Database Design

Anna Monreale

Need to design

- Database are often born designless,from a huge spreadsheet
- Anomalies arise, because of redundancy
- Redundancy generate errors
- Design must involve the user

Anomalies

Name	Surname	Address	StudId	Subject	Date	Grade
Mario	Addis	Via Roma	354765	BD	1/1/13	28
Luca	Bini	Via Pola	354234	BD	2/3/12	18
Mario	Addi	Via Roma	354765	Alg	1/1/13	27
Luca	Bini	Via Pola	354234	Pro	2/5/12	30
Luca	Bini	Via Bari	354234	Lab	3/4/12	24

Phases for DB realization

- **User requirements analysis & specification**
  - collecting **user needs** and normalizing them according to standards
- **Conceptual design**
  - is the phase in which requirements are formalized and integrated into a **global conceptual schema**
  - using a DBMS-independent conceptual language
- **Logical design**
  - the conceptual schema is mapped into a **logical schema** using the data model supported by the DBMS chosen for the implementation
- **Physical design**
  - concerns the selection of the **data structures** used to store and retrieve the data.

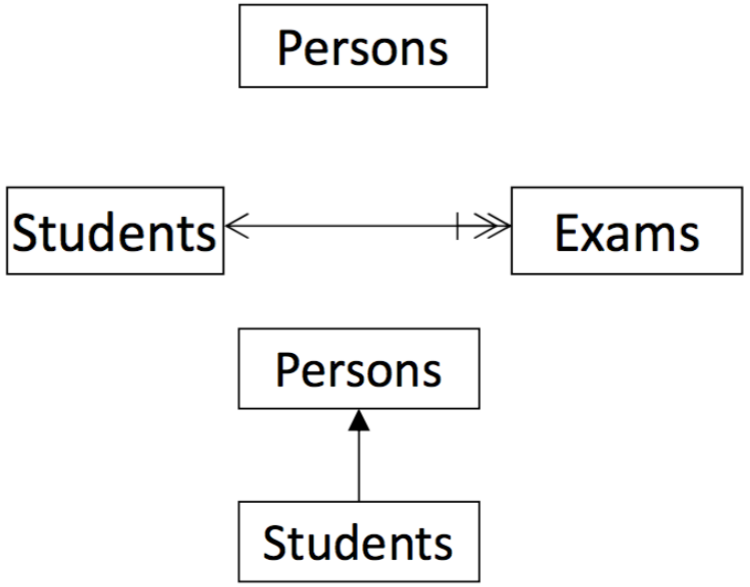
User Requirements Analysis

- Hard activity because hard to standardize

- Suggestions
  - Involve the users many times for continuous checks
  - Consider the point of view of the applications users
  - To be sure that you are using a common language
  - Identify case studies that you can discuss in details to identify the properties to be captured by the model

An Object Oriented Language for data design

- Realization of a **diagram** representing the conceptual model of the database
- Components:
  - **Classes** (collections)
  - **Relationships** among classes
  - **Sub-collections** links



Class Diagram

- Phase of Analysis
  - Need to adopt the right level of abstraction
- In particular
  - We don’t need all attributes
  - Type of attributes is not necessary

Example: University DB

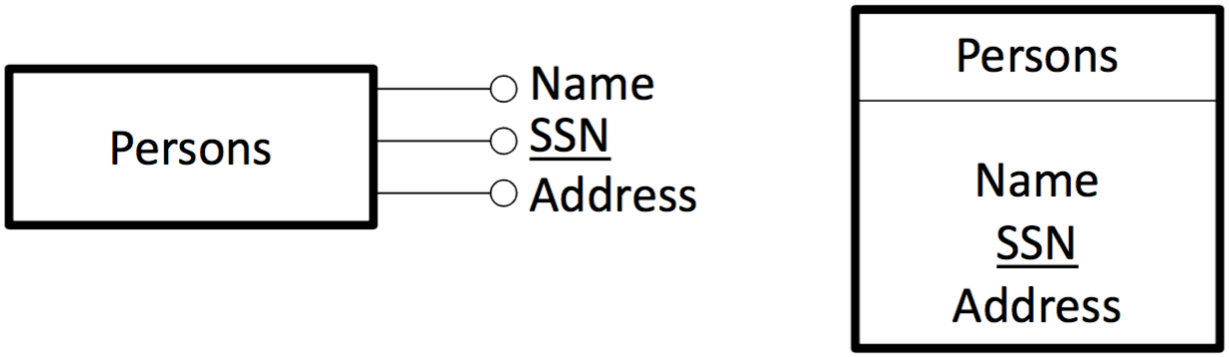
- We need to design the database for managing data about **courses** of computer science degree at the University of Pisa
- The system must manage data about **students** of the master program and bachelor program. For each one we need to maintain data related to the students **exams**.
- We need to record data about **courses** and the **students exams** for each course.
- For each course we want to record **teachers**, who may be more than one. Moreover a **teacher** may be **internal** or **external**.
- For each teacher we have one or more **phone numbers**.
- For each student we need to record the supervisor (a **teacher**). Bachelor students may ask a supervisor only when they are attending the third year.
- Lastly the system must maintain information about the tutoring activities of master students, that help bachelor students.

