Lecture 07.03.18

Today

- Database modeling
- E/R (entity/relationship) diagrams
- Modeling example
- Normalization

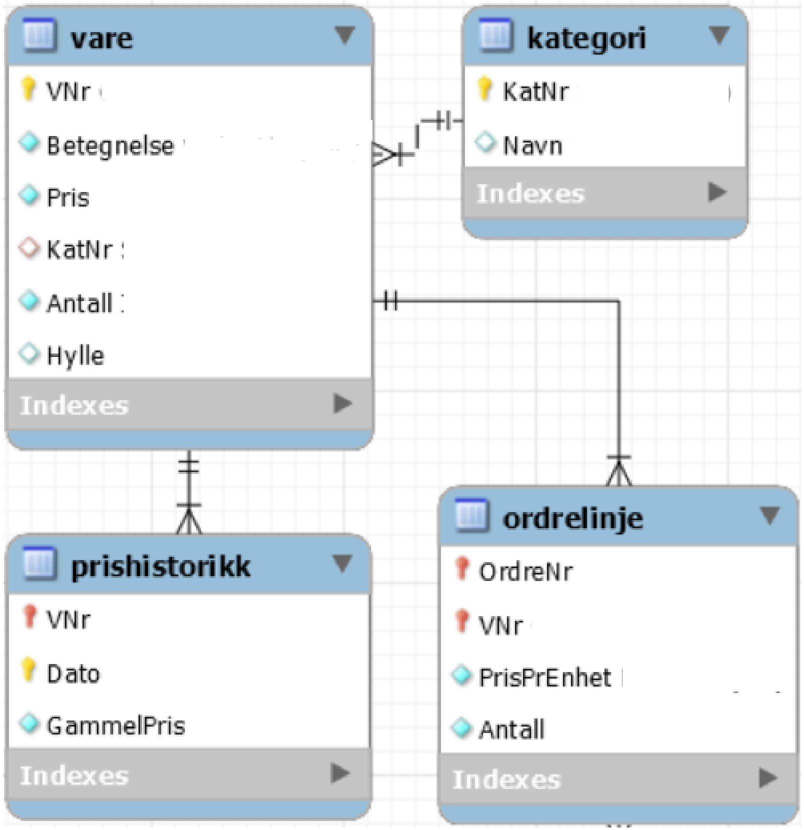
Covers chapter 7 and 8.

Purpose of database modeling

- Simpler representation of a database
- Makes planing and designing databases easier
- Makes explaining the database easier

- Entity/relationship (E/R) diagrams is a visual tool to design and explore databases with

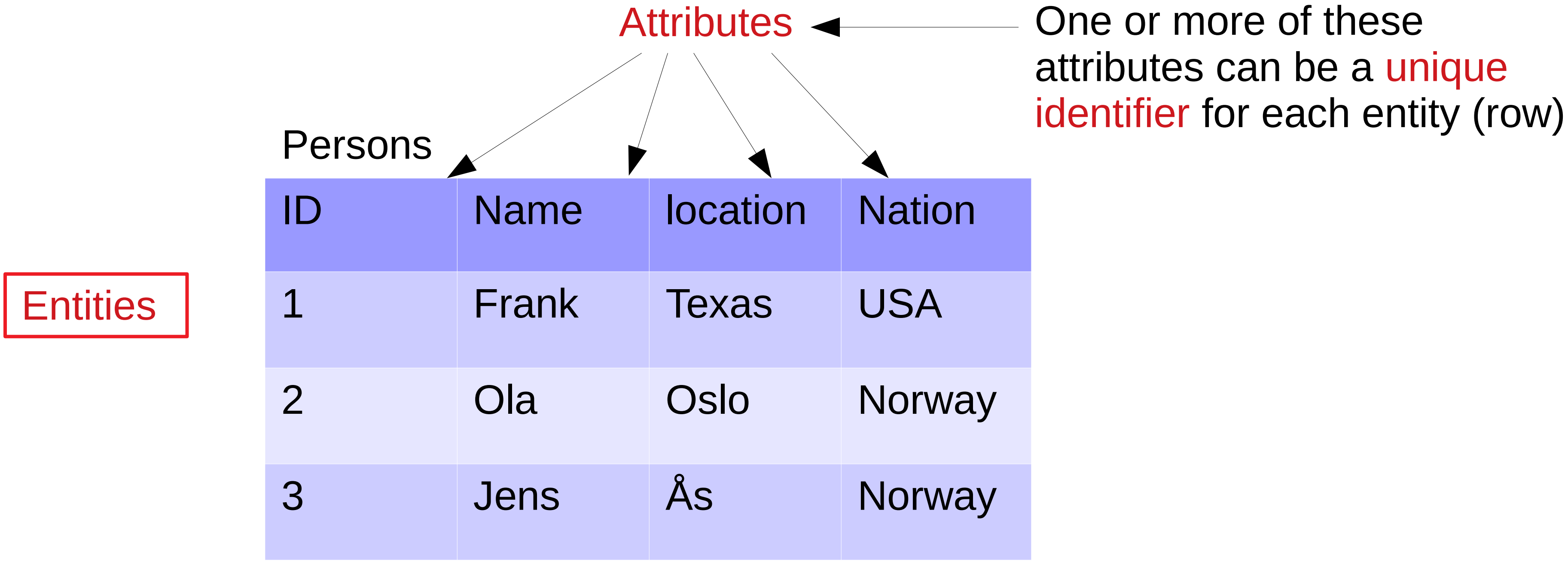
E/R diagram example



From model to database

- Conceptual design
  - Technology independent (excel,database,python)
  - E/R (Entity/Relationship) is popular
- Logical design
  - DBMS independent
- Physical design
  - DBMS dependent
  - Contains all details about implementation (SQL code)

Table representation from before



Attributes

One or more of these attributes can be a unique identifier for each entity (row)

Persons

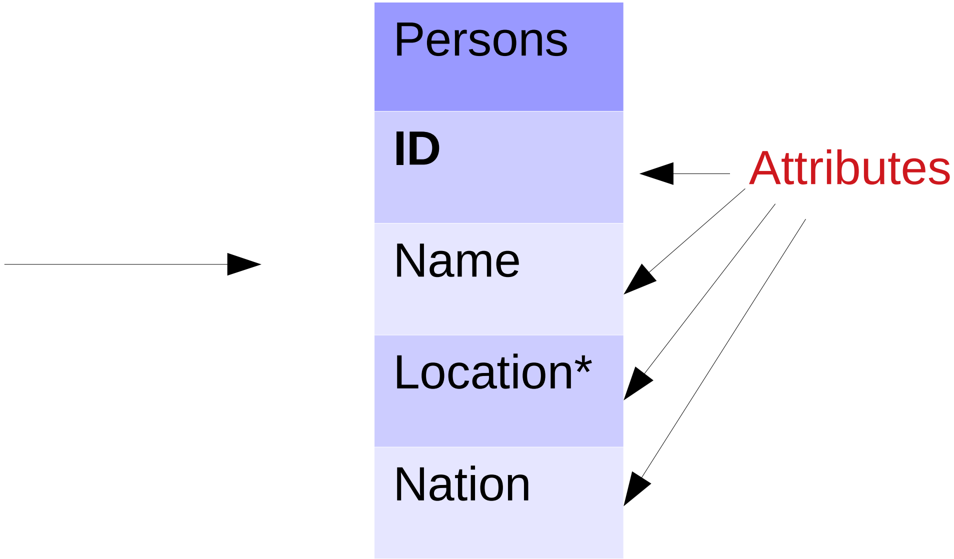
Entities

ID	Name	location	Nation
1	Frank	Texas	USA
2	Ola	Oslo	Norway
3	Jens	Ås	Norway

Table representation

Persons

ID	Name	location*	Nation
1	Frank	Texas	USA
2	Ola	Oslo	Norway
3	Jens	Ås	Norway



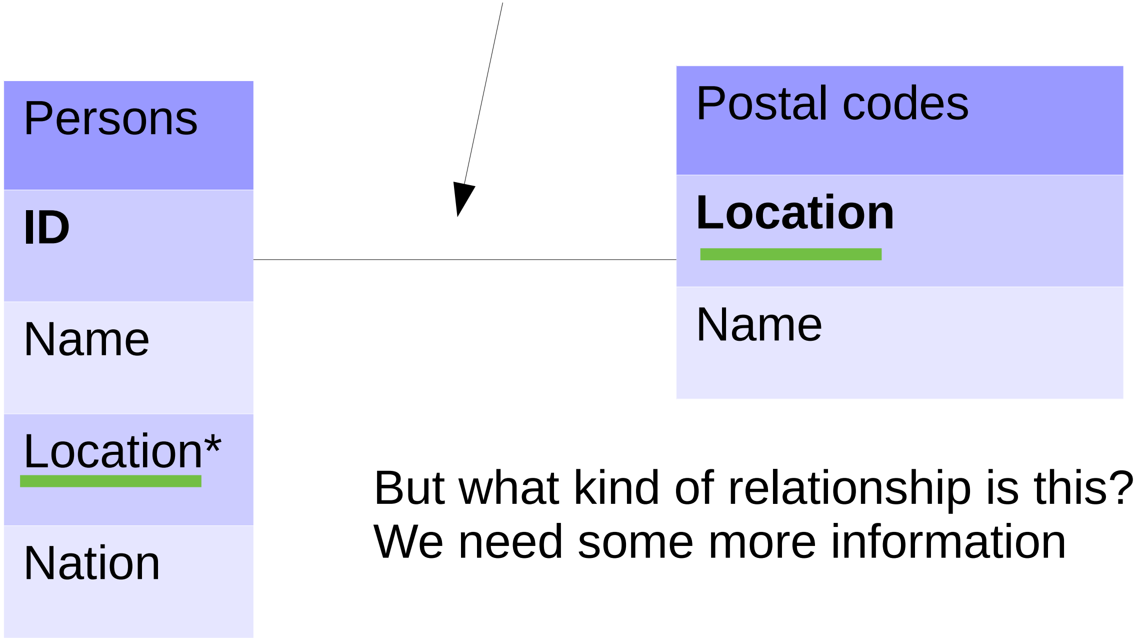
Each entity (row) of the table belongs to the Entity type

Entity type refers to the table as a whole

Relationship notation

A simple line symbolizes that there is a relationship

Foreign keys are the cause of the relationship



Persons

ID

Name

Location*

Nation

Postal codes

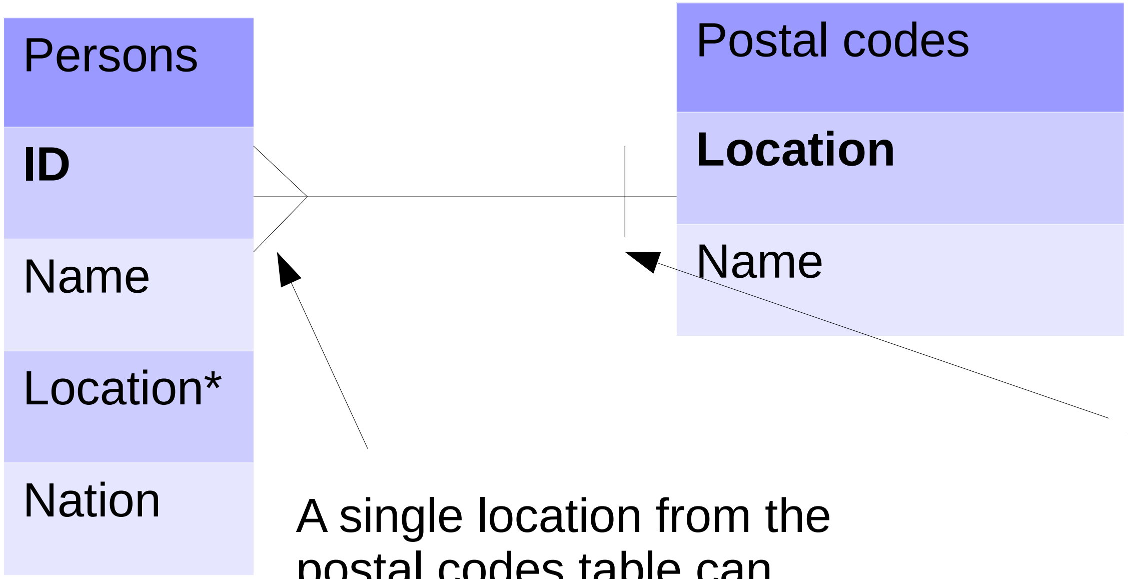
Location

Name

But what kind of relationship is this?
We need some more information

Relationship type (cardinality)

Solved by asking the question:
How many times can an entity exist in a table it is related to?



Persons

ID

Name

Location*

Nation

Postal codes

Location

Name

This example is called a one to many relationship

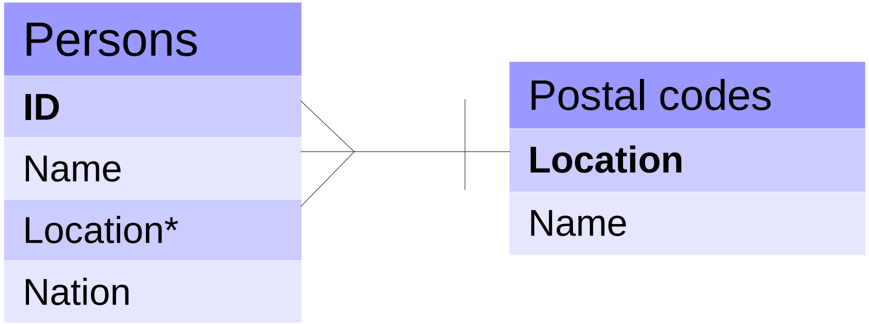
A single location from the postal codes table can exists multiple (**many**) times in the persons table

A location in the persons table can only exist **once** in the postal codes table

Relationship type (cardinality)

There are 3 relationship types (Cardinalities)

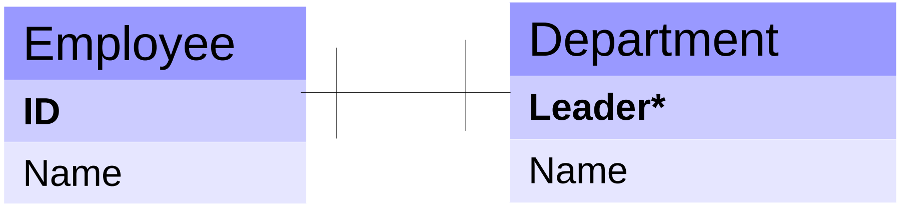
One to many (1:N)



Persons
ID
Name
Location*
Nation

Postal codes
Location
Name

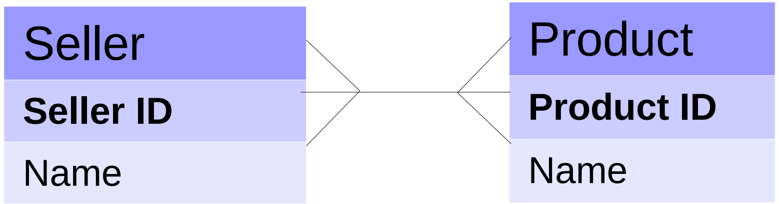
One to one (1:1)



Employee
ID
Name

Department
Leader*
Name

Many to many (M:N)



