# INF115 Lecture 9: *From Models to Databases*

Adriaan Ludl
Department of Informatics
University of Bergen

Spring Semester 2021

# Hand-In Assignment 1

- **You have to hand-in *your own solution* by Monday 1.3.2021.**
- Check the announcements on mitt.uib about what you should include in the submission.

- This week's **group sessions** will help you with the assignment.

- If you have questions outside the group sessions, you may contact your group leaders via mitt.uib.

  In that case **please be patient** and allow circa 1 day for them to answer.

- Ask questions ahead of time.

  ***Time is your friend.***

  *Do not wait until the last minute !*

- You can also discuss on the *Discord* channel: https://discord.gg/34vkUY52PC

# Chapter 8: *From Models to Databases*

**Learning Goals:**

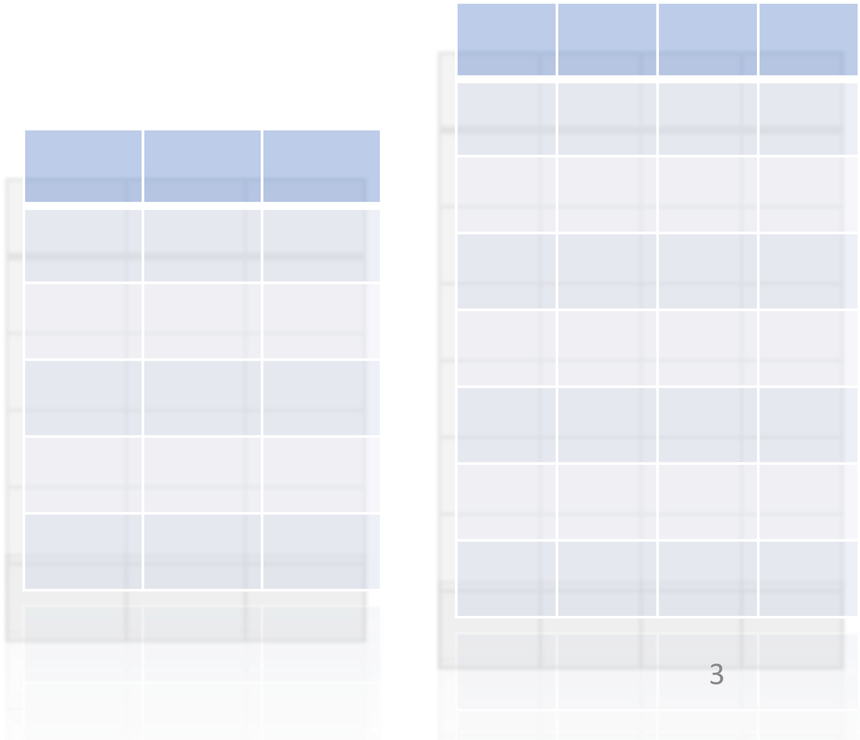
➢ **Translate data models to logical table structures**

- Weak **entities** and identifying **relationships**
- Resolving relationships as **coupling tables**
- Implementing **subtypes**
- Modelling **non-atomic attributes**

**Next lecture:**

➢ Normalise tables to avoid redundance

➢ How to use views in database design



# Logical Data Models

How to go from a **conceptual data model** to a **logical data model** ?

- Which tables do we get ?
- Which columns does each table contain ?
- What are the primary and foreign keys ?

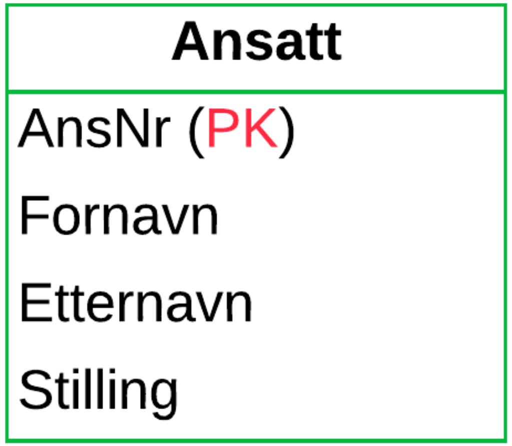
❖ **ER diagrams** can be used **to draw both conceptual and logical data models**.