Classes

- **“Concepts” of the reality to be modelled**
  - facts, people, things,
  - examples: student, course, exam, teacher
- **Instances of a class**
  - entities, objects of the reality to be modelled
- **Classes have attributes**
  - Properties relevant for the application

Class with attributes

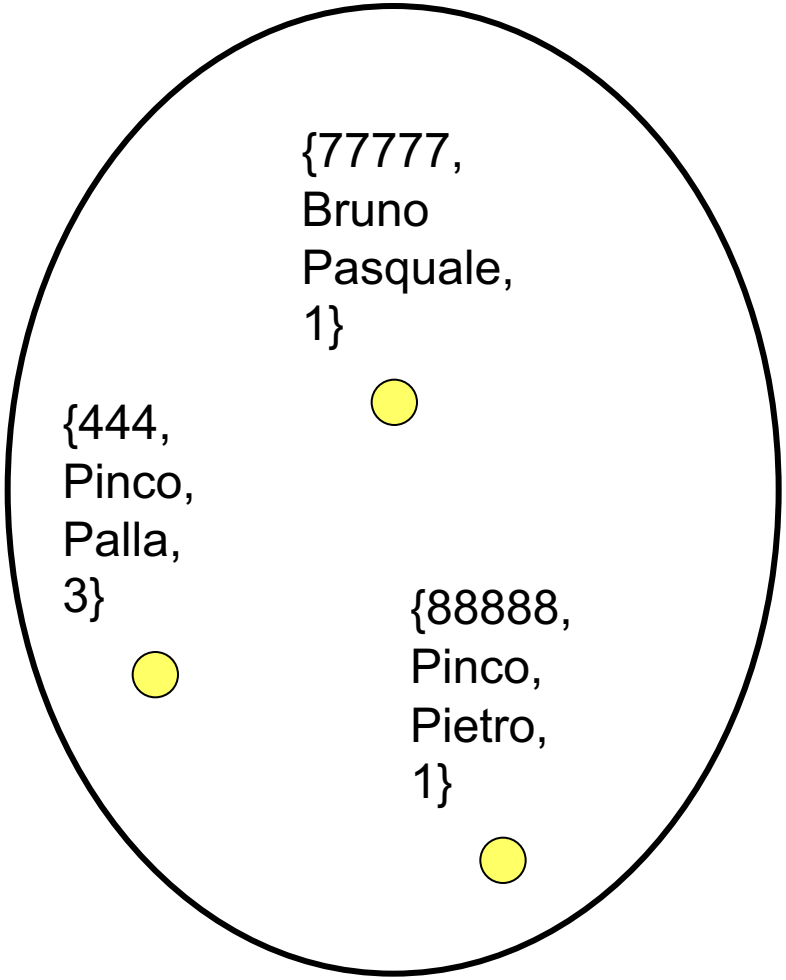
- A person class,with attributes:
  - Name
  - SSN (key)
  - Address



