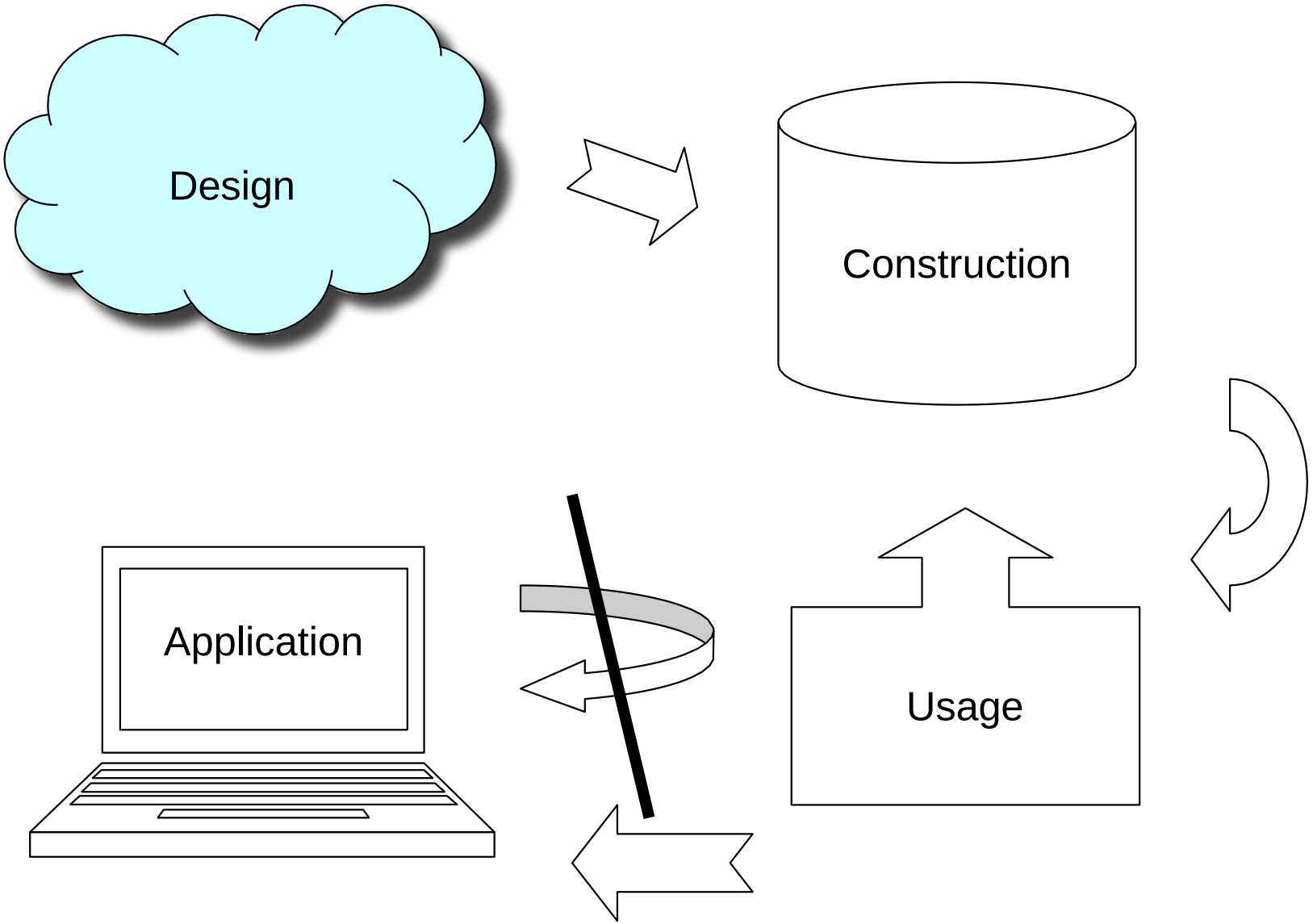
Lecture 3

Database design

The Entity-Relationship model

Course Objectives



The Entity-Relationship approach

- Design your database by drawing a picture of it – an *Entity-Relationship diagram*
  - Allows us to sketch the design of a database informally (which is good when communicating with customers)
- Use (more or less) mechanical methods to convert your diagram to relations.
  - This means that the diagram can be a formal specification as well

