Data Modeling Using the Entity-Relationship (ER) Model

Khalid Mahmood

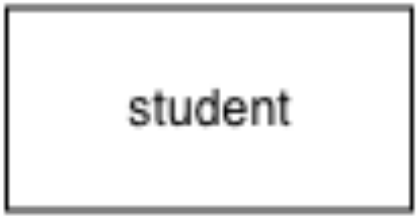
Khalid.mahmood@it.uu.se

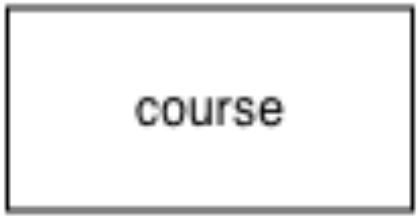
Entity-Relationship Model Concepts

- Entities
  - Specific ***things*** or ***objects***, e.g. student, course
  - Analogous to *class* of *OOP*
  - Holds Set/Collection of objects: *Entity set*
  - Same as *table* in DBMS

Entity-Relationship Model Concepts

- Entities
  - Specific ***things*** or ***objects***, e.g. student, course
  - Analogous to *class* of *OOP*
  - Holds Set/Collection of objects: *Entity set*
  - Same as *table* in DBMS





ER Model Concepts

- Attributes
  - ***Properties*** to *describe* an entity
  - ***Columns*** of a table (i.e. Entity)

ER Model Concepts

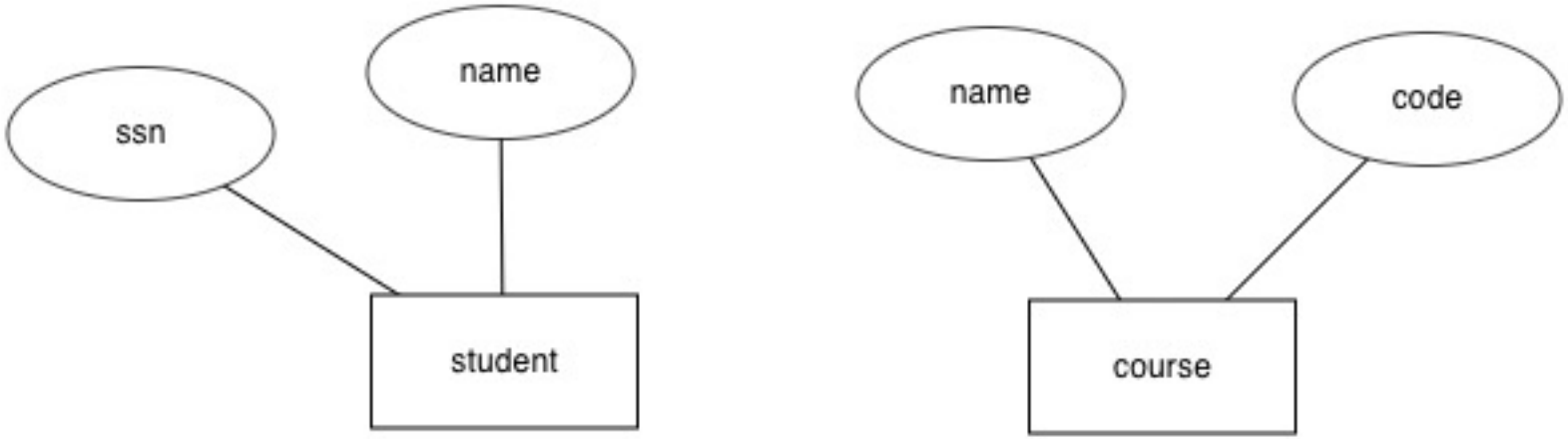
- Attributes
  - ***Properties*** to *describe* an entity
  - ***Columns*** of a table (i.e. Entity)
  - ***Elements*** in each entity have ***value*** for each *attributes*

ER Model Concepts

- Attributes
  - ***Properties*** to *describe* an entity
  - ***Columns*** of a table (i.e. Entity)
  - ***Elements*** in each entity have ***value*** for each *attributes*
  - *Each **attribute** has a value set (or data type) – e.g. integer, string, date, ..*

ER Model Concepts

- Attributes
  - ***Properties*** to *describe* an entity
  - ***Columns*** of a table (i.e. Entity)
  - ***Elements*** in each entity have ***value*** for each *attributes*
  - *Each **attribute** has a value set (or data type) – e.g. integer, string, date, ..*



Types of Attributes

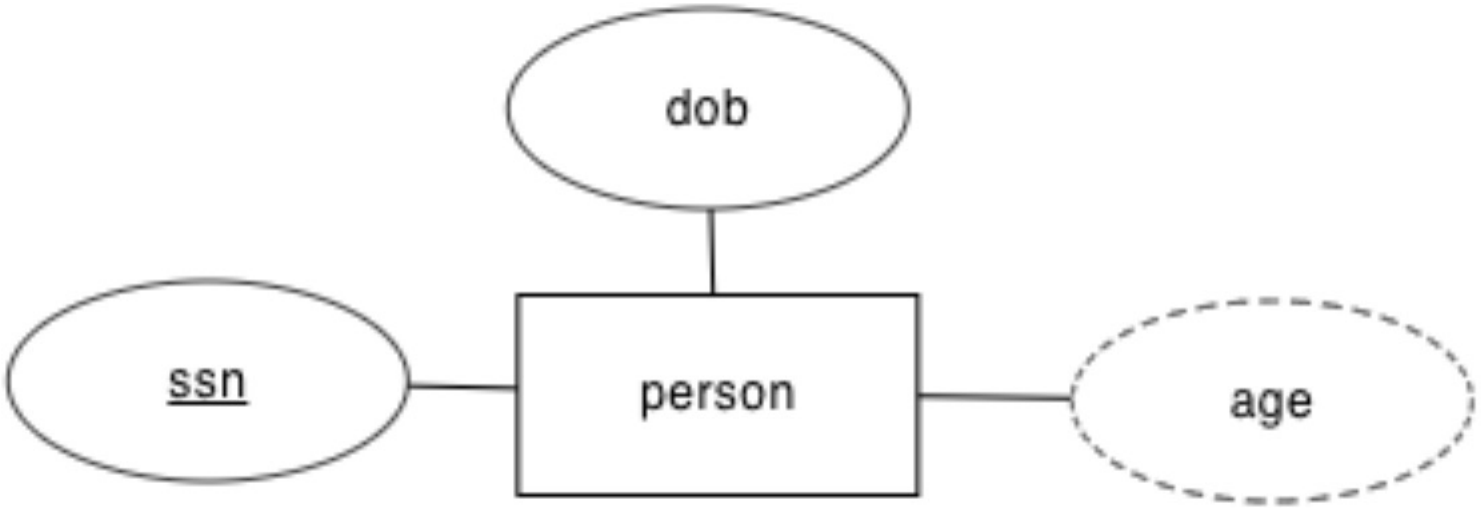
- Simple: single/atomic value, e.g, SSN or Age

- Derived: Derived from another attribute: age from date of birth.

Types of Attributes

- Simple: single/atomic value, e.g, SSN or Age

- Derived: Derived from another attribute: age from date of birth.



