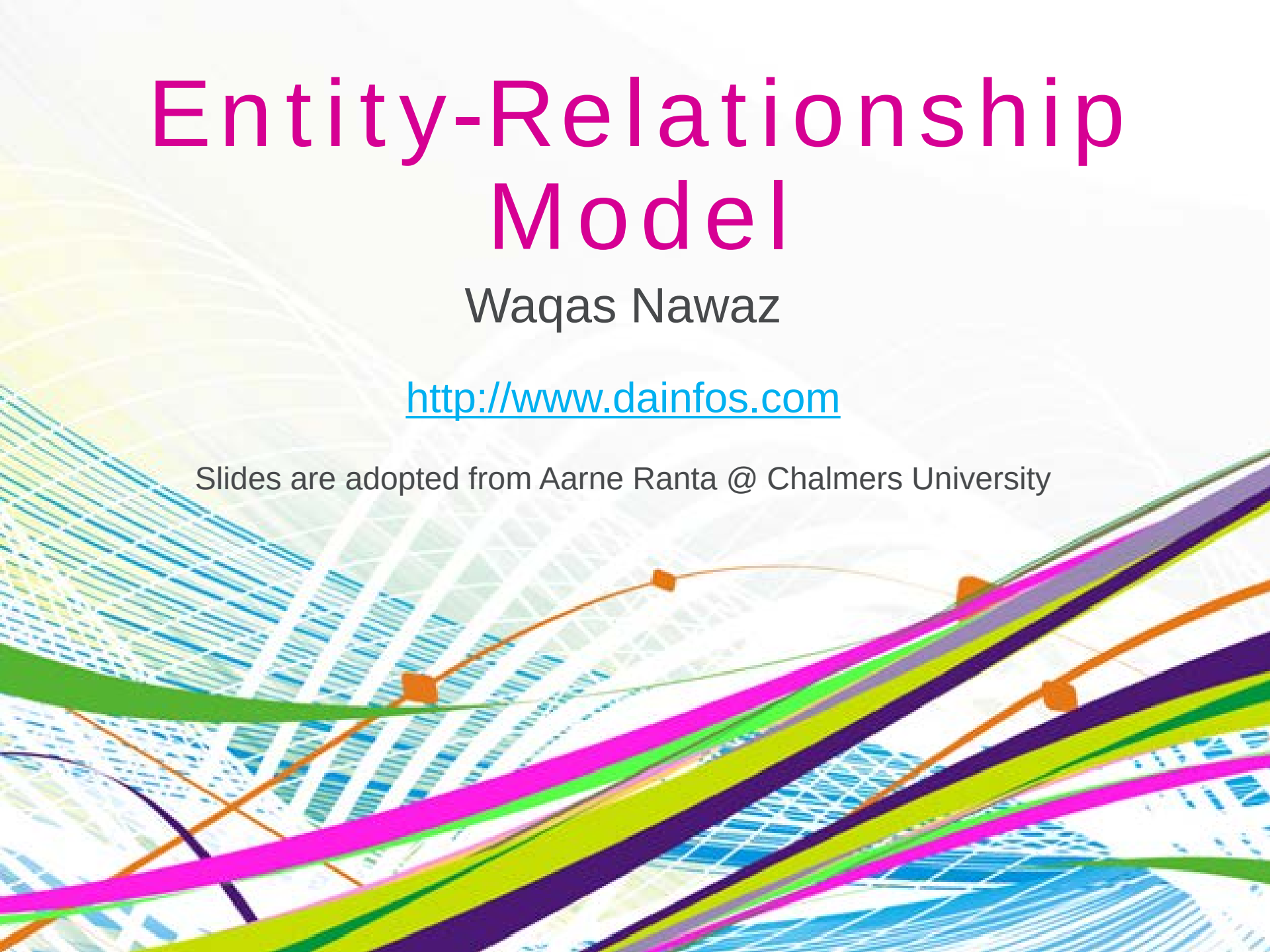
Entity-Relationship Model

Waqas Nawaz

http://www.dainfos.com

Slides are adopted from Aarne Ranta @ Chalmers University



Entity-Relationship approach

- Design your database by drawing a picture of it- an Entity-Relationship diagram
- Allows us to sketch the design a database informally
- Use mechanical methods to convert your diagram to relations -> This means that the diagram can be a formal specification as well

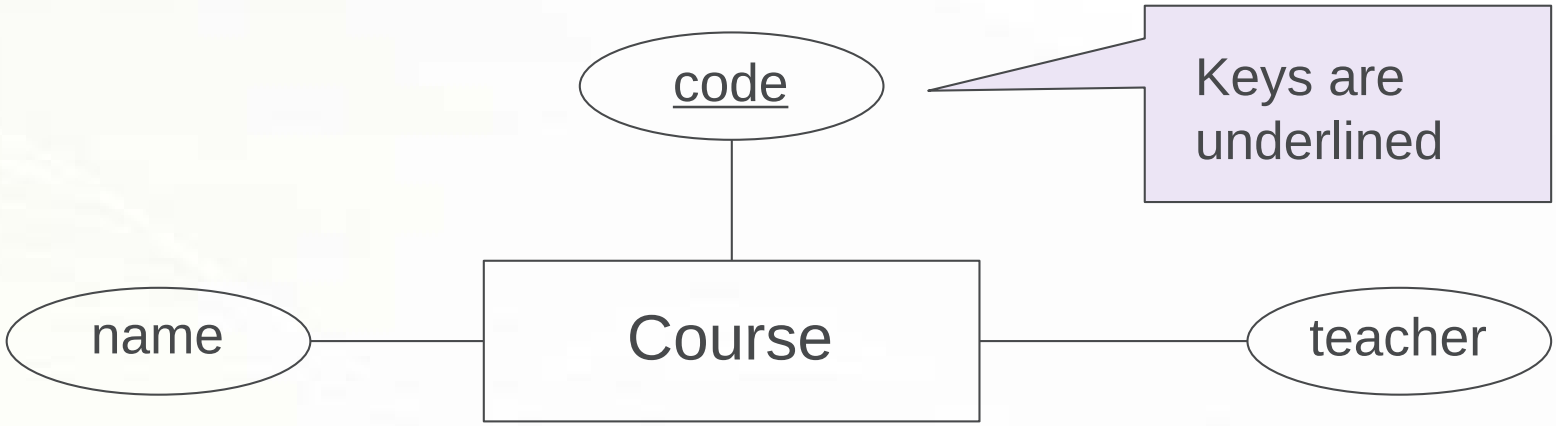
Entities and entity sets

- Entity = “thing” or object
  - course, room etc.
- Entity set = collection of similar entities
  - all courses, all rooms etc.
- Entities are drawn as rectangles

Attributes

- Entities have attributes
- All entities in an entity set have same attributes (though not the same values)
- Attributes are drawn as ovals connected to the entity by a line

