Lecture 5

Database design

Special Relations

Subclassing and weak entities

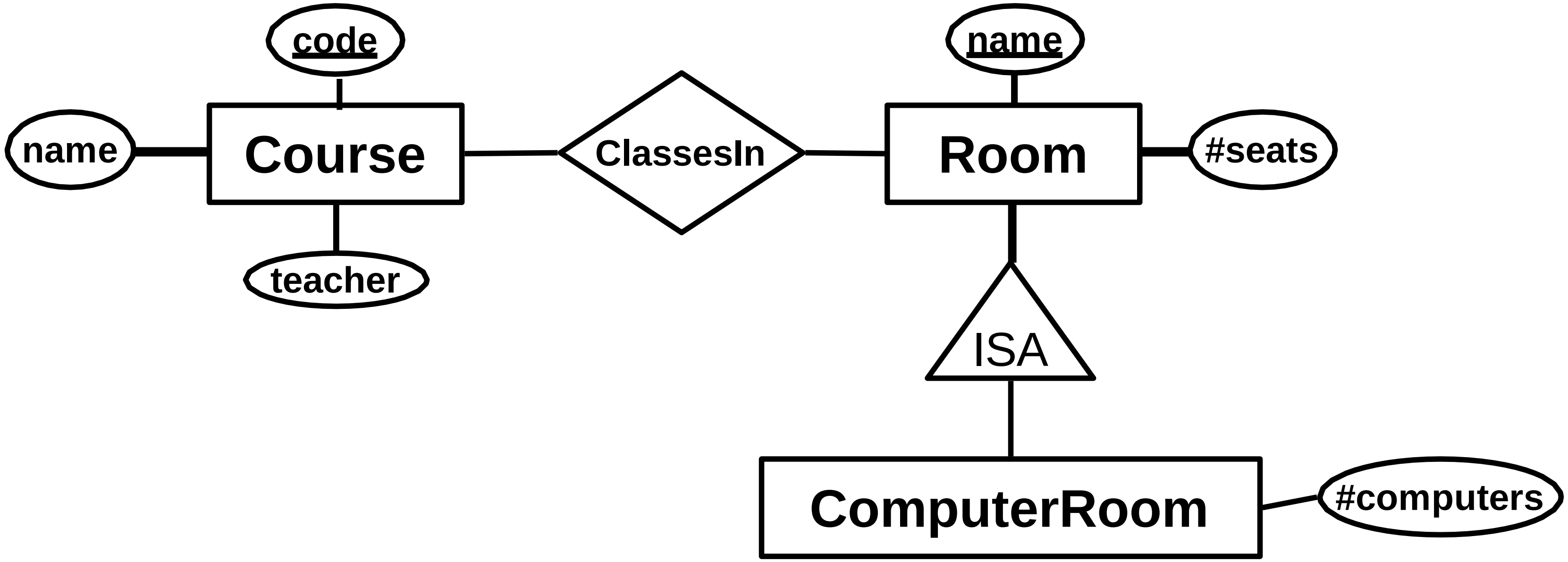
SPECIAL RELATIONSHIPS

Subclassing

- Subclass = sub-entity = special case.
- A subclass is a subset of an entity set.
- More attributes and/or relationships.
- A subclass shares the key of its parent.

- Drawn as an entity connected to the superclass by a special triangular relationship called *ISA.* Triangle points to superclass.
  - ISA = ”is a”

Example:



- – A computer room *is a* room.
- – Not all rooms are computer rooms.
- – Computer rooms share the extra property that they have a number of computers.

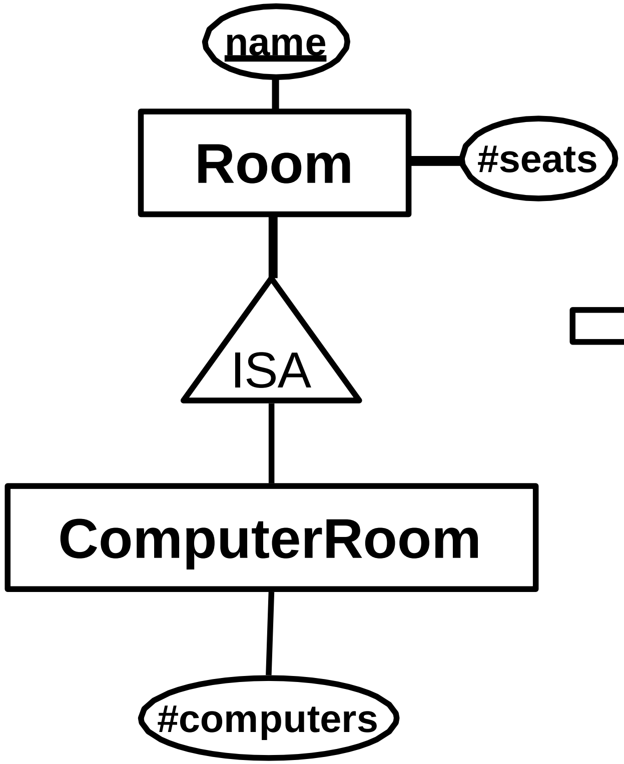
Subclass/Superclass Hierarchy

- We assume that subclasses form a tree hierarchy.
  - A subclass has only one superclass.
  - Several subclasses can share the same superclass.
    - E.g. Computer rooms, lecture halls, chemistry labs etc. could all be subclasses of Room.