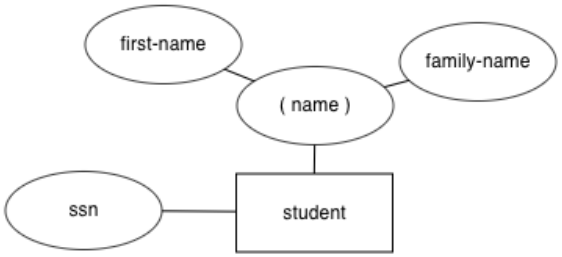
Types of Attributes

- Composite: ***Combination*** of ***simple attributes, e.g. name***

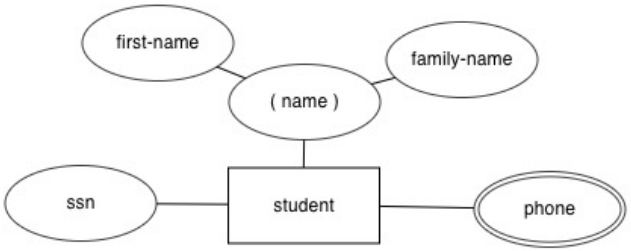
- Multi-valued: *Multiple values* for a attribute. E.g., phone-no

Types of Attributes

- Composite: ***Combination*** of ***simple attributes, e.g. name***



- Multi-valued: *Multiple values* for a attribute. E.g., *phone*



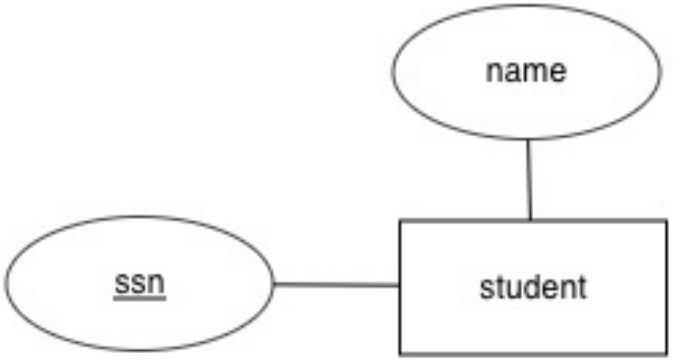
Key Attribute

- Key: an attribute which has unique values, e.g. SSN

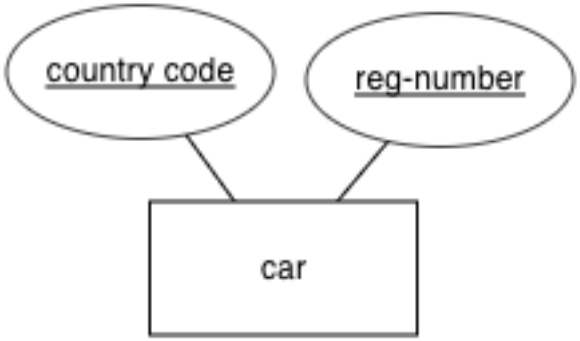
- Composite key: **Combination** of attributes makes it unique

Key Attribute

- Key: an attribute which has unique values, e.g. SSN



- Composite key: **Combination** of attributes makes it unique

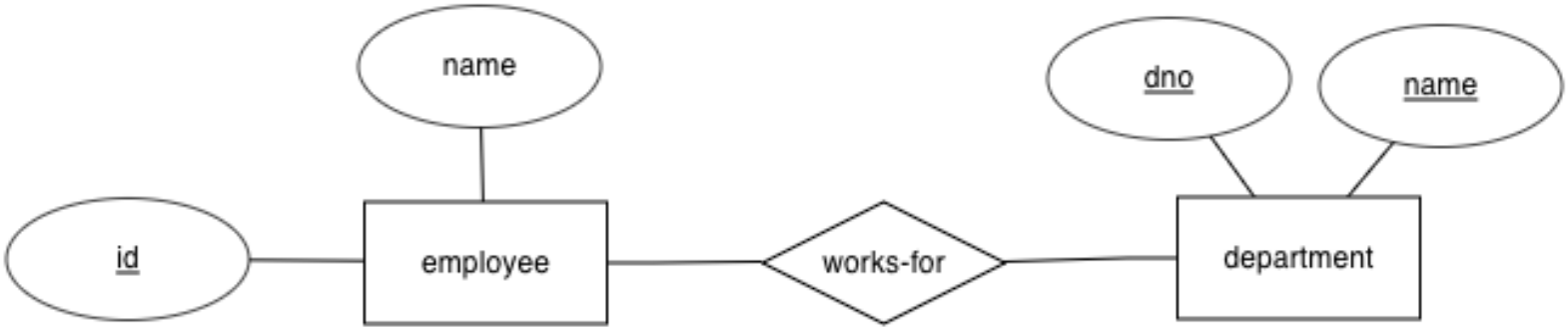


Relationship

- Relationship: ***relates*** two entity, e.g. *employee works-for a department*
- *Degree: two…more*

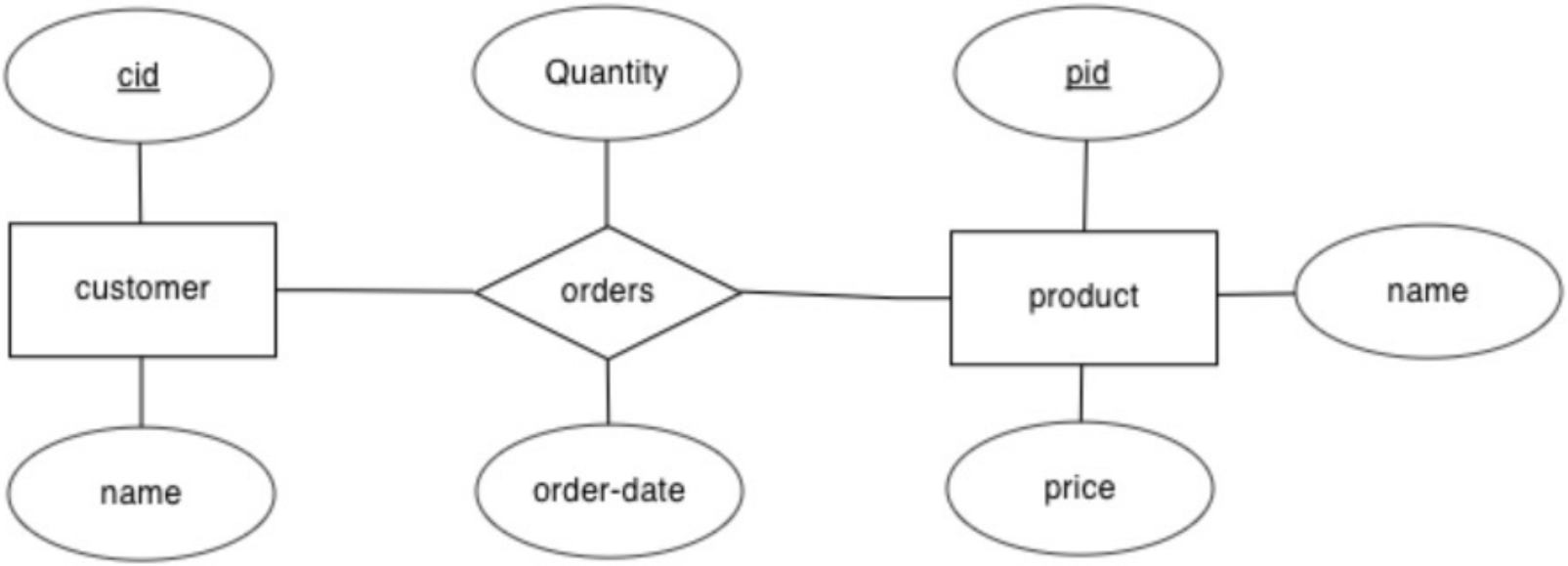
Relationship

- Relationship: ***relates*** two (or more) entity, e.g. *employee works-for a department*
- *Degree: two…more*