Classes

- Instances of the classes

Students
studentID
surname
name
year

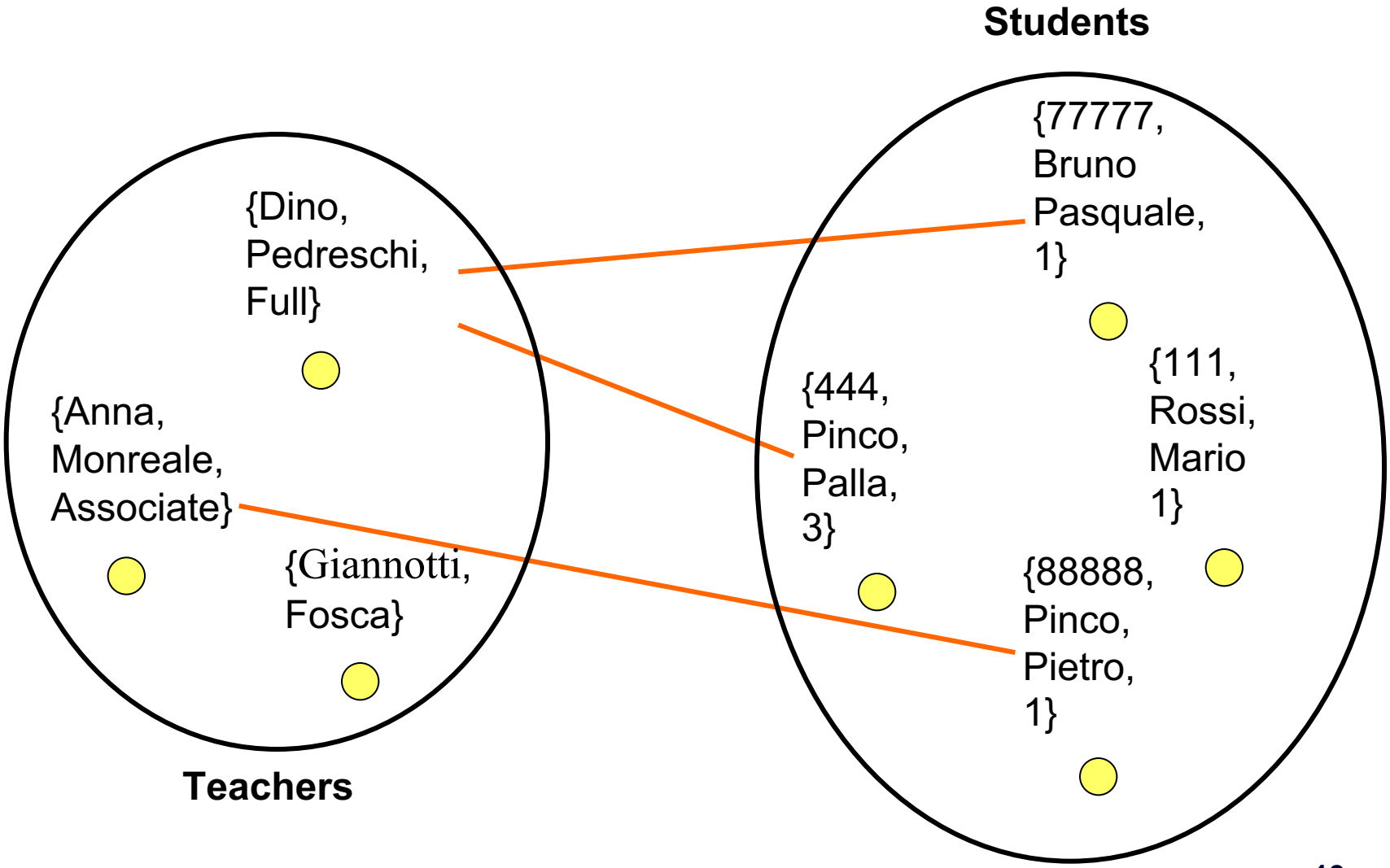


Relationship

- **Relationship between classes**
  - Logic link relevant for the application
  - ex: **teaching** between teacher and course
  - ex: student **passes** an exam

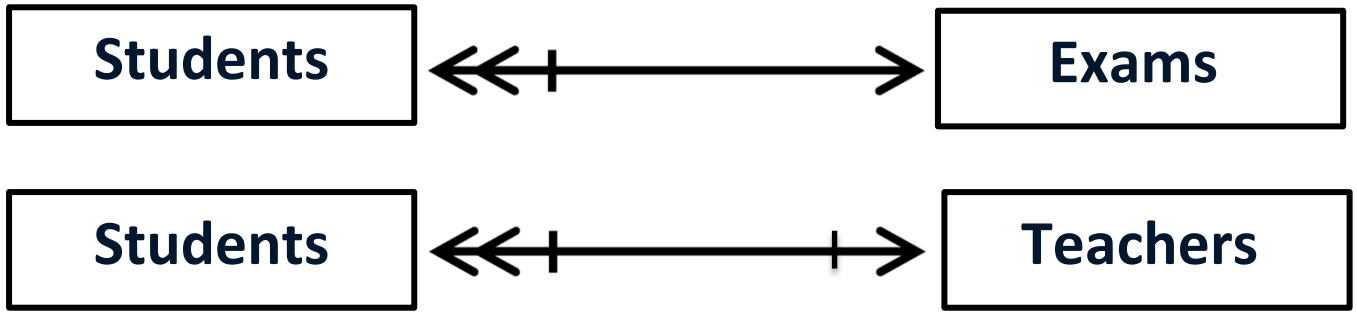
- **Instance of a relationship**
  - A set of edges between instances belonging to the involved classes