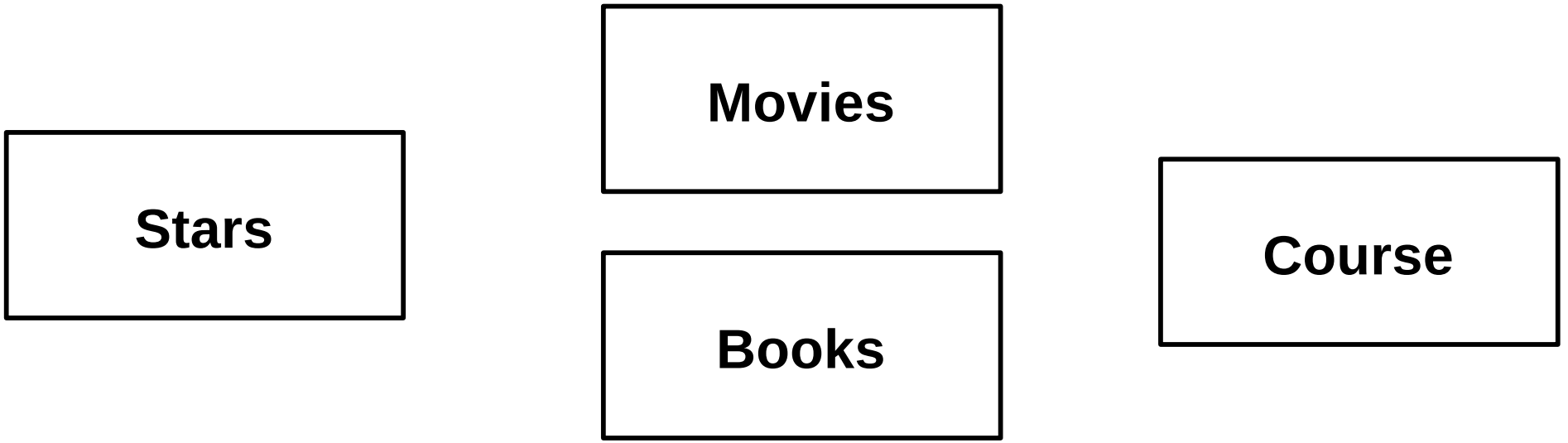
ER BASICS

E/R Model

- Three main element types:
  - Entity sets
  - Attributes, and
  - Relationships

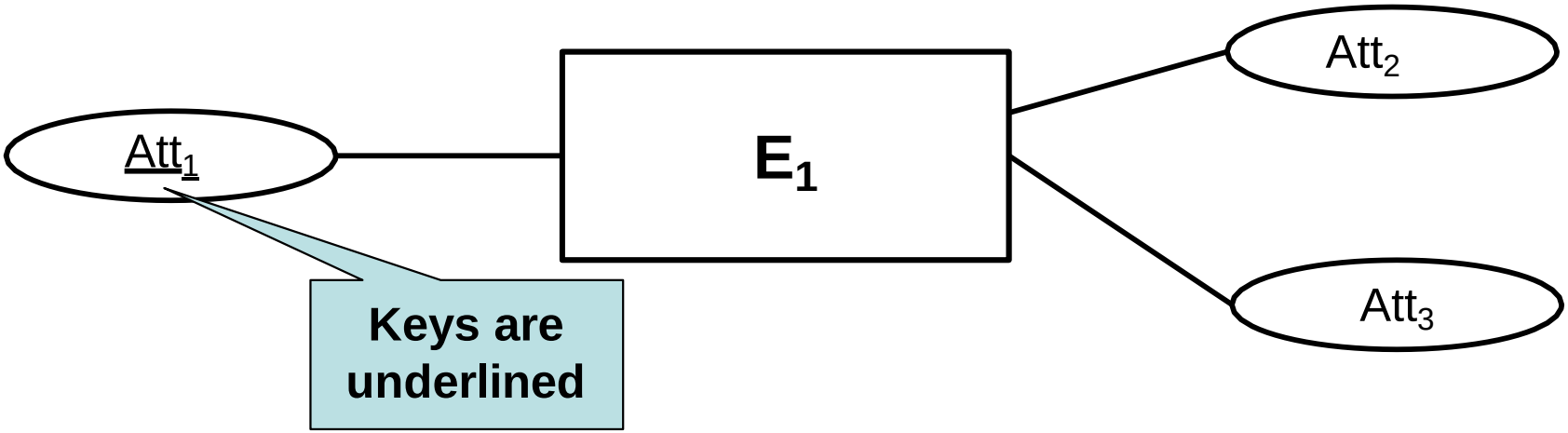
Entity Sets

- *Entity* = object that exists and distinguishable from other entities
  - course, room, person, customers, books, etc.
- *Entity set* = collection of similar entities
  - all courses, all rooms etc.
- Entities are drawn as rectangles



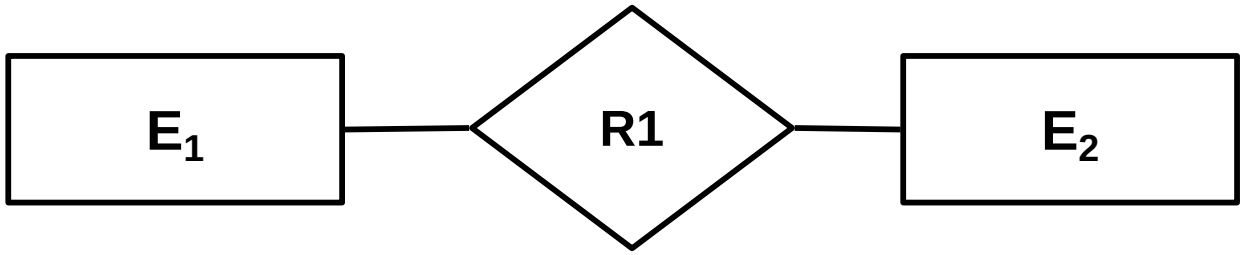
Attributes

- Entity sets have the same attributes (though not the same values)
- Attributes are drawn as ovals connected to the entity by a line.