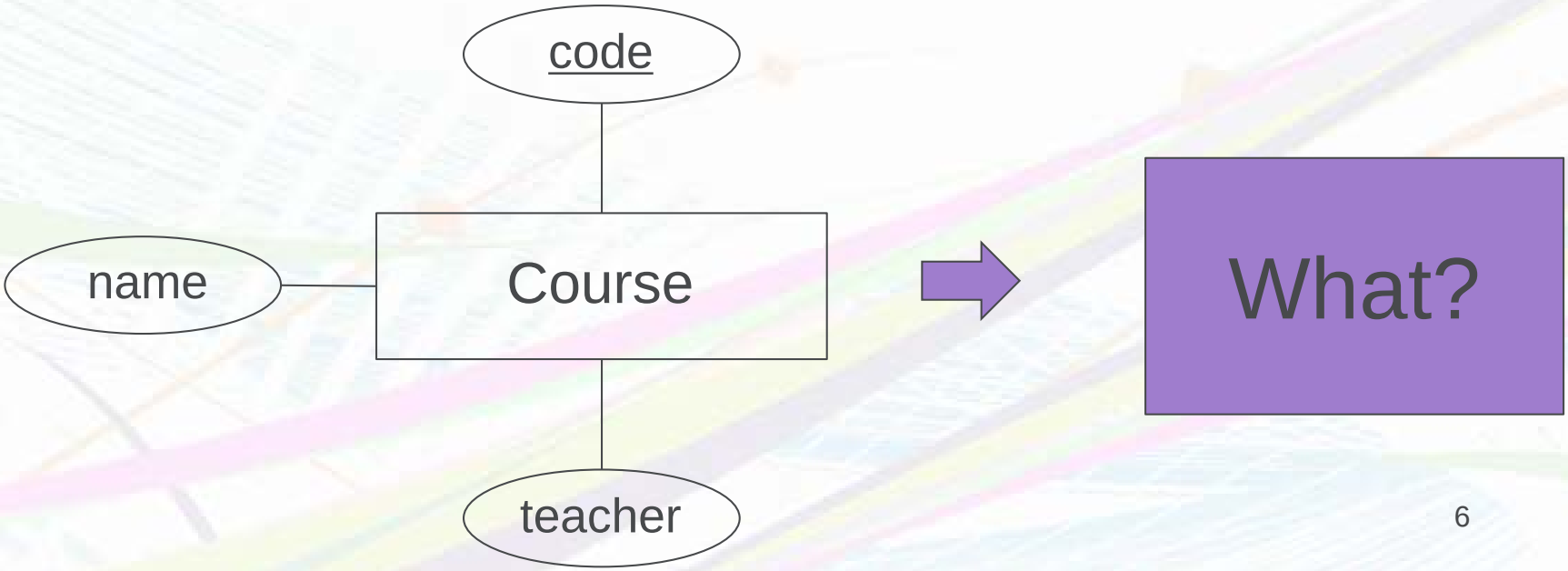
Example:



- A course has three attributes – the unique course code, a name and the name of the teacher
- All course entities have values for these three attributes, e.g. (CS100, Databases, Qiang Qu)

Translation to relations

- An ER diagram can be mechanically translated to a relational database schema.
- An entity becomes a relation, the attributes of the entity becomes the attributes of the relation, keys become keys.

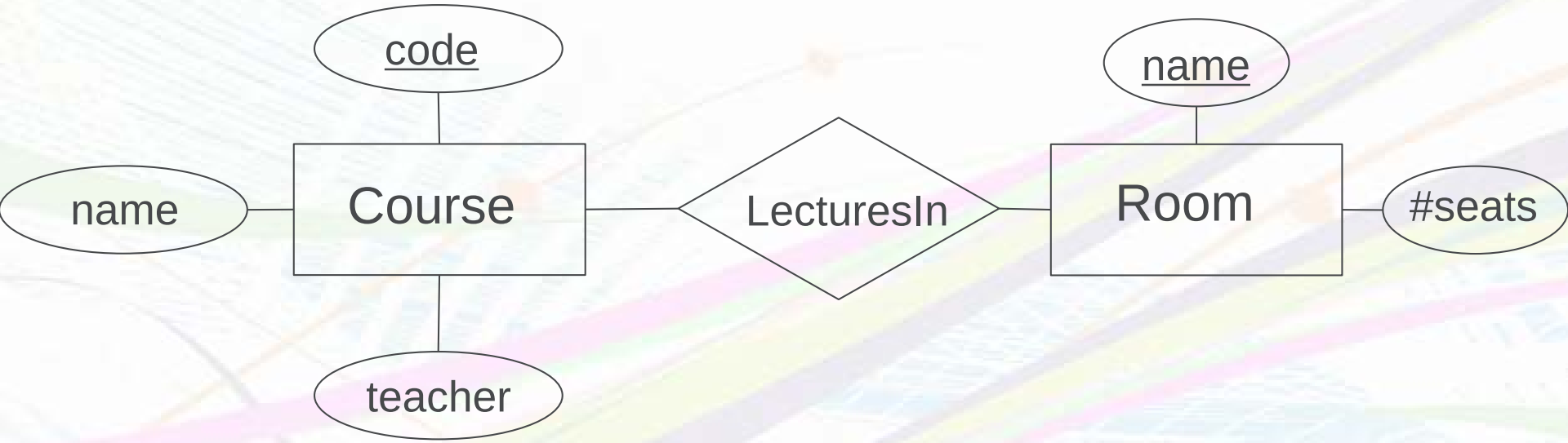


Relationships

- A relationship connects two (or more) entities.
- Drawn as a diamond between the related entities, connected to the entities by lines.
- Note: Relationship ≠ Relation

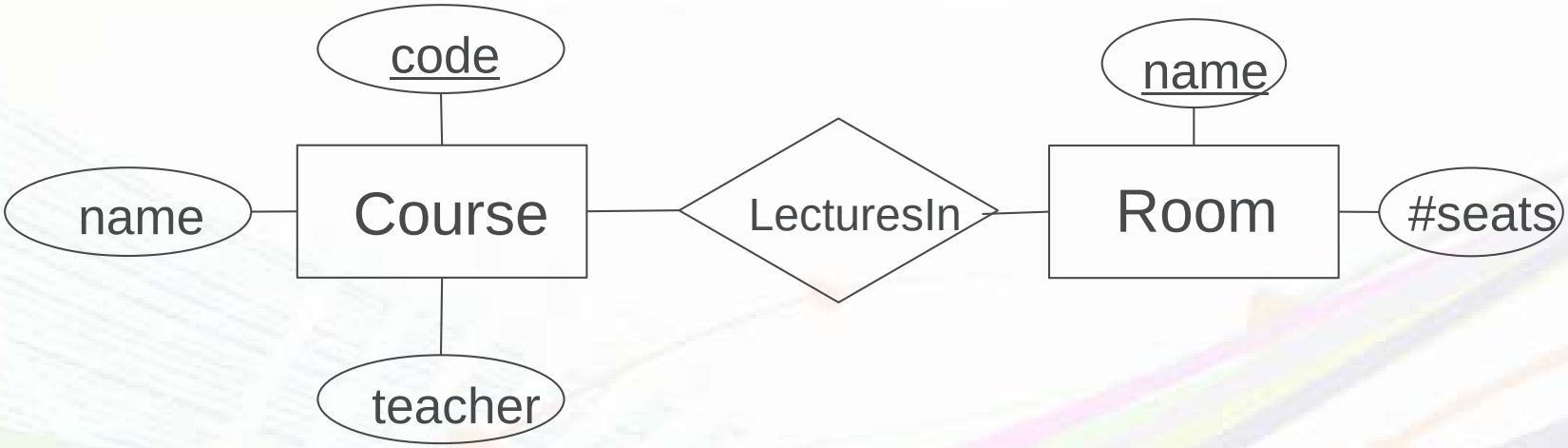
Example

- A course has lectures in a room.
- A course is related to a room by the fact that the course has lectures in that room.
- A relationship is often named with a verb.



Translations to relations

- A relationship between two entities is translated into a relation, where the attributes are the keys of the related entities.



What?

References

- Courses (code, name, teacher)
  Rooms (name, #seats)
  LecturesIn (code, name)

- We must ensure that the codes used in **LecturesIn** matches those in **Courses**
  - Introduce references between relations
  - e.g. the course code used in **LecturesIn** reference those in **Courses**.

- Courses (code, name, teacher)
  Rooms (name, #seats)
  LecturesIn (code, name)
  code -> Courses.code
  name -> Rooms.name

“Foreign” Keys

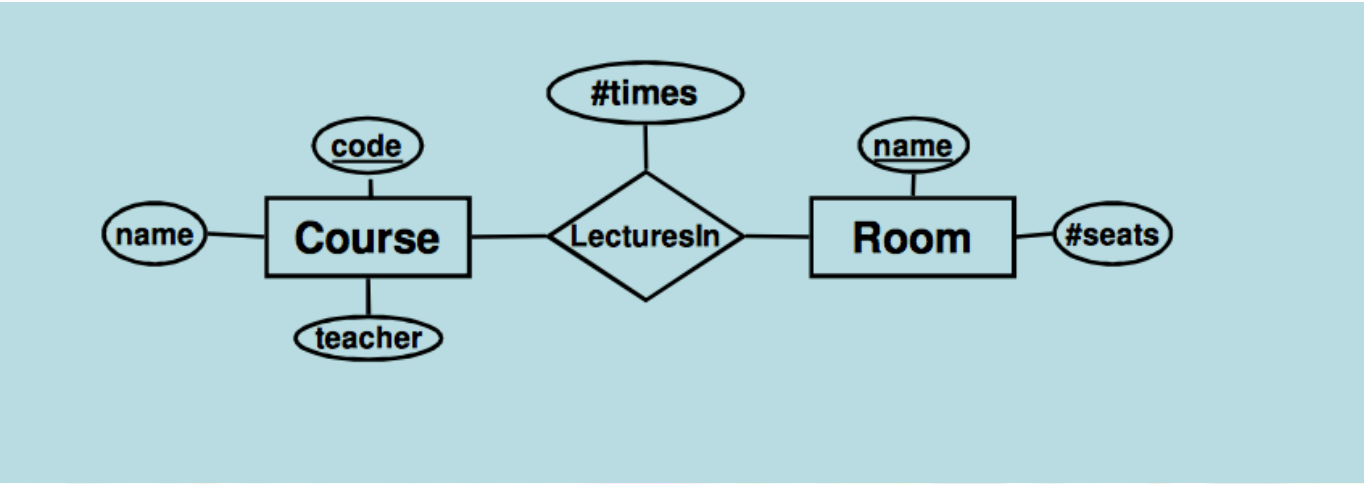
- Usually, a reference points to the key of another relation
  - e.g. **name** in **LecturesIn** references the key **name** in **Rooms**
  - **name** is said to be a foreign key in **LecturesIn**

Relationship (non-)keys

- Relationship relations have no key attributes of their own
  - The “key” of a relationship relation is the combined keys of the related entities
  - Follows from the fact that entities are either related or not
  - If you at some point think it makes sense to put a key on a relationship, it should probably be an entity instead

Quiz

- Suppose we want to store the number of times that each course has a lecture in a certain room. How do we model this?

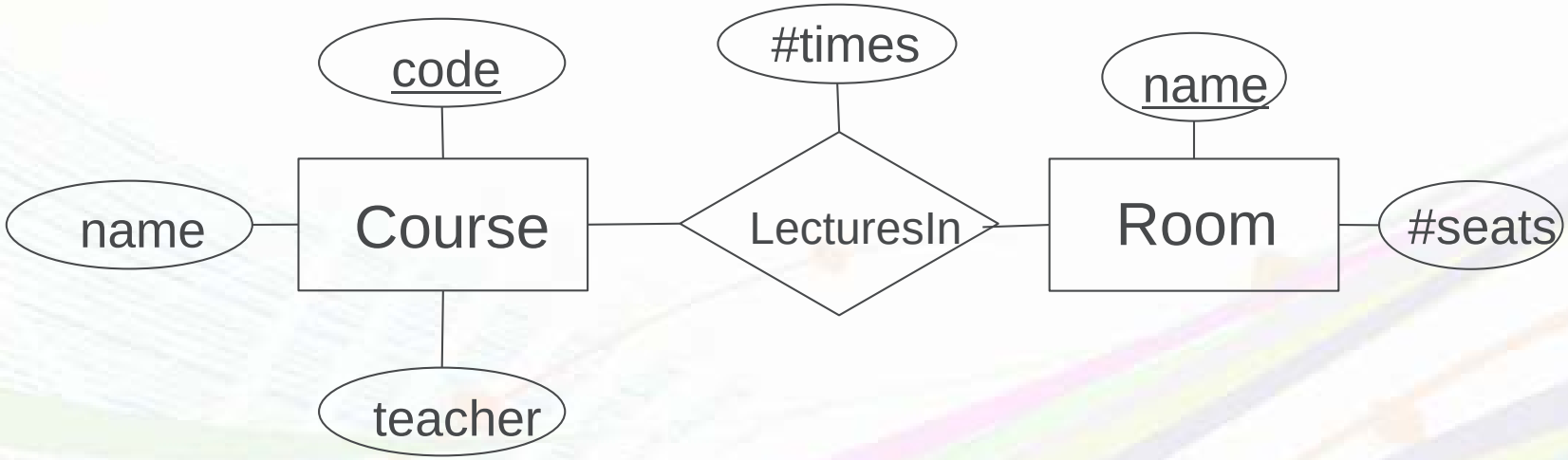


Attributes on relationships

- Relationships can also have attributes
- Represent a property of the relationship between the entities
  - e.g. **#times** is a property of the relationship between a course and a room

Translation to relations

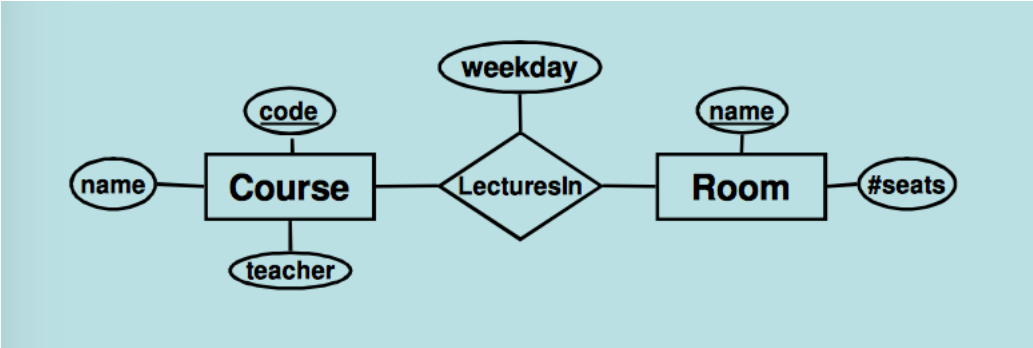
- A relationship between two entities is translated into a relation, where the attributes are the keys on the related entities, plus any attributes of the relationship.



What?

Quiz

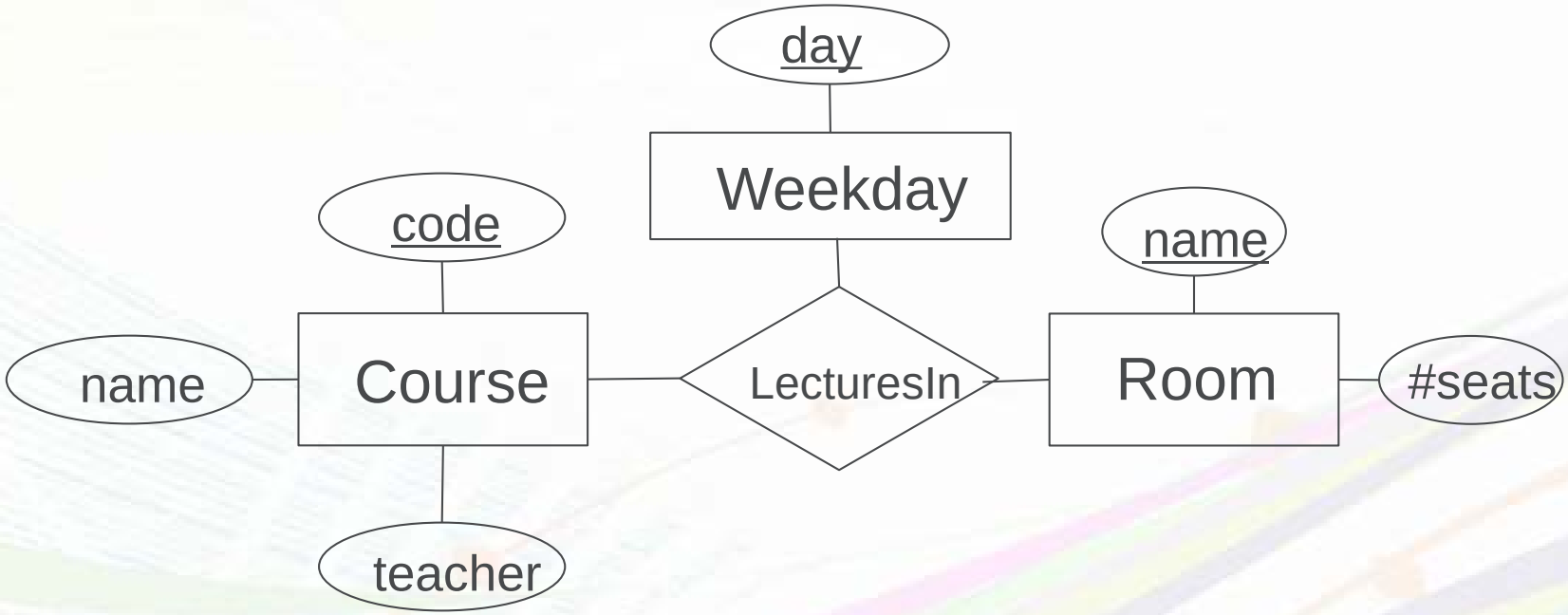
- Why could we not do the same for weekday?



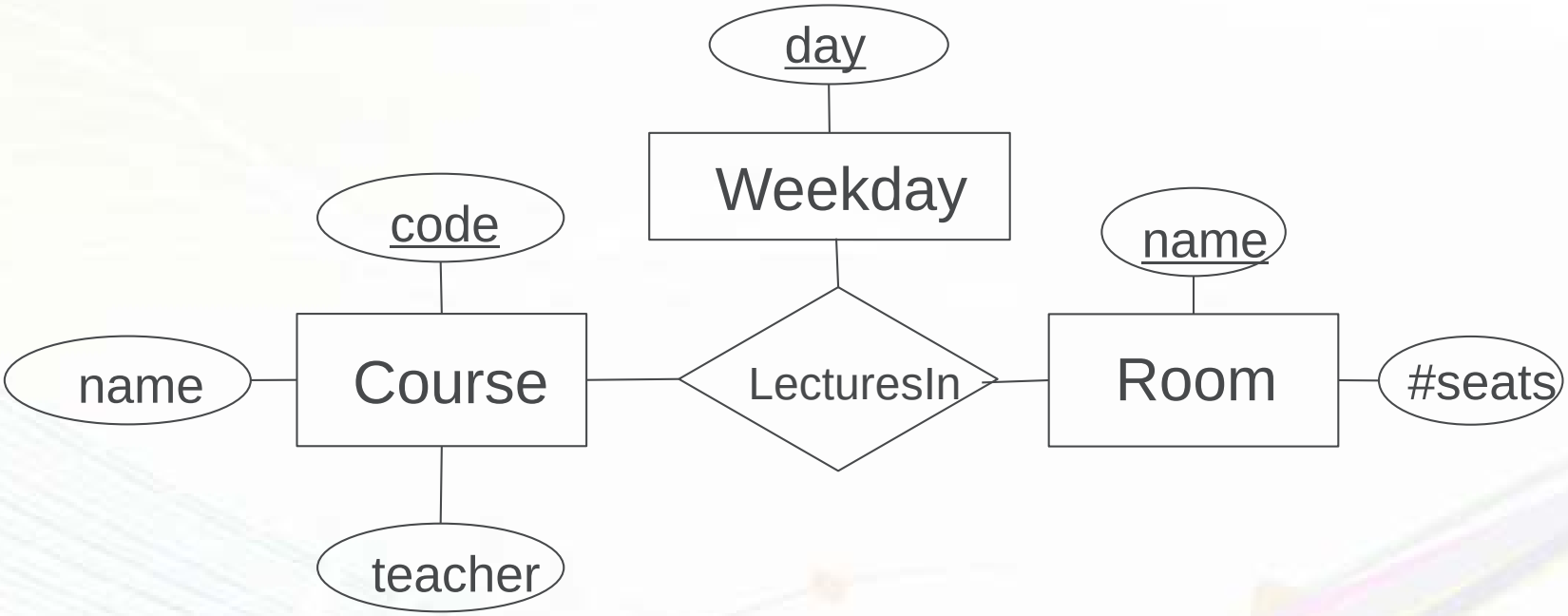
- Not a property of the relationship – a course can have lectures in a given room on several weekdays
- A pair of entities either related or not

Multiway relationships

- A course has lectures in a given room on different weekdays



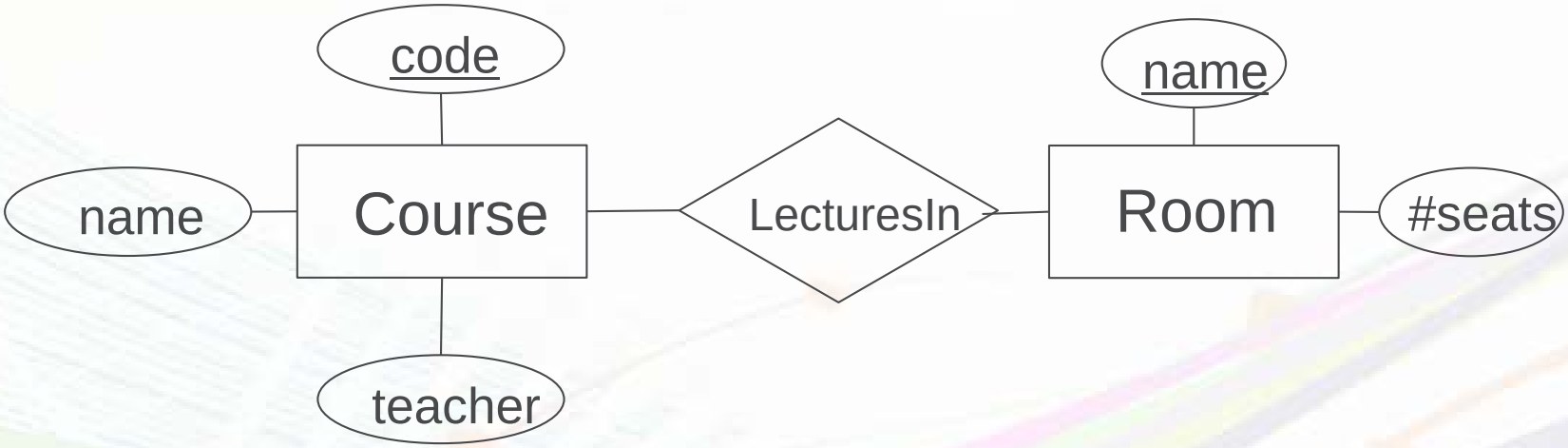
Translating to relations:



What?

Many-to-many relationships

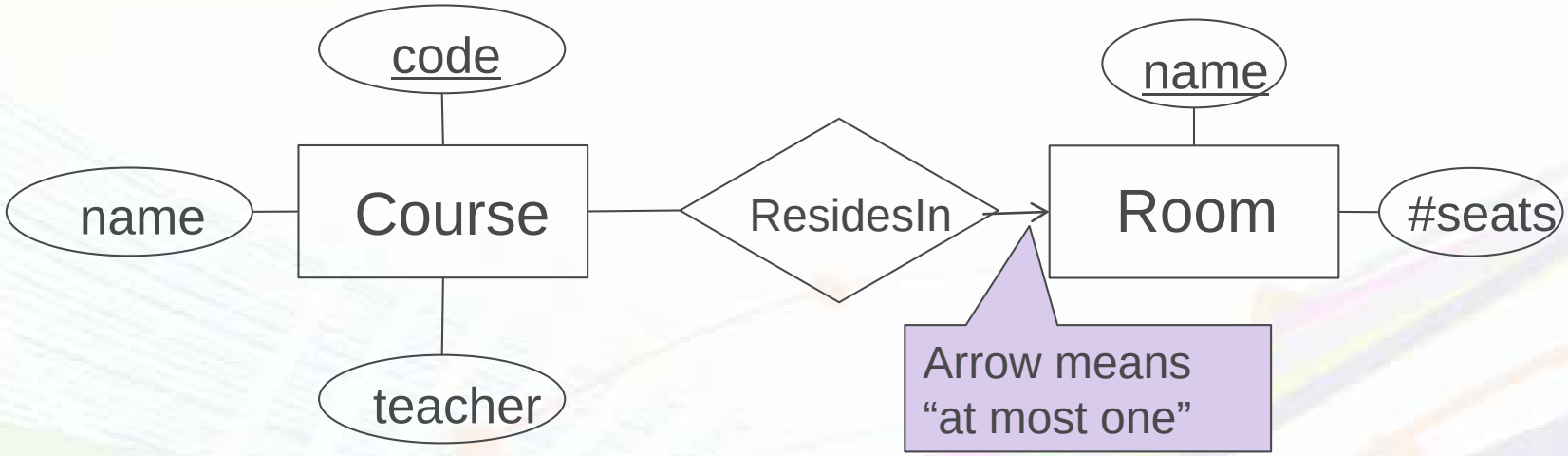
- Many-to-many (n-to-m, N-M) relationships
  - Each entity in either of the entity sets can be related to any number of entities of the other set.



  - A course can have lectures in many rooms.
  - Many courses can have lectures in the same room.

Many-to-one relationships

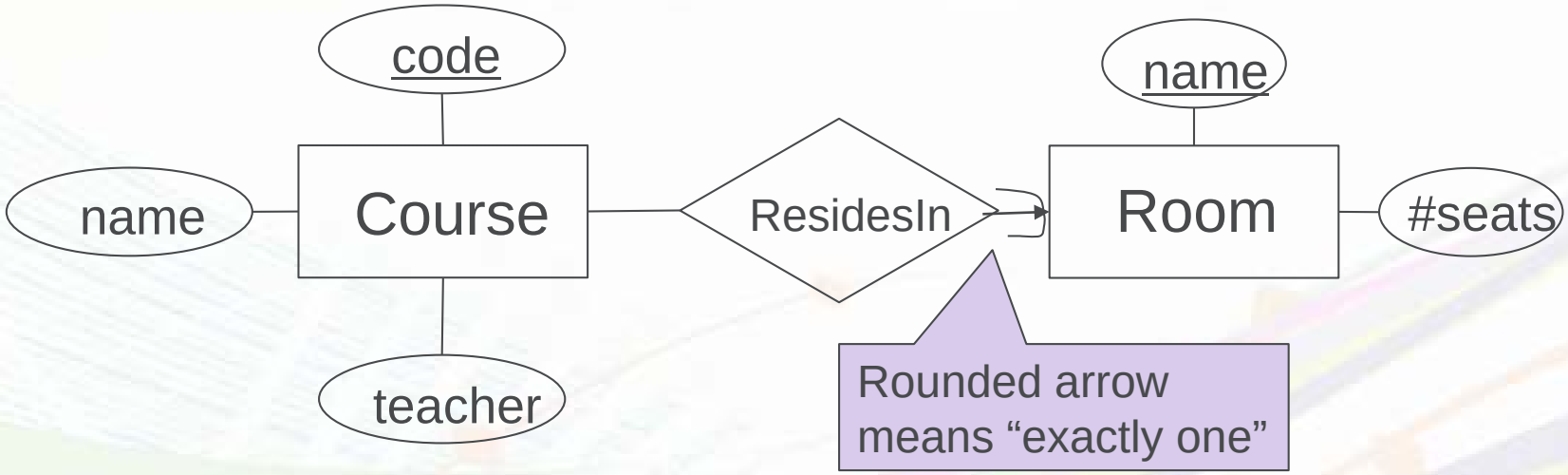
- Many-to-one relationships
  - Each entity on the “many” side can only be related to (at most) one entity on the “one” side.



  - Courses have all their lectures in the same room.
  - Many courses can share the same room

Many-to-"exactly one"

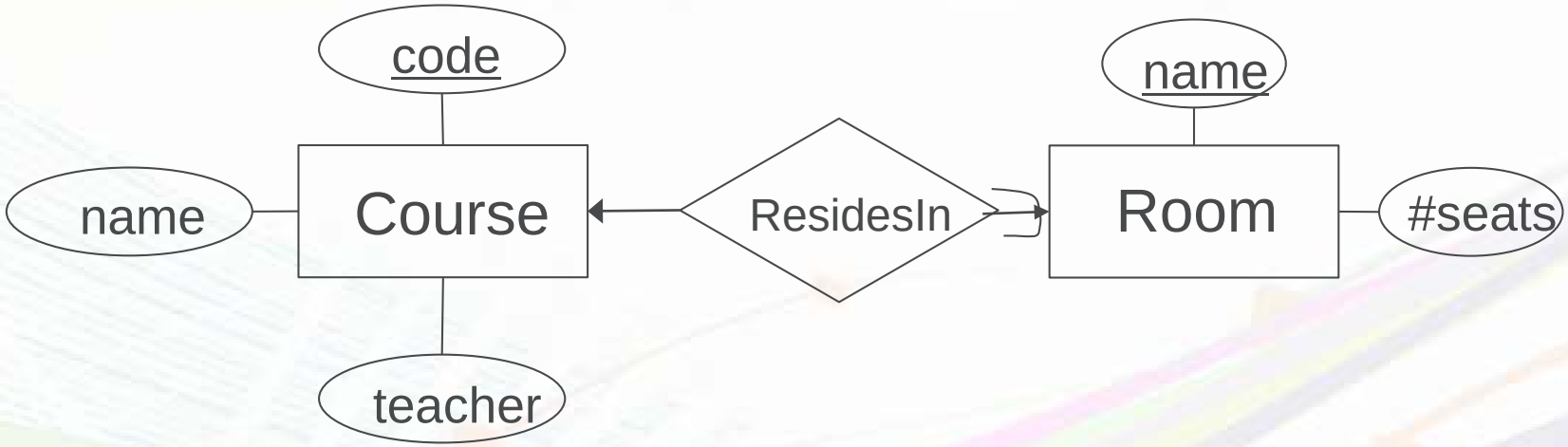
- All entities on the "many" side must be related to one entity on the "one" side.
  - This is also known as total participation



- Courses have all their lectures in some room.
- Many courses can share the same room.

