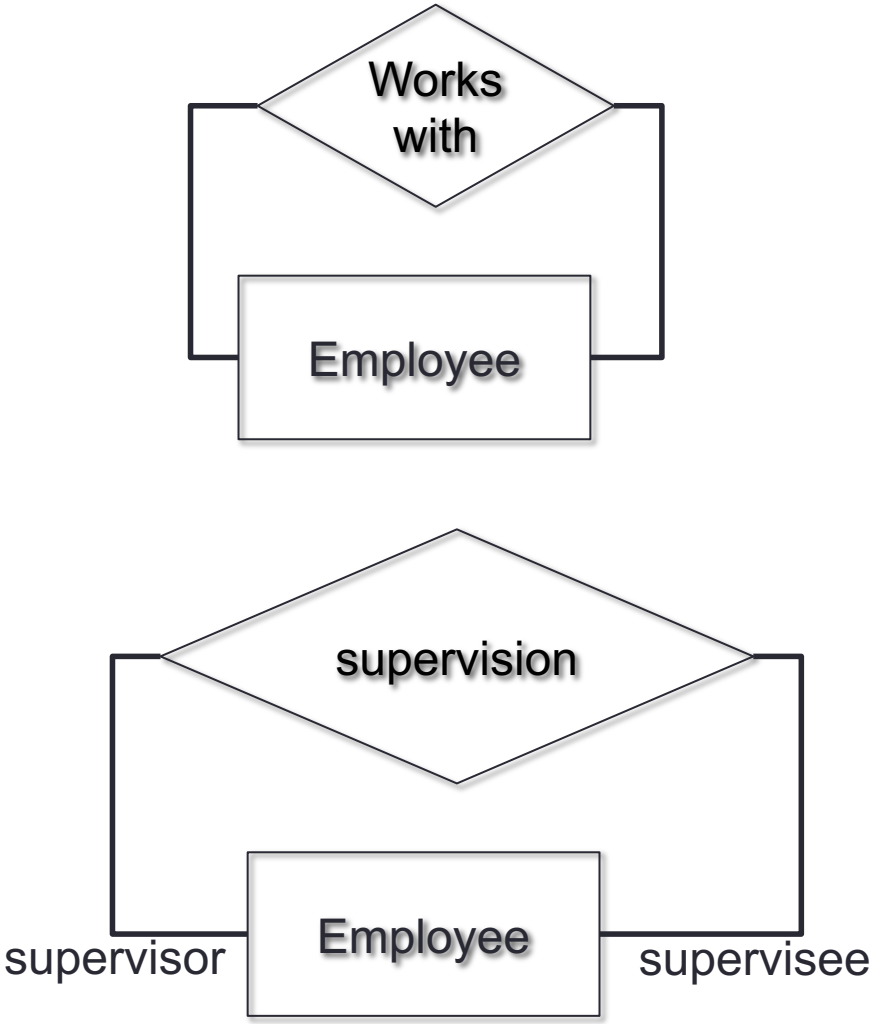
Relationships Examples



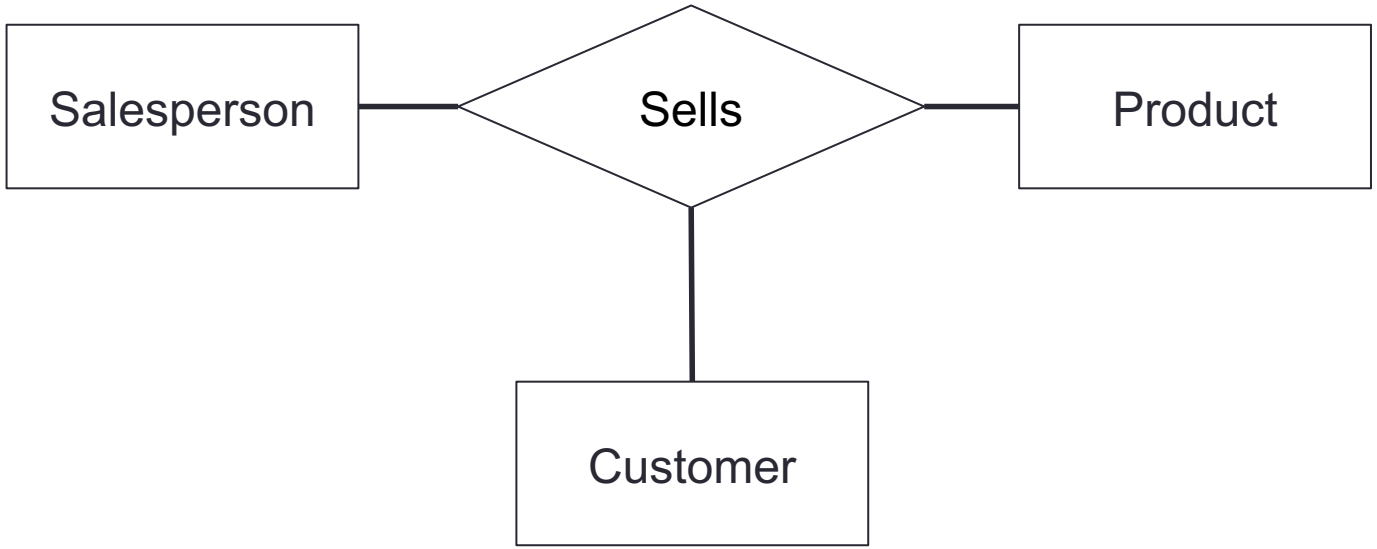


Relationships can have attributes too!

Recursive Relationships Examples



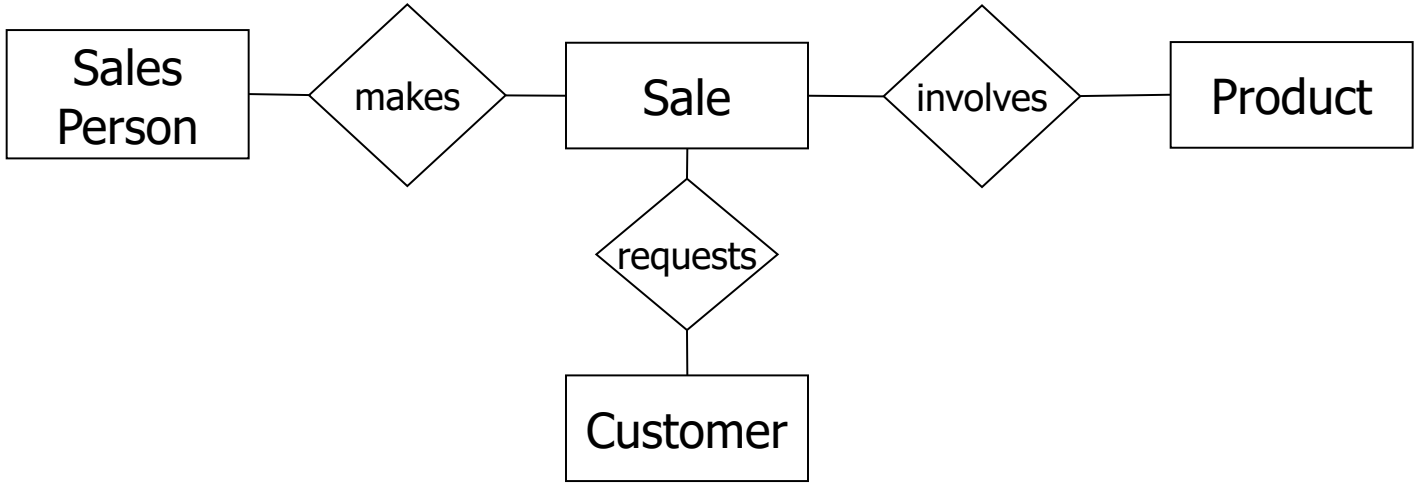
Ternary Relationships



Ternary relationships should not be left in the conceptual model.
They should be replaced with a series of binary relationships before the model is completed.

Ternary Relationships

- **General rule:** replace the relationship with an entity type.
  - This entity type will have binary relationships with the other entity types that were in the original ternary relationship.



Optionality of Relationships

- A relationship can be **optional** or **mandatory**.
- If a relationship is mandatory:
  - An entity at one end of the relationship must be related to another entity on the other end.
  - Also known as **total participation**.
- If a relationship is optional:
  - An entity on one end of the relationship may exist without being related to another entity on the other end.
  - Also known as **partial participation**.
- This can be different for opposite ends of the same relationship. For example:
  - A student must be registered for a course: this is mandatory.
  - But a course can exist before any students have enrolled: this is optional.

Optionality of Relationships

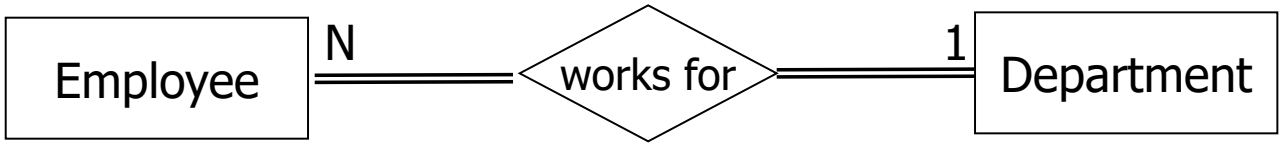
- Mandatory participation in a relationship is shown with a **double line**.
- Here, every department must have an employee that manages it: **mandatory**.
- However, not every employee is required to manage a department: **optional**.