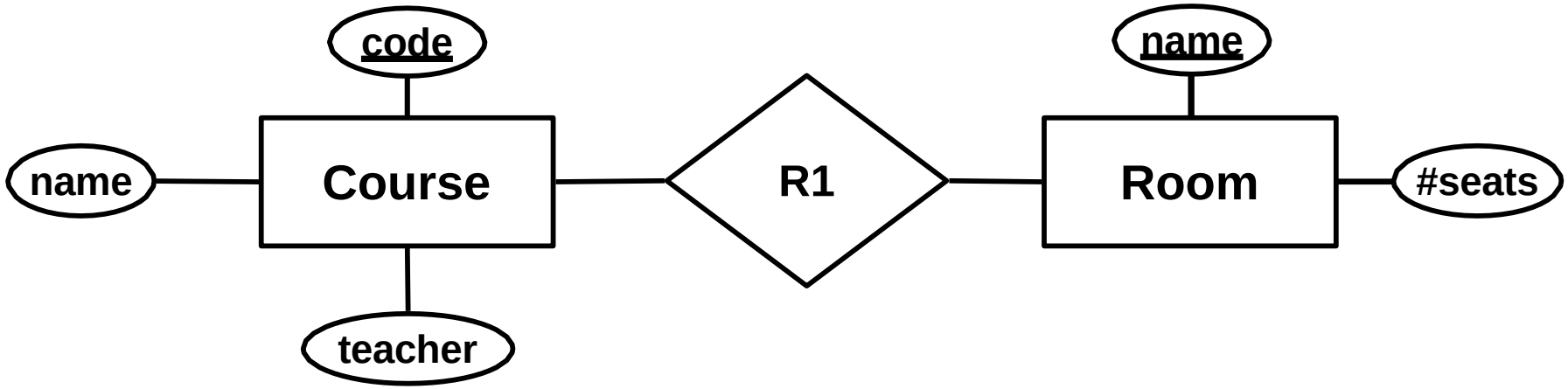


Relationships

- A relationship is an association among several entities
- Drawn as a diamond between the related entities, connected to the entities by lines.
- Note: Relationship ≠ Relation!!



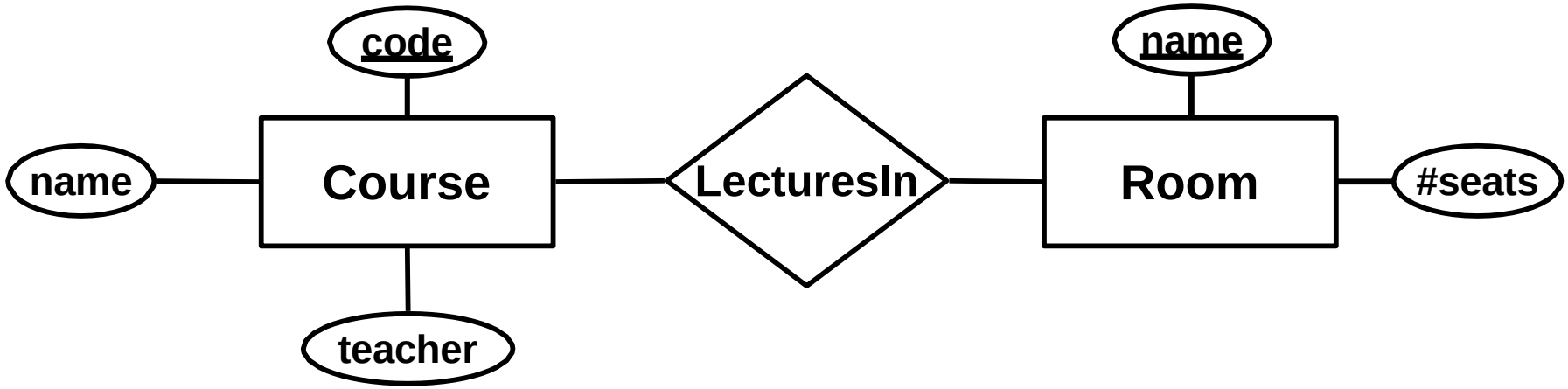
Examples:



- 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.

- Both entities are related through the relationship named ”R1”

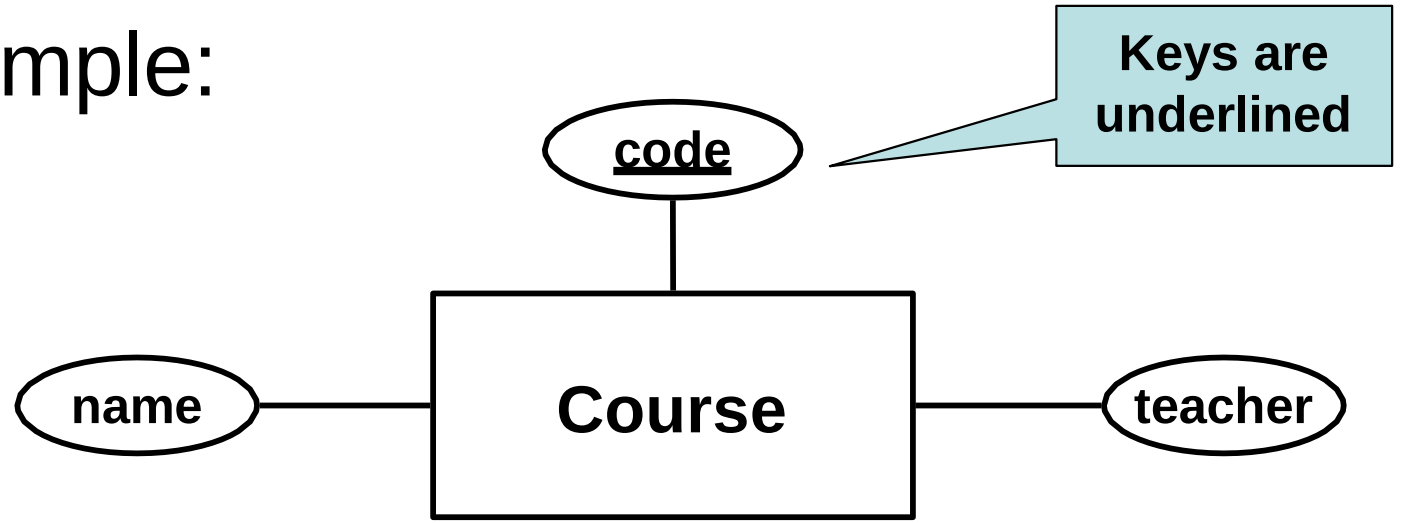
Example:



- 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 form (LecturesIn)

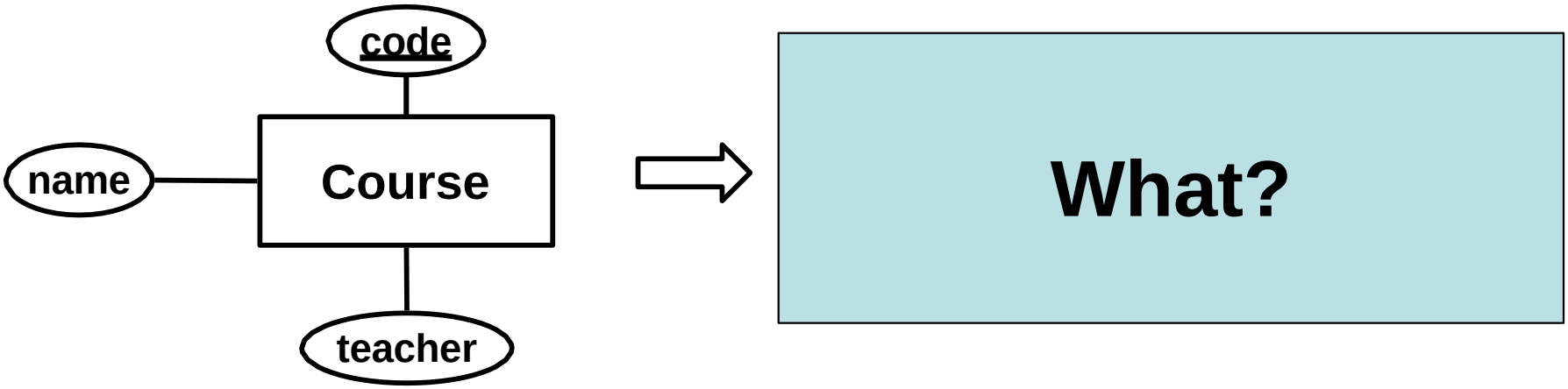
Example:



- 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. (TDA357, Databases, Mickey).

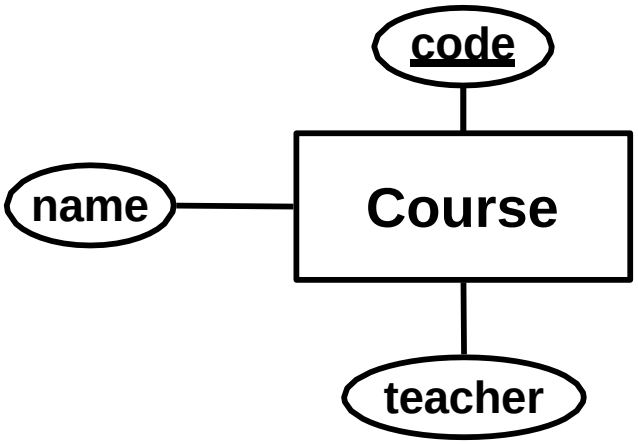
Translation to relations

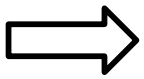
- An E-R diagram can be mechanically translated to a relational database schema.
- An entity becomes a relation, the attributes of the entity become the attributes of the relation, keys become keys.



Translation to relations

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





Courses(<u>code</u>, name, teacher)

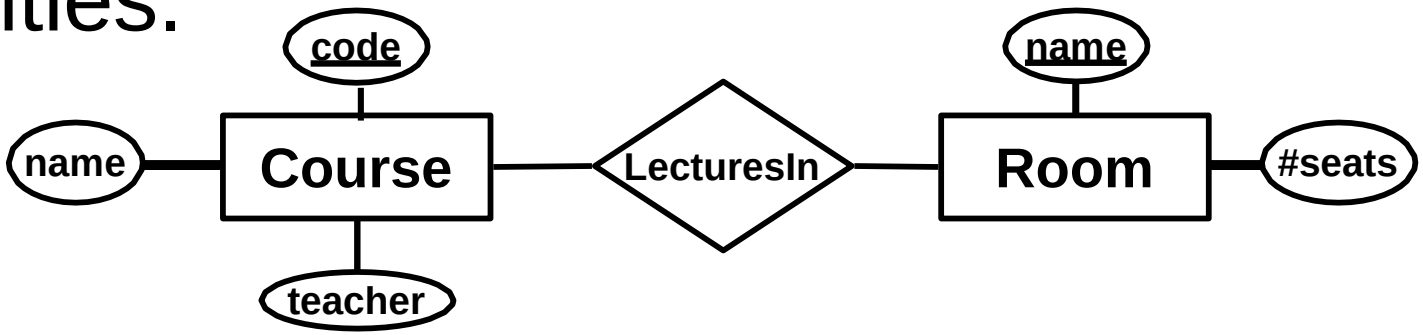
Books(<u>title</u>, <u>gender</u>)
Customers(<u>name</u>, address)
Movies(<u>title</u>, star, lenght)

A note on naming policies

- My view: A rectangle in an E-R diagram represents an entity, hence it is put in singular (e.g. Course).
  - Fits the intuition behind attributes and relationships better.
- The book: A rectangle represents an entity set, hence it is put in plural (e.g. Courses)
  - Easier to mechanically translate to relations.