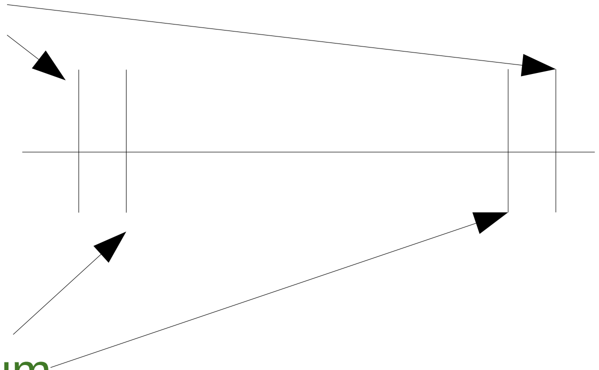
Seller
Seller ID
Name

Product
Product ID
Name

Minimum/maximum cardinality

Outermost is maximum

Maximum can be either one |
or many >



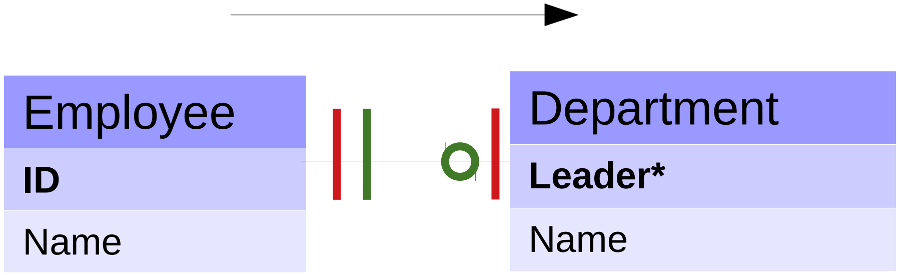
Innermost is minimum

Minimum can be either one | or zero ○

Minimum/maximum cardinality

How to read the relationships with minimum/maximum cardinality

An employee can be a leader of minimum zero and maximum one department



Employee

ID

Name

Department

Leader*

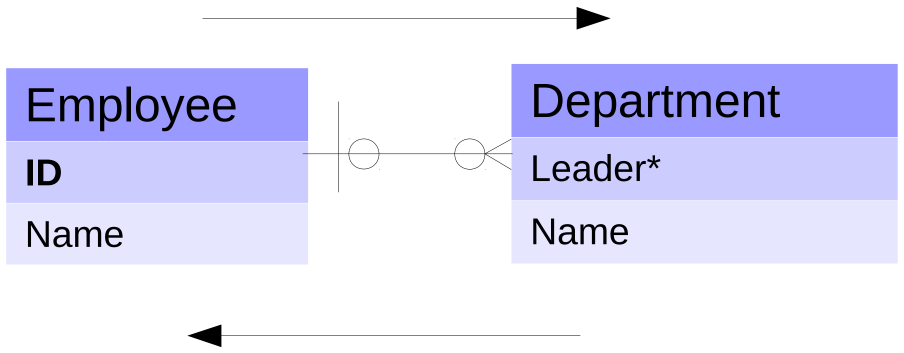
Name

A leader can be minimum one and maximum one employee

Minimum/maximum cardinality

If leader isn't primary key or unique and can be NULL

An employee can be the leader of zero, one or many departments



Employee

ID

Name

Department

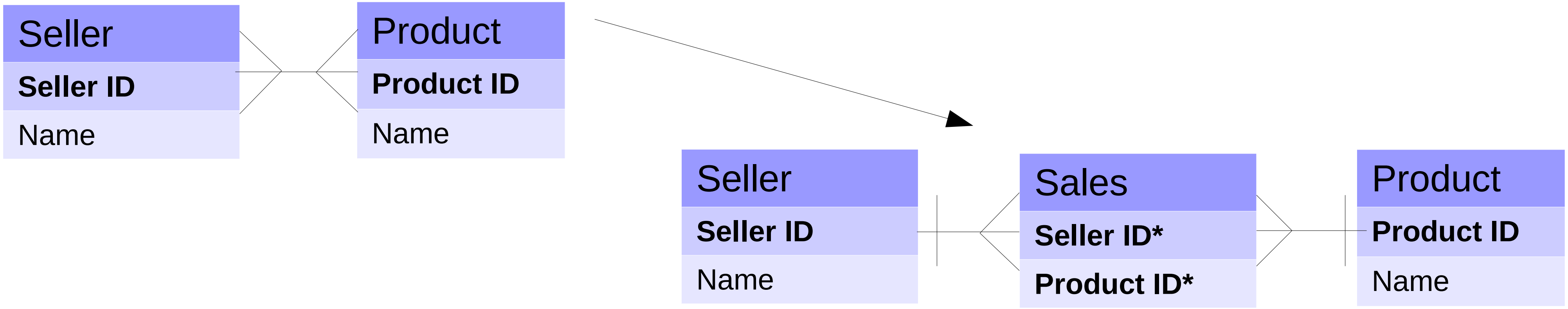
Leader*

Name

A department can have zero or one leader

Many to many

- When making the actual physical database, there is no such thing as a many to many relation.
- They are broken up into two or more one to many (1:N) relations



Seller
Seller ID
Name

Product
Product ID
Name

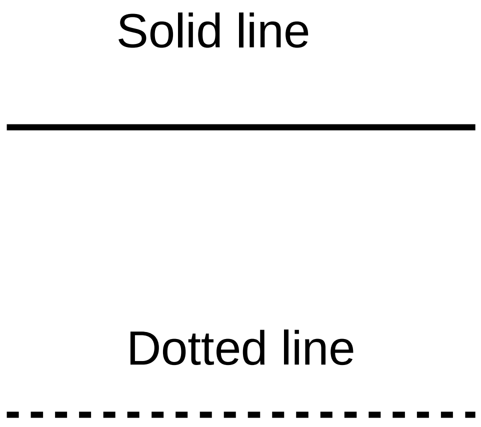
Seller
Seller ID
Name

Sales
Seller ID*
Product ID*

Product
Product ID
Name

Identifying and non-identifying

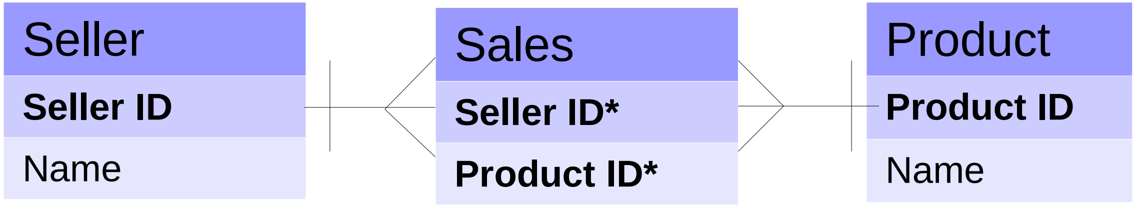
- Identifying relations
  - Foreign key is part of the primary key
  - Table cannot exist without the relation
- Non-identifying
  - Foreign key is not part of the primary key
  - Table can exist without relation
- Weak entity
  - An entity dependent on a identifying relationship is called a weak entity



Solid line

Dotted line

Identifying



Seller

Seller ID

Name

Sales

Seller ID*

Product ID*

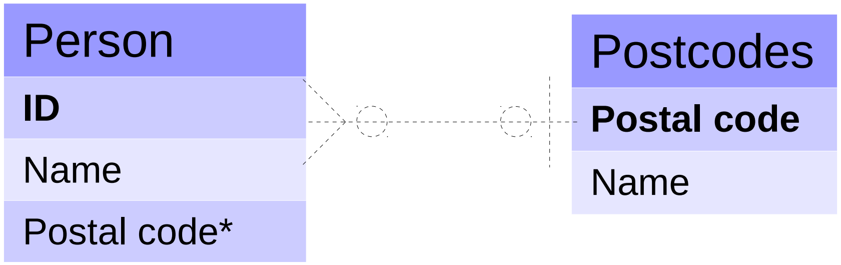
Product

Product ID

Name

Sales cannot exist without either the product or the seller

Non-identifying



A person can exists without a postal code

This is only the case if the postal code can be NULL

Relations

- In general
  - Foreign keys decide if there is a relation
  - Primary key/Unique and NULLable decides which relationship (cardinality) there is
  - Read the relation both ways
  - Split many to many

Cardinality confusion

- Cardinality is a term also used for column "Uniqueness"

Person
ID
Name
gender

ID is completely unique – High cardinality

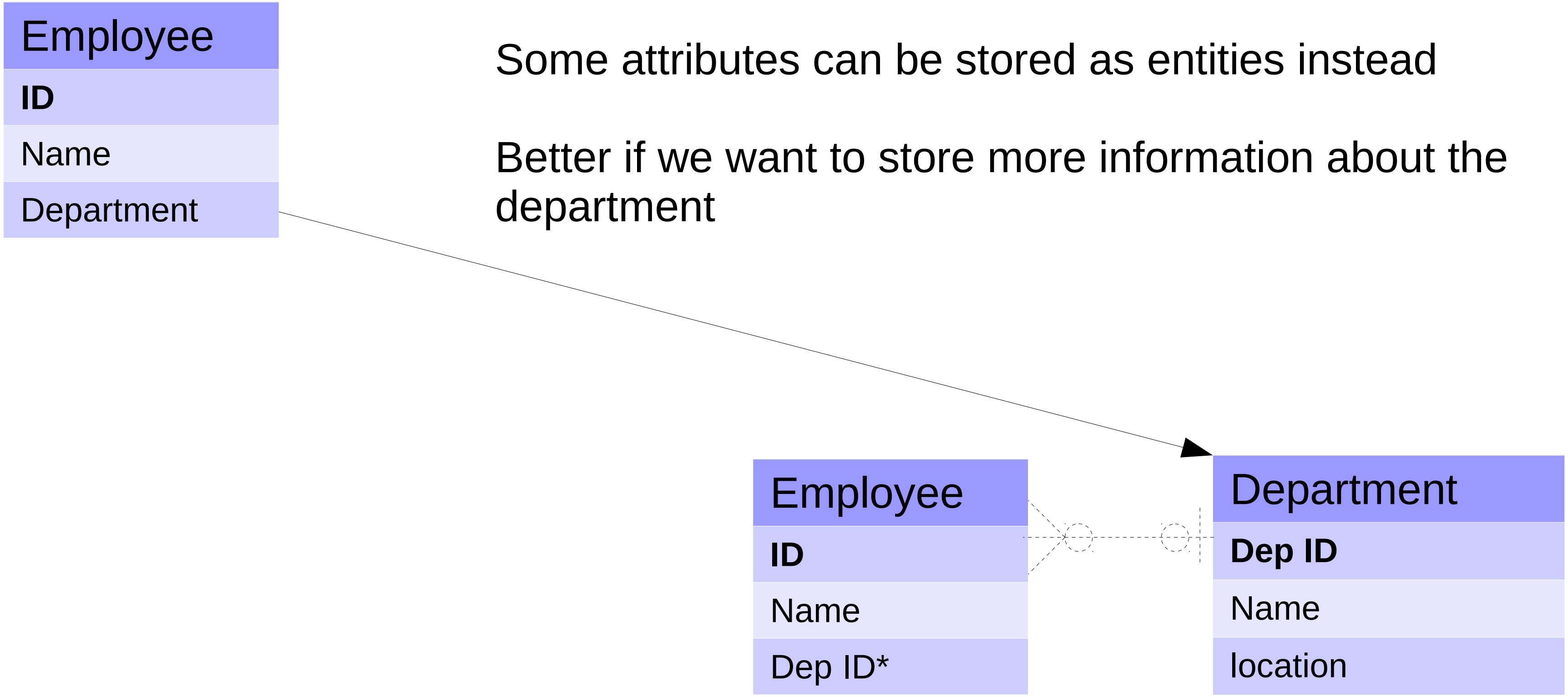
Name is somewhat unique – Medium cardinality

Gender is not very unique – Low cardinality

Entity or attribute?

Some attributes can be stored as entities instead

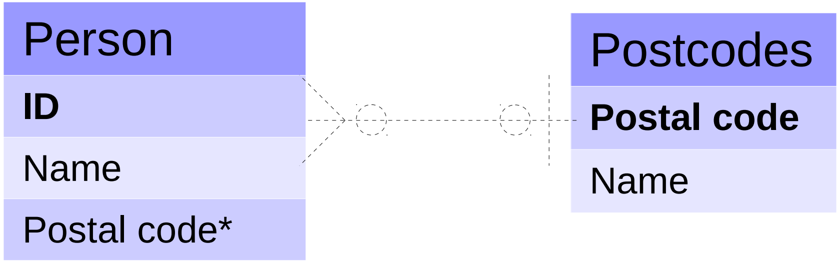
Better if we want to store more information about the department