Translating ISA to relations

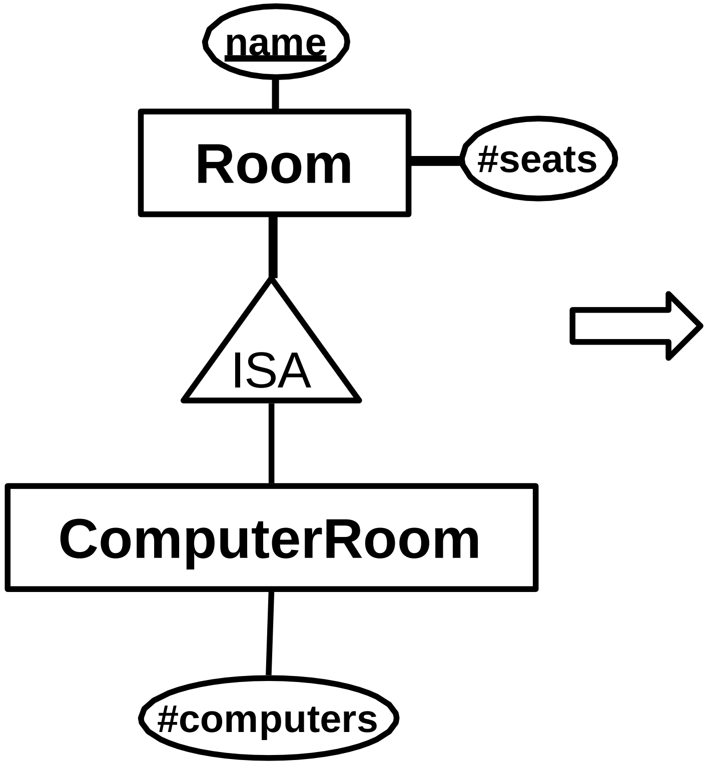
- Standard approach:
  - An ISA relationship is a standard one-to-”exactly one” relationship. Each subclass becomes a relation with the key attributes of the superclass included.
  - Also known as the E-R approach.

The E-R approach:



What?

The E-R approach:



```
Rooms(name, #seats)
ComputerRooms(name, #computers)
  name -> Rooms.name
```

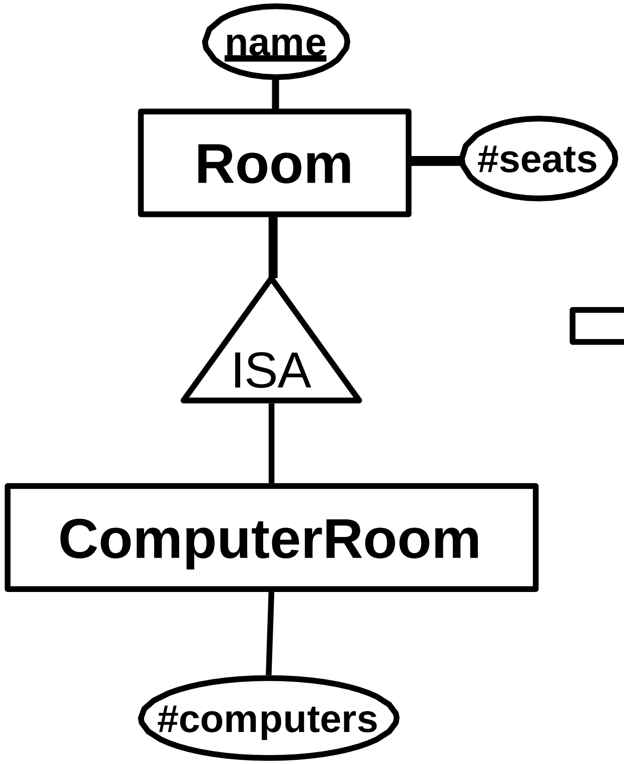
name	*#seats*
VR	216
ED6225	52

name	*#computers*
ED6225	26

Alternate ISA translations

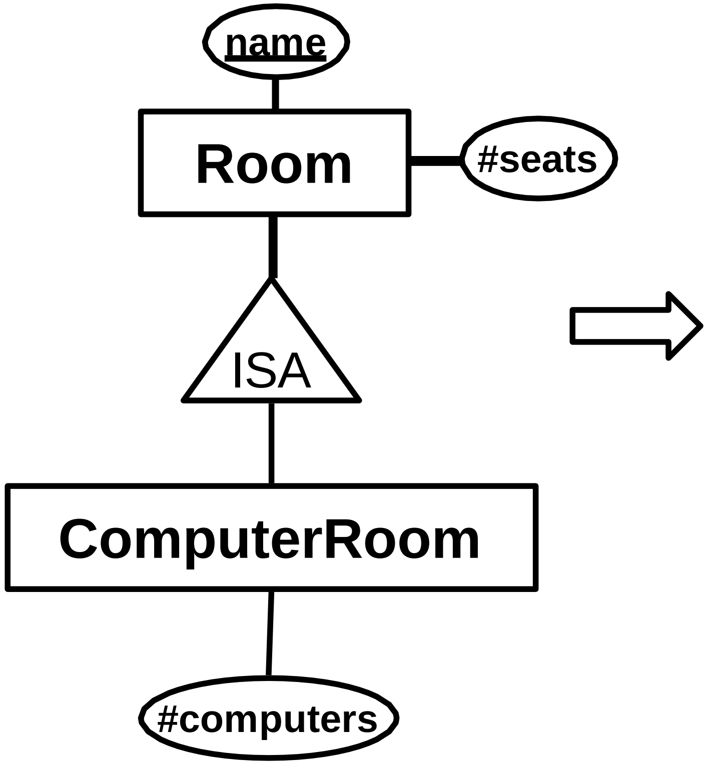
- Two alternate approaches
  - *NULLs:* Join the subclass(es) with the superclass. Entities that are not part of the subclass use NULL for the attributes that come from the subclass.
  - *Object-oriented:* Each subclass becomes a relation with all the attributes of the superclass included. An entity belongs to either of the two, but not both.

