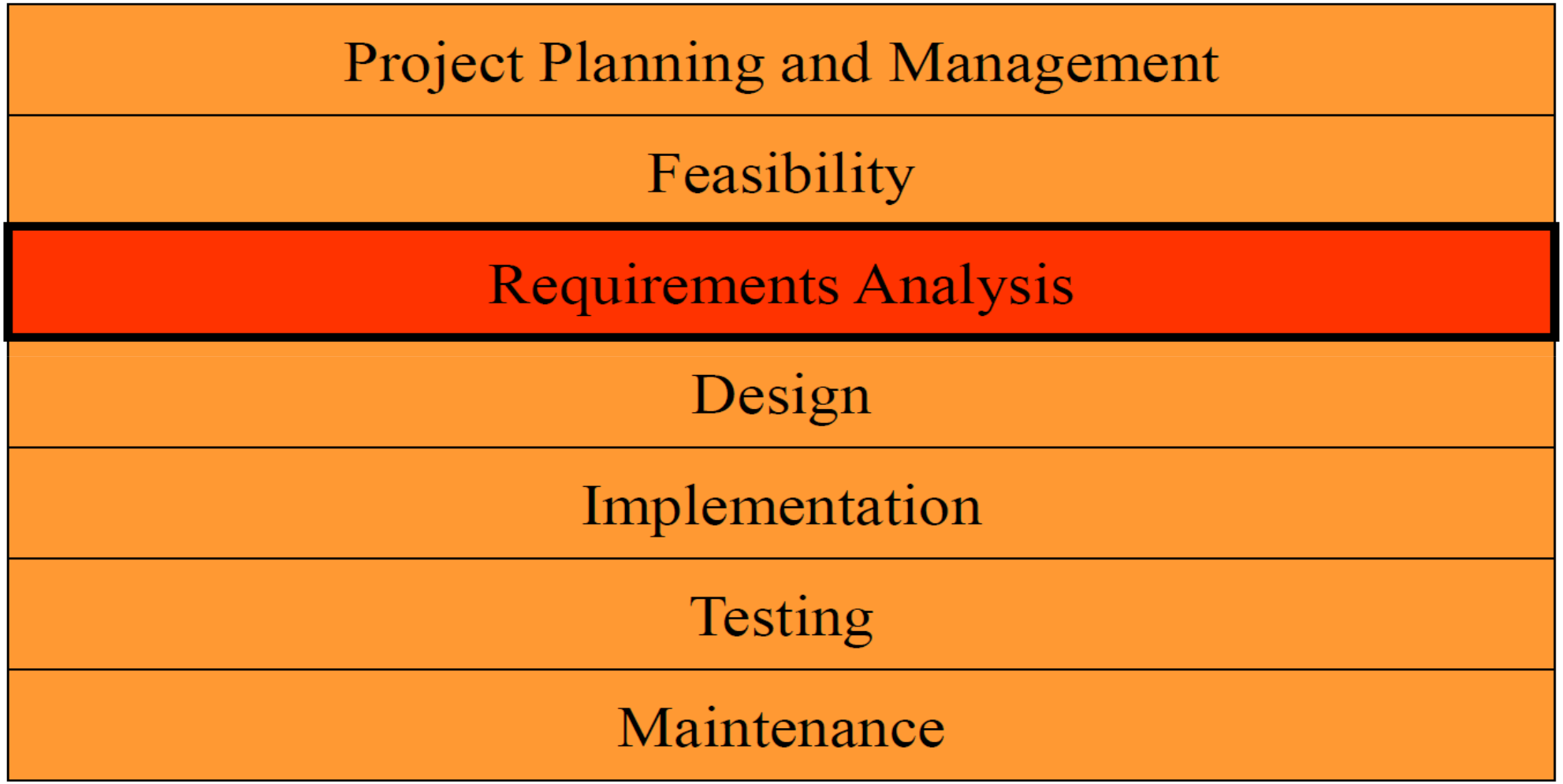
2. Database Design

Master I – Software Engineering

Dr. Imed Bouchrika
Dept of Mathematics & Computer Science
University of Souk-Ahras
imed@imed.ws

Thanks are due to Nishadha & Griffin for their notes

Software Lifecycle



Database design lifecycle

- **Requirements analysis**
  - User needs; what must database do?
- Conceptual design
  - High-level description; often using E/R model
- Logical design
  - Translate E/R model into (typically) relational schema
- Schema refinement
  - Check schema for redundancies and anomalies
- Physical design/tuning
  - Consider typical workloads, and further optimise

Requirement Analysis

- During this phase, we need to know as much as possible about :
  - End-Users.
  - Tasks/Functionalities
  - Context

- So that we can identify the aim of the user as well as their needs

- This phase is extremely IMPORTANT because more then 50% of software bugs/errors are attributed to errors committed at the requirement analysis phase.

Requirement Analysis

- Problems for the Requirement Analysis
  - Users don’t know what they want !
  - Users suppose that YOU know everything
  - Requirements are different from Person to Person
  - Political or Social Factors !
  - Things can change rapidly.

Requirement Analysis

- Functional Specification
  - What the user wants to do
  - What the user wants to see

- Non-Functional Specification
  - Security
  - Efficiency
  - Performance
  - Availability
  - Easiness of use.
  - ....