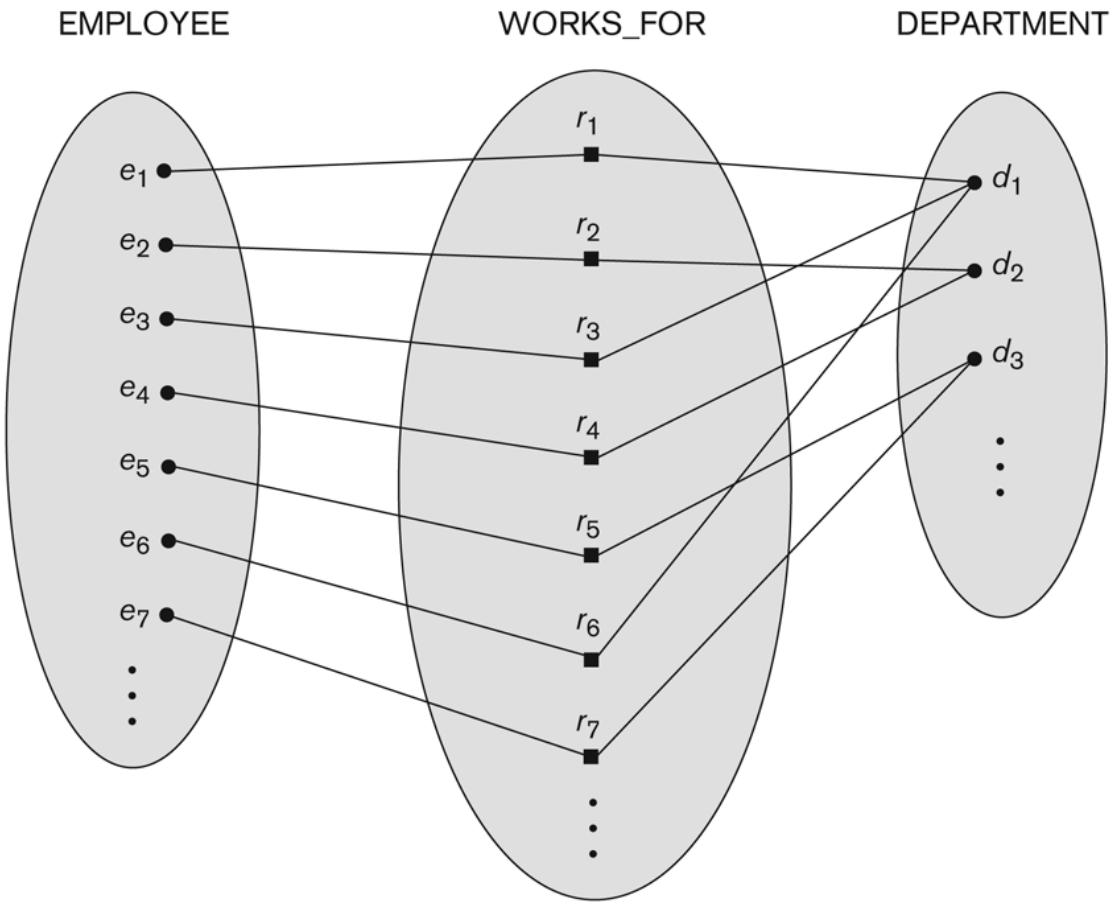
Relationship

- Relationship: ***relates*** two (or more) entity, e.g. *employee works-for a department*
- *Degree: two…more*



Relationship

- Relationship: ***relates*** two entity, e.g. *employee works-for a department*

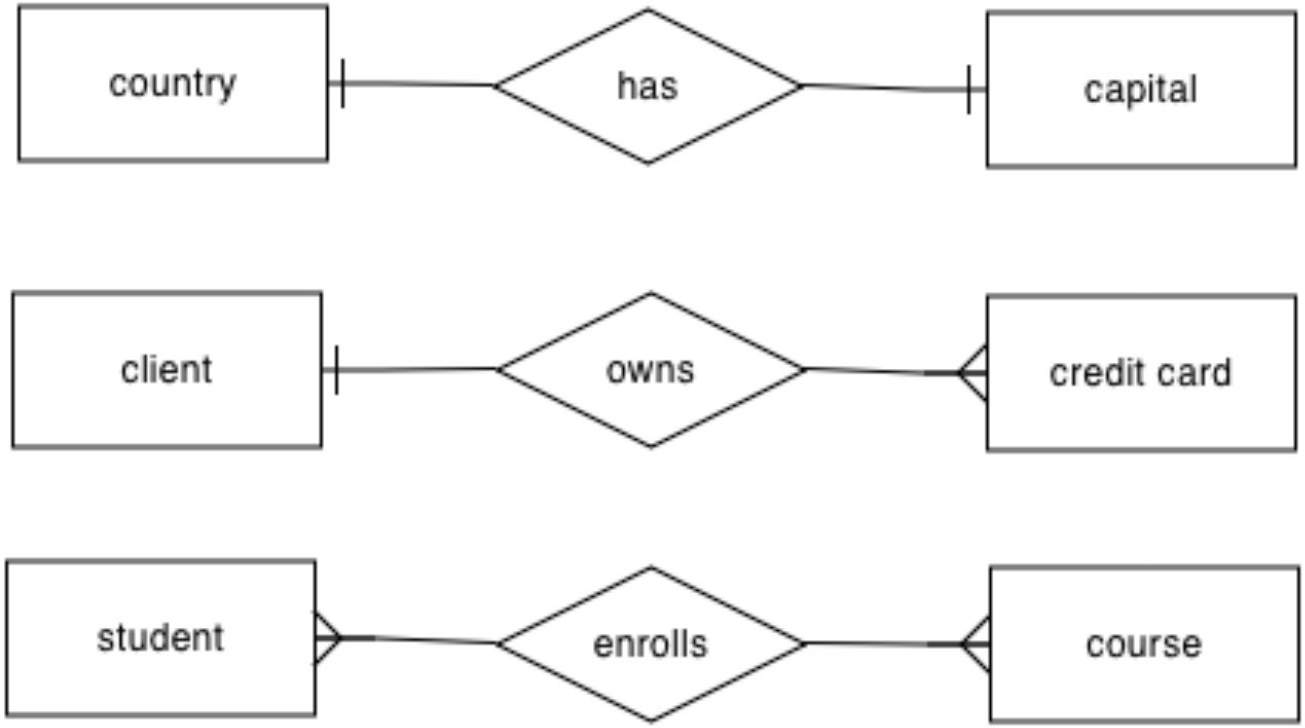


Constraints on Relationships

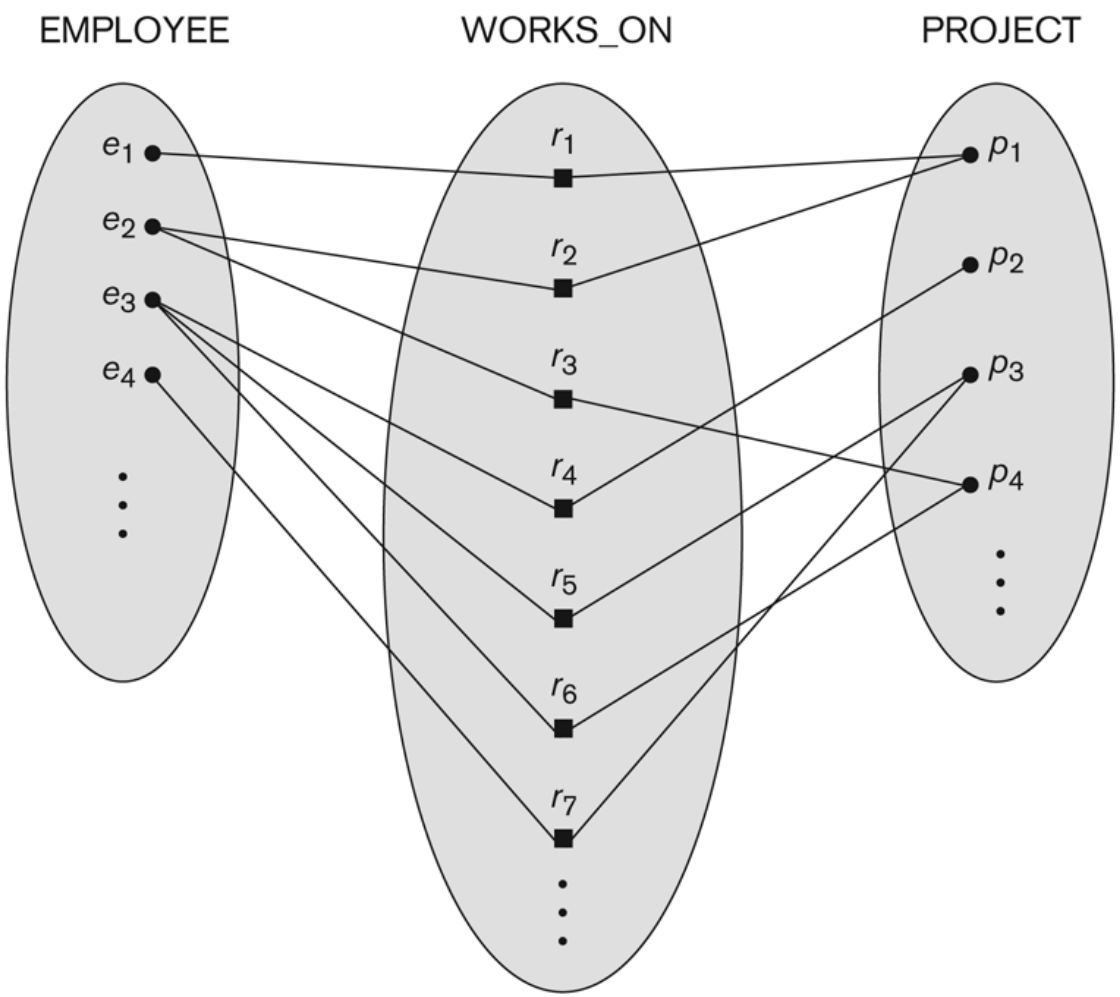
- **Cardinality Ratio** / Ratio constraints
  - One-to-one (1:1)
  - One-to-many (1:N) or Many-to-one (N:1)
  - Many-to-many (M:N)