The NULLs approach:



What?

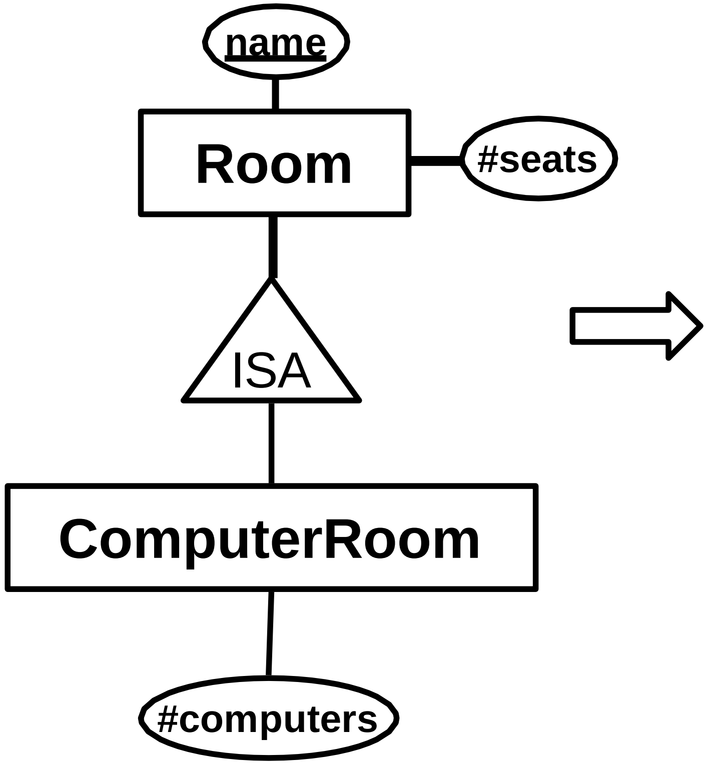
The NULLs approach:



Rooms(<u>name</u>, #seats, #computers)

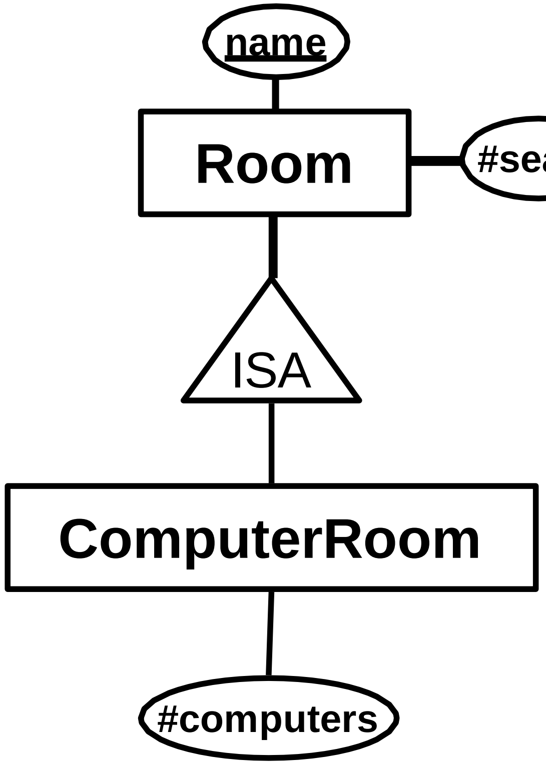
<u>name</u>	*#seats*	*#computers*
VR	216	NULL
ED6225	52	26

The object-oriented (OO) approach:



What?

The object-oriented (OO) approach:



```
Rooms(name, #seats)
ComputerRooms(name, #seats,
                         #computers)
```

name	*#seats*
VR	216

name	*#seats*	*#computers*
ED6225	52	26

Comparison – E-R

- E-R approach
  - Always works.
  - Use unless you have a good reason not to.

Comparison – OO

- OO approach
  - Good when searching for general information about entities in a subclass only.
    - *”List the number of seats in all computer rooms”*
  - Does *not* work if superclass has any relationships.
    - An entity belonging to the subclass does not belong to the superclass as well, so foreign keys would have no single table to refer to.

Comparison – NULLs

- NULLs approach
  - Could save space in situations where most entities in the hierarchy are part of the subclass (e.g. most rooms have computers in them).
  - Reduces the need for *joins.*
  - Not suited if subclass has any relationships.
    - Would lose the constraint that only the entities in the subclass can participate in the relationship.