Requirement Analysis

- Functional Specification
  - What the user wants to do
  - What the user wants to see

- Non-Functional Specification
  - Security
  - Efficiency
  - Performance
  - Availability
  - Easiness of use.
  - ....

Requirement Analysis

- Group Activities : Try to produce a set of requirements for the following software systems:
  - Personal Library.
  - University Library.
  - Hotel Booking
  - Small Search Engine

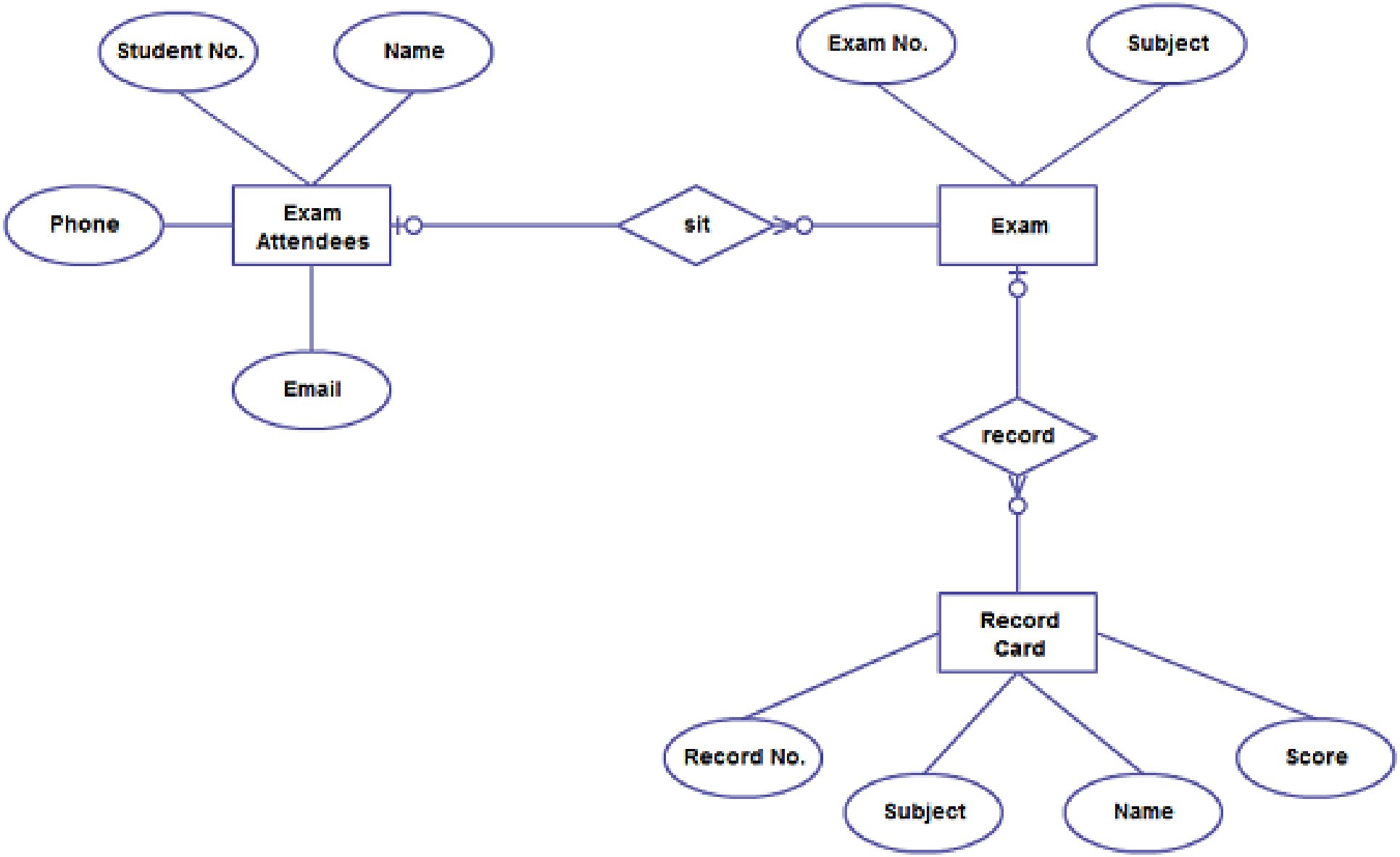
Database design lifecycle

- Requirements analysis
  - User needs; what must database do?
- **Conceptual design**
  - High-level description; often using E/R model
- Logical design
  - Translate E/R model into (typically) relational schema
- Schema refinement
  - Check schema for redundancies and anomalies
- Physical design/tuning
  - Consider typical workloads, and further optimise

Entity-Relationship Diagram (ERD)

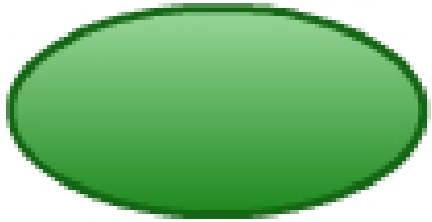
- ERD is a graphical language being used often for database design.
- ERD is used for provide a graphical representation for the logical structure of a database.
- ER diagrams are easy to understand and do not require a person to undergo extensive training.
- Designers can use ER diagrams to easily communicate with developers, customers, and end users.
- ER diagrams are readily translatable into relational tables which can be used to quickly build databases.

Entity-Relationship Diagram (ERD)

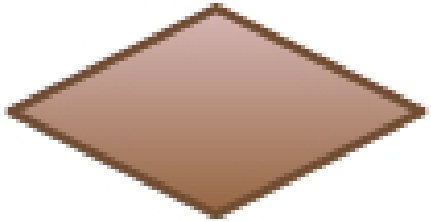


Entity-Relationship (E/R) basics

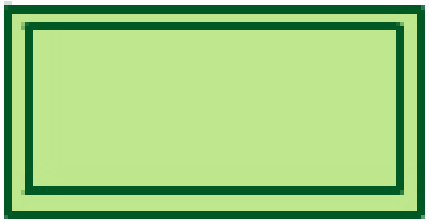
Entity



Attribute

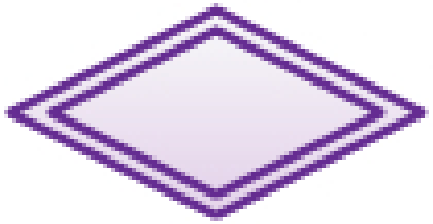


Relationship



Weak Entity

Multivalued Attribute



Weak Relationship

Entity-Relationship (E/R) basics

- An **entity** is a real-world object that is distinguishable from other objects

- An entity can be a person, place, event, or anything that has properties.

- For example, a school system may include :
  - students,
  - teachers,
  - courses,
  - Subjects ....

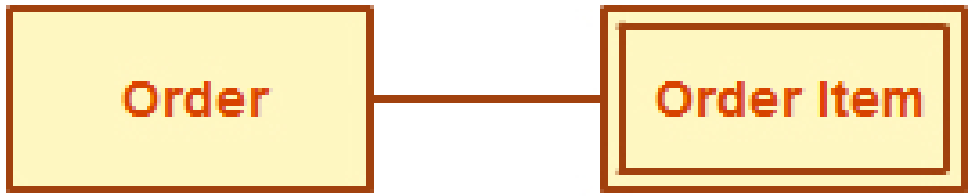
Entities and attributes

- **Entity** types are drawn as ***rectangle***

- Each entity has **attributes**

Weak Entity

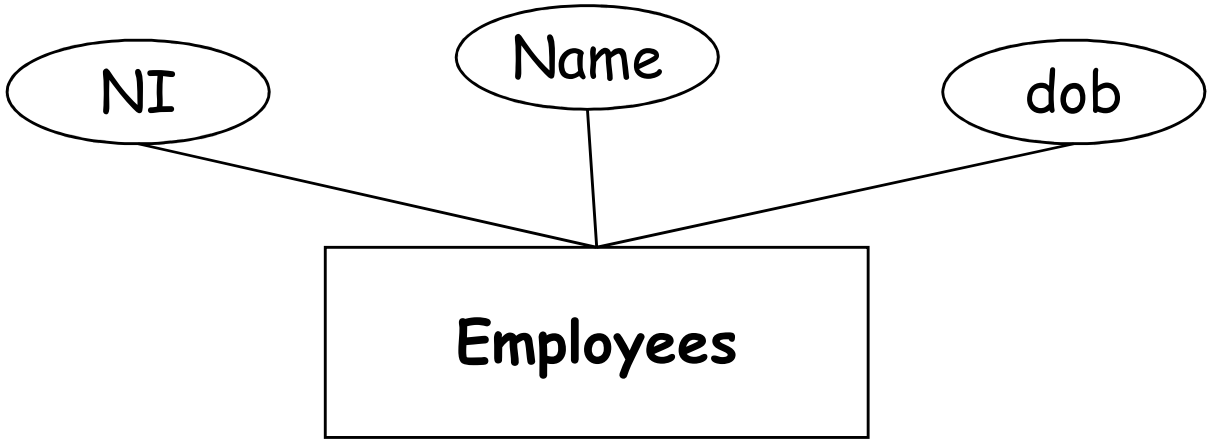
- **weak entity** is an entity that **depends** on the **existence** of another entity
- *Weak* entity cannot be identified by its own attributes.
- It uses a foreign key combined with its attributed to form the primary key
- Weak Entity drawn inside a **doubled line** rectangle



Entity-Relationship (E/R) basics

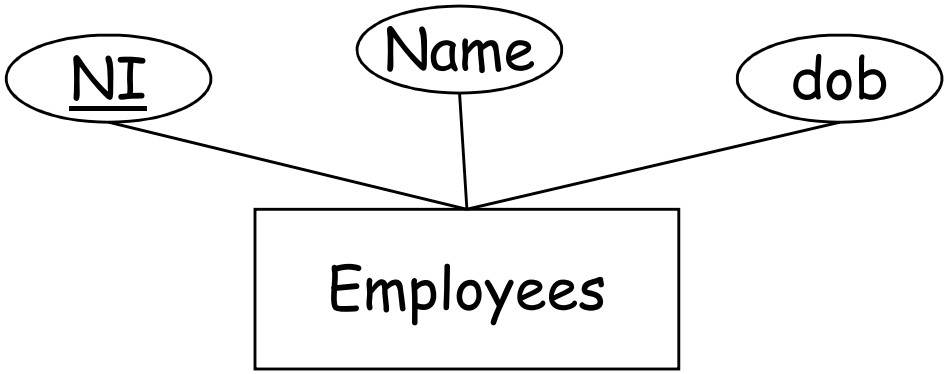
- An **attribute** is a property, trait, or characteristic of
  - an entity.
  - relationship,
  - or another attribute

