



The Relational Model

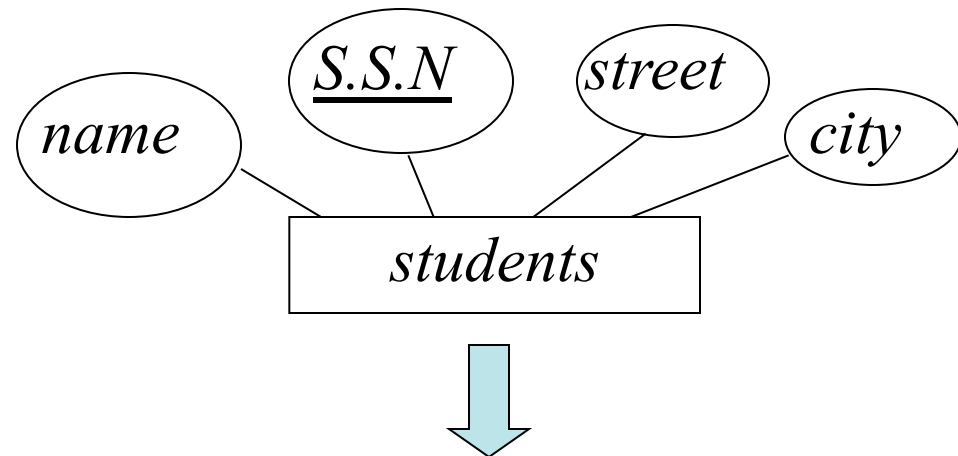
Week 2

Relations

A **relation** is a more concrete construction, of something we have seen before, the ER diagram.

A relation is (just!) a table!

We will use **table** and **relation** interchangeably, except where there is a possibility of confusion.



<i>name</i>	<u><i>S.S.N</i></u>	<i>street</i>	<i>city</i>
Lisa	1272	Main	Fairfax
Bart	5592	Apple	Manassas
Lisa	7552	Ox	Fairfax
Sue	5555	Lee	Vienna

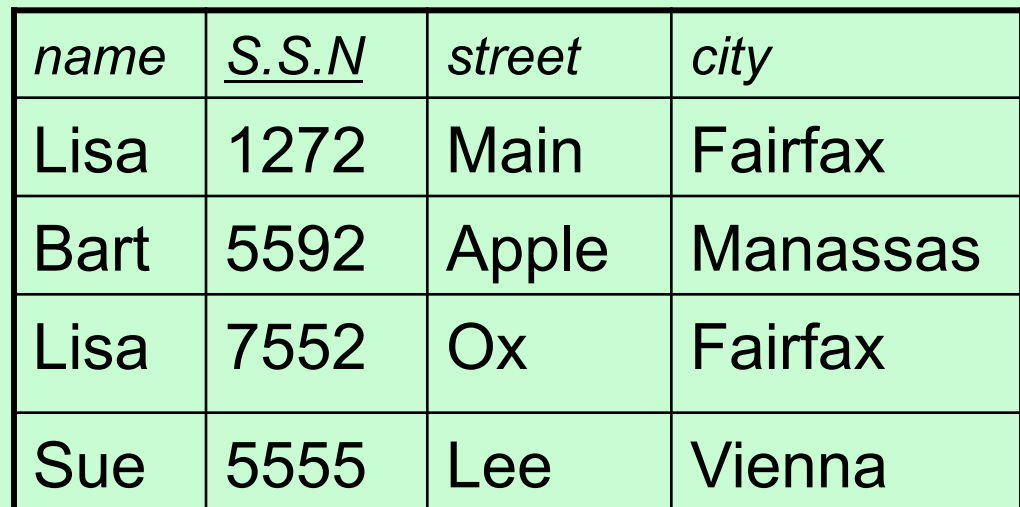
The students relation

A **relation** consists of a **relational schema** and a **relational instance**.

A **relation schema** is essentially a list of column names with their data types. In this case...

`students(name : string, S.S.N : string, street : string, city : string)`

- A **relation instance** is made up of zero or more **tuples** (rows, records)



<i>name</i>	<u><i>S.S.N</i></u>	<i>street</i>	<i>city</i>
Lisa	1272	Main	Fairfax
Bart	5592	Apple	Manassas
Lisa	7552	Ox	Fairfax
Sue	5555	Lee	Vienna

A schema specifies a relation's name.

students(*name* : string, *S.S.N* : string, *street* : string, *city* : string)

A schema also specifies the name of each **field**, and its domain.

Fields are often referred to as columns, attributes, dimensions

A minor, but important point about relations, they are unordered.

<i>name</i>	<u><i>S.S.N</i></u>	<i>street</i>	<i>city</i>
Lisa	1272	Main	Fairfax
Bart	5592	Apple	Manassas
Lisa	7552	Ox	Fairfax
Sue	5555	Lee	Vienna

<i>name</i>	<u><i>S.S.N</i></u>	<i>city</i>	<i>street</i>
Lisa	1272	Fairfax	Main
Bart	5592	Manassas	Apple
Lisa	7552	Fairfax	Ox
Sue	5555	Vienna	Lee

This is not a problem, since we refer to fields by name.

However sometimes, we refer to the fields by their column number, in which case the ordering becomes important. I will point this out when we get there.

Also, the tuples are unordered too!

Note that every tuple in our instance is unique. This is not a coincidence. The definition of relation demands it.

Later we will see how we can represent weak entities in relations.

<i>name</i>	<u><i>S.S.N</i></u>	<i>street</i>	<i>city</i>
Lisa	1272	Main	Fairfax
Bart	5592	Apple	Manassas
Lisa	7552	Ox	Fairfax
Sue	5592	Lee	Vienna

The number of fields is called the **degree** (or **arity**, or dimensionality of the relation).

Below we have a table of degree 4.

The number of tuples = **cardinality** of the relation

Of course, we don't count the row that has the labels!

To the right we have a table of cardinality 3.

<i>name</i>	<u><i>S.S.N</i></u>	<i>street</i>	<i>city</i>
Lisa	1272	Main	Fairfax
Bart	5592	Apple	Manassas
Lisa	7552	Ox	Fairfax

students(*name* : string, *S.S.N* : string, *street* : string, *city* : string)

Note that relations have primary keys, just like ER diagrams. Remember that the primary key might not be one field, it may be a combination of two or more fields.

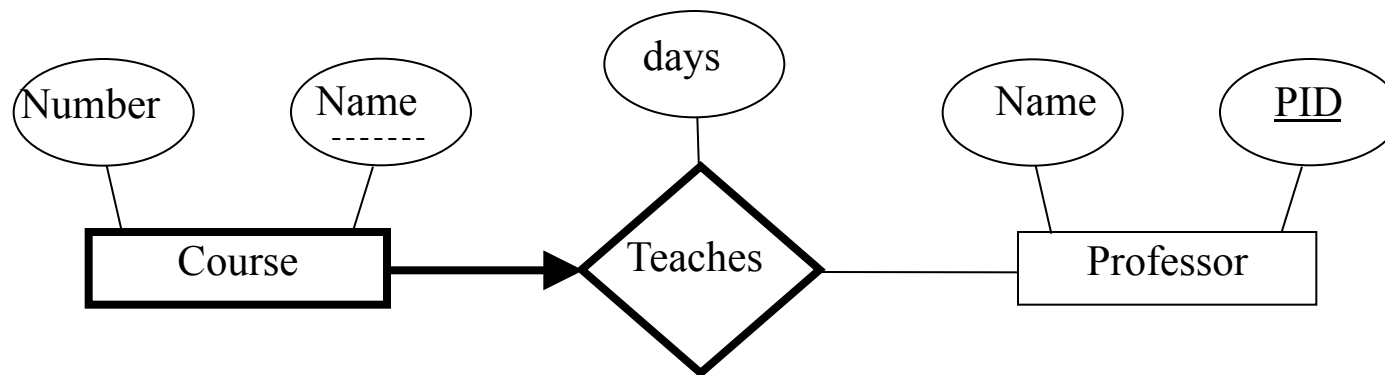
<i>name</i>	<u><i>S.S.N</i></u>	<i>street</i>	<i>city</i>
Lisa	1272	Main	Fairfax
Bart	5592	Apple	Manassas
Lisa	7552	Ox	Fairfax
Sue	5555	Lee	Vienna

Translating ER diagrams into Relations

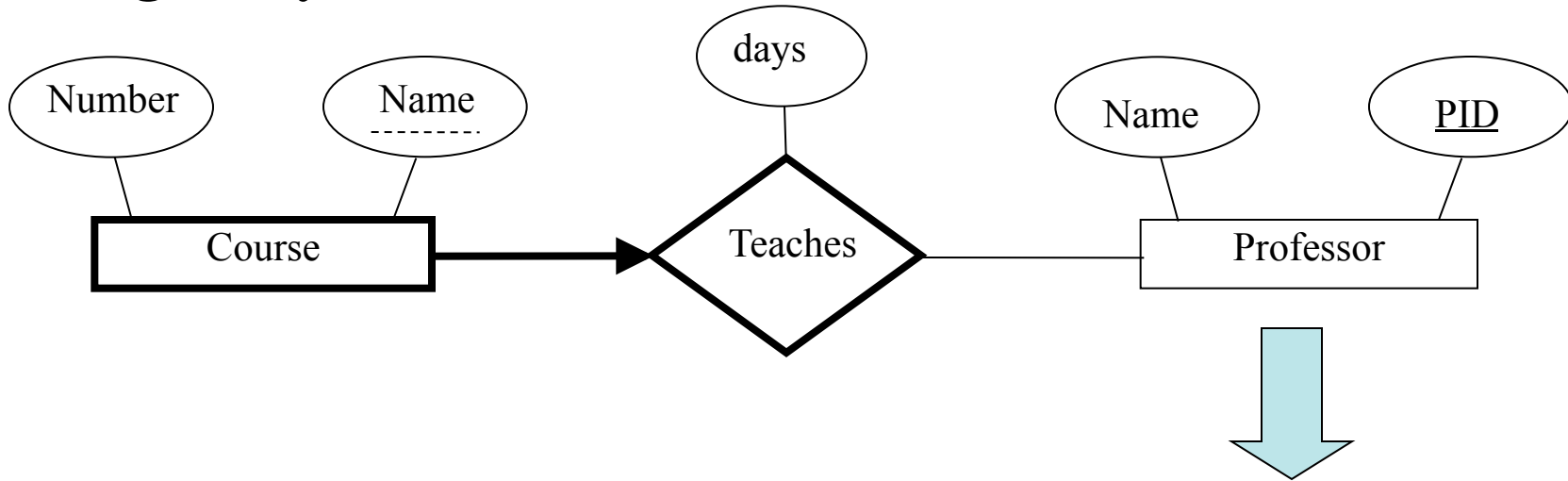
We need to figure out how to translate ER diagrams into relations.

There are only three cases to worry about.