Translation to relations

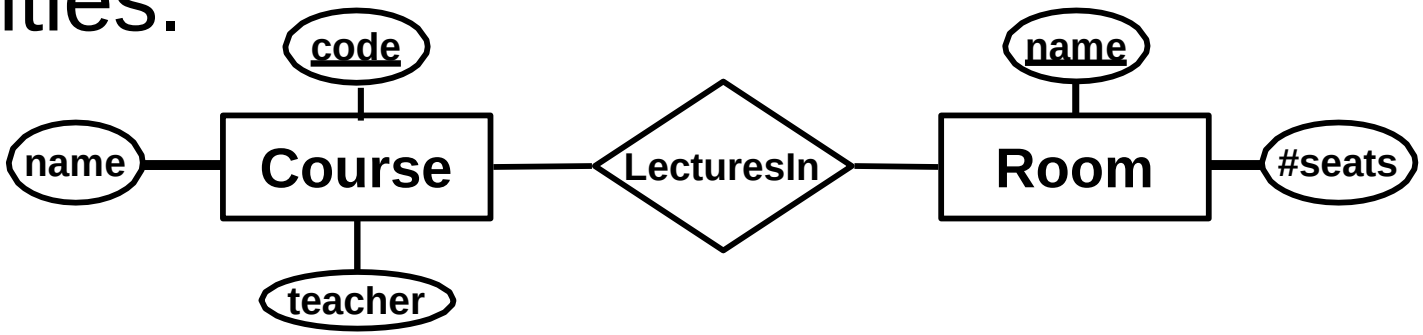
- A relationship between two entities is translated into a relation, where the attributes are the *keys* of the related entities.