*MySQL Workbench:*

- Does not make a distinction between *conceptual* and *logical design*.
- You work on the table structure right from the start.

# Entities and Attributes

- Every **entity** becomes a **table** with the same name.
- Every **attribute** becomes a **column** with the same name.



**Ansatt**

AnsNr (PK)

Fornavn

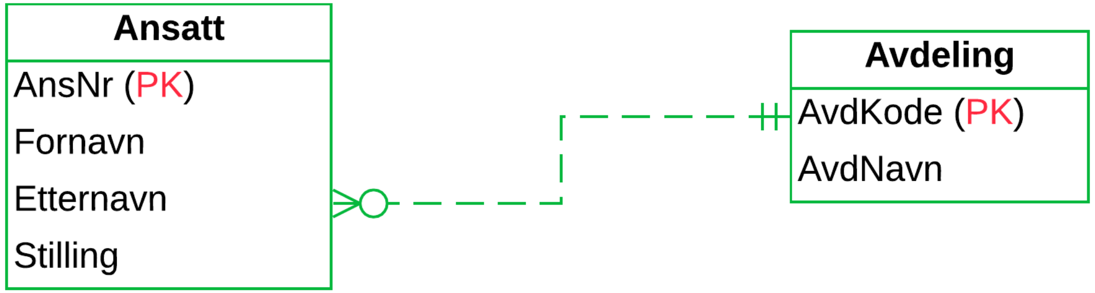
Etternavn

Stilling

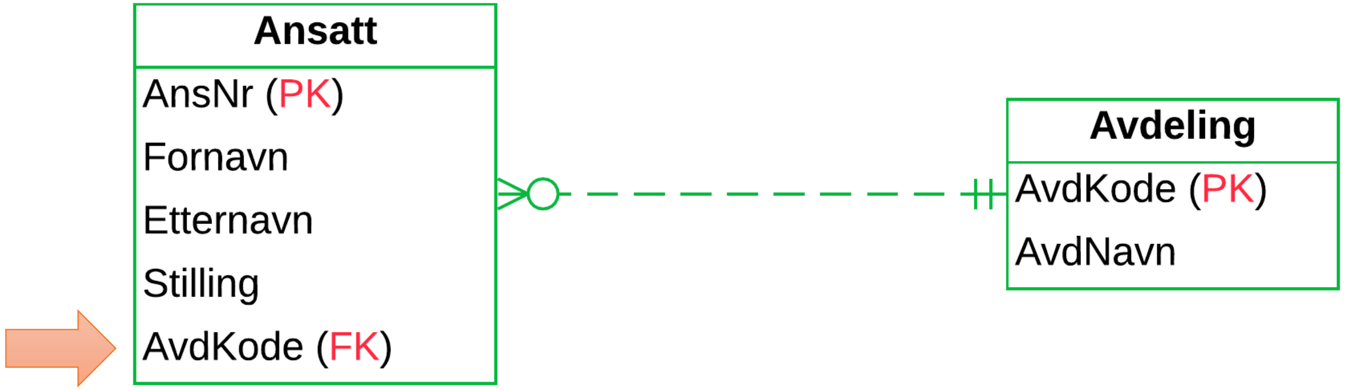
**The logical diagram corresponds directly to the table structure:**

Ansatt(<u>AnsNr</u>, Fornavn, Etternavn, Stilling)

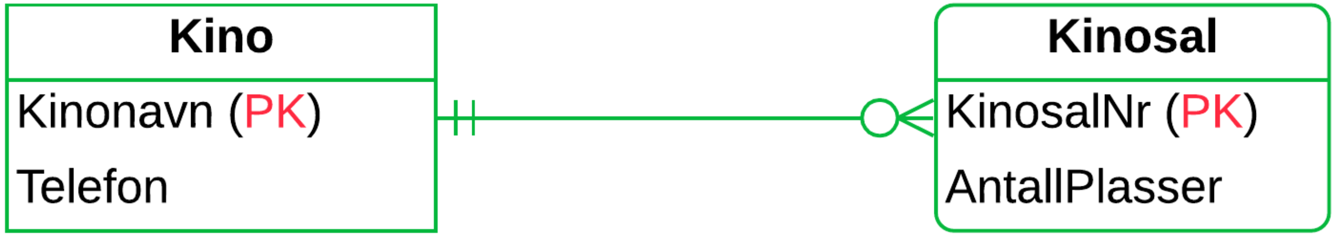
# One-to-Many Relationships



- The ***identifiers*** ***(PK)*** on the «one-side» of the relationship are copied and **added as *columns* on the «many-side»** . . .
- … where they become **foreign keys** (FK) linking back to the «one-side».