- Strong entity sets
- Weak entity sets
- Relationship sets



- Strong entity sets

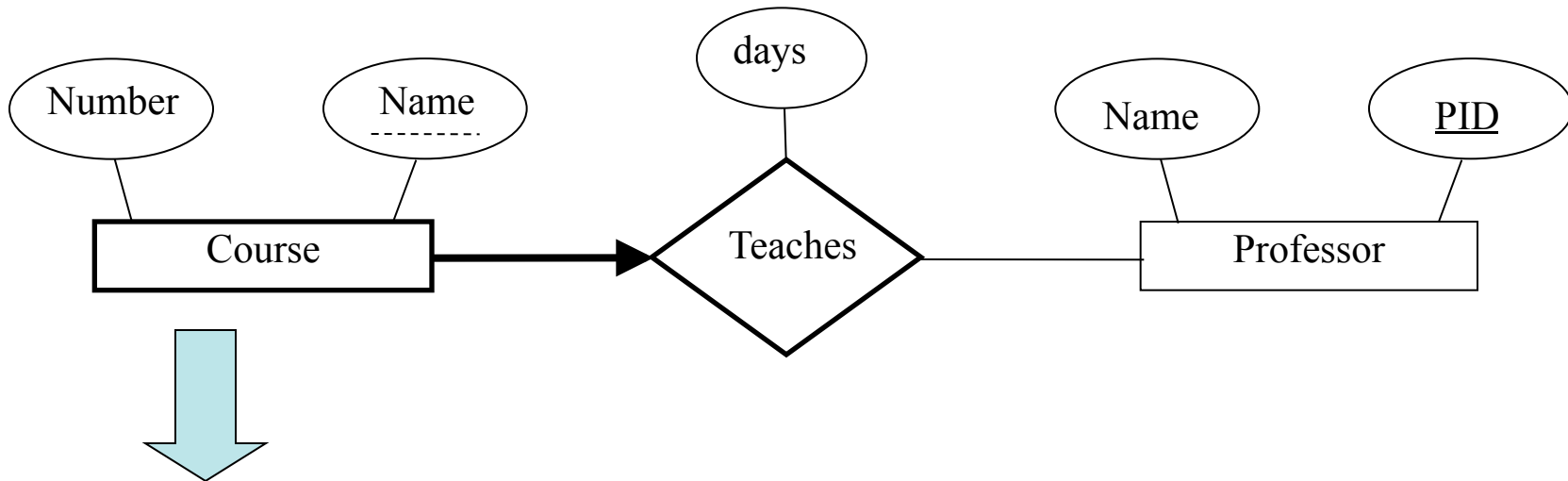


professor(PID : string, name : string)

This is trivial, the primary key of the ER diagram becomes the primary key of the relation. All other fields are copied in (in any order)

<u>PID</u>	name
1234	Lin
3421	Lee
2342	Smyth
4531	Lee

- **Weak entity sets**



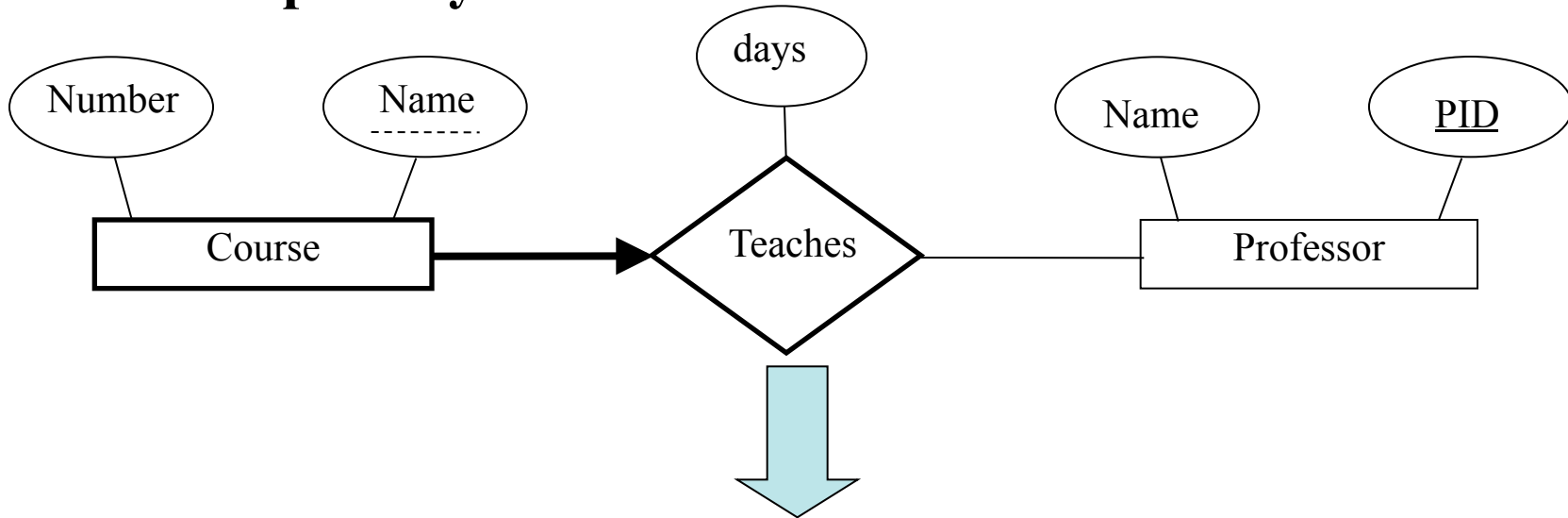
course(PID : string, *number* : string, name : string)