Relationship: Instances

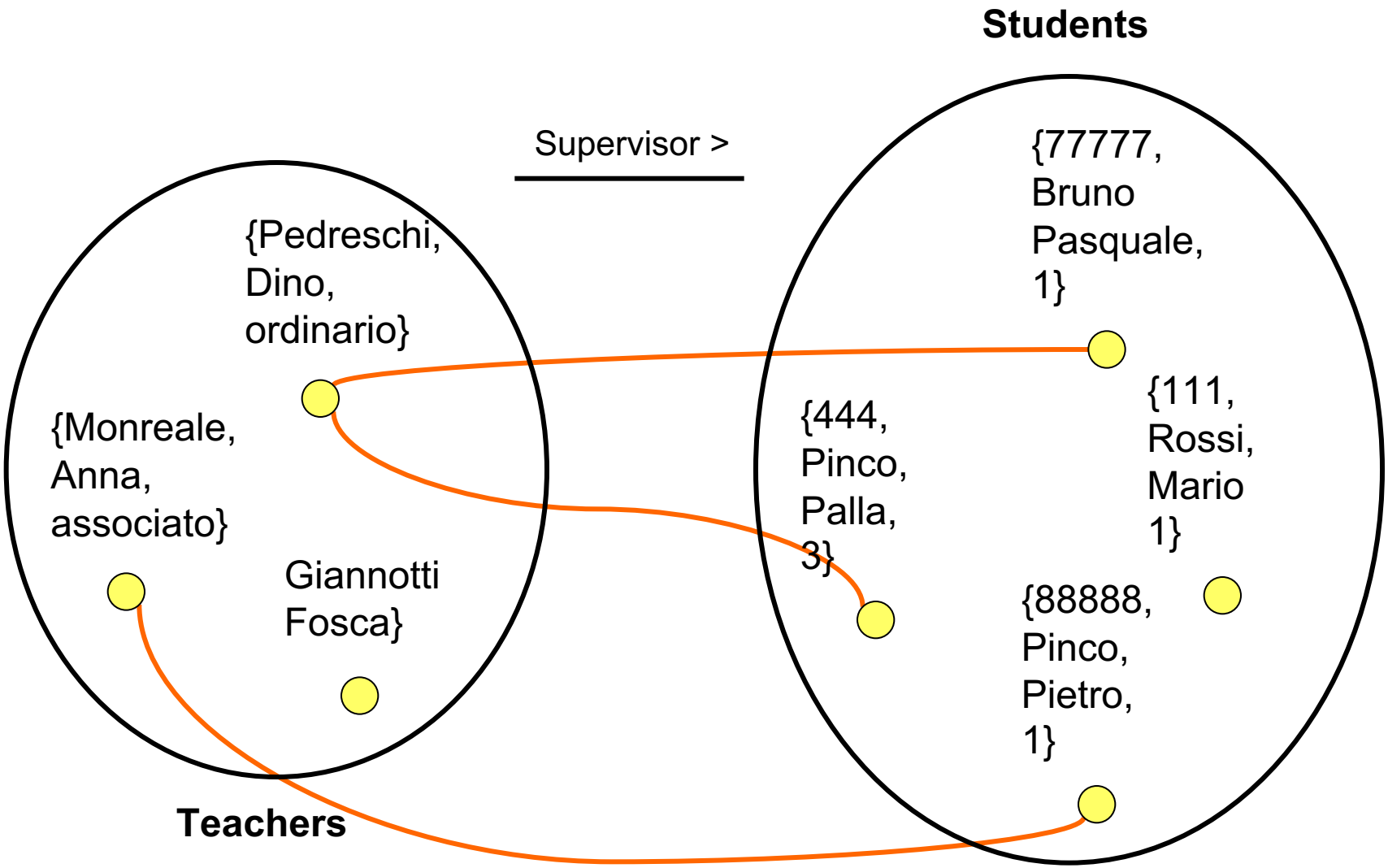


Cardinality

- **Constraints on relationships**
  - Constraints on the number of edges between instances of classes
- **Minimal Cardinality**
  - 0 or 1
- **Maximal Cardinality**
  - 1 or many



Cardinality

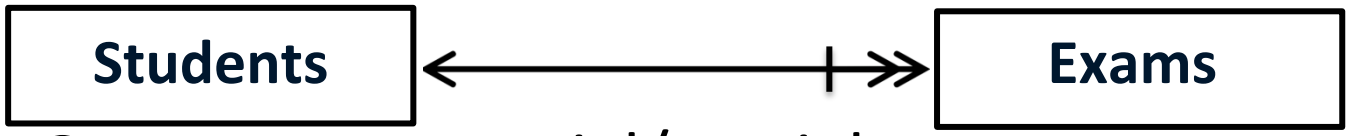


Cardinality (upper bound)

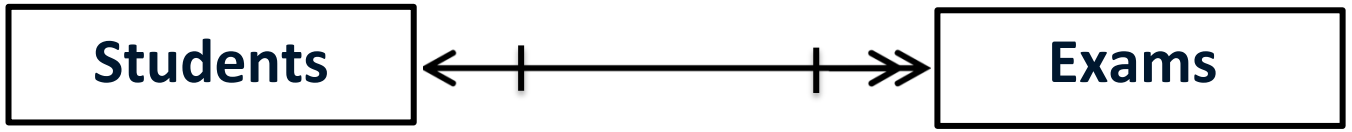
- Classification of the relationships wrt the cardinality
  - **One to One**: maximal cardinality equal to **1** for both classes
    - Manages[Managers, Departments]
  - **One to Many**: maximal cardinality equal to 1 for a class and **many (N)** for the other one
    - Owns[Persons, Cars]
  - **Many to Many**: maximal cardinality equal to **N** for both classes
    - Teaching[Course, Teacher]

Cardinality (lower bound)