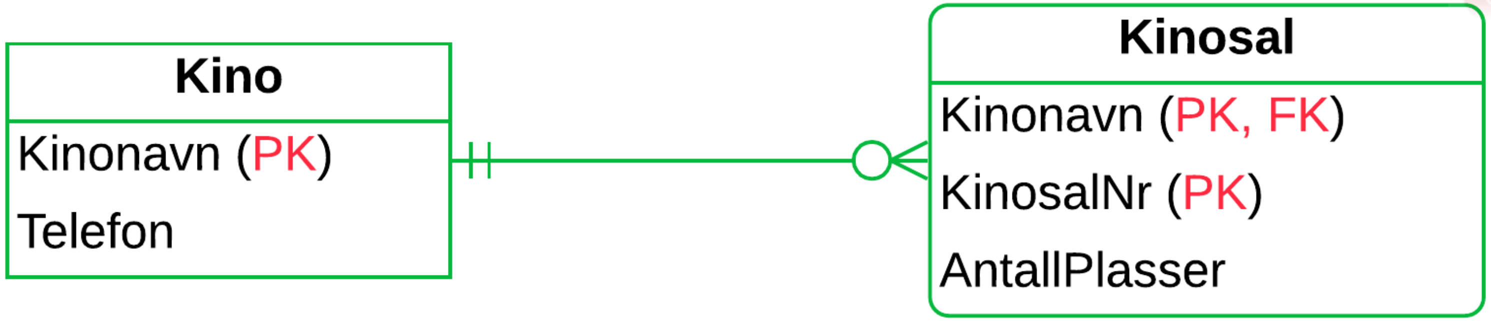
# Weak Entities / Identifying Relationships



**Kino**
Kinonavn (PK)
Telefon

**Kinosal**
KinosalNr (PK)
AntallPlasser

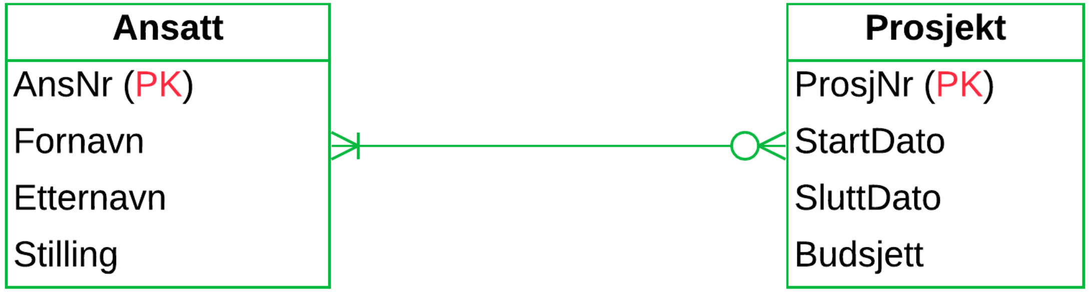
- ➢ ***Weak Entities* inherit** their **primary keys** from the entities they depend on.
- ➢ All one-to-many relationships yield *foreign keys*.
- ➢ **Identifying relationships** yield in addition **primary keys**.