One-to-one relationships

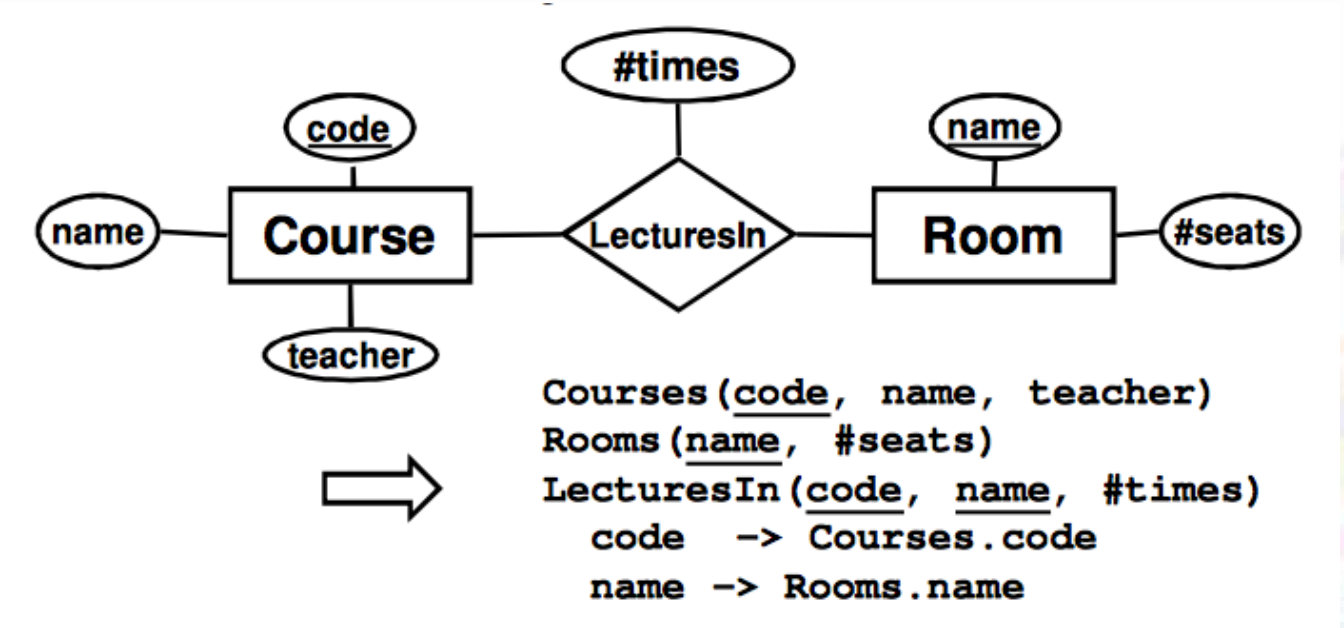
- One-to-one (1-to-1, 1-1) relationships
  - Each entity on the either side can only be related to (at most) one entity on the other side



  - Courses have all their lectures in the same room.
  - Only one course in each room
  - Not all rooms have courses in them

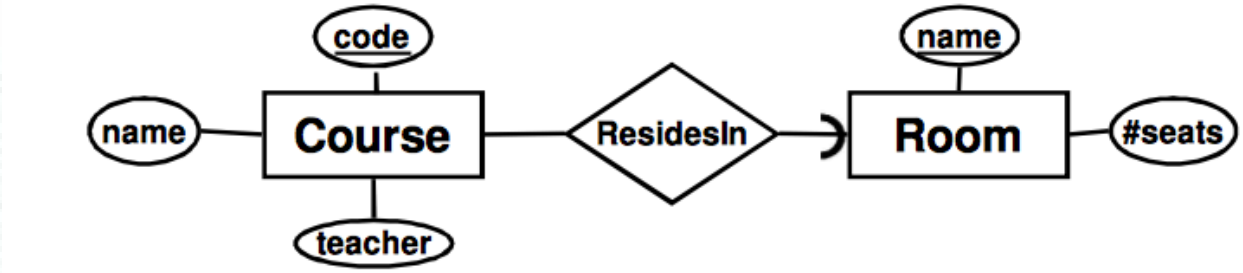
Translating multiplicity

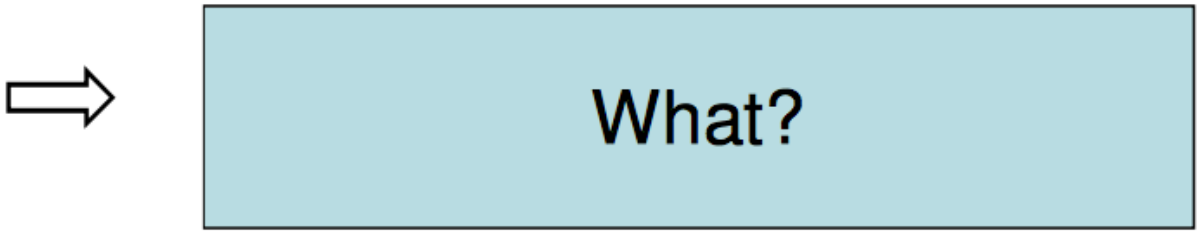
- A many-to-many relationship between two entities is translated into a relation, where the attributes are the *keys* of the related entities, and any attributes of the relation.



Translating multiplicity

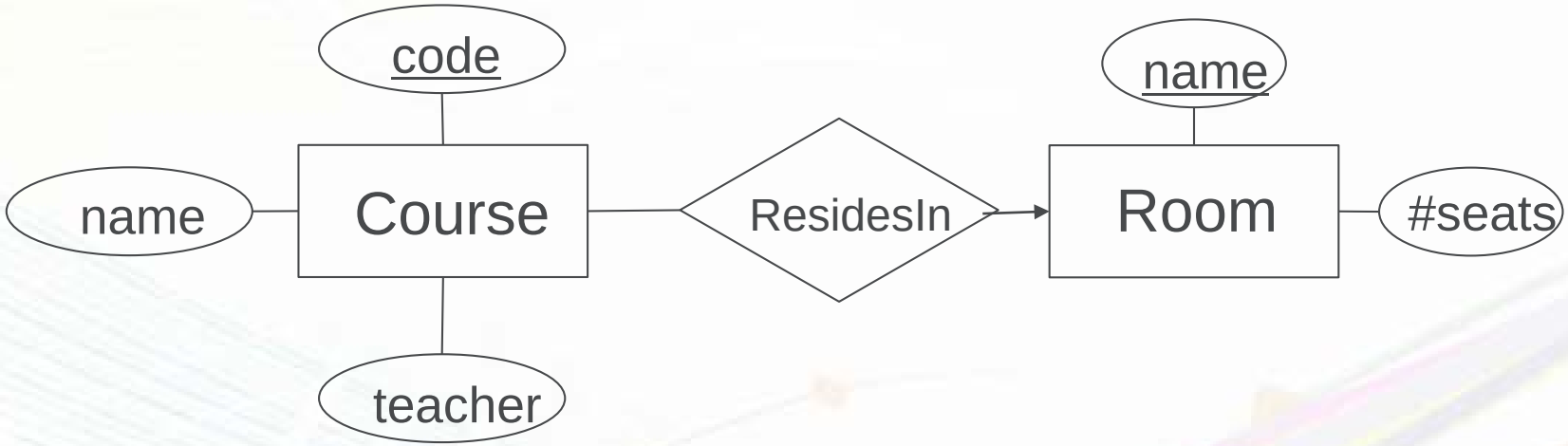
- A N-to-”exactly one” relationship between two entities is translated as part of the “many”-side entity





Quiz

- How do we translated N-to-one (meaning “at most one”) relationship?




```
Courses(code, name, teacher, room)
Room(name, #seats)
```

or

```
Courses(code, name, teacher)
Room(name, #seats)
ResidesIn(code, room)
```

?

Translation comparison

```
Courses(code, name, teacher, room)
Rooms(name, #seats)
```