Cardinality of Relationships

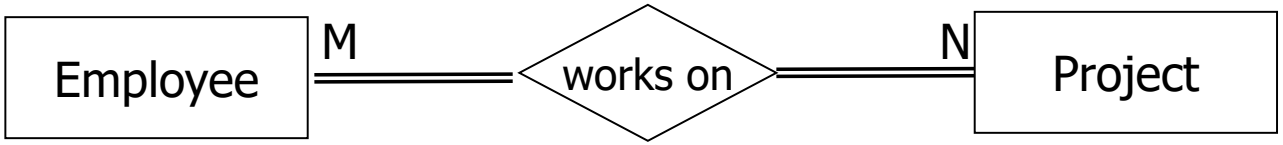
- The **cardinality** of a relationship refers to the number of entity occurrences that are involved in the relationship.
- There are three main types of cardinality, in each case, we talk about the **maximum** number of other entities that an entity can be related to:
  - **1-to-1 (1:1):**
    - Each employee can manage one department at most.
    - Each department can be managed by one employee at most.

Cardinality of Relationships

- **1-to-Many (1:N):**
  - Each employee works for one department at most.
  - Each department can have many employees working for it.



- **Many-to-Many (M:N):**
  - Each employee can work on many projects.
  - Each project can have many employees working on it.



Strong and Weak Entities

- Most entity types are **strong entity types**, they have their own primary key and can exist by themselves.
- However, some entity types are **weak entity types**, which cannot exist without a relationship to another entity type.

- An example:
  - Our company stores data about employee's dependents (spouse and children) for insurance purposes.
  - We store the following information about each:
    - Name
    - Sex
    - Relationship to employee
    - Date of birth