- **Attributes** are drawn as ***oval***



Key attributes

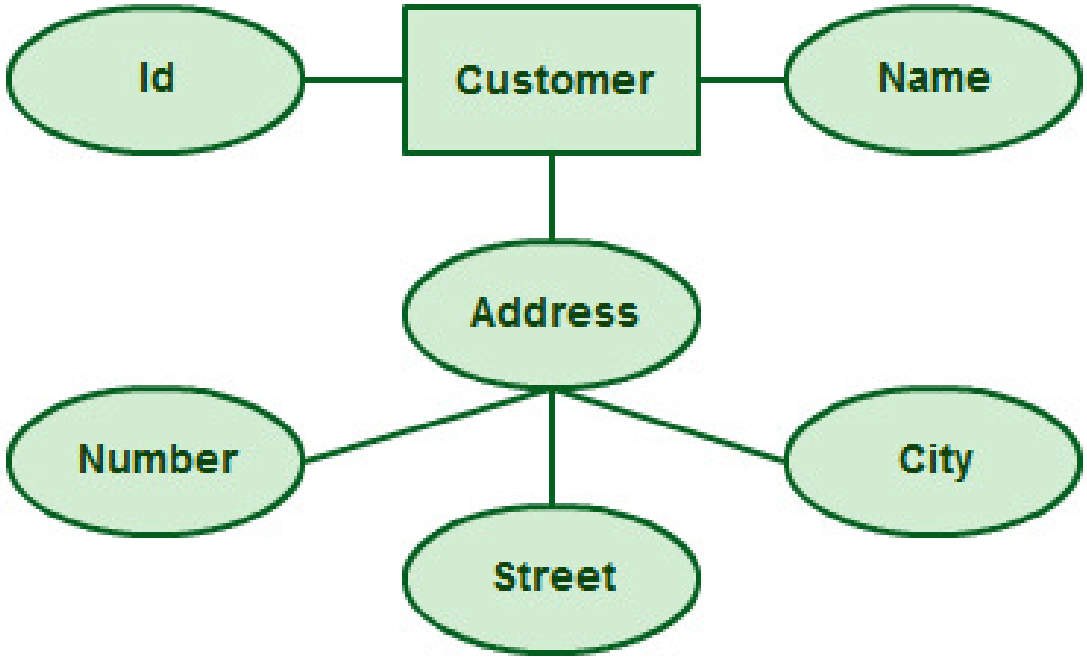
- A **key attribute** of an entity type is an attribute whose values are **distinct** for each entity



- We **underline** key attributes
- **NI** is the key attribute in the example above.
- Sometimes several attributes (a composite attribute) together form a key
  - NB: Such a composite should be **minimal**

Composite attributes

- Attributes can also have **their own** specific attributes.
- For example, the attribute “customer address” can have the attributes : *number*, *street*, *city*, and *state*.
- These are called **composite attributes**.



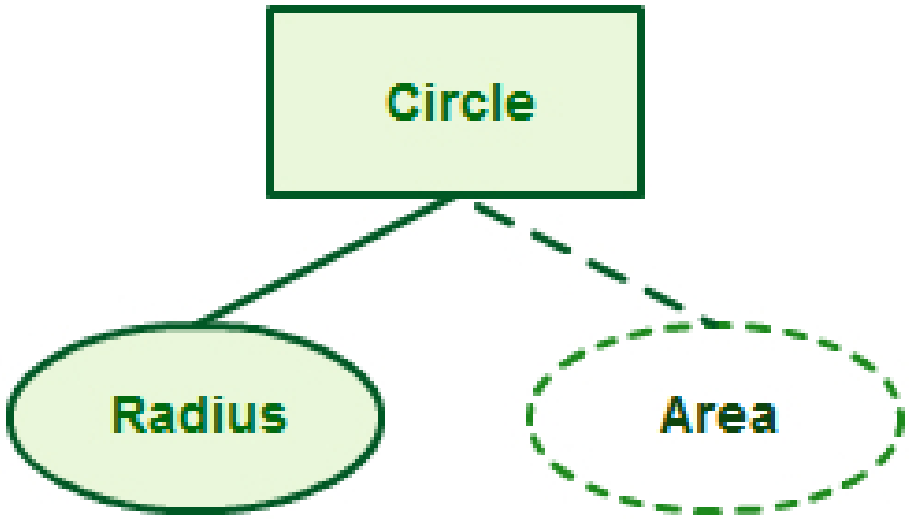
Multivalued Attribute

- If an Entity can have multiple Attribute, we use a **DOUBLE lined oval** for the attribute.

- For example a teacher entity can have multiple subject values.

Derived Attribute

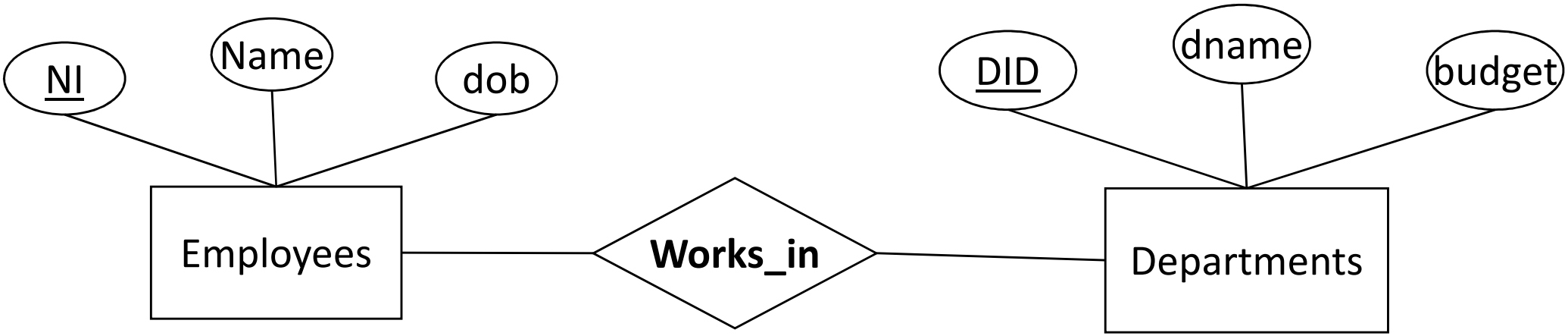
- Derived Attribute : that can be computed from another attribute
- For example for a circle the area can be derived from the radius.



Relationships in E/R

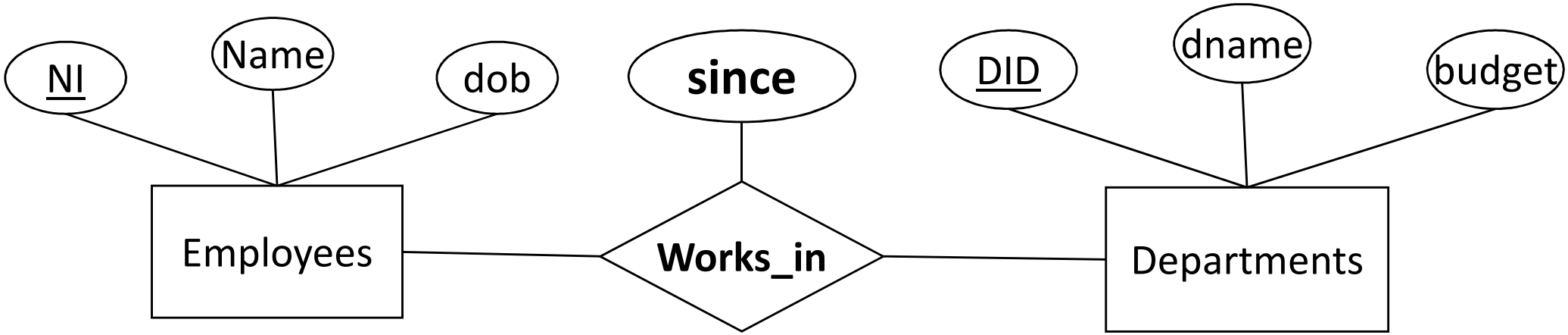
- A **relationship** describes how entities interact.
- Relationship types are represented by ***diamond***
- They connect the participating entity types with ***straight line***s

Relationships in E/R



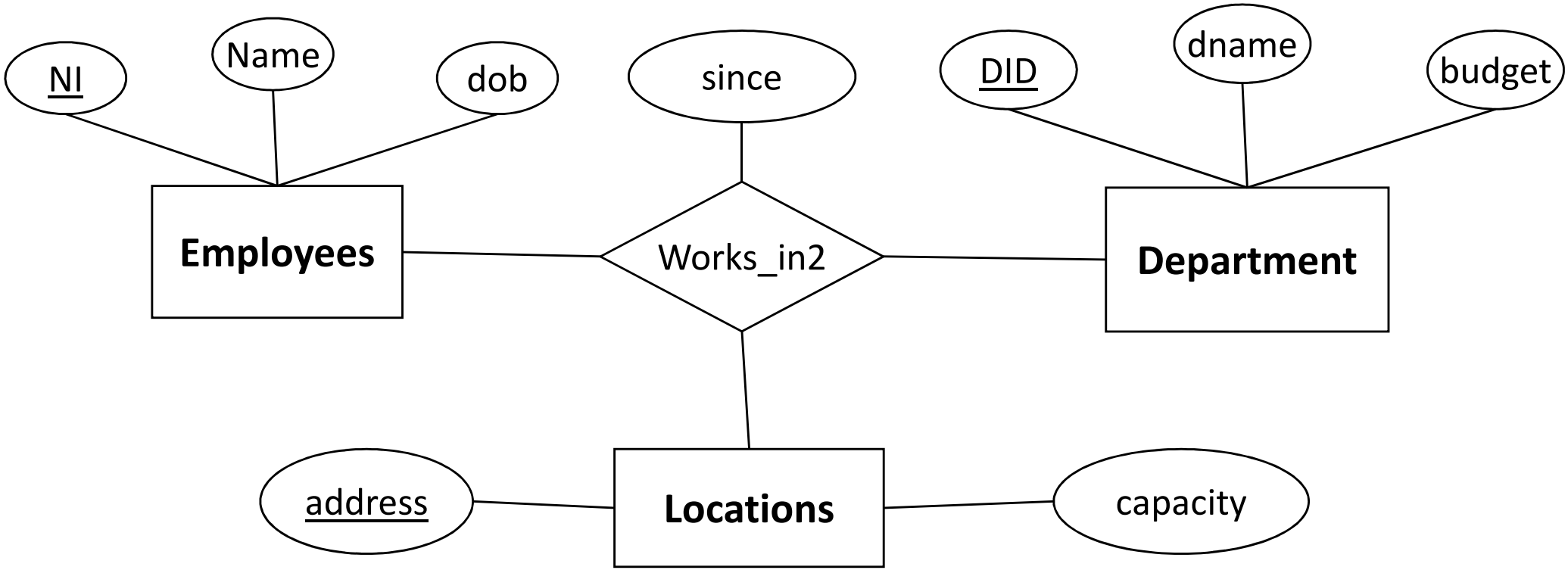
Relationship attributes

- Relationships can also have **attributes**



N-ary relationships

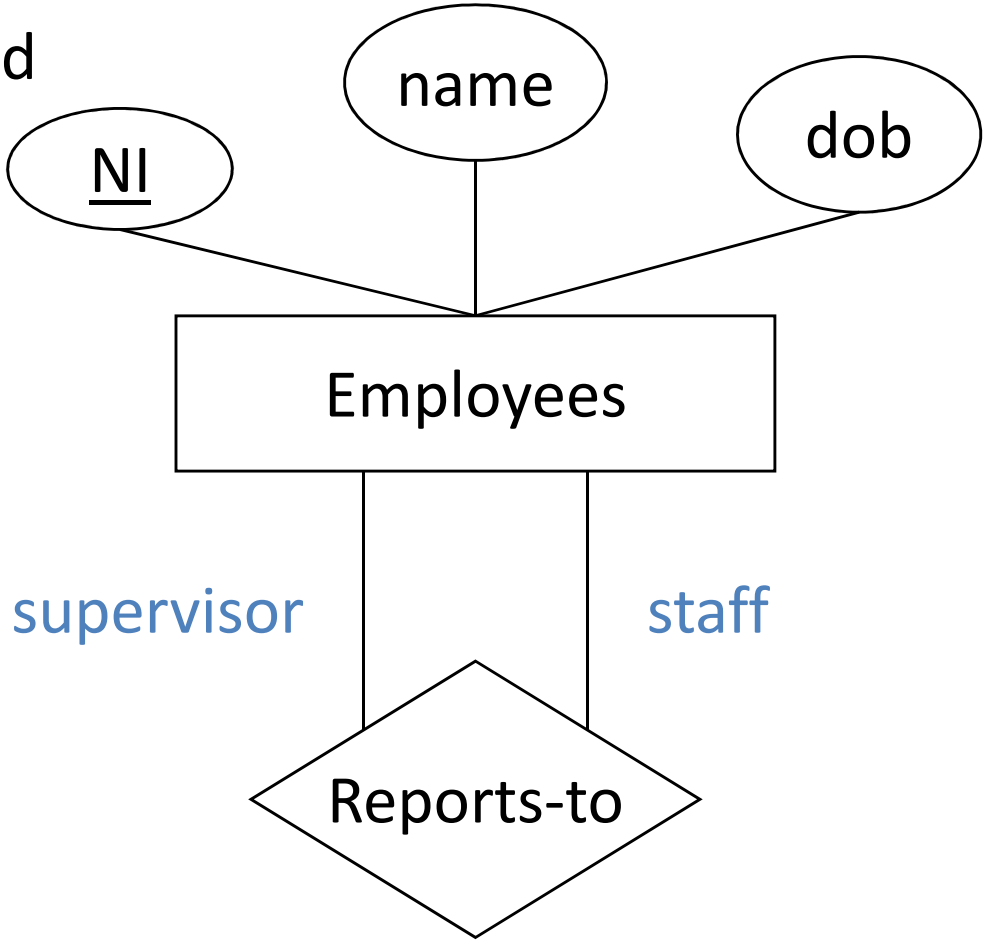
- We can have **n-ary** relationships instead of Binary.
- The example for **Ternary** relationship.



Recursive relationships

- **Recursive relationship** is when an entity type plays more than one role in the relationship type

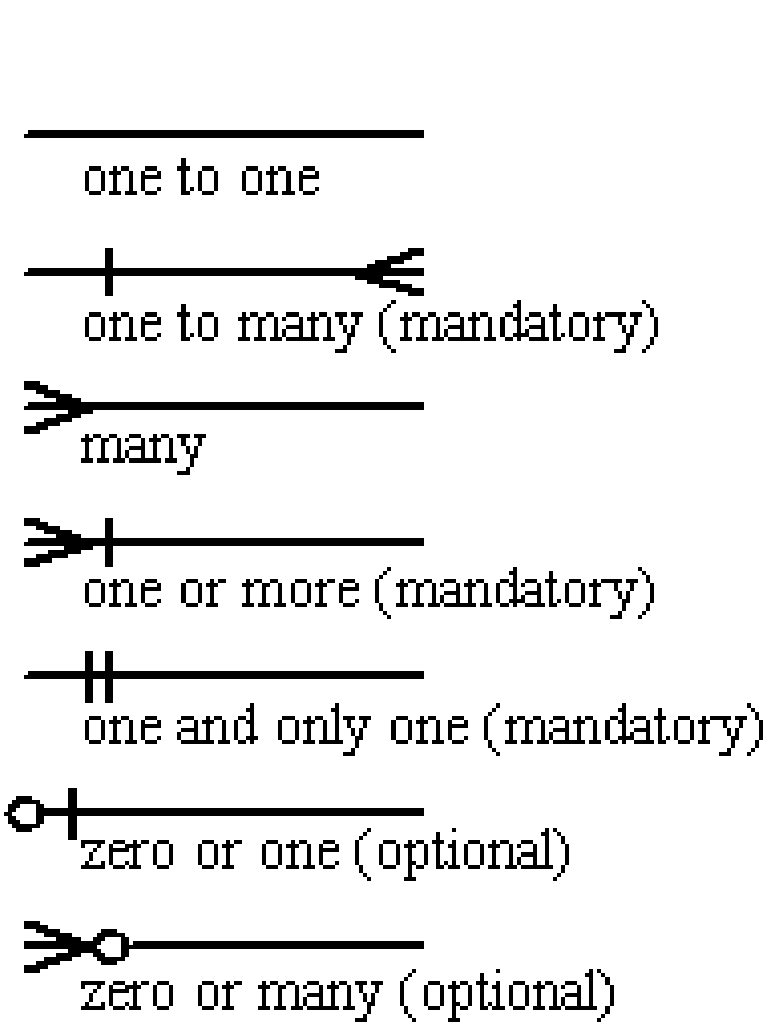
- The ***role name*** is required

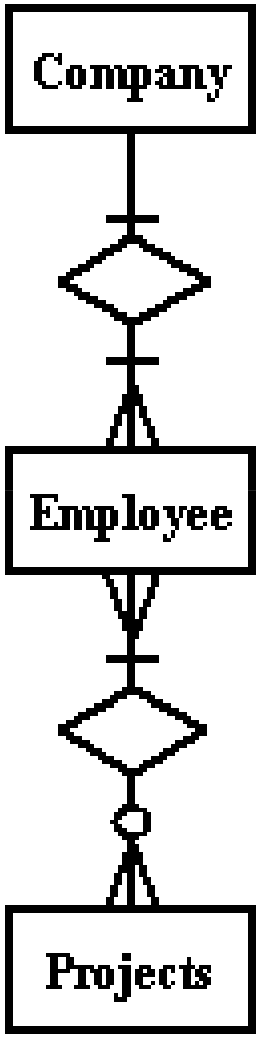


ERD Cardinality

- For example:
  - An employee can work in many departments; a department can have many employees
  - In contrast, each department has at most one manager
- The possible ratios are: **1:1**, **1:N**, **N:1**, **M:N**

ERD Cardinality





Participation constraints

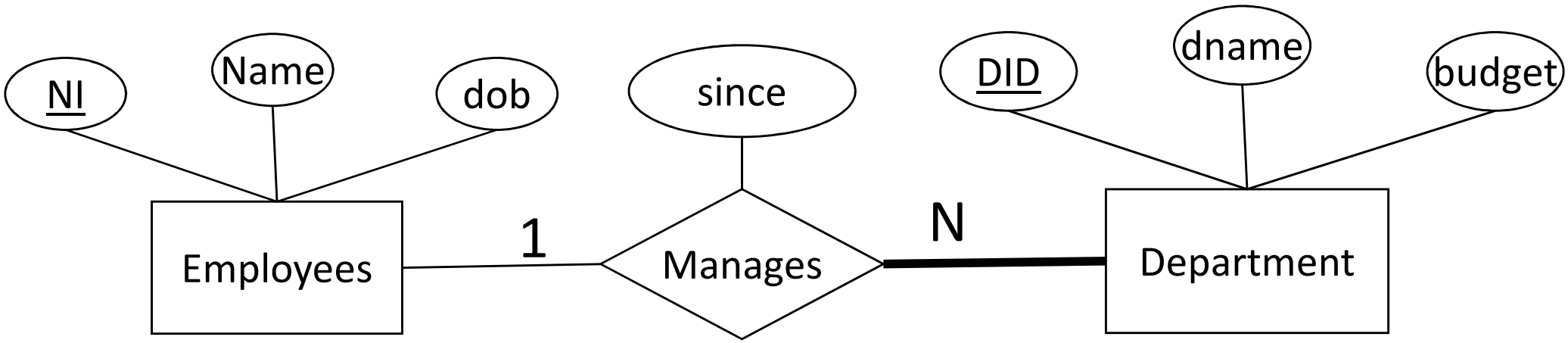
Every department must have a manager

- This is an example of a **participation constraint**
- The participation of an entity set, E, in a relationship set R is said to be **total** if

  → every entity in E participates in **at least one** relationship in R.
- If not its participation is said to be **partial**

Participation in E/R diagrams

- Total participation is displayed as a **bold** line between the entity type and the relationship
  - NB. Sometimes this is written as a **double line**



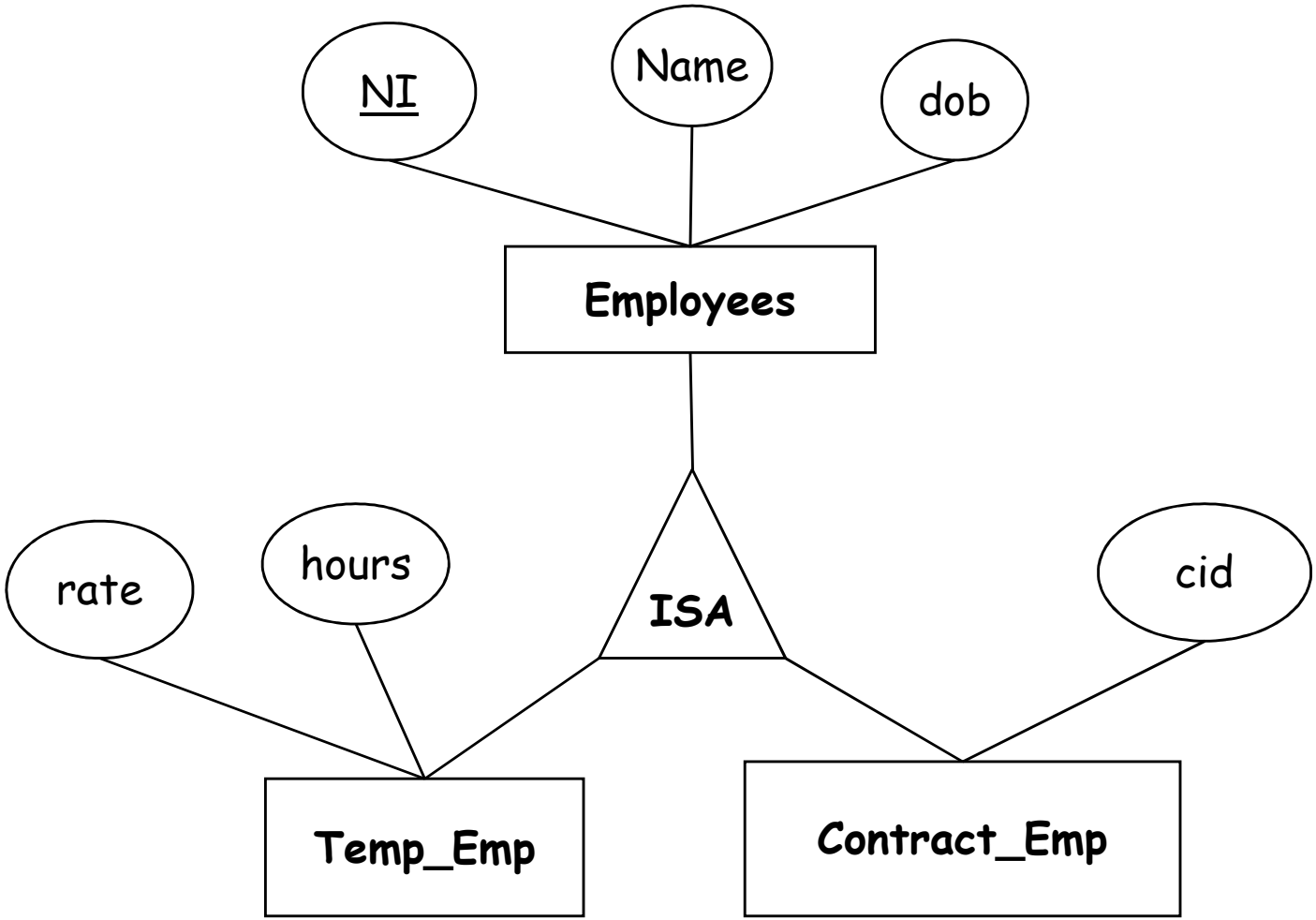
Extended E/R modelling (EERD)

- Over the years a number of features have been added to the model and the modelling process

- These features include:
  - Sub- and super-classes
  - Specialisation
  - Generalisation
  - Categories
  - Higher/Lower-level entity sets
  - Attribute inheritance
  - Aggregation

ISA hierarchies

- We can devise **hierarchies** for our entity types



Attribute inheritance

- ***attributes*** of superclasses are inherited by the subclasses.
- Thus: Temp_Emp also has attributes NI, Name and dob
- subclasses inherit ***relationships*** too

UML

- *The Unified Modeling Language (UML) is a graphical language for communicating design specifications for software.*

- *UML Diagrams :*
  - *Structure diagrams*
    - *Class, Component, Deployment ...*
  - *Behavior Diagram*
    - *Activity, state, Use case*
  - *Interaction Diagram*
    - *Sequence, Communication ...*

For you to search !

- **Database Normalisation .**
- **Chen Notation.**
- ***Software Artifacts***