Constraints on Relationships

- **Cardinality Ratio** / Ratio constraints
  - One-to-one (1:1)
  - One-to-many (1:N) or Many-to-one (N:1)
  - Many-to-many (M:N)

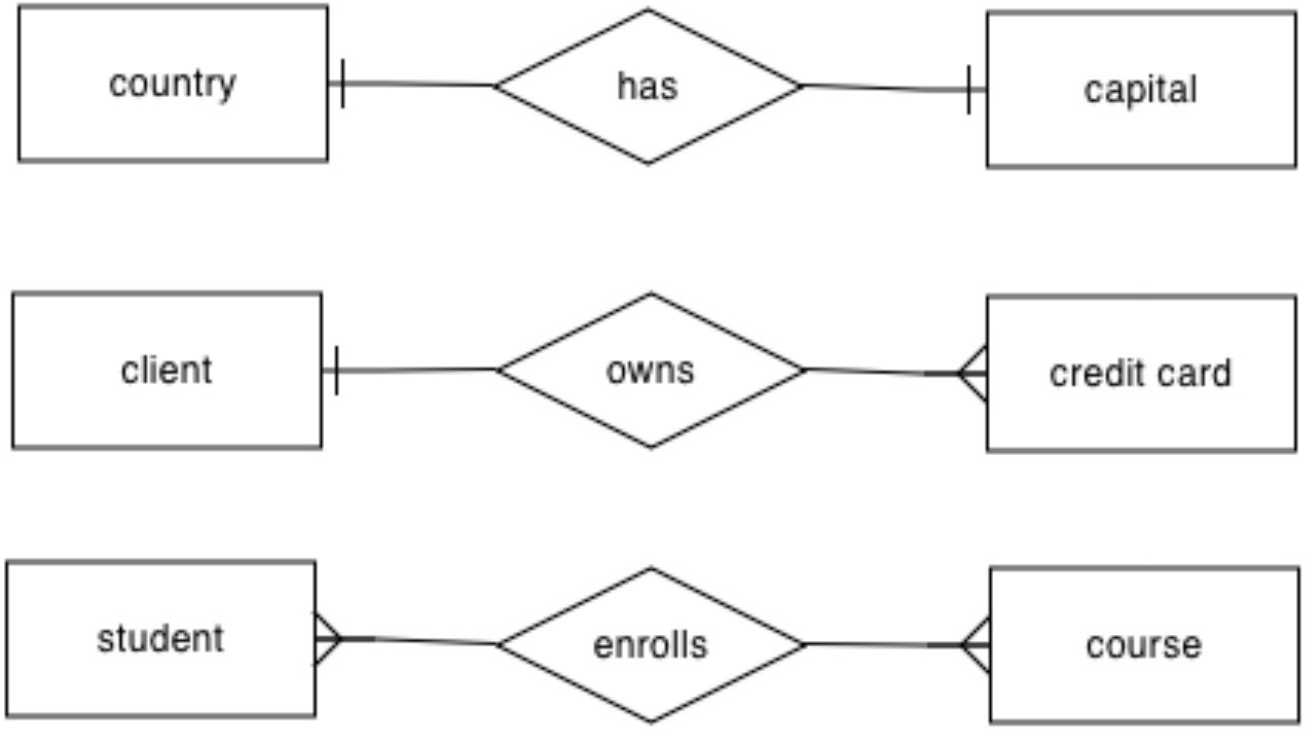


Many-to-Many Relationship



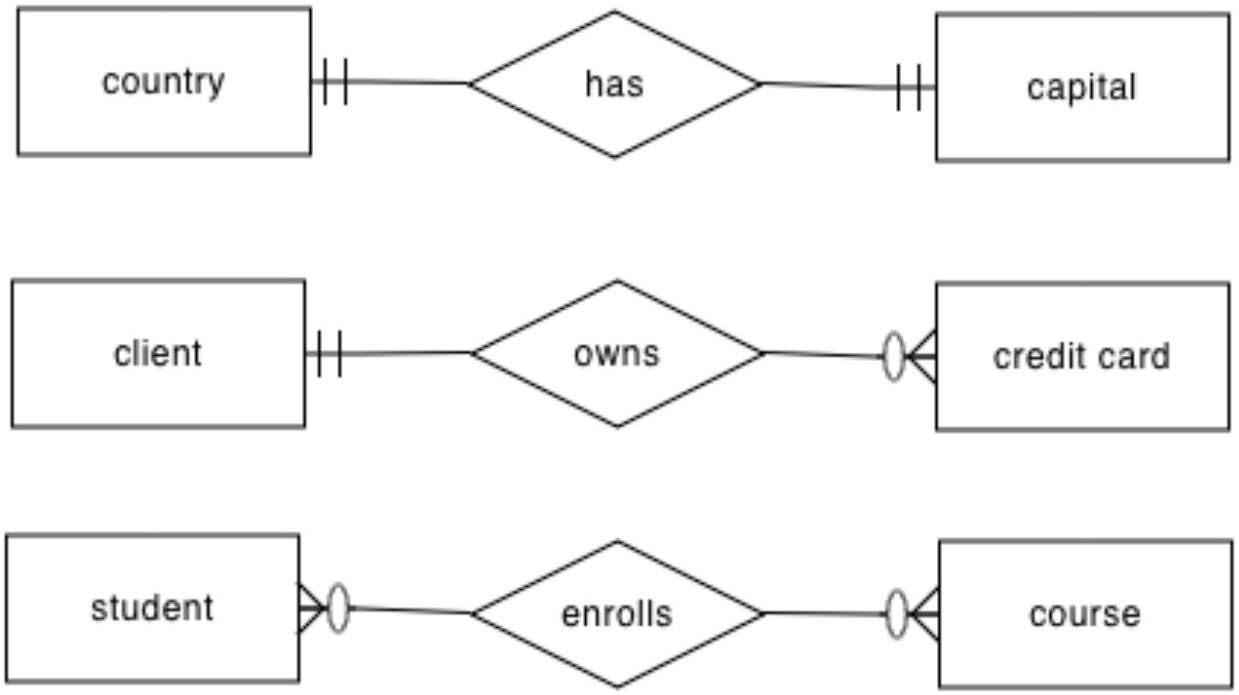
Constraints on Relationships

- **Participation Constraint /** Dependency Constraint**:** *minimum* participation
  - Zero/optional
  - One or more/mandatory/total-participation



Constraints on Relationships

- **Participation Constraint /** Dependency Constraint**:** *minimum* participation
  - Zero/optional
  - One or more/mandatory/total-participation