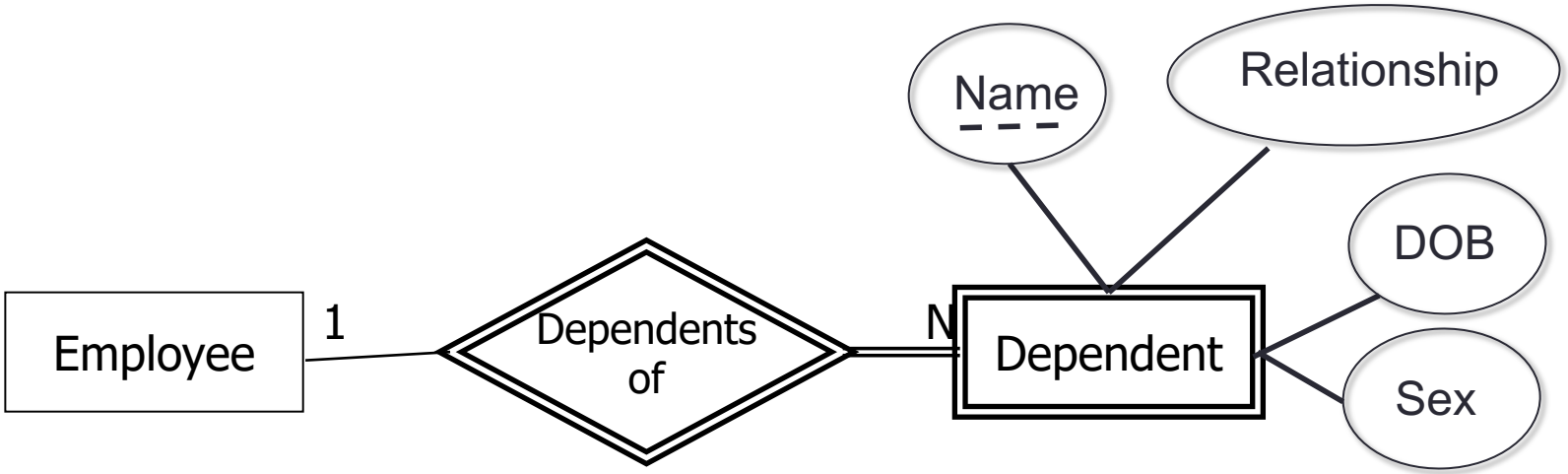
Strong and Weak Entities

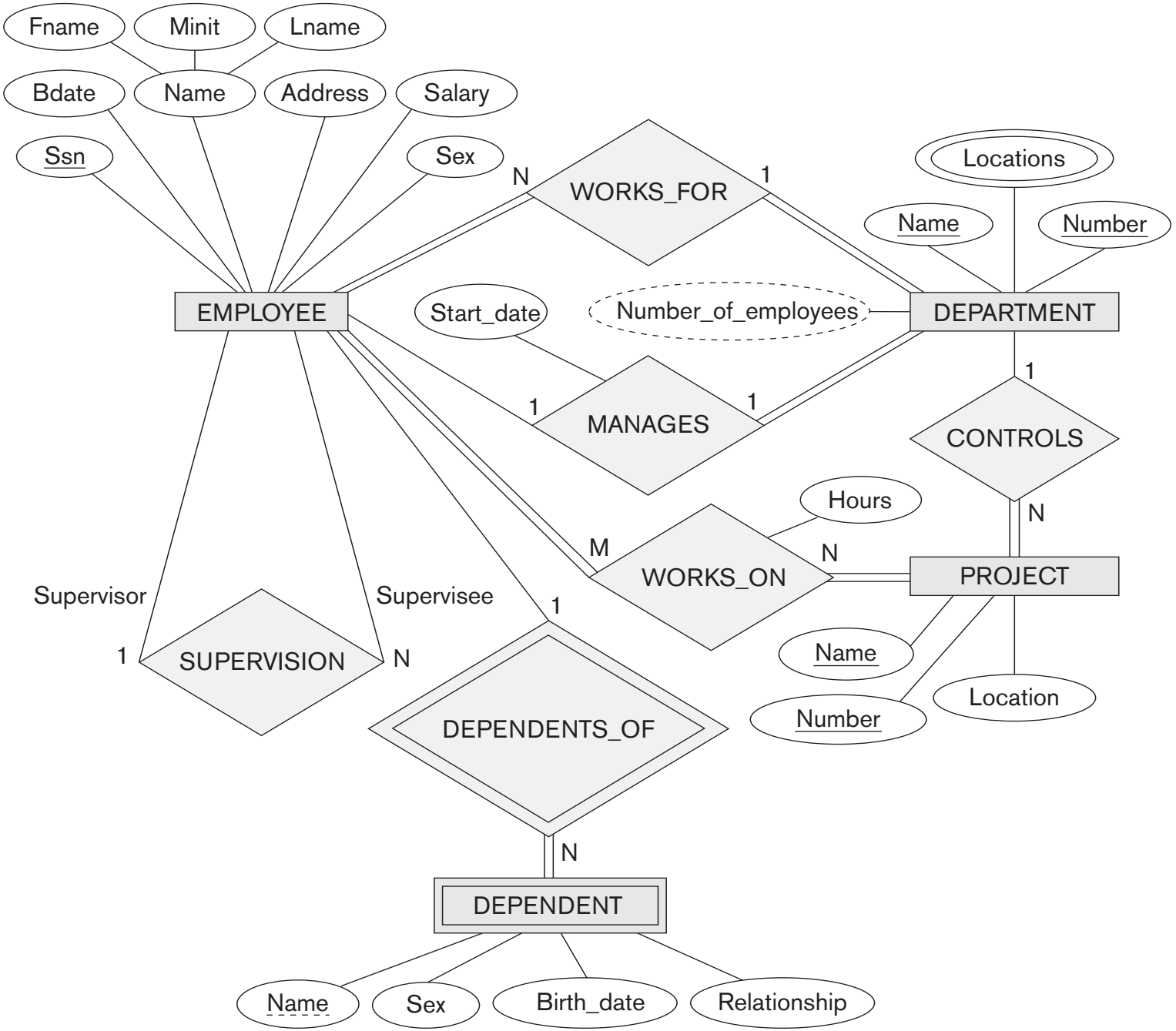
- In this schema, the Dependent entity type has no appropriate key.
  - Two employees may have a dependent with the same name, sex, date of birth and relationship, even though they are two different people.
  - We only see them as distinct entities when we know which employee they are associated with (their *owner* entity).

- For weak entities, we define a **partial key**. That is, something that can be used as a key *if we know which owner entity it is related to*.
  - If we assume that an employee will not have two dependents with the same name, we can use “name” as a partial key.

Strong and Weak Entities

- An attribute that is part of a **partial key** has its named underlined with a **dashed line**.
- The **weak entity**'s box has a **double line**.
- The **relationship** that identifies its owner entity also has a **double line**.




Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
WORKS_ON
N
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship