Tomas Sandnes
tomas.sandnes@kristiania.no

Week 7

DB1102 / PGR 111 – DATABASES

Today's topics

(Today's chapters: 7 + 8.1 in Norwegian book, 6 in English)

Only one (large) topic for this lesson:

– (and it continues through the next lesson)

- ER modelling

ER modelling

ER modelling, introduction

- So far, we have primarily looked at what it is like to write SQL statements for existing schedules / databases.

- Tables haven been defined for us. We have focused on:
  - Select statements (including advanced stuff like joins and subqueries).
  - Insert into / update / delete from statements.
  - Create table / alter table / drop table statements.

- Sometimes we want to be the architect behind the database.
  - That's when ER modelling becomes relevant.

ER modelling

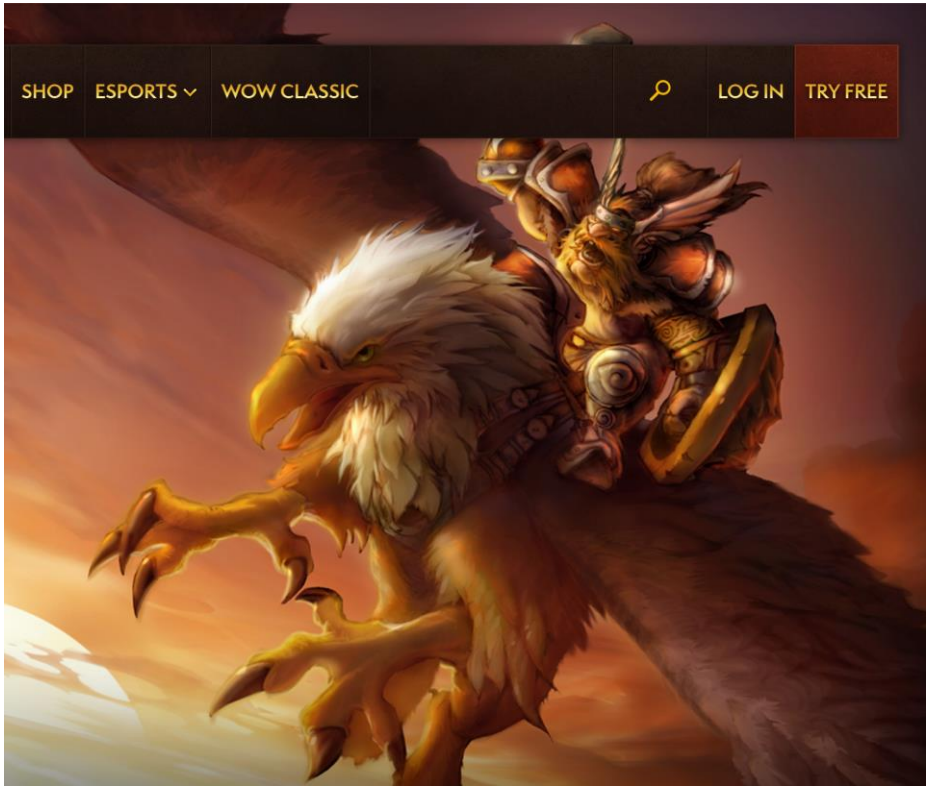
- ER modelling:
  - ER = Entity Relationship

- Note, regarding Relation vs. Relationship, as mentioned in earlier lessons:
  - Relation = another word for Entity, which is the same as Table.
  - Relationship = how two Entities are connected.

- Then there are some slight variations of this, like EAR and EER.
  - EAR: Entity Attribute Relationship. (Attribute here means "column".)
  - EER: Extended Entity Relationship. MySQL Worbench uses this term.

Case study: World of Warcraft

- When Blizzard was working on *WoW Classic* some years ago, knowledge of SQL and databases, and especially ER modeling, turned out to be extremely important!
  - Read about how WoW uses an SQL database here: Dev Watercooler: World of Warcraft Classic
  - We will talk about this in class as well. :-)



Case study: World of Warcraft – cont.

- In World of Warcraft (WoW), several types of player characters (PCs) can cast magic spells ("Spell").
  - Spells consists of 1-3 immediate effects ("Spell Effect").
  - Immediate effects can be 0-1 damage effects and 0-2 startups of over-time effects ("Spell Aura").
  - Spell auras can be damage over time (DoT) and/or something else (for example, a "slow" effect).

- *How can we create one or more tables to store this info?*

Case study: World of Warcraft – cont.

ID	Name	Effect One	Effect Two	Effect Three	Aura One	Aura Two	Effect Damage One	Aura Damage One	Aura Damage Two
1	Fireball	Deal Damage	Apply Aura	Nothing	Nothing	Deal Damage Periodically	30	Nothing	3
2	Frost Bolt	Deal Damage	Apply Aura	Nothing	Nothing	Slow	20	Nothing	Nothing

VS.

Table Name: Spell

ID	Name
1	Fireball
2	Frostbolt

+

Table Name: Spell Effect

ID	SpellID	Effect	Damage
1	1	Damage	30
2	2	Damage	25

+

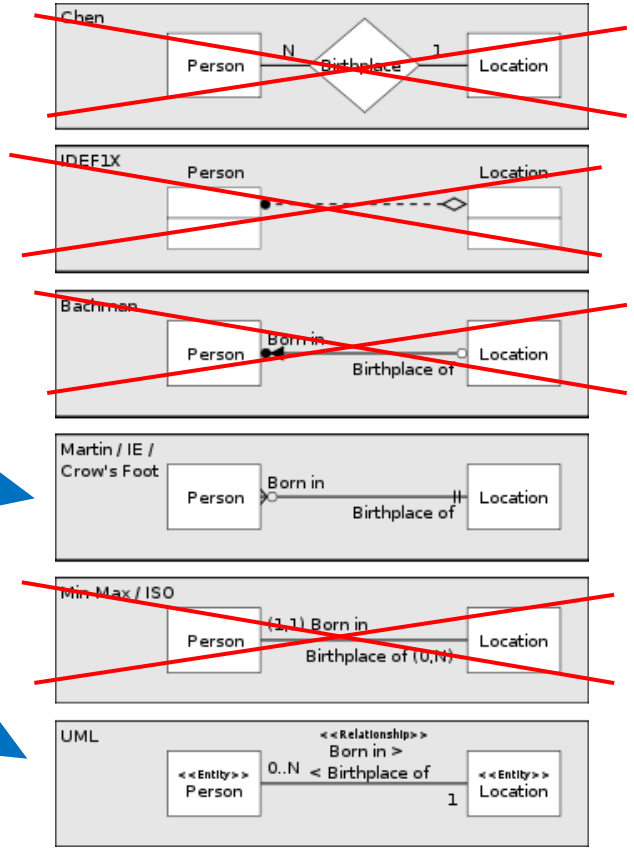
Table Name: Spell Aura

ID	SpellID	Aura	Damage
1	1	Deal Damage Periodically	3
2	2	Slow	Nothing

Regarding notations

- There are a number of ER notations.
- In this course, you can choose whether you want to use Crow's Foot or UML.
  - But use a notation consistently.
- MySQL Workbench uses Crow's Foot.
- On my slides I mostly use UML.
- These are the most common notations.
  - We do *not* look at the rest in this course.

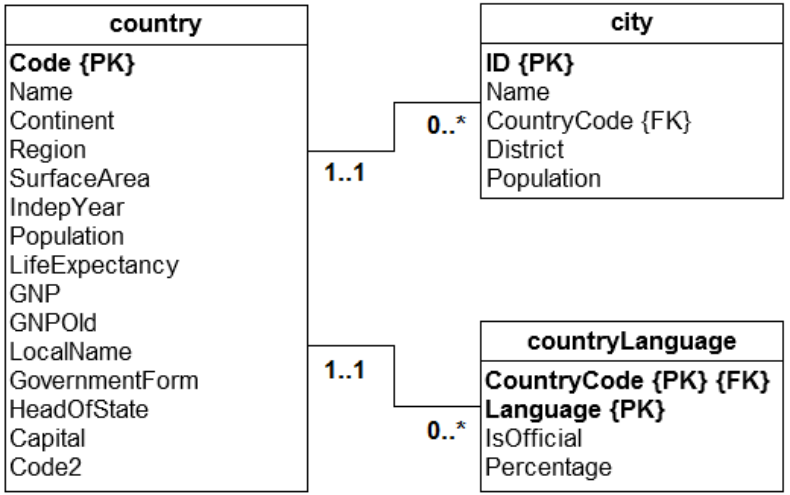
(Source: Wikipedia)



ER diagrams (models)

- Our `World` database shown as an ER diagram: (UML notification)
  - Entity
  - Relationship

- This modell is made with Gliffy: gliffy.com
  - You can also model in MySQL Workbench.
  - A couple other options are lucidchart.com/pages/ and app.diagrams.net.
  - Or you could draw with pen and paper.



Example case: Project management

- A company wants to get an overview of its projects. The company has hired us to create a database that will fix this.

- They specifically want to get an overview of the following:
  - Which department (number, name) owns each project (number, title)?
  - Which projects involve which employees (id, name)?
  - How much time does each employee spend per project? (This one is a bit difficult to place in the model, we will get back to that.)

- Modeling questions:
  - Which entities (tables) do we need?
  - What attributes (columns) should be placed in the entities?
  - What are the relationships between the entities? How do they belong together? One-to-one (1:1), one-to-many (1:M) or many-to-many (M:M)?

Isolate the details

- A company wants to get an overview of its projects. The company has hired us to create a database that will fix this.

- They specifically want to get an overview of the following:
  - Which department (number, name) owns each project (number, title)?
  - Which projects involve which employee (id, name)?
  - How much time does each employee spend per project? (This one is a bit difficult to place in the model, we will get back to that.)

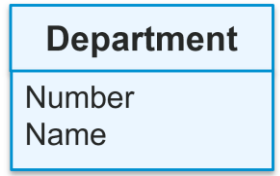
- Modeling questions:
  - Which entities (tables) do we make?
  - What attributes (columns) should be placed in the entities?
  - What are the relationships between the entities? How do they belong together? One-to-one (1:1), one-to-many (1:M) or many-to-many (M:M)?

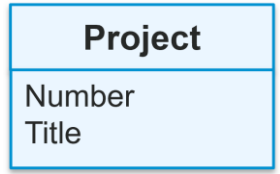
Entities and attributes

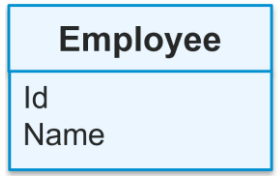
- Based on the specification, we find the following entities:
  - Department
  - Project
  - Employee

- Furthermore, we need the following attributes:
  - Department: number, name
  - Project: number, title
  - Employee: id, name

- And then time spent (per employee per project) has to be placed somewhere. (Getting back to that.)





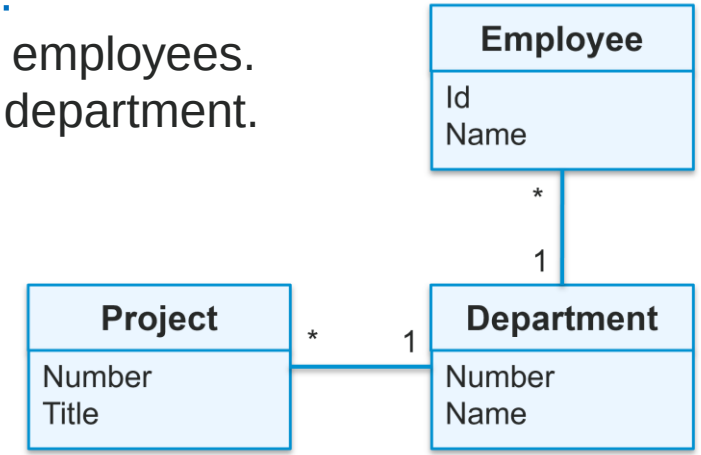


Relationships

- The relationships are not stated in the specification, but it is logical(?) to assume they are as stated below.
  - (We should get these confirmed by the customer early on in the project!)

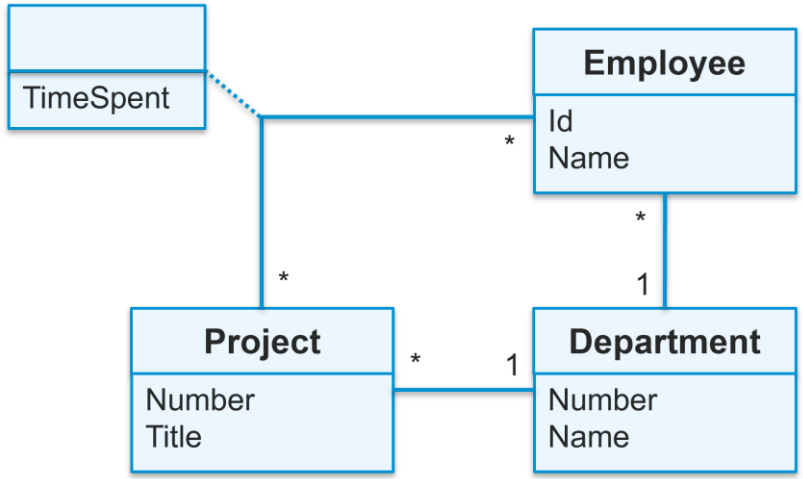
- Relationship department <--> employee:
  - A department can have many (symbol: *) employees.
  - An employee belongs to one (symbol: 1) department.