⇨ **What?**

Translation to relations

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



⇨

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

References

```
Courses(code, name, teacher)
Teacher(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 codes used in **`LecturesIn`** *reference* those in **`Courses`**.

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

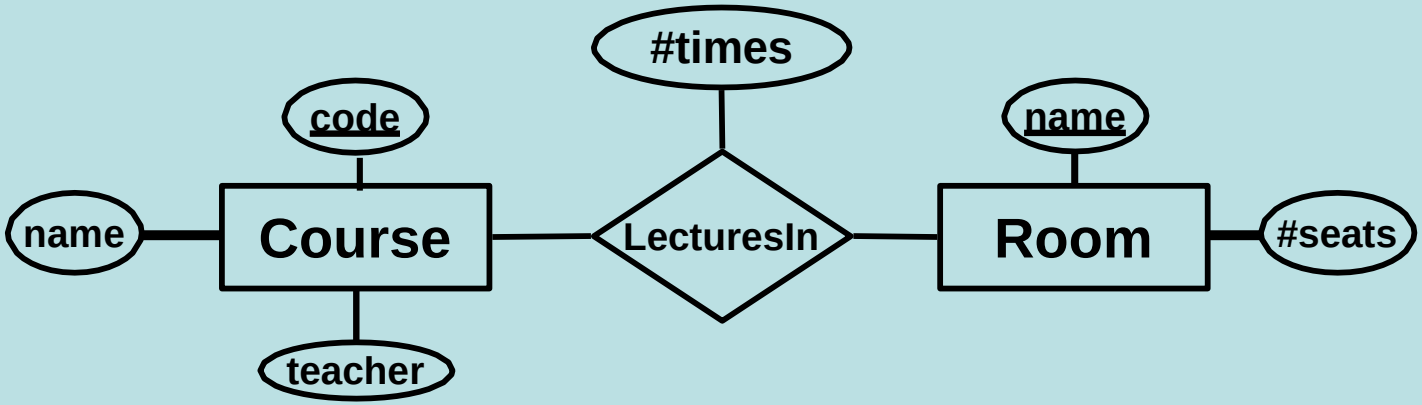
References

”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`**.

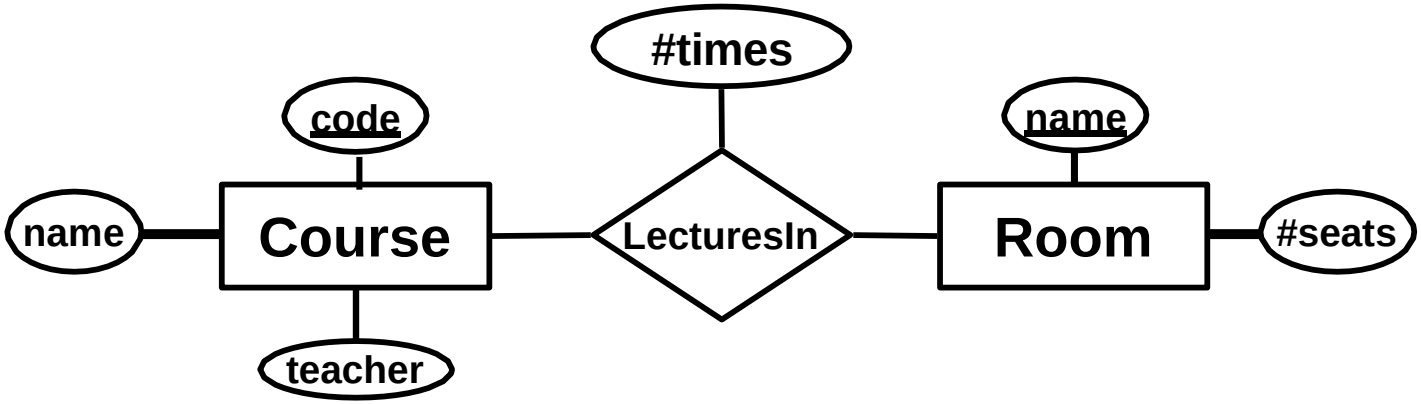
Quiz

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



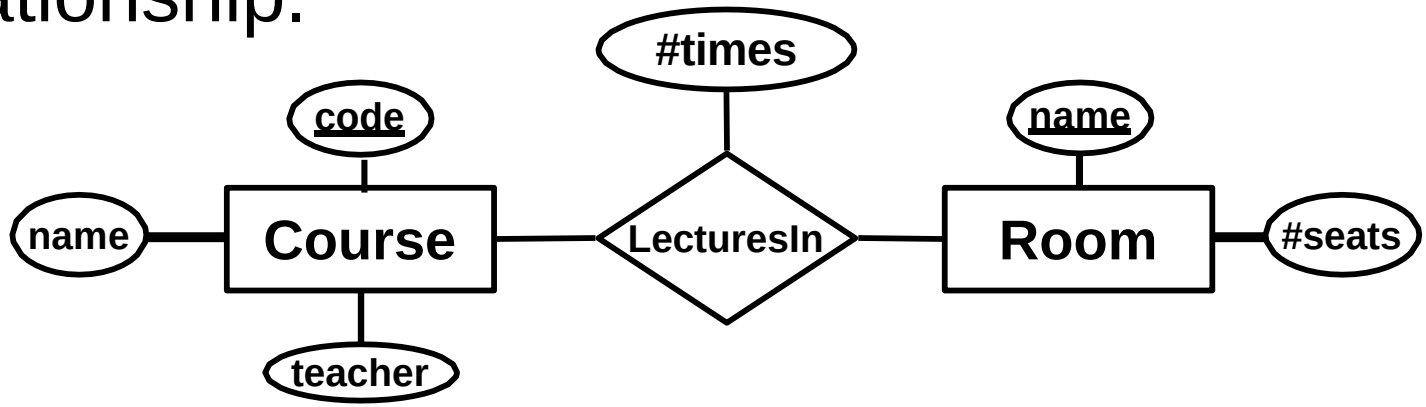
Attributes on relationships

- 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* of the related entities, plus any attributes of the relationship.

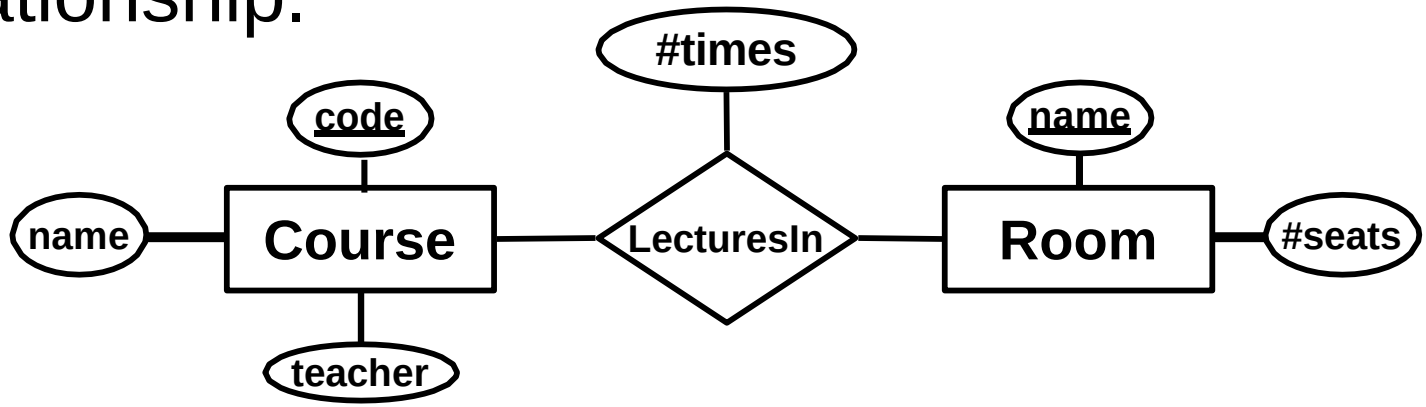


⇨

What?

Translation to relations