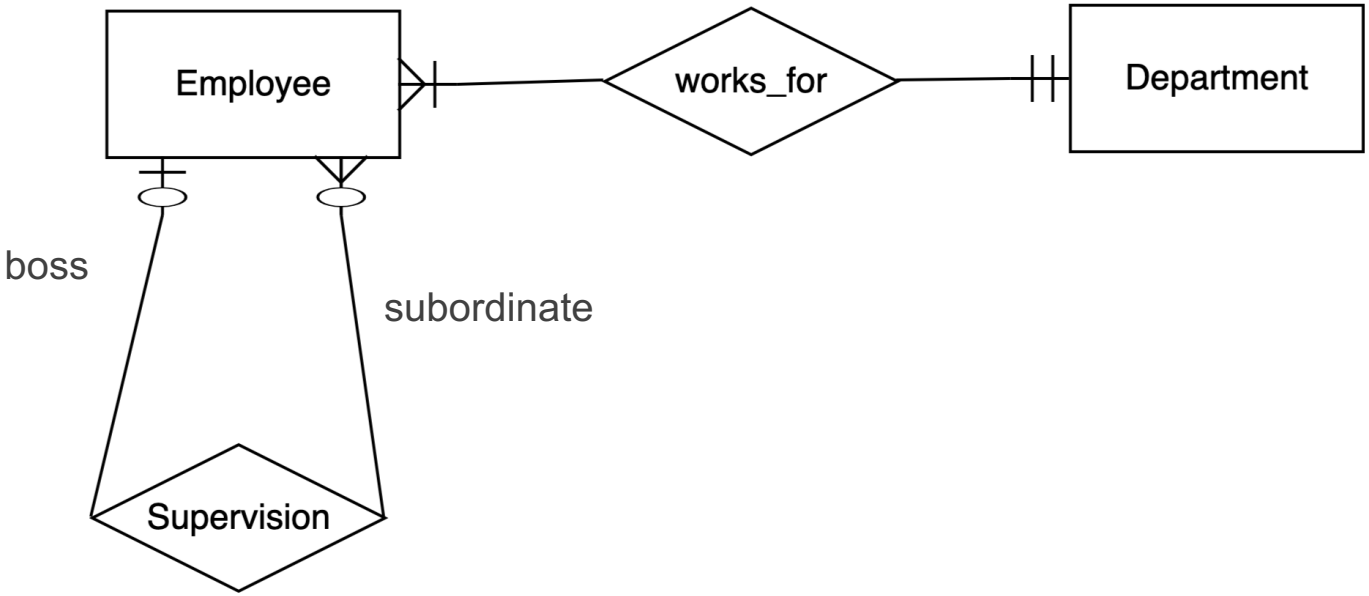


Recursive Relationship

- A relationship between same entity,
  - ***supervisor (or boss) role vs supervisee (or subordinate) role***

Recursive Relationship

- A relationship between same entity,
  - *supervisor (or **boss**) role vs supervisee (or **subordinate**) role*

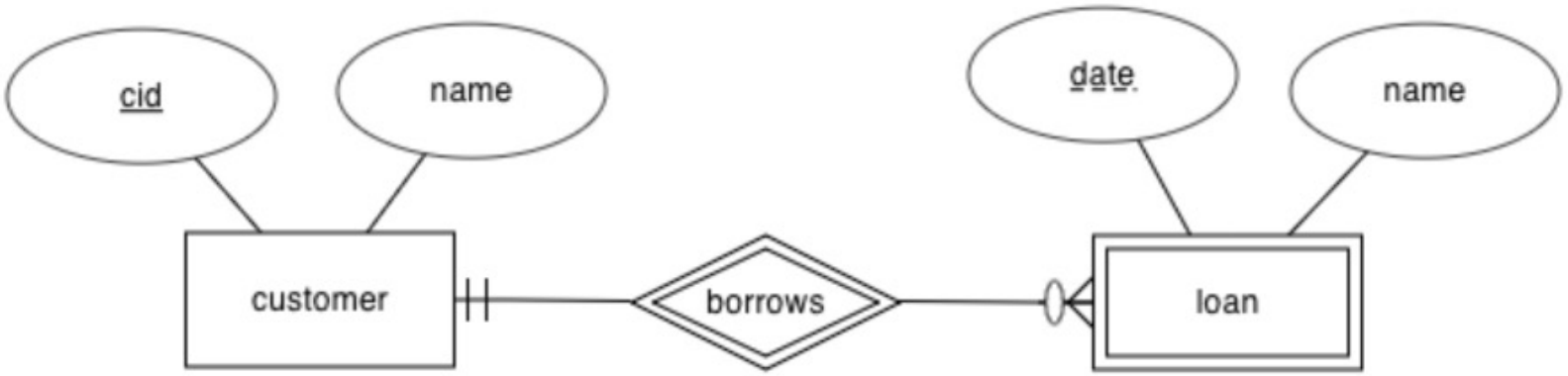


Weak Entity

- An entity **without** a key. But has **partial-key(s)**
- Dependent on other entity (owner entity)
- **Participation:** mandatory/total-participation

Weak Entity

- An entity **without** a key. But has **partial-key(s)**
- Dependent on other entity (owner entity)
- **Participation:** mandatory/total-participation



Exercise 1:

- Each department has exactly one manager. An employee might be a manager of maximum one department.

Exercise 1:

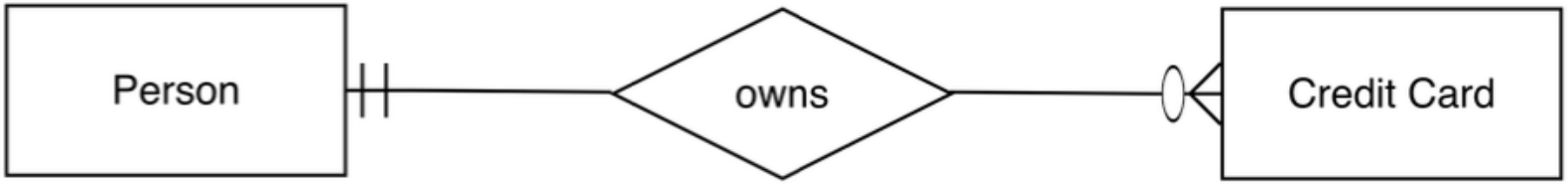
- Each department has exactly one manager. An employee might be a manager of maximum one department.

Exercise 2:

- One person might own many credit cards, each credit card must have one owner.

Exercise 2:

- One person might own many credit cards, each credit card must have one owner.



Exercise 3:

- A teacher might teach many courses this term, each course must have one or more teachers.

Exercise 3:

- A teacher might teach many courses this term, each course must have one or more teachers.

Exercise 4:

The company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department and a number of other people working for it. 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.

For each employee, the following information is kept: name, social security number, address, salary and sex. 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.

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.

Exercise 4:

The company is organized into DEPARTMENTs. Each **department** **has** a **name, number** and an **employee** who ***manages*** the department and a number of other people working for it. We keep track of the **start date** of the department manager. A department may have several **locations**.

Each department ***controls*** a number of **PROJECT**s. Each project has a **unique name, unique number** and is located at a single location.

For each employee, the following information is kept: name, social security number, address, salary and sex. Each employee may *have* a number of **DEPENDENT**s. For each **dependent**, the DB keeps a record of **name, sex, birthdate**, and relationship to the employee. 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

Exercise 4:

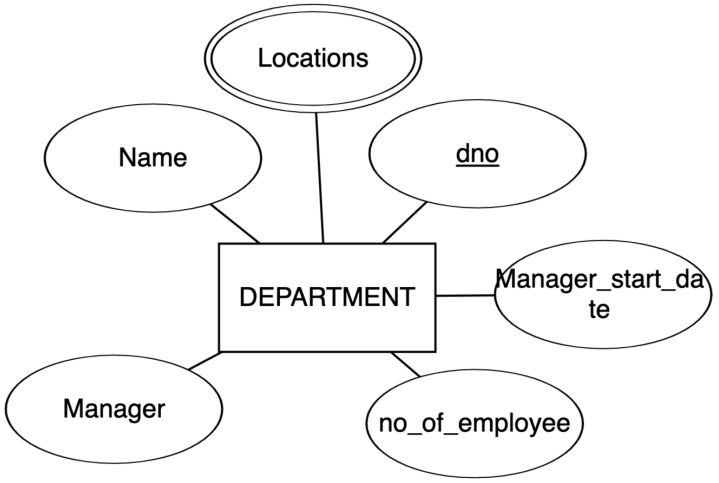
Exercise 4:

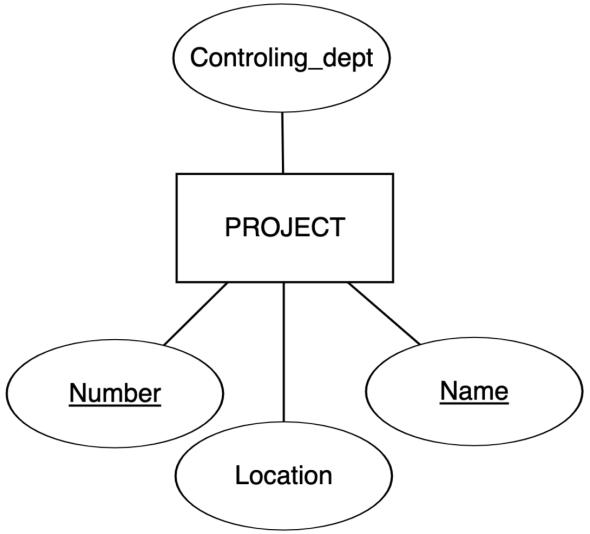
The company is organized into DEPARTMENTs. Each **department has** a **name, number** and an **employee** who ***manages*** the department and a number of other people working for it. We keep track of the **start date** of the department manager. A department may have several **locations**.