- Relationship project <--> department:
  - A project belongs to one (symbol: 1) department.
  - A department can have many (symbol: *) projects.



Relationships – cont.

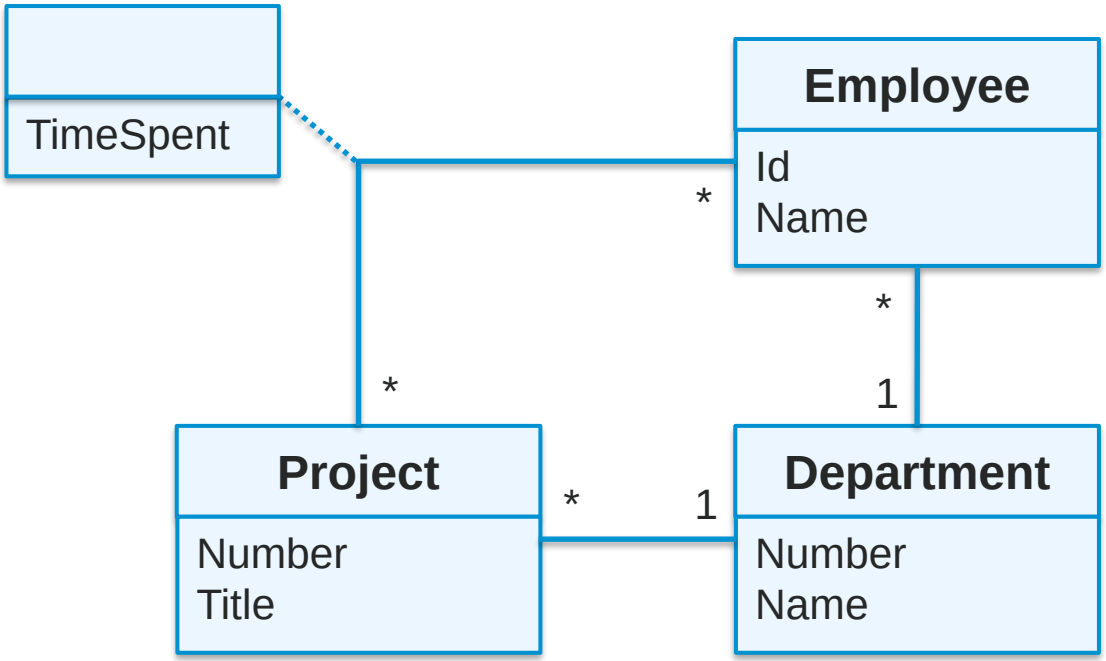
- Relationship project <--> employee:
  - A project can be staffed by many (*) employees.
  - An employee can work in parallel on many (*) projects.



- The model is starting to come together! :-)

- But we have one attribute left:
  - We need to know "time spent per employee per project".
  - But put this where?
  - On the project <--> employee *relationship*!