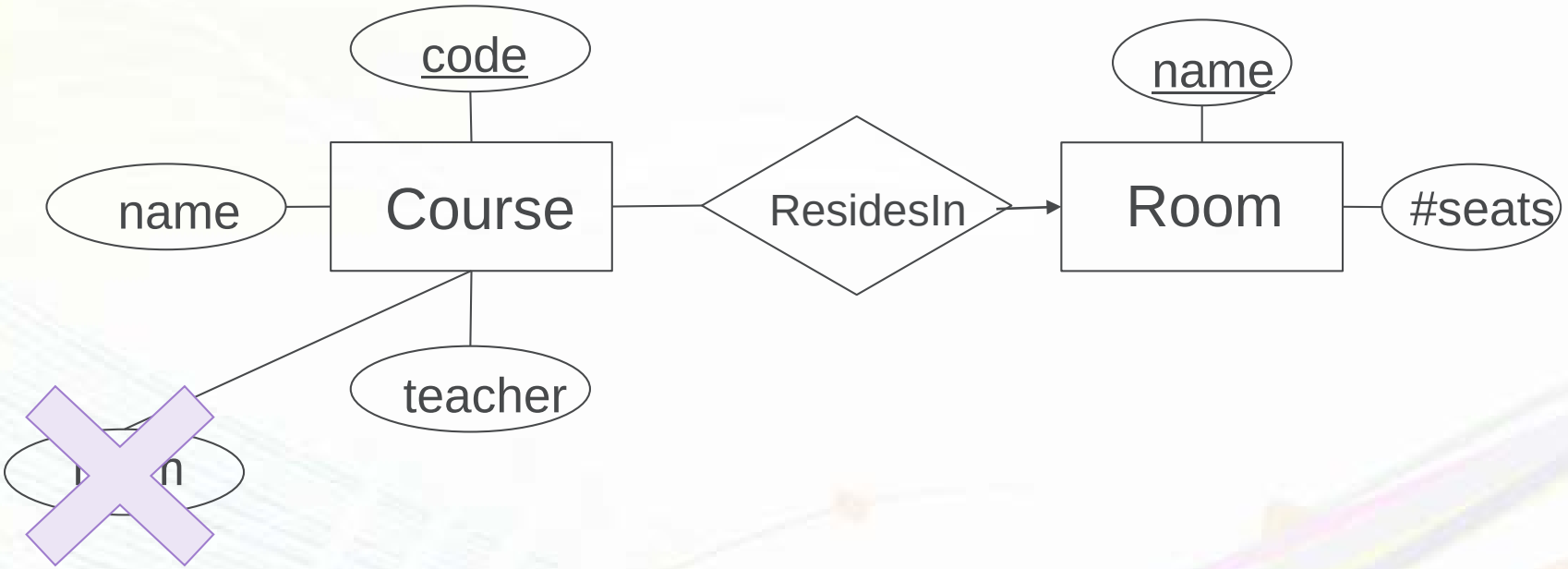
- Will lead to NULLs for courses that have no room.
- Typically used when *not* having a room is the exception to the rule.

```
Courses(code, name, teacher)
Rooms(name, #seats)
ResidesIn(code, room)
```

Note that "name" is not a key here

- No NULLs anywhere.
- May lead to much duplication of the course code.
- Typically used when *having* a room is the exception to the rule.

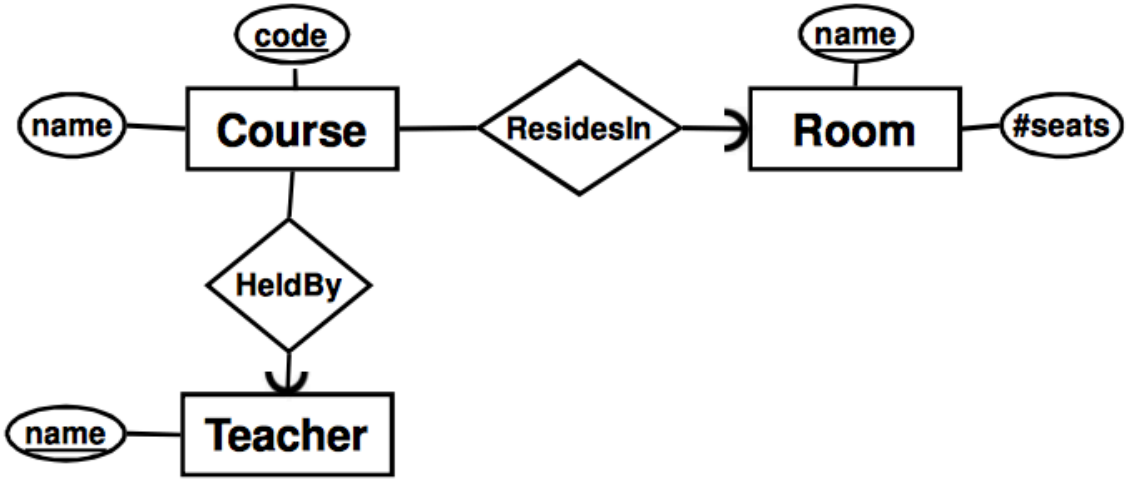
Bad E-R design



- Room is a related entity – not an attribute as well

Attribute or related entity?

What about teacher? Isn't that an entity?



Quiz

- When should we model something as an entity in its own right (as opposed to an attribute of another entity)?

At least one of the following should hold:

- Consists of more than a single (key) attribute
- Used by more than one other entity
- Part of an X-to-many relation as the many side
- Generally entity-ish, is important on its own

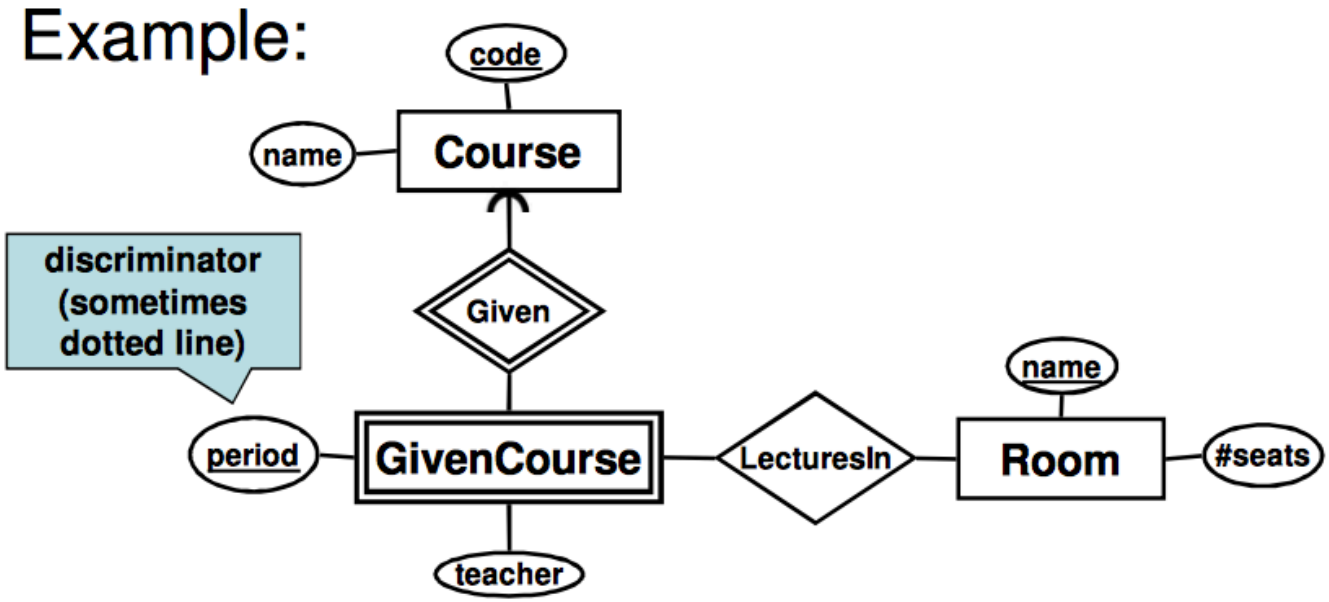
Weak entities

- 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.