Each department ***controls*** a number of **PROJECT**s. Each project has a **unique name, unique number** and is located at a single location.

For each employee, the following information is kept: name, social security number, address, salary and sex. Each employee may *have* a number of **DEPENDENT**s. For each **dependent**, the DB keeps a record of **name, sex, birthdate**, and relationship to the employee. 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.





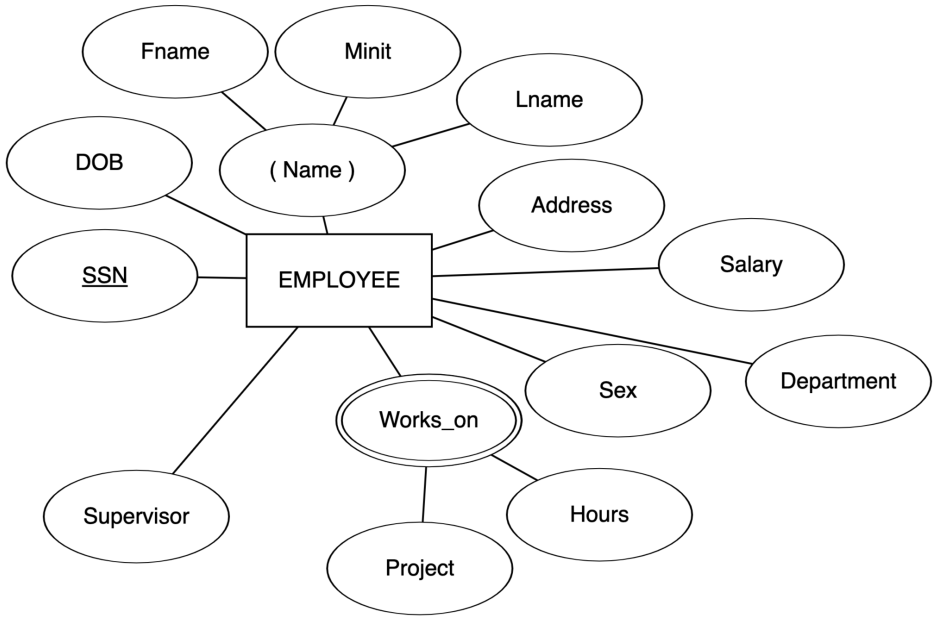
Exercise 4:

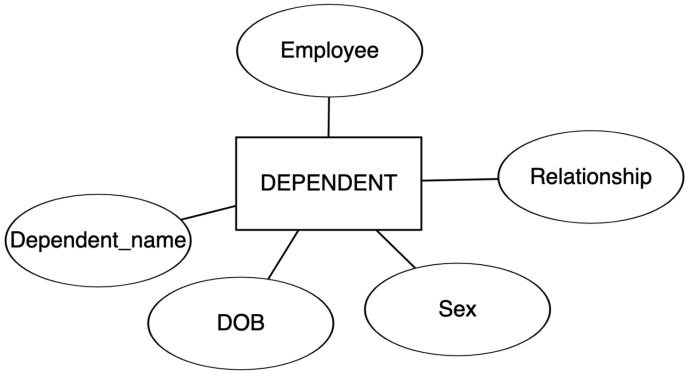
The company is organized into DEPARTMENTs. Each **department has** a **name, number** and an **employee** who ***manages*** the department and a number of other people working for it. We keep track of the **start date** of the department manager. A department may have several **locations**.

Each department ***controls*** a number of **PROJECT**s. Each project has a **unique name, unique number** and is located at a single location.

For each employee, the following information is kept: name, social security number, address, salary and sex. Each employee may *have* a number of **DEPENDENT**s. For each **dependent**, the DB keeps a record of **name, sex, birthdate**, and relationship to the employee. 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.





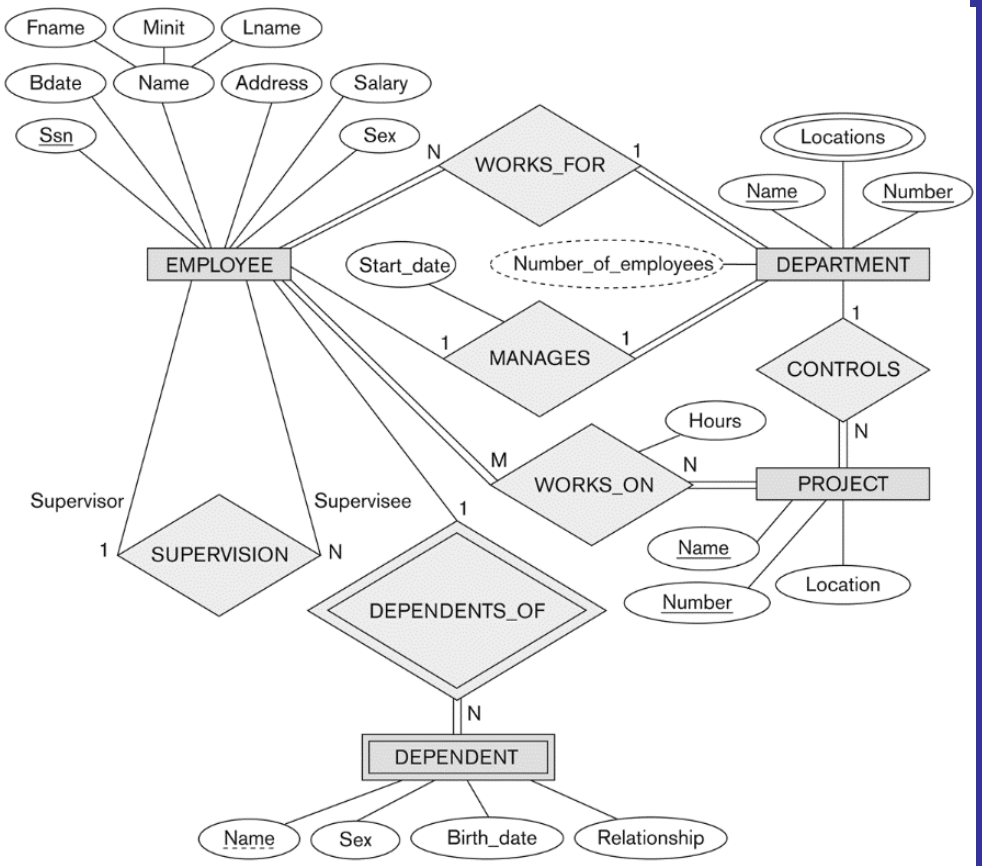
Exercise 4 (Using Different notation):

The company is organized into DEPARTMENTs. Each **department has** a **name, number** and an **employee** who ***manages*** the department and a number of other people working for it. We keep track of the **start date** of the department manager. A department may have several **locations**.