Extra clearity

Can add descriptive text for the relations

Person has poscode



Person

ID

Name

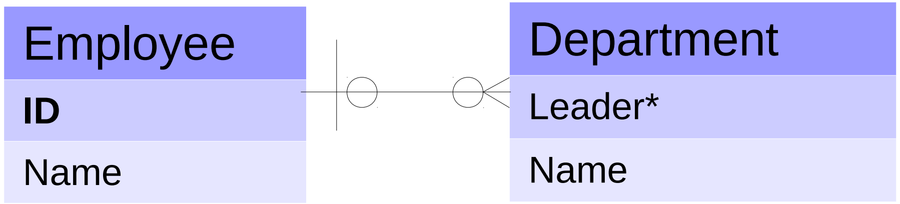
Postal code*

Postcodes

Postal code

Name

Employee is leader of department



Employee

ID

Name

Department

Leader*

Name

Common modelling problems

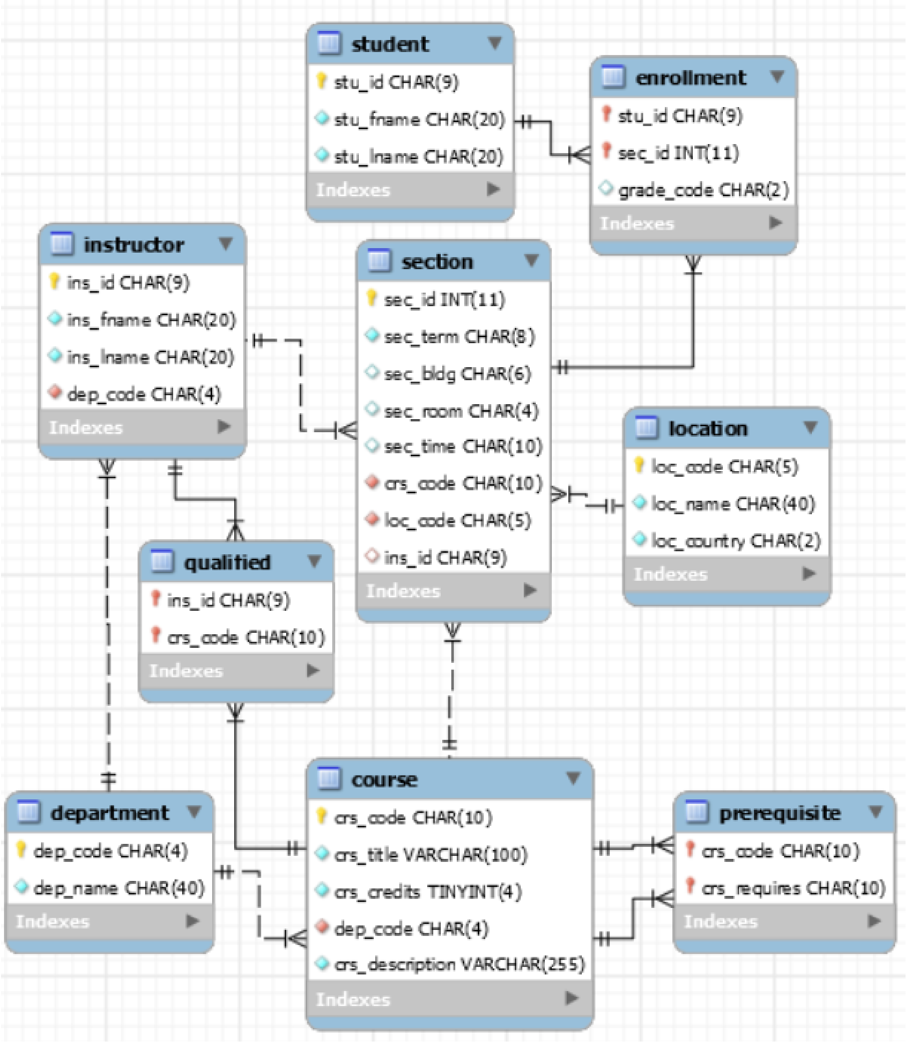
- Saving historical data
  - Leads to a many to many (M:N) relation

From model to database

- Modeling tools can generate physical design from logical/conceptual design
  - Identifiers become primary keys
  - Relationships become foreign keys
  - Many to many relationships are split

Modeling using Workbench

- Lets recreate the University database in workbench
  - Note that workbench is not a true E/R modeling tool as it jumps directly to the physical representation

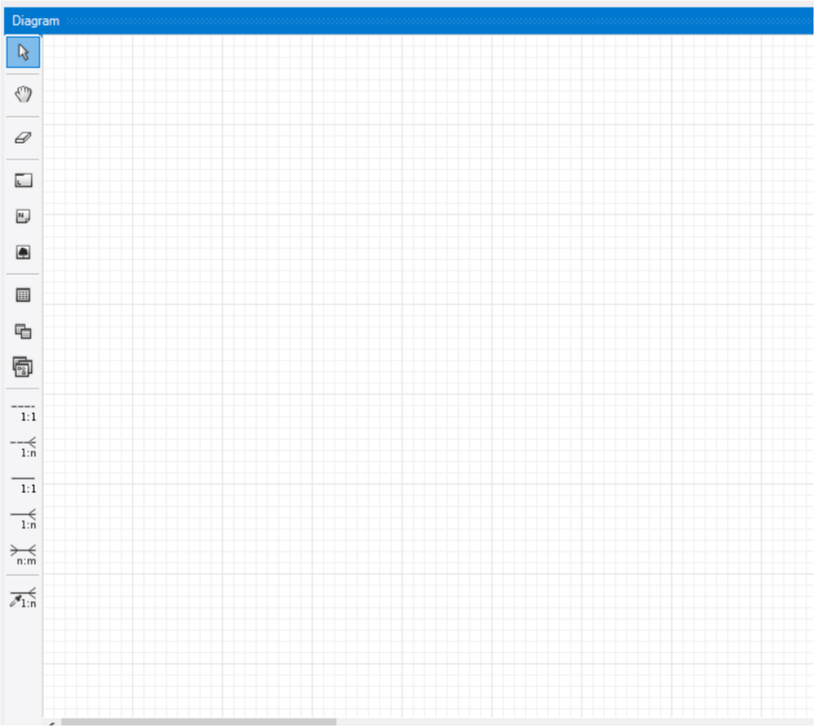

student
stu_id CHAR(9)
stu_fname CHAR(20)
stu_lname CHAR(20)
Indexes
enrollment
stu_id CHAR(9)
sec_id INT(11)
grade_code CHAR(2)
Indexes
instructor
ins_id CHAR(9)
ins_fname CHAR(20)
ins_lname CHAR(20)
dep_code CHAR(4)
Indexes
section
sec_id INT(11)
sec_term CHAR(8)
sec_bldg CHAR(6)
sec_room CHAR(4)
sec_time CHAR(10)
crs_code CHAR(10)
loc_code CHAR(5)
ins_id CHAR(9)
Indexes
location
loc_code CHAR(5)
loc_name CHAR(40)
loc_country CHAR(2)
Indexes
qualified
ins_id CHAR(9)
crs_code CHAR(10)
Indexes
course
crs_code CHAR(10)
crs_title VARCHAR(100)
crs_credits TINYINT(4)
dep_code CHAR(4)
crs_description VARCHAR(255)
Indexes
department
dep_code CHAR(4)
dep_name CHAR(40)
Indexes
prerequisite
crs_code CHAR(10)
crs_requires CHAR(10)
Indexes