Weak entities

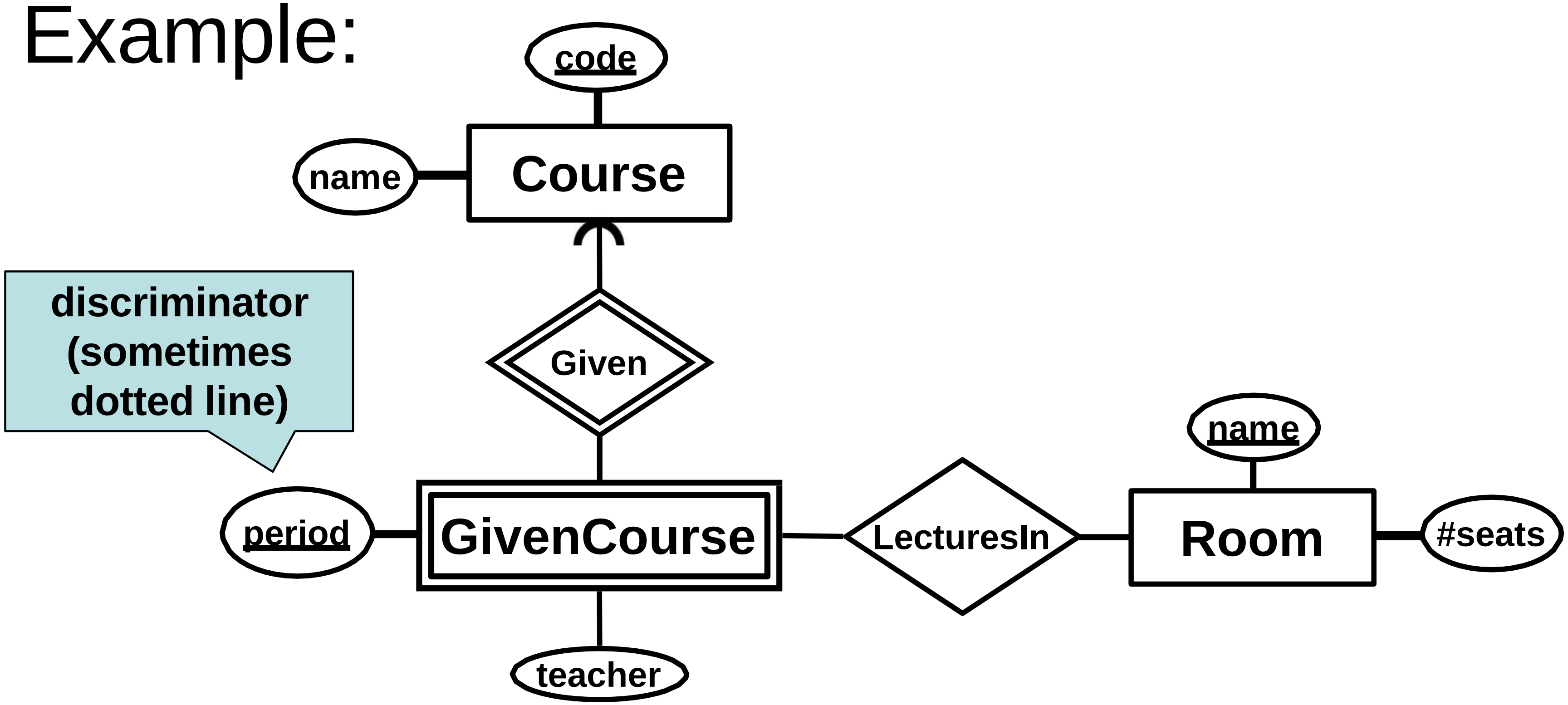
- Some entities depend on other entities.
  - A course is an entity with a code and a name.
  - A course does not have a teacher, rather it has a teacher for each time the course is given.
  - We introduce the concept of a given course, i.e. a course given in a particular period. A given course is a *weak entity*, dependent on the entity course. A given course has a teacher.

Weak entities

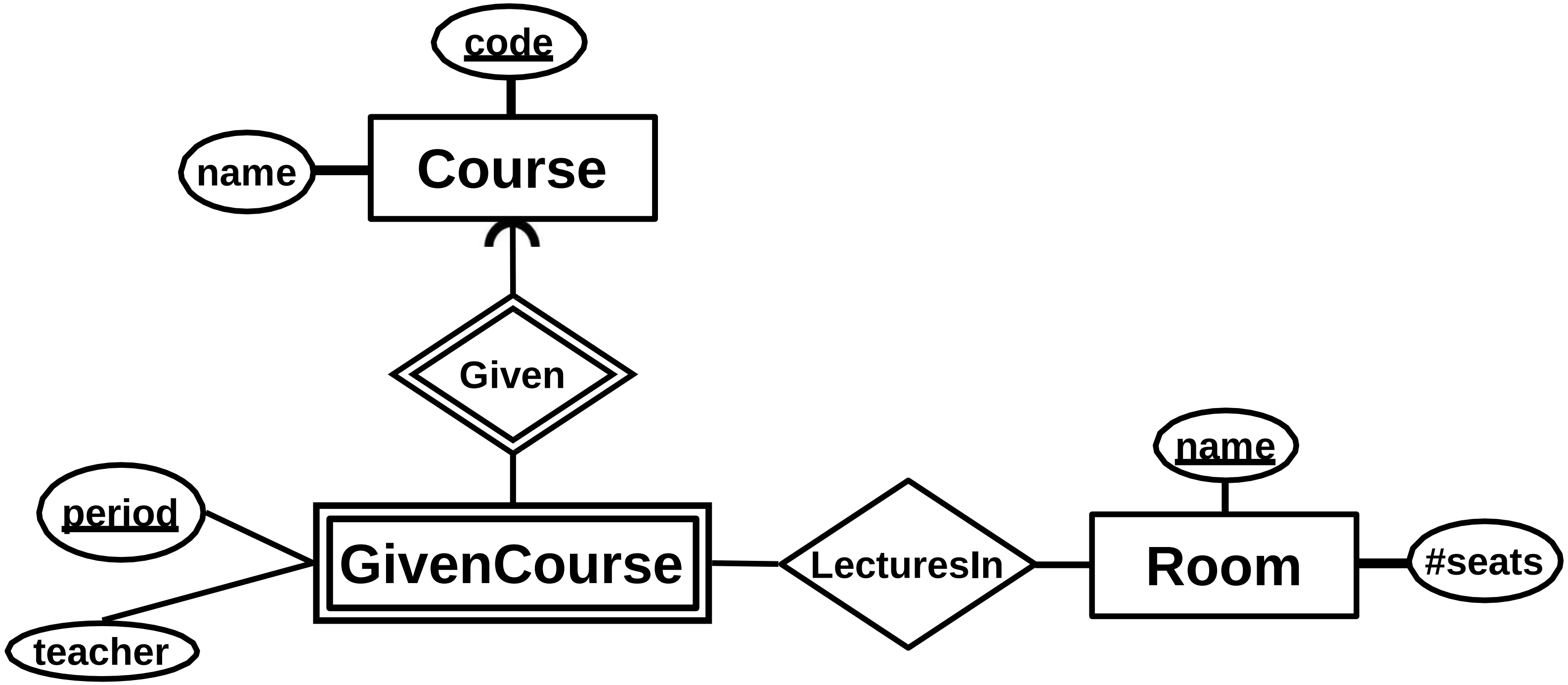
- A *weak entity* is an entity that depends on another entity for help to be ”uniquely” identified.
  - E.g. an airplane seat is identified by its number, but is not uniquely identified when we consider other aircraft. It depends on the airplane it is located in.
- Drawn as a rectangle with double borders.
- Related to its *supporting entity* by a *supporting relationship*, drawn as a diamond with double borders. This relationship is always many-to-”exactly one”.