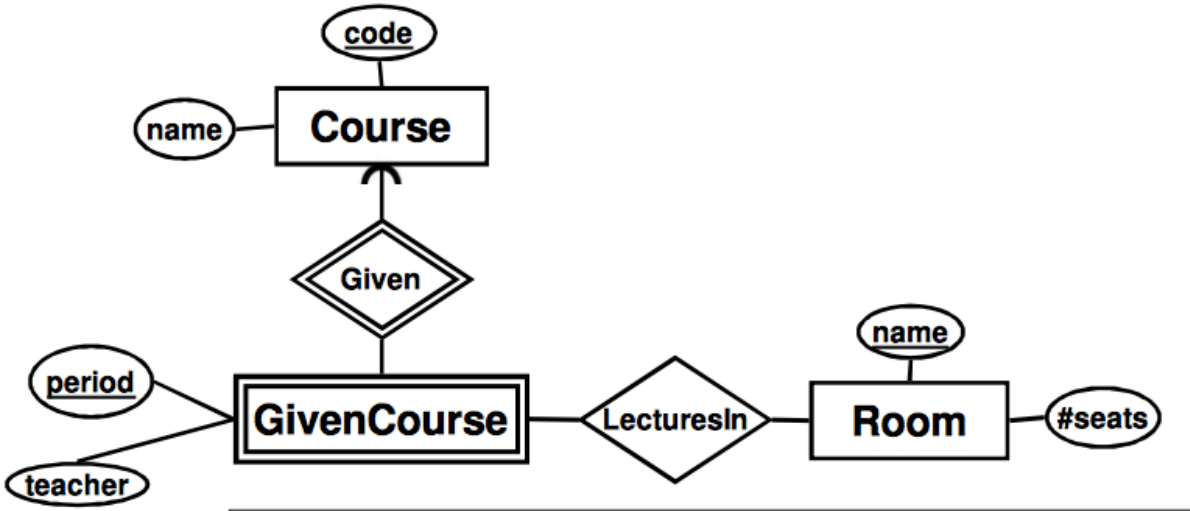
Weak entities

- 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 ER diagram



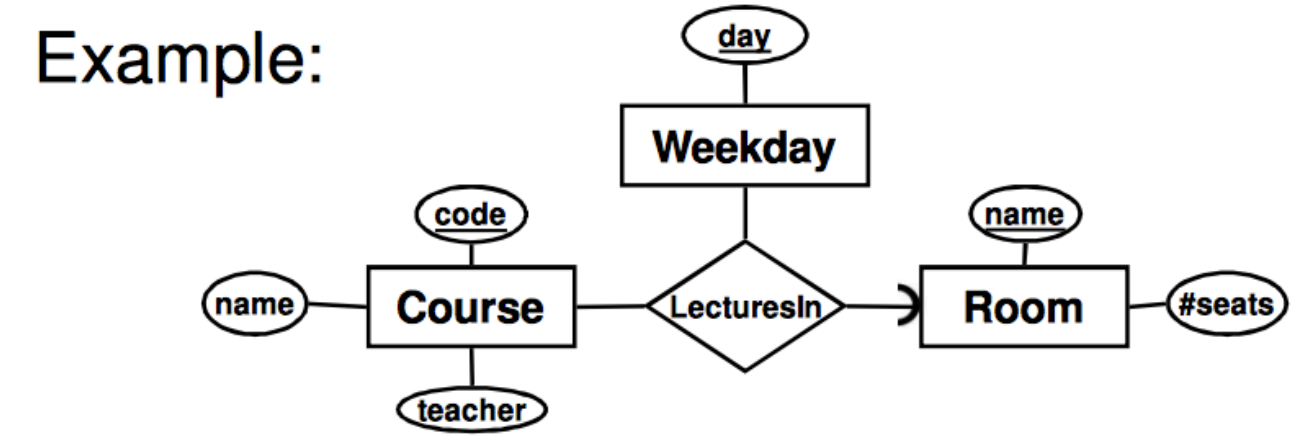
Translating to relations

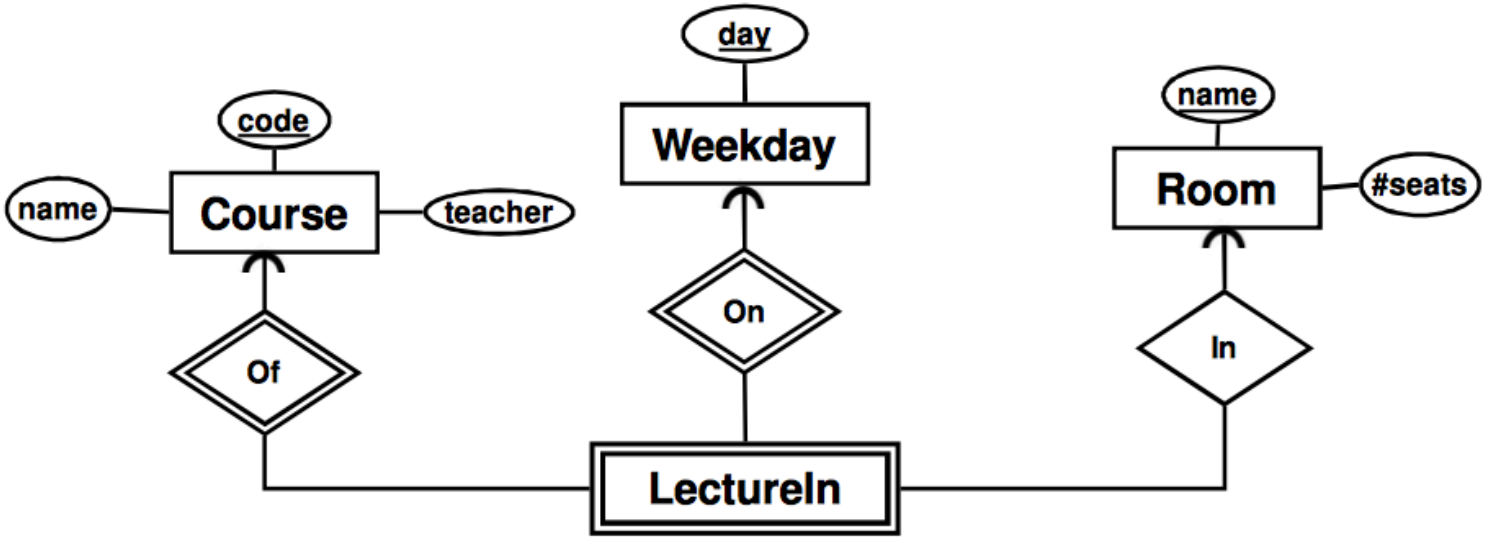


Multiway relationships as WEs

- Multiway relationships can be transformed away using weak entities
  - Substitute 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

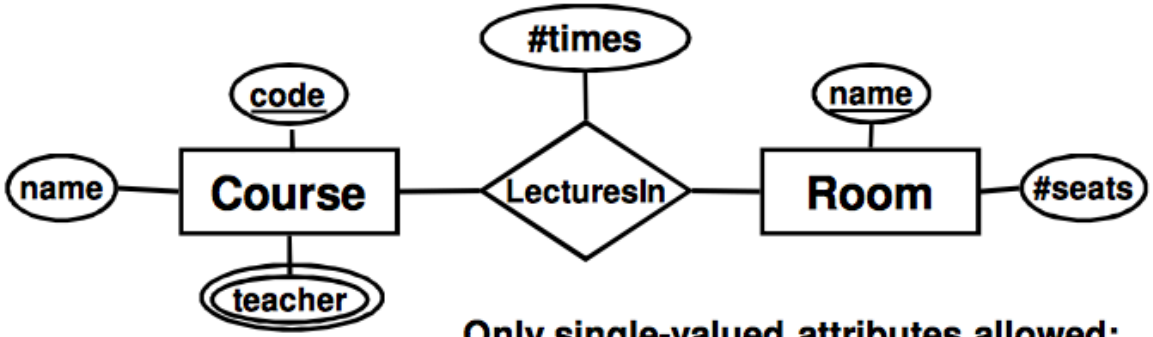

day
Weekday
code
name
name
Course
teacher
Room
#seats
On
Of
In
LectureIn

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

- If an attribute can have more than one value it is called multivalued:



Only single-valued attributes allowed:

```
Courses(code, name)

Teachers(code,t_name)
  code  -> Courses.code
Rooms(name, #seats)
LecturesIn(code, name, #times)
  code  -> Courses.code
  name  -> Rooms.name
```

Assignment#2

- (50 pts.) Design an Entity-Relationship Model that represents academic structure of Innopolis University.
  - Courses offered in Spring semester.
  - There are different groups of students and some courses are only offered to specific groups of students.
  - Instructors and classrooms for the courses. Remember there are multiple instructors for a course in IU but only 1 primary instructor for a given course.