Full ER Modell

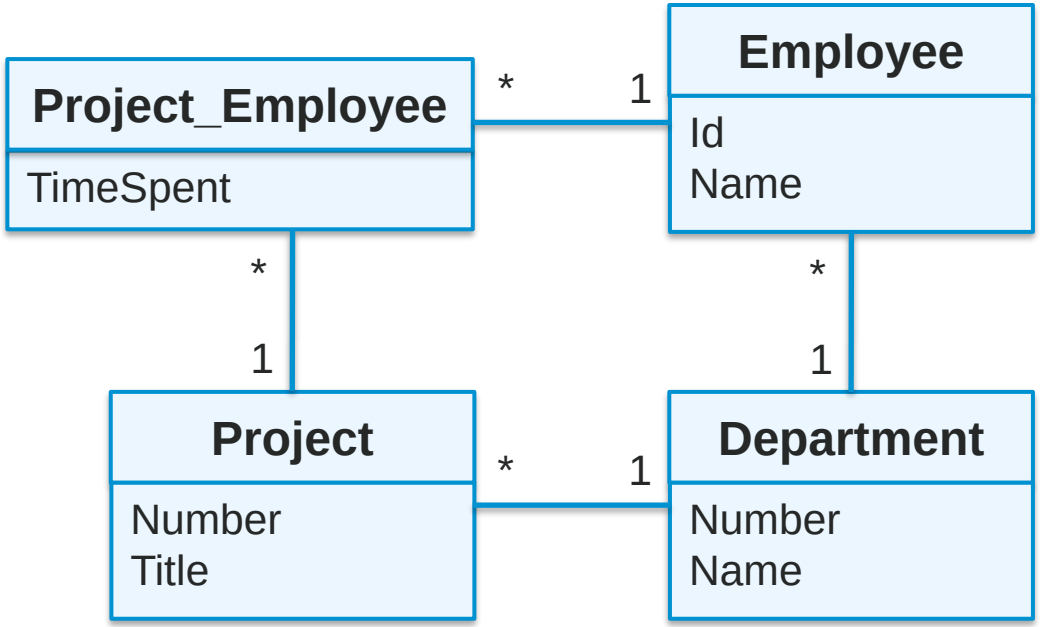


The composite entity

- When we have a many-to-many relationship, we introduce a composite entity.
  - Any relationship-attribute (TimeSpent in our example) is placed on this entity.

- The original M:M relationship is split into two relationships.
  - Both becomes 1:M relationships to the composite entity. (With the 'M' end of each connecting to the composite entity.)

Modell, including the composite entity



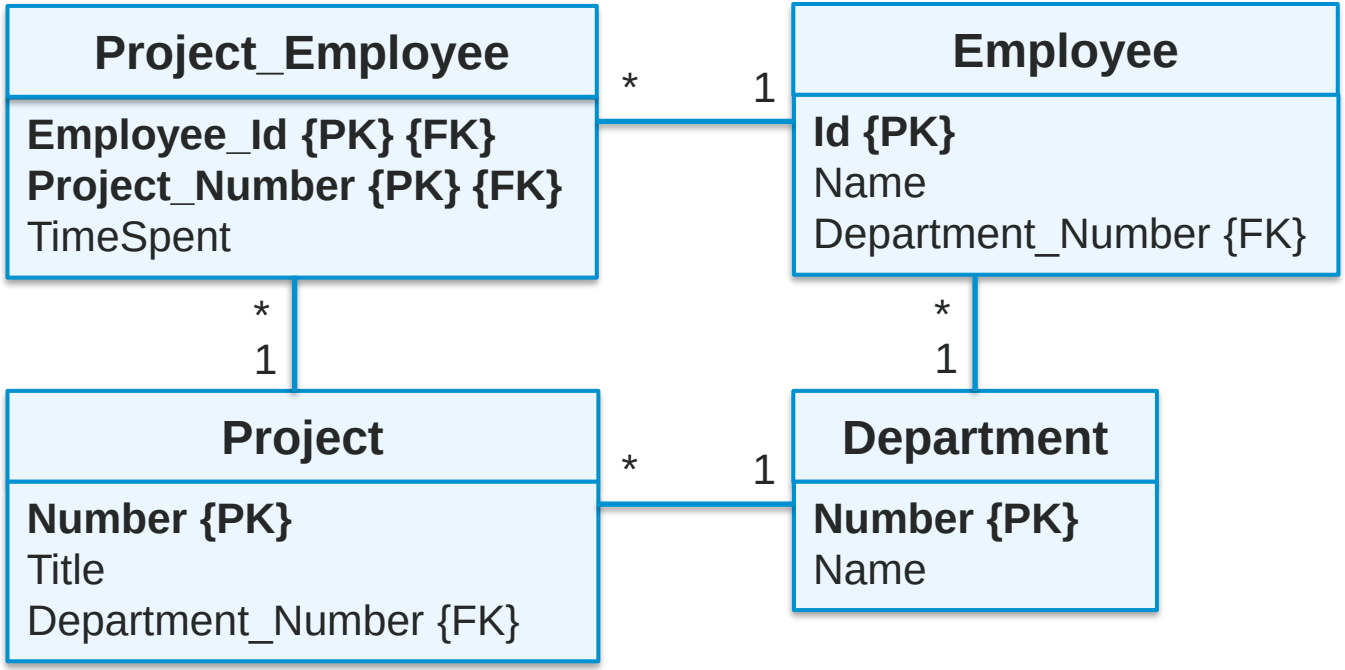
Primary keys and foreign keys

- Before we can implement the model in SQL, we also need to figure out primary keys and foreign keys.

- Primary key:
  - A unique attribute (column) in each entity (table).