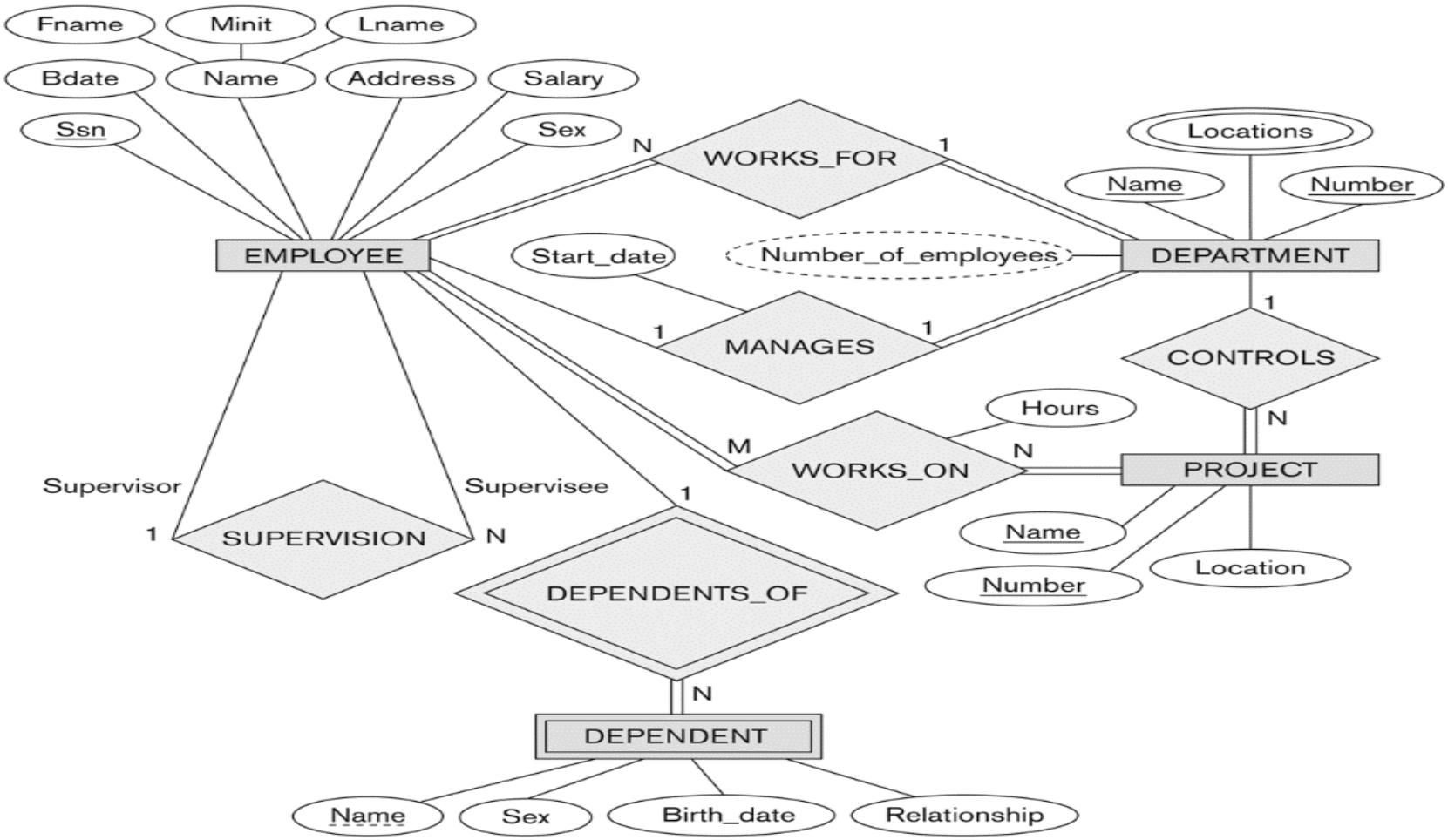
Each department ***controls*** a number of **PROJECT**s. Each project has a **unique name, unique number** and is located at a single location.

For each employee, the following information is kept: name, social security number, address, salary and sex. Each employee may *have* a number of **DEPENDENT**s. For each **dependent**, the DB keeps a record of **name, sex, birthdate**, and relationship to the employee. 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.

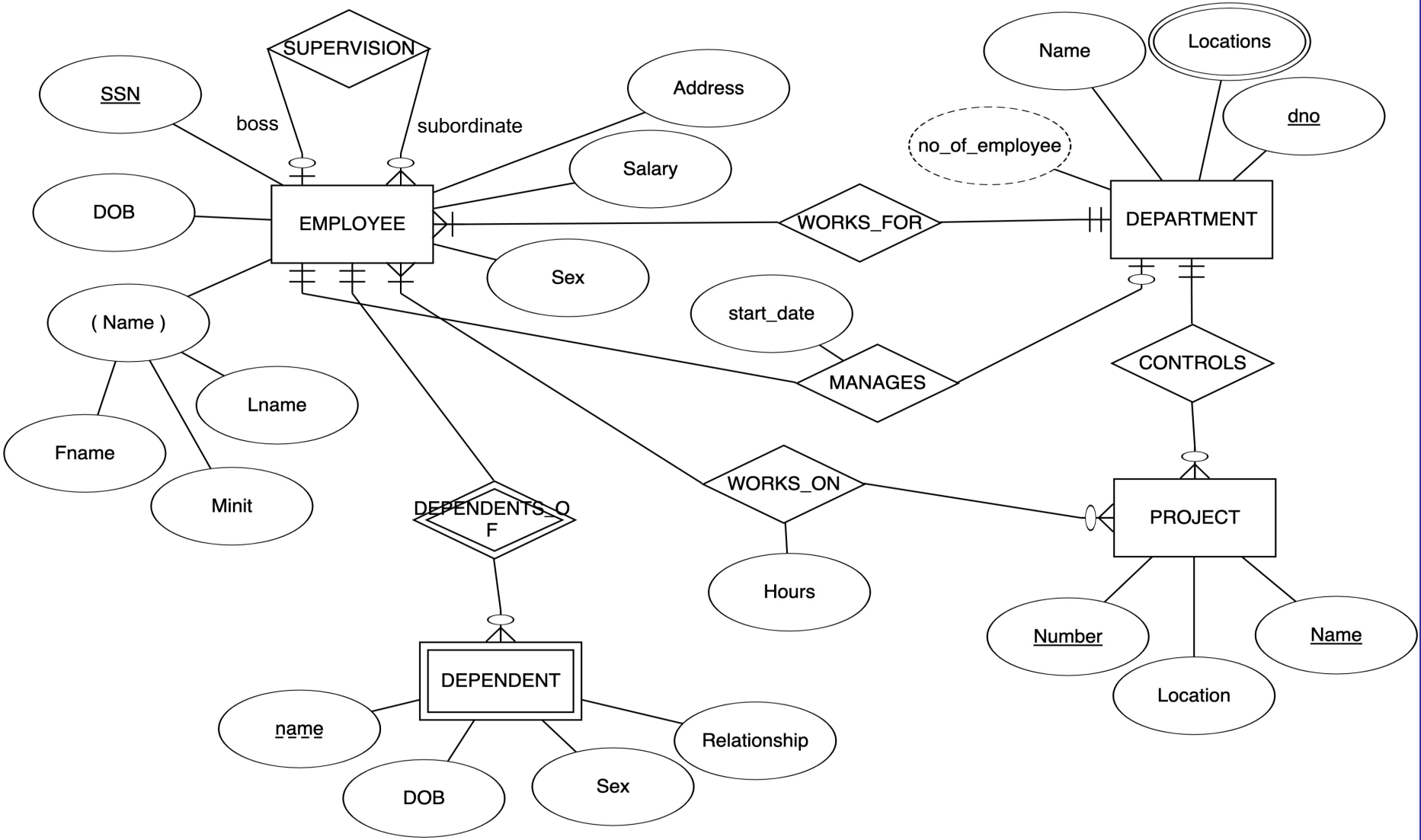
It is required to keep track of the *direct supervisor* of each employee.