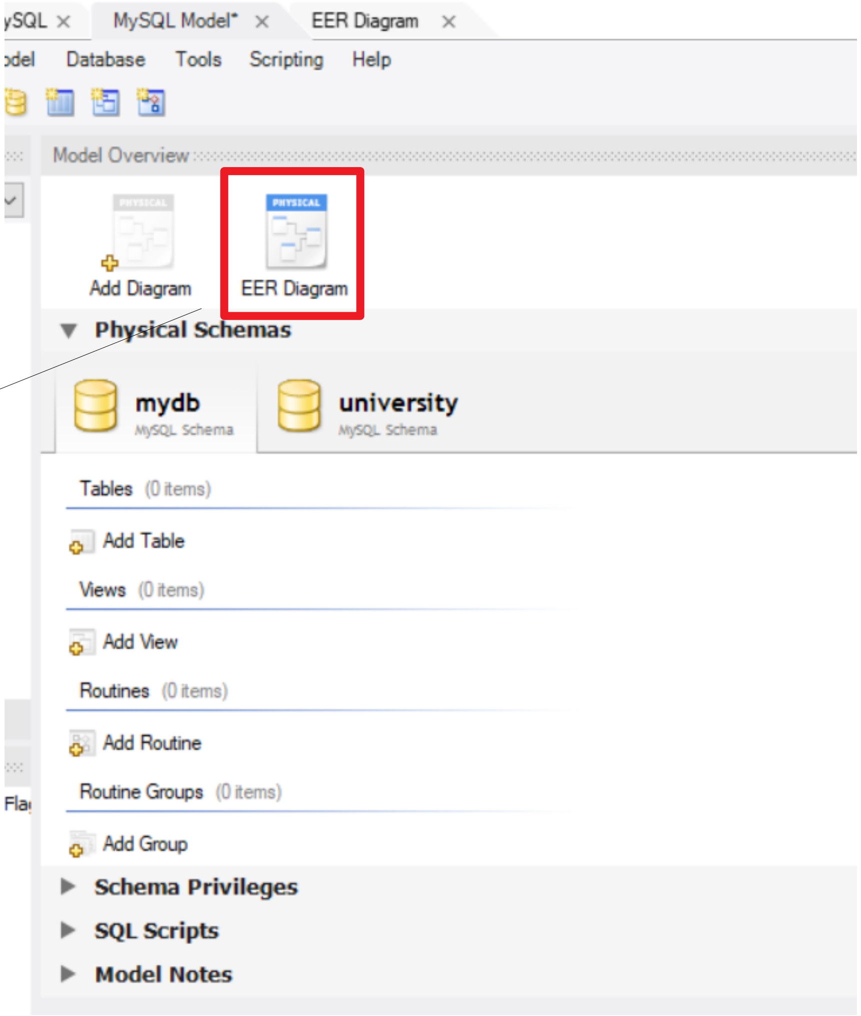
University

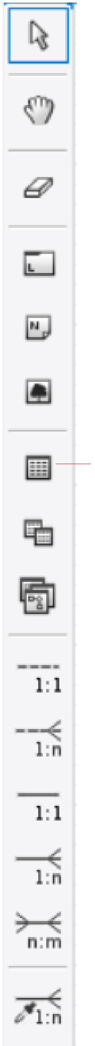
- Base tables (entity types)
  - Student
  - Instructor
  - Department
  - Course
  - Location
  - section