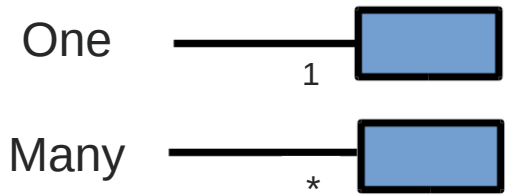
- Foreign key:
  - Need 1 per *relationship*.
  - Mirrors a unique attribute (the PK) on the other side of the relationship.
  - An FK on one side of a relationship forces a '1' on the other side of the same relationship. (For 1:M relationships, an easier way of saying this is: *"The FK is on the M side of any 1:M relationship – always!"*)

ER Modell, ready for DB implementation



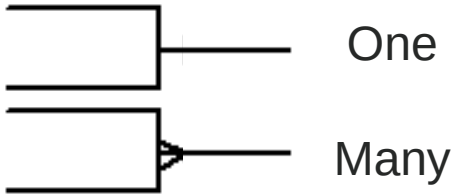
UML and Crow's Foot, relationships

UML



Source: vertabelo.com/blog/uml-notation/

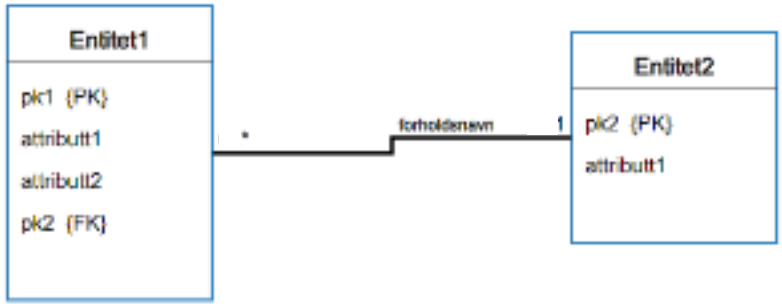
Crow's Foot



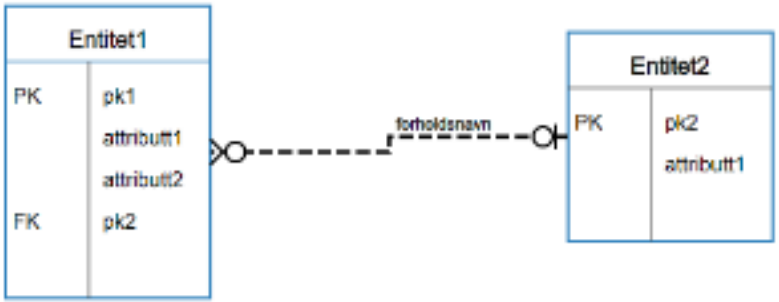
Source: tdan.com/crows-feet-are-best/7474

Comparison of UML and Crow's Foot

- The two notifications are quite similar. 2 main differences:
  - Placement of PK and FK notifications.
  - Symbols/text explaining relationships (1:1, 1:M, and M:M).



UML



Crow's Foot

Modelling tools

- We can do ER modelling with different tools.
  - A great way to start is by using pen and paper. :-)
  - (Or in my case, as in this lesson: On a whiteboard.)

- We can use several types of computer programs:
  - ER model drawing programs: gliffy.com, lucidchart.com/pages/, app.diagrams.net.
  - EER modelling in MySQL Workbench. (In menu: File -> New Model)
  - Or special, pedagogical modelling software, like LearnER. *Note:* Learner is in *Norwegian only*, so will probably not suit all of you? :-\

Optional tool LearnER (Norwegian only)

- If you want to use LearnER:
  - Login (during user registration: check spam for confirmation mail)
  - General landing-page
  - Introduction video (in Norwegian only)

- If you want to use the LearnER "Innstillinger" matching my lessons:
  - "Løs logisk fase"
  - "UML"

LearnER demo

- I'll solve a LearnER task or two together with you, live.

- FYI, regarding LearnER:
  - Main creator is our Norwegian textbook author: Bjørn Kristoffersen.
  - It's still in active development. (So expect minor bugs, or the like.)
  - I am a co-author on some research articles regarding this tool, so I'm *not* totally unbiased about it myself.

Today's exercises & looking ahead

- Now: 2 hours of exercises.

- Exercises are on Canvas, as usual. Short summary:
  - Modell databases on your own, based on the textual description given in the exercises texts.

- Main contents for the next lesson:
  - ER modelling, part 2 (the remaining topics on modelling).

The