Weak entities in E-R diagrams

Example:



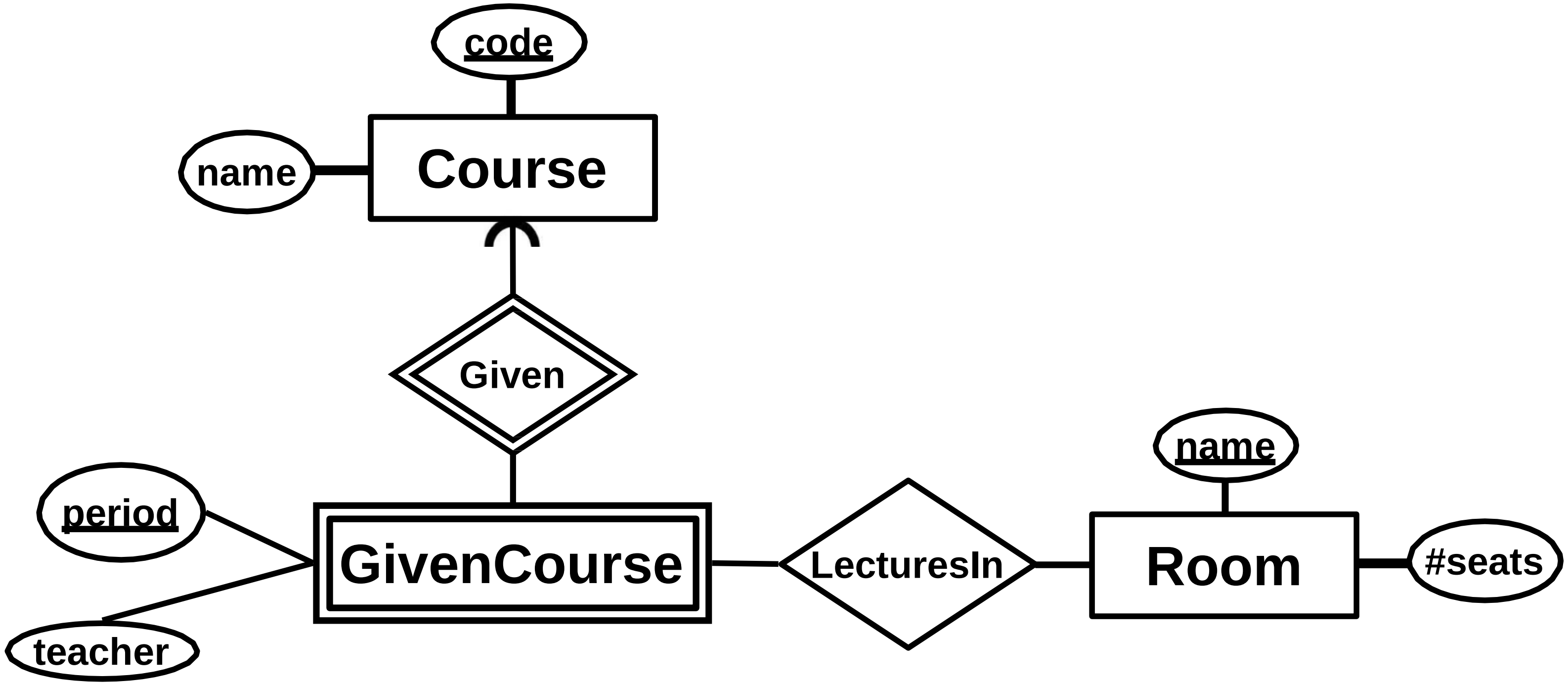
Translating to relations:



⇨

What?