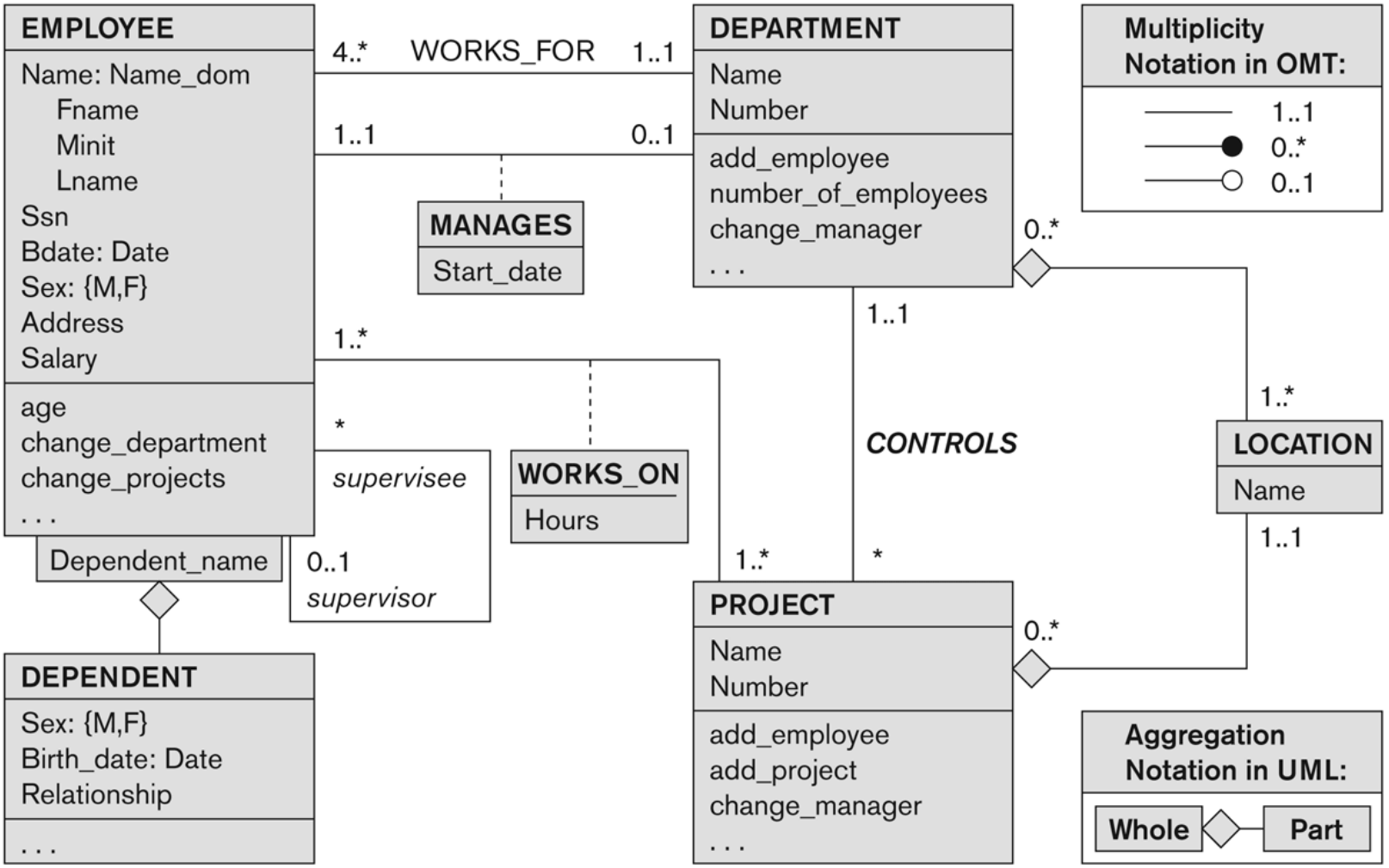
Exercise 4 (model from prev slide)



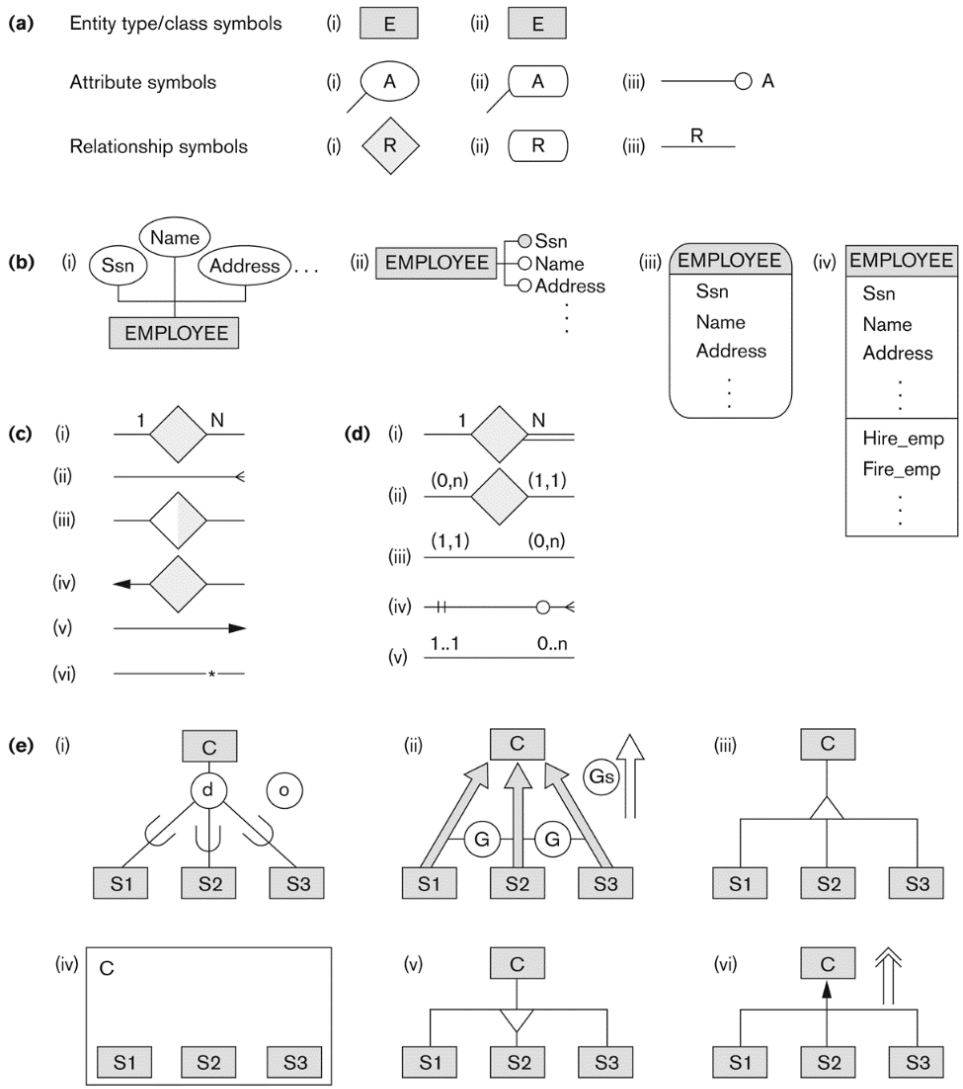
Exercise 4 (Another notation):



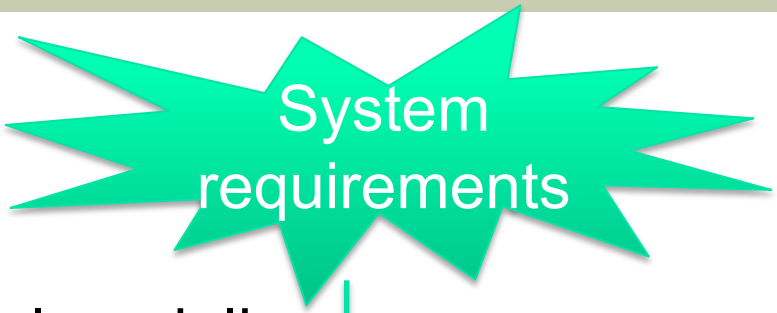
UML class diagram for COMPANY database schema



Other alternative diagrammatic notations



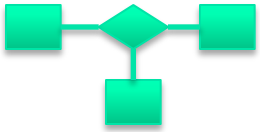
Database design: full picture



A university offers courses to students.
For every student we register name, surname, address, year and student ID.
Every course instance is given on a specific year by a professor,
for whom we store the income and the supervisor.
For each exam we save the date and the grade of the student.

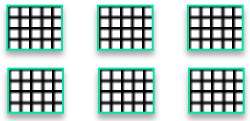
Natural language

Conceptual modeling



More formal
Not ambiguous

Logical modeling



Entity-Relationship diagram

No implementation or DBMS specific details

Physical design

Relational model

Why a conceptual model (e.g. ER)?

- More formal than natural language.
  - Avoid misconceptions/multiple interpretations.
- Implementation independent (of DBMS).
  - Less technical details.
- High-level description.
  - Easier for people without a technical background
- Documentation.
- Comes with model transformations to be mapped to an implementation data model.
- CASE Tools automates ER to DB design

Published: Ex-1.1-ER Diagrams.pdf

For drawing ER-Models: www.erdplus.com

Full slide from book authors:

Chapter03.pdf

May be useful for revision/exam for exam (?)