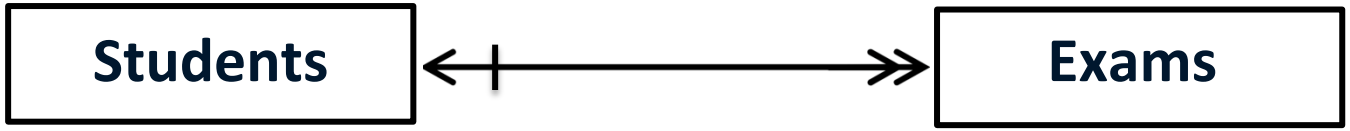
- Sixteen combintions:
  - One to many total/partial



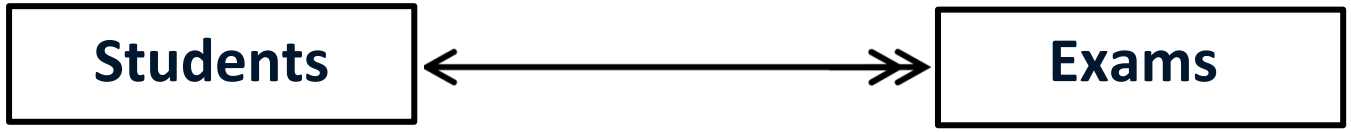
  - One to many partial/partial



  - One to many partial/total



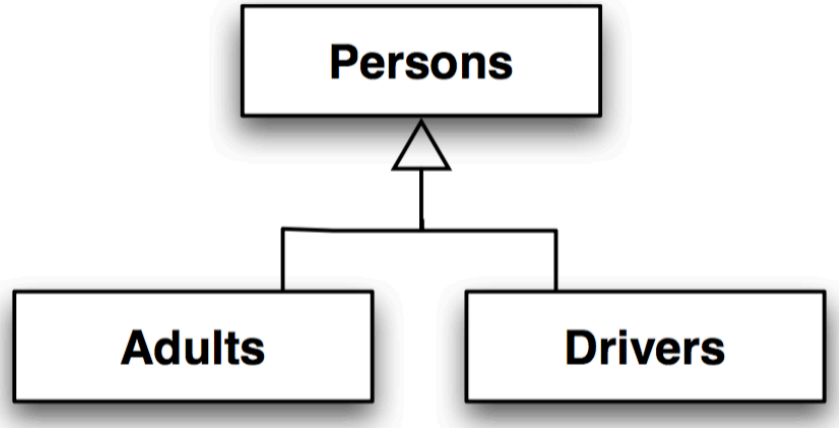
  - One to many total/total



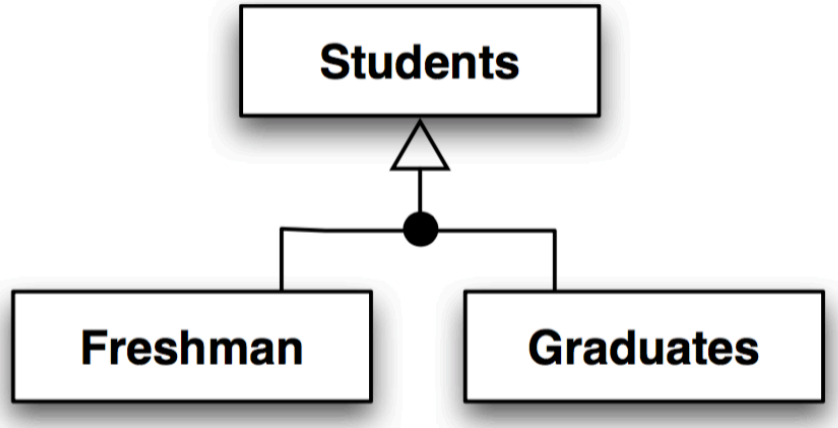
Class Hierarchy

- A subclass:
  - a subset of class elements, for which we plan to collect more information:
  - ex: Students is subclass of Persons
  - ex: **Internal** and **external** teachers are subclasses of the generic concept "**teacher**"