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

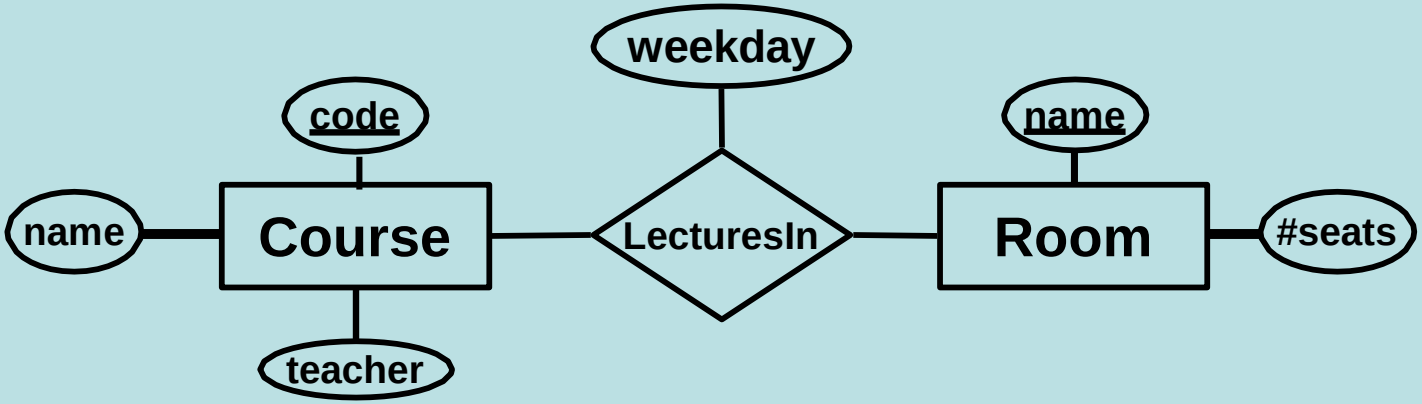


⇨

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

Quiz

Why could we not do the same for weekday?



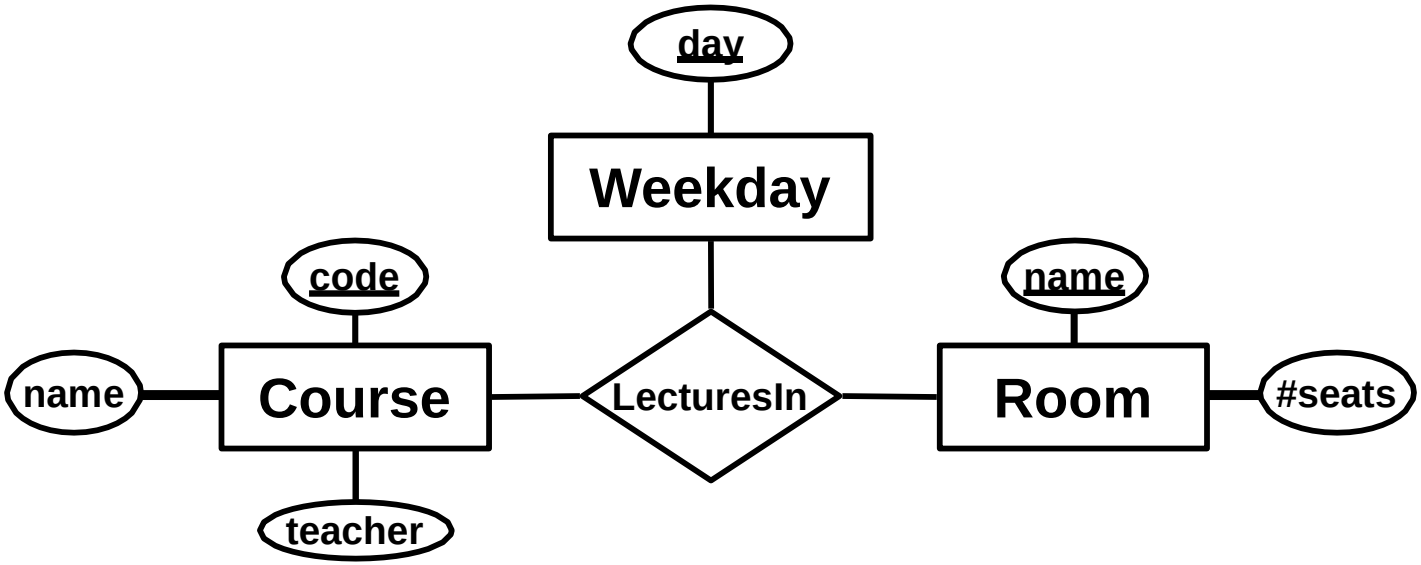
- Not a property of the relationship – a course can have lectures in a given room on several weekdays!
- A pair of entities are either related or not.

Relationship (non-)keys

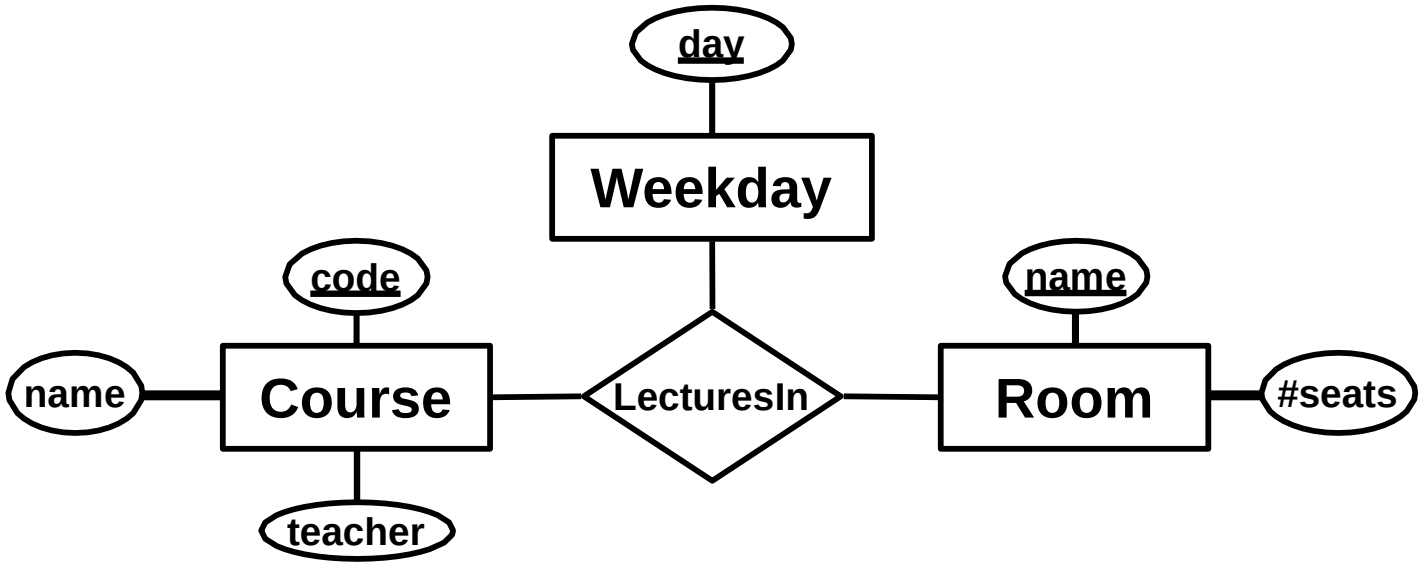
- Relationships have no keys of their own!
  - The ”key” of a relationship is the combined keys of the related entities
  - Follows from the fact that entities are either related or not.
  - If you at some point think it makes sense to put a key on a relationship, it should probably be an entity instead.

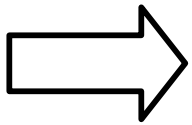
Multiway relationships