- Derived tables (entity types)
  - qualified
  - enrollment
  - prerequisite

- Workbench view



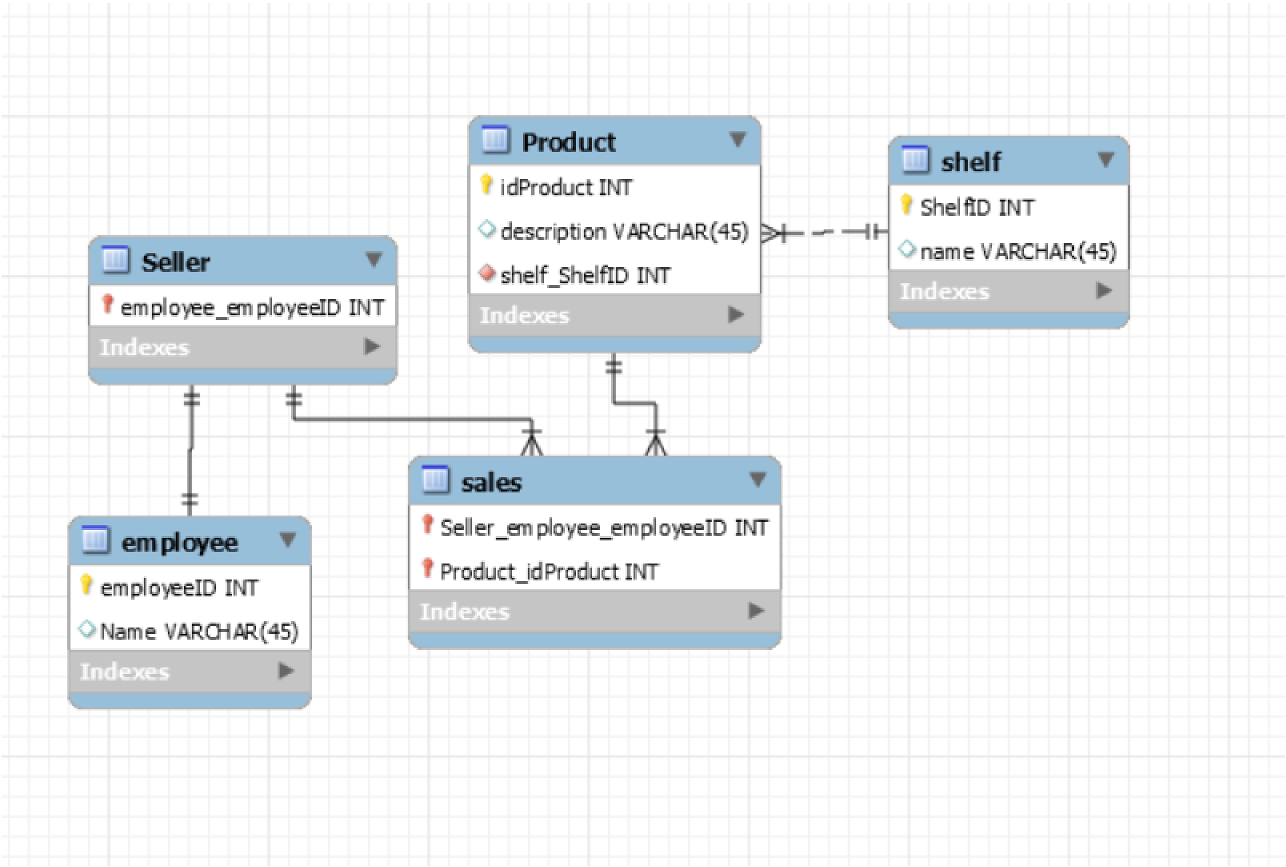




New table

Relations

Try yourself



Normalization

- We want to avoid redundancy
  - Save space
  - Reduce inconsitency
- Normalization is a set of states and rules to achive those states

Normalization

- There are four normal forms
  - 0. normal form (kinda)
  - 1. normal form
  - 2. normal form
  - 3. normal form

The book mentions 2 more, but they are not necessary

0. normal form

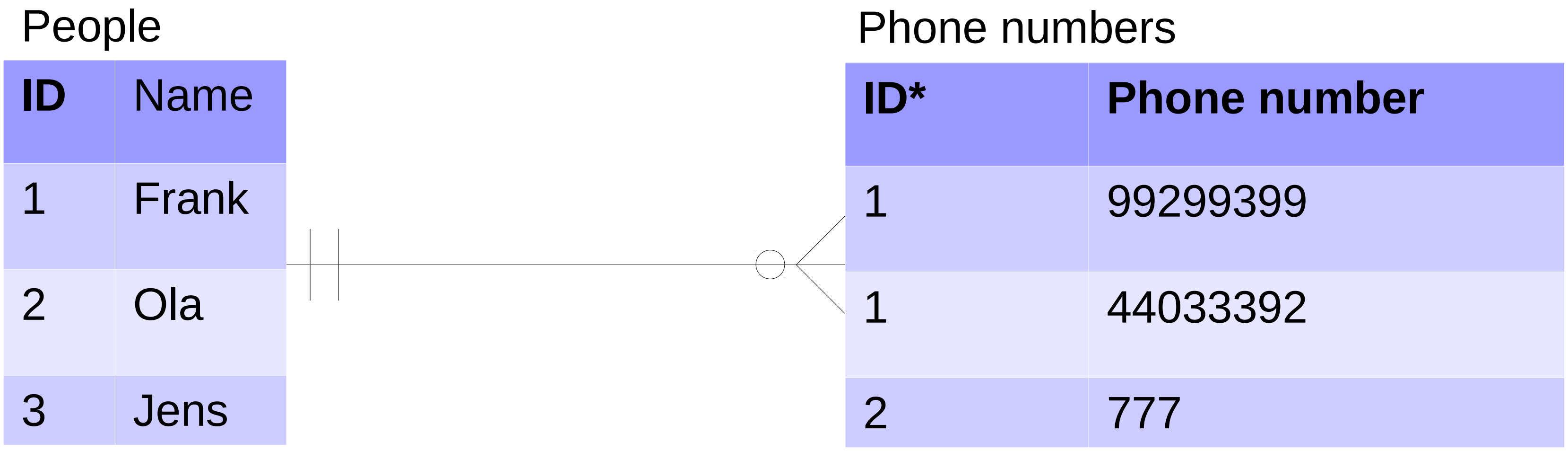
- Contains non atomic values
  - Columns that contain more than one value or multiple columns that store the same type of values

ID	Name	Phone number
1	Frank	99299399;44033392
2	Ola	4432;991;1142
3	Jens	777;886

ID	Name	Phone number 1	Phone 2
1	Frank	99299399	4430202
2	Ola	4432	1131342
3	Jens	777	1102944

1. normal form

- Does not contain non atomic values
  - Achieved by decomposing the table



People

ID	Name
1	Frank
2	Ola
3	Jens

Phone numbers

ID*	Phone number
1	99299399
1	44033392
2	777

2. normal form

- On the 1. normal form
- Does not have partial dependencies

Partial dependency

Ordernumber	**Item Number**	Date	Description
1	12	14.02	Hat
2	13	12.02	Cat
3	19	18.02	Fat

Partial dependency:
Part of the primary key determines a column in the table

Item number determines description

3. normal form

- No transitive dependencies

ID	name	Zip code	Zip adress
1	Ola	1442	Oslo
2	Frank	1112	Ås
3	James	7764	Moss