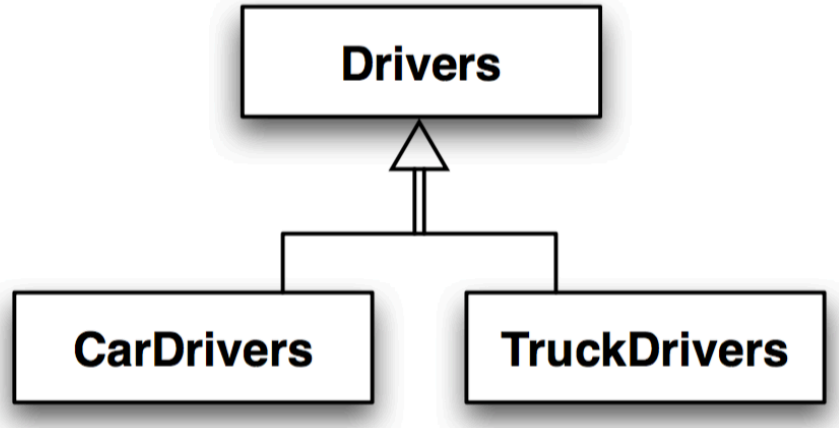
Class Hierarchy



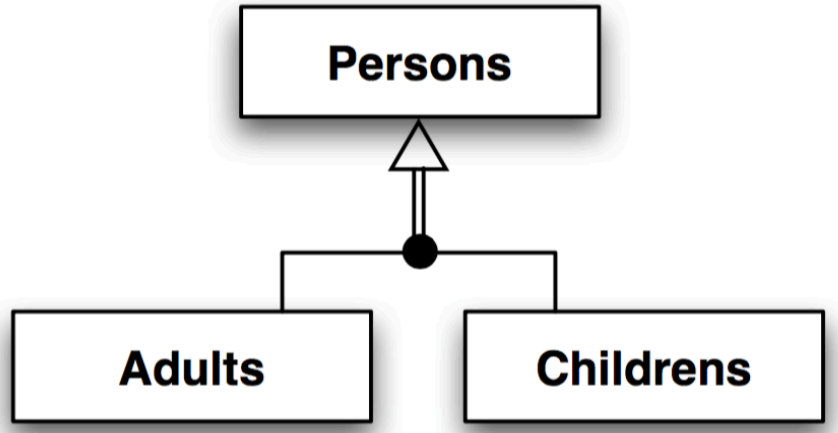
Overlapping subsets



Non overlapping subsets



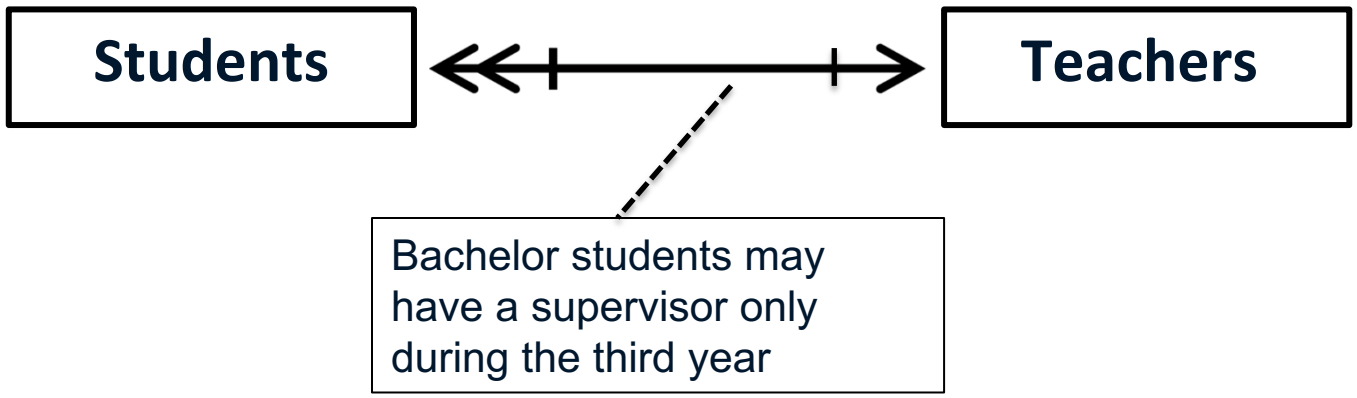
Overlapping cover



Non overlapping cover

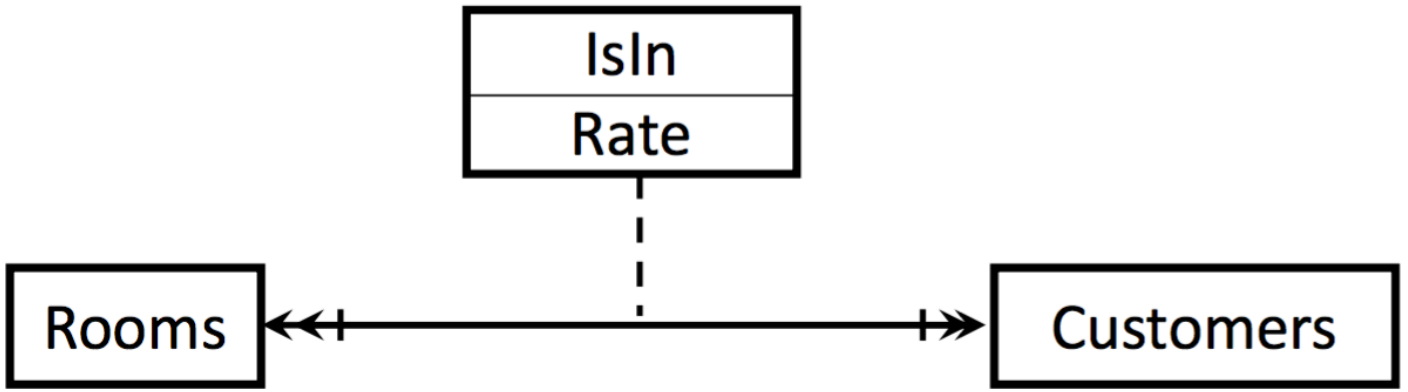
Notes

- Sometime it is necesary to add notes in the diagram to express some constraints
  - Ex: Bachelor students may ask a supervisor only when they are attending the third year.