Translating to relations:

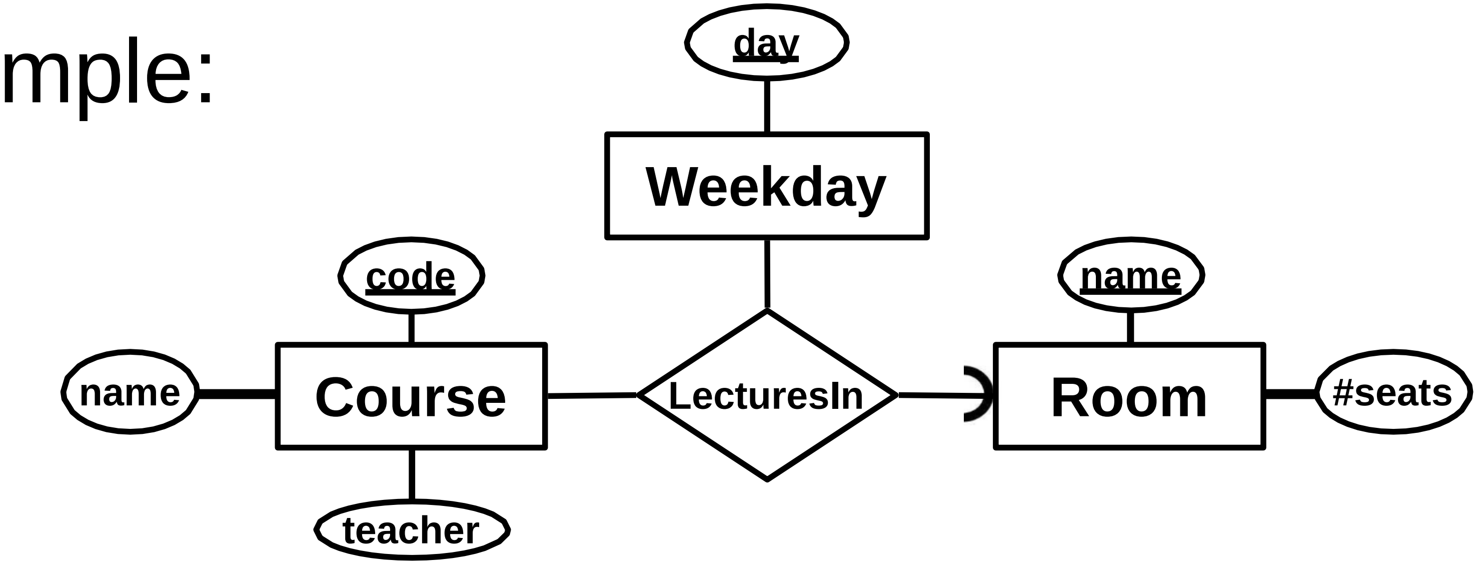


```
Courses(code, name)
GivenCourses(course, period, teacher)
  course -> Courses.code
LecturesIn(course, period, room)
  (course, period) -> GivenCourses.(course, period)
  room             -> Rooms.name
Rooms(name, #seats)
```