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



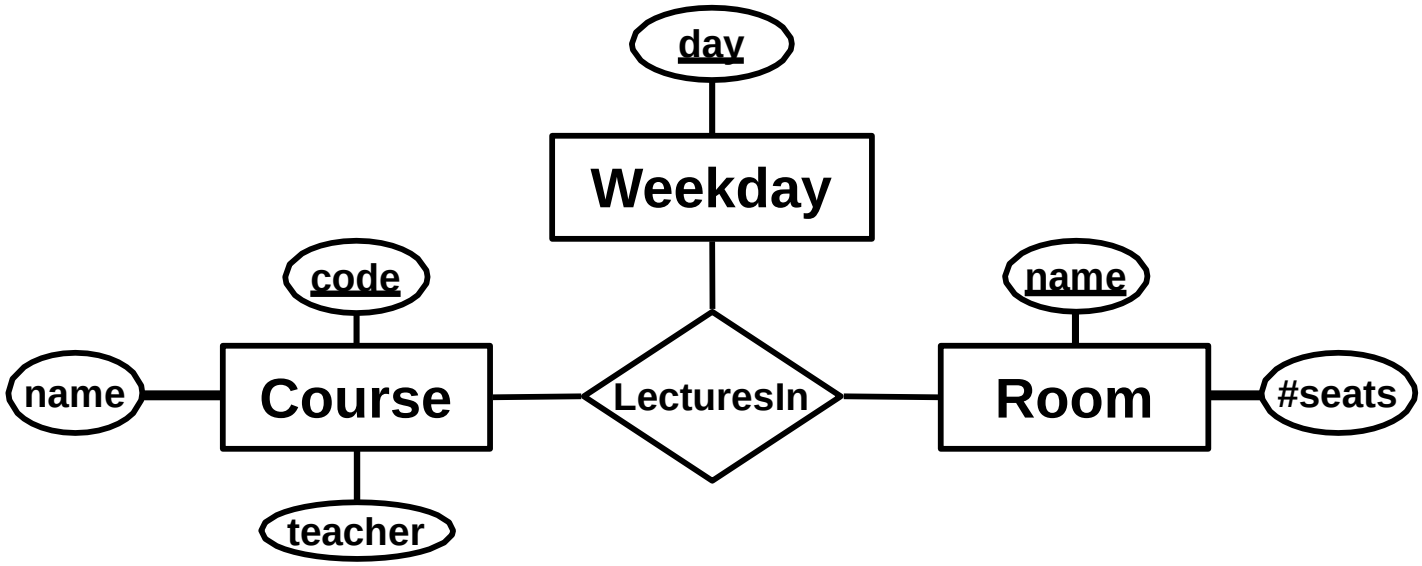
- Translating to relations:

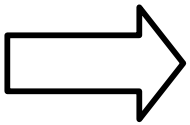




What?

- Translating to relations:

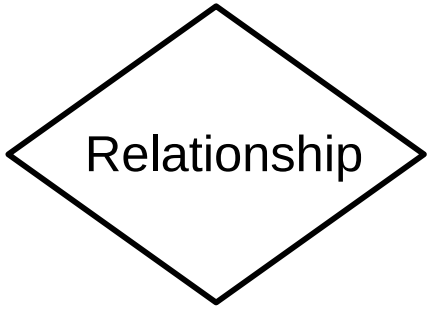




```
Courses(code, name, teacher)
Rooms(name, #seats)
Weekdays(day)
LecturesIn(code, name, day)
  code  -> Courses.code
  name -> Rooms.name
  day   -> Weekdays.day
```

ER Cheatsheet 1

Entity



ENTITY = noun/thing

- Exist on their own
- Have their own keys

RELATIONSHIP = verb

- Only exist in relation to an entity
- No own keys, only foreign keys
- Reference the entity keys with `->`

```
Course(code, name, teacher)
Room(name, #seats)
Weekday(day)
```

```
HasLecturesIn(code,name,day,#times)
   code -> Course.code
   name -> Room.name
   day -> Weekday.day
```

Both entities and relationships can have attributes!