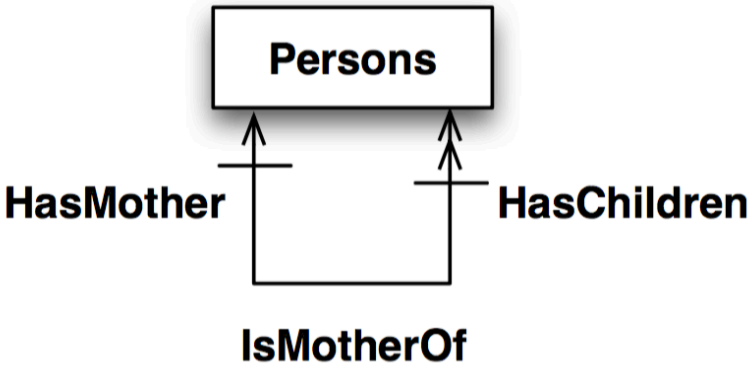


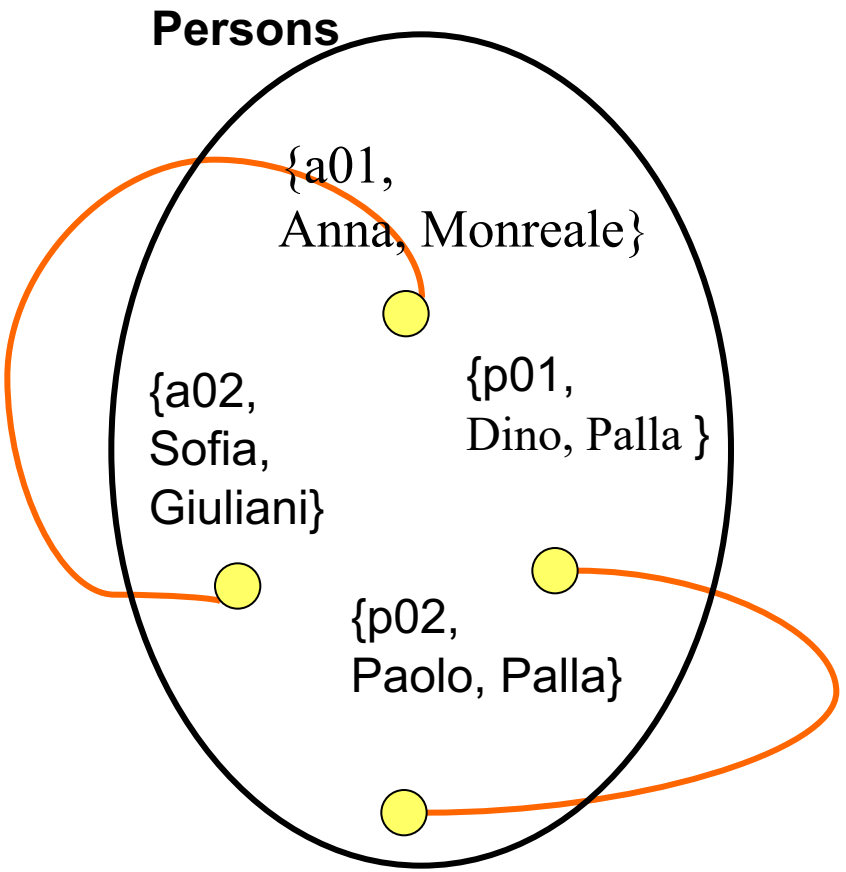
Relationshp with attributes

- Sometimes a relationship may have some properties that characterize each instance of the relationship
- **“John is occupying the room 105 at Le Meridien - Houston, at a $145 rate”**
- This is a relationship instance between persons and rooms, with a rate attribute