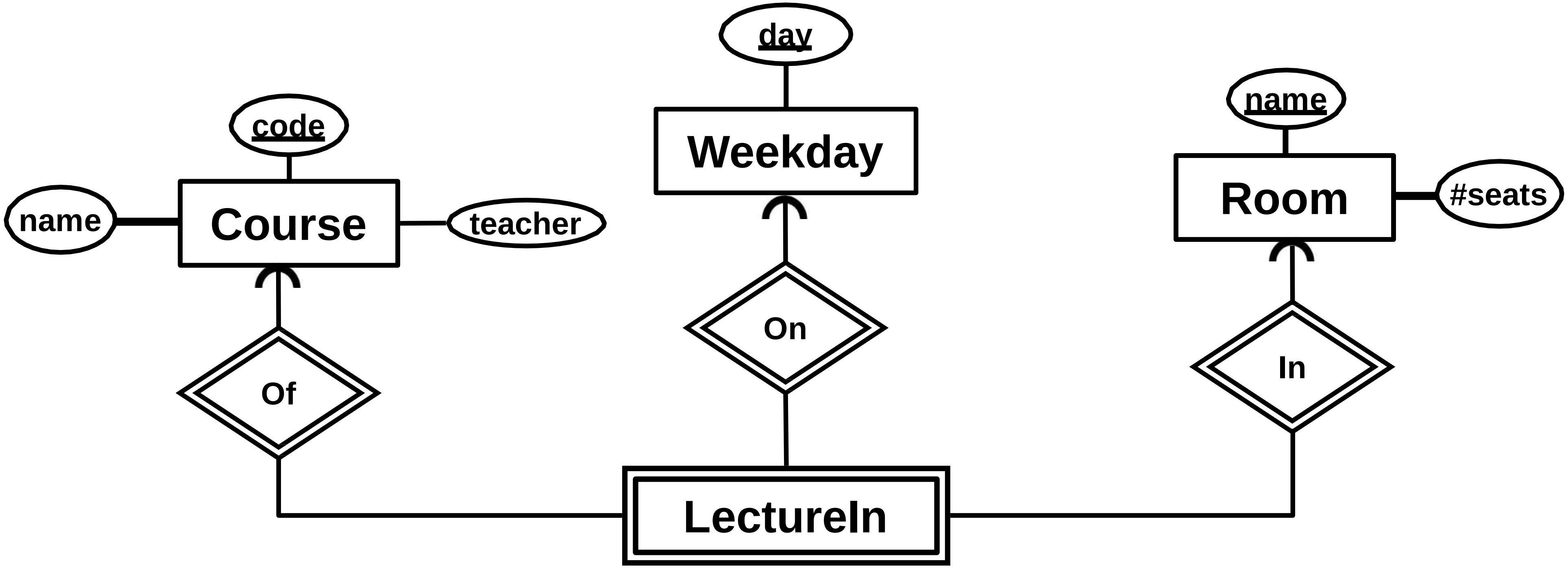
Multiway relationships as WEs

- Multiway relationships can be transformed away using weak entities
  - Subtitute the relationship with a weak entity.
  - Insert supporting relationships to all entities related as ”many” by the original relationship.
  - Insert ordinary many-to-one relationships to all entities related as ”one” by the original relationship.

Example:



day
Weekday
code
name
name
Course
LecturesIn
Room
#seats
teacher



day
Weekday
name
code
Room
#seats
name
Course
teacher
On
In
Of
LectureIn

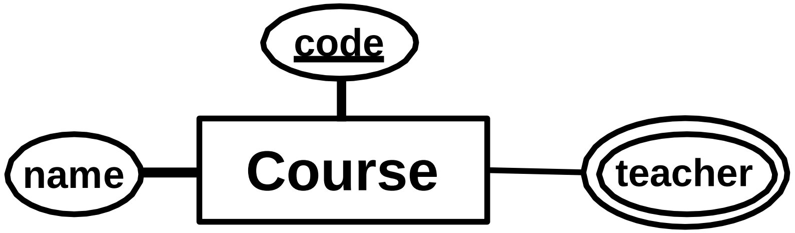
What’s the point?

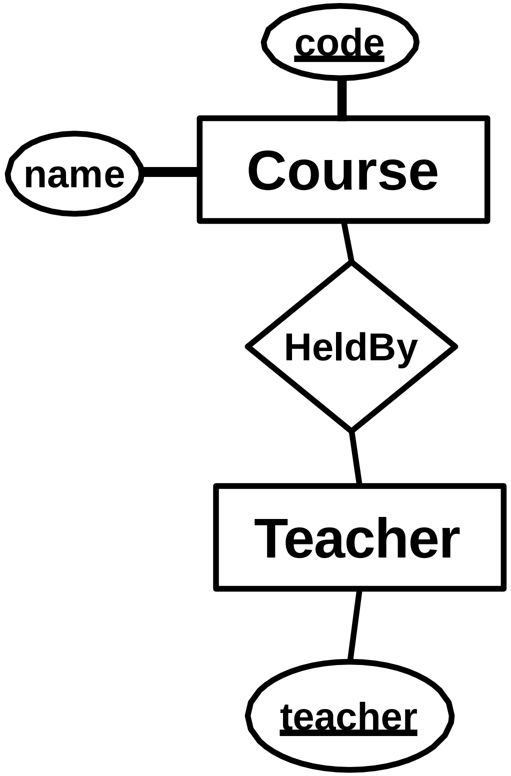
- Usually, relationships work just fine, but in some special cases, you need a weak entity to express all multiplicity constraints correctly.
- A weak entity is needed when a **part** of an entity’s key is a foreign key.

“Multivalued” attributes and “flag” attributes

THINGS NOT TO DO…

”Multivalued” attributes





```
Courses(code,name)
HeldBy(code,teacher)
  code -> Courses.code
```

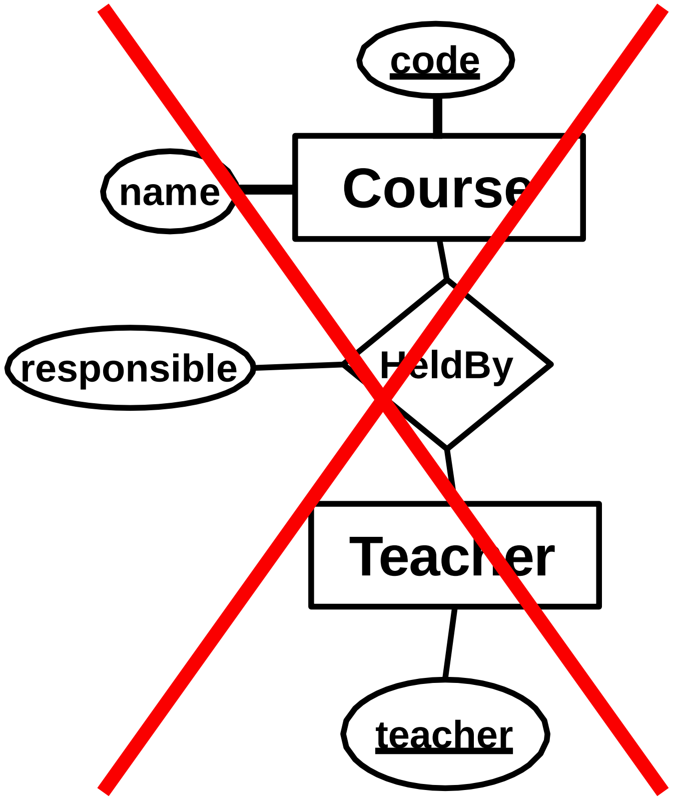
```
Courses(code,name)
Teachers(teacher)
HeldBy(code,teacher)
  code    -> Courses.code
  teacher -> Teachers.teacher
```