<u>PID</u>	<i>number</i>	<u>name</u>
1234	CS12	C++
3421	CS11	Java
2342	CS12	C++
4531	CS15	LISP

The primary key of the relation consists of the union of the primary key of the strong entity set and the discriminator of the weak entity set. The “imported key from the strong entity set is called the **foreign key**.

All other fields are copied in (in any order)

• Relationship entity sets



teaches(PID : string, *days* : string)

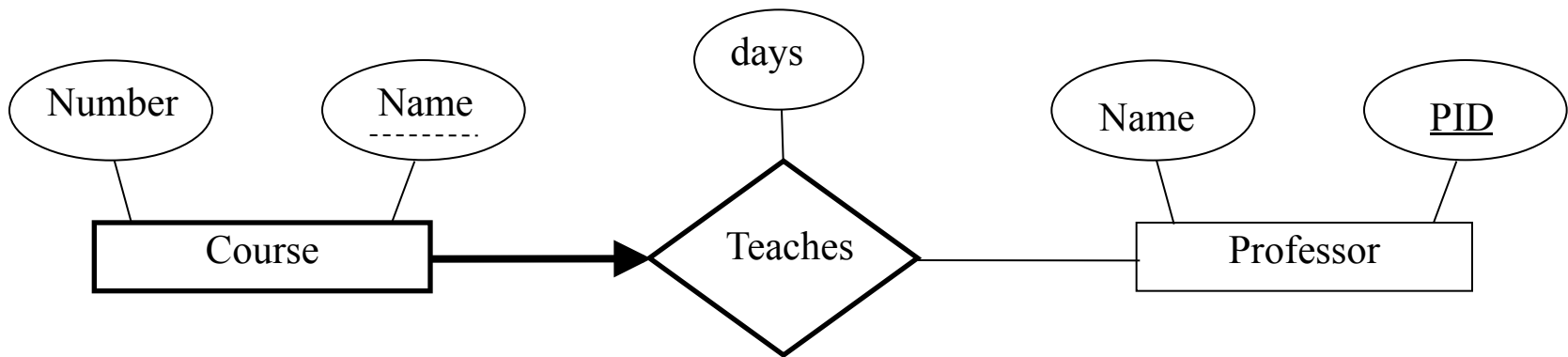
For one-to-one relationship sets, the relation's primary key can be that of either entity set.

- For many-to-many relationship sets, the union of the primary keys becomes the relation's primary key

- For other cases, the relation's primary key is taken from the strong entity set.

<u>PID</u>	<i>days</i>
1234	mwf
3421	wed
2342	tue
4531	sat

So, this ER Model...



... maps to this **database schema**

professor(PID : string, name : string)

course(PID : string, number : string, name : string)

teaches(PID : string, days : string)

Later we'll see how we can take advantage of the key constraint and come up with a better design.