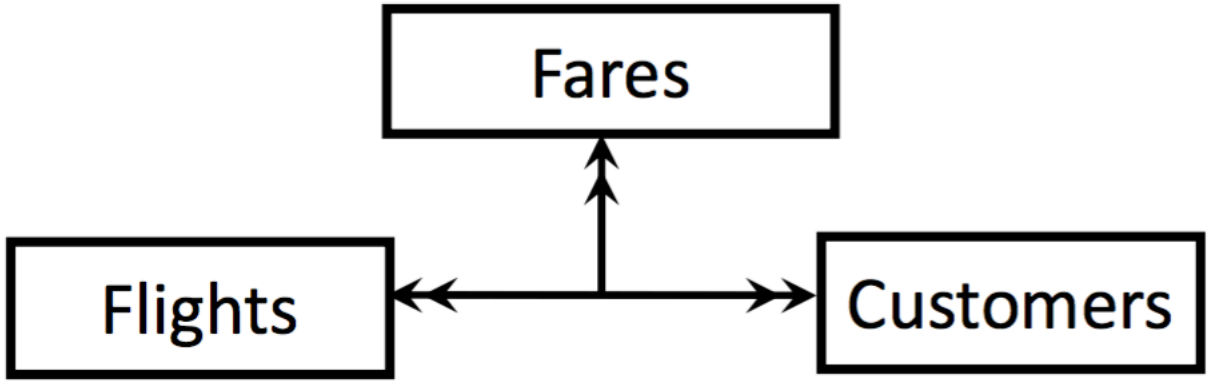
Recursive Relationships





Ternary Relationship

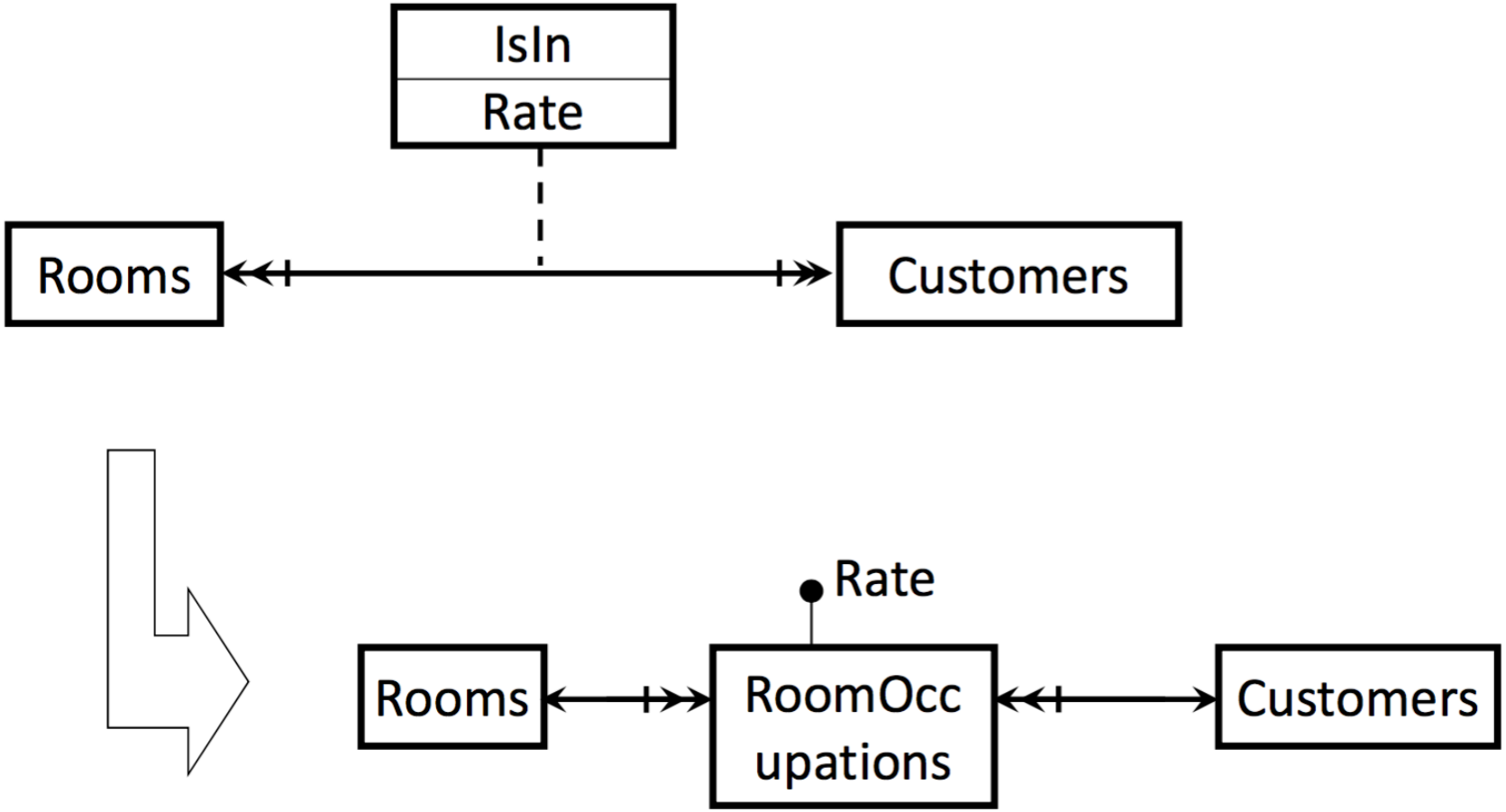
- Ternary facts exist also
- “John booked flight FK354/13-6-2000 with Y2 fare”



Keep it simple

- Whenever it makes sense, upgrade a **relationship with attributes**, or a **ternary** one, to a collection

From Attributes to classes



From ternary to new class