”Multivalued” attributes

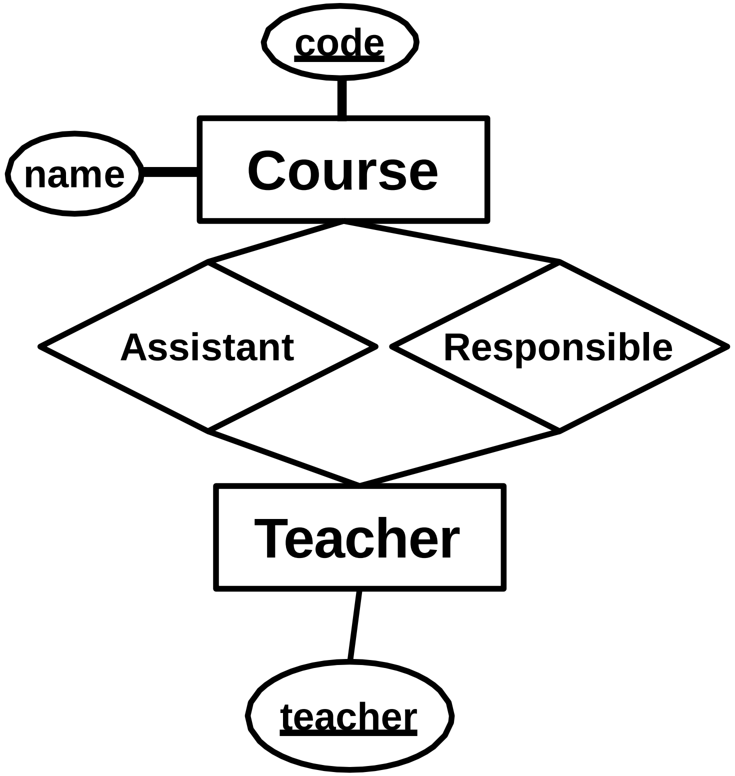
- Inflexible if you later want more attributes on teachers.
- No guarantees against e.g. spelling errors of teacher names.
  - less flexible to insert a constraint on what values are allowed than to use an extra table.
- Tables are cheap – references are cheap
  - No reason NOT to use an entity.

- Rule of thumb: Don’t use multivalued attributes!!

”Flag” attributes on relationships



vs.



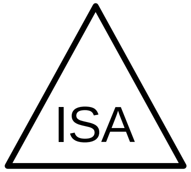
”Flag” attributes on relationships

- Less intuitively clear.
- Inflexible if later you need more roles.
- Tables are cheap, union of two tables is a cheap operation (O(1)) – filtering can be expensive (O(n))!

- Only benefit: automatic mutual exclusion (a teacher can only be *either* responsible *or* an assistant).
  - If important, can be recovered via assertions (costly).

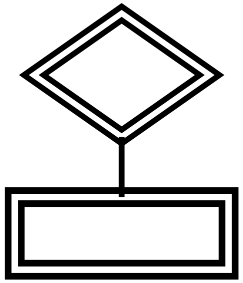
- Rule of thumb: Don’t use flag attributes on relationships!

ER cheatsheet 3



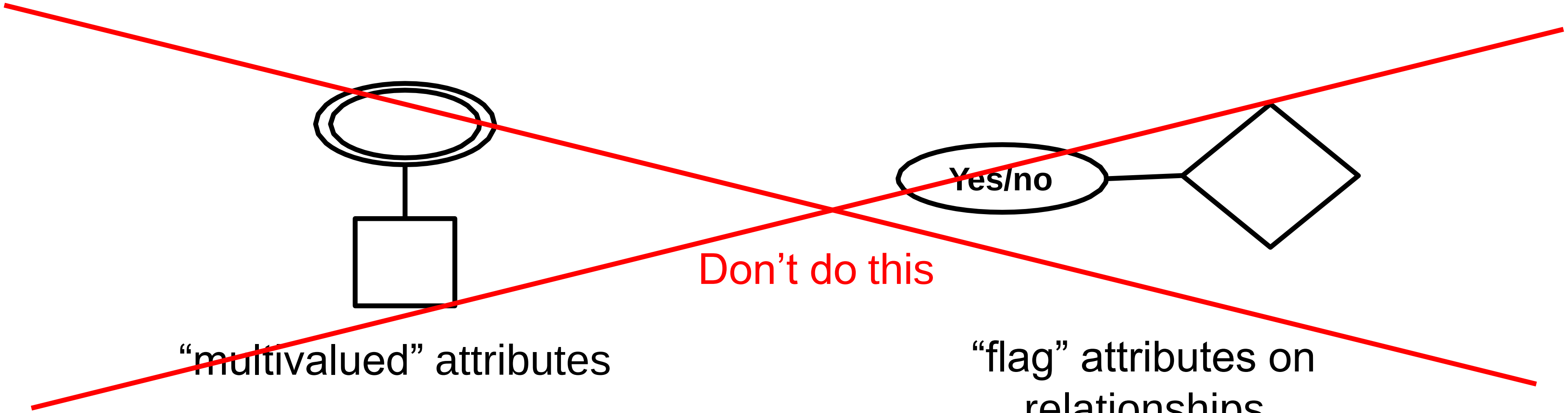
Subclassing
sub-entity extends super-entity

- ER-approach
- NULL-approach
- OO-approach



Weak entities, identifying relationship
Weak entity "is part of" entity

- Composite key with foreign key



Don't do this

"multivalued" attributes

"flag" attributes on relationships