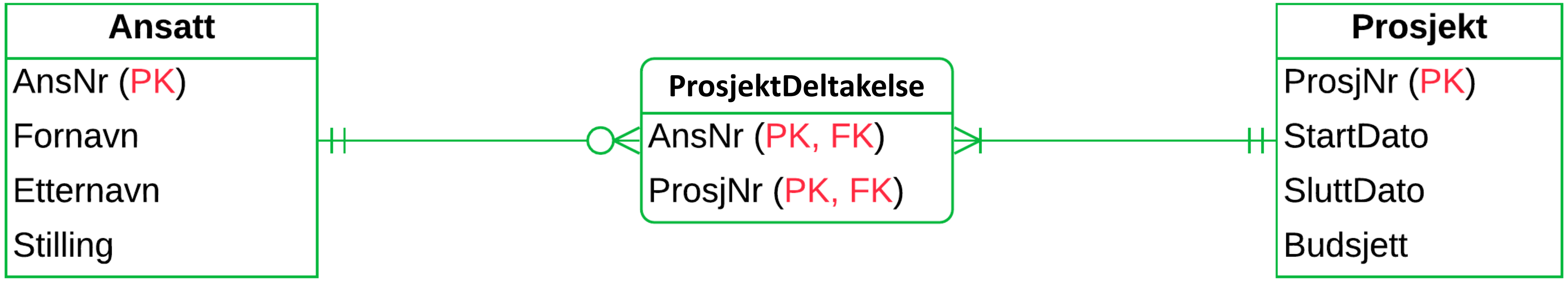
**Kino**
Kinonavn (PK)
Telefon

**Kinosal**
Kinonavn (PK, FK)
KinosalNr (PK)
AntallPlasser

# Many-to-Many Relationships



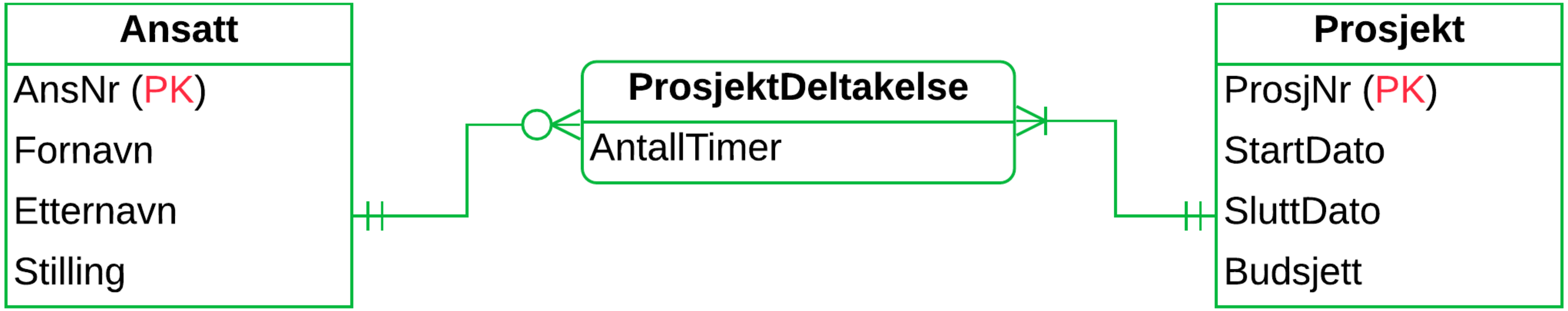
- Many-to-many relationships become «**coupling tables**» (koblingstabeller)
- **Identifiers** of the involved entities become **composite primary keys**.
- These relationships can be resolved in the model – or when generating the database.



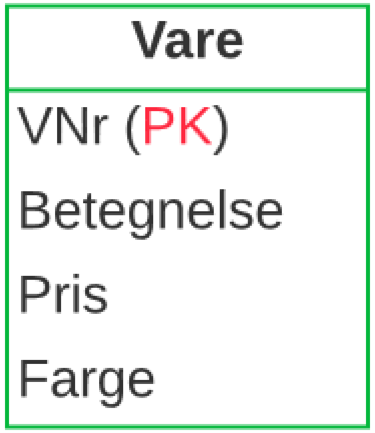
# Resolving Many-to-Many Relationships

- Resolve many-to-many relationships if you wish to add **attributes** to the relationship.
- Coupling entities will then have **two many-to-one relationships**.

- Usually, we get a **weak coupling entity** as shown below.
  - It is also possible to introduce a serial number (løpenummer) as primary key in a coupling entity. A non-identifying many-to-many relationship will lead to this solution.



# « Code tables »



Vare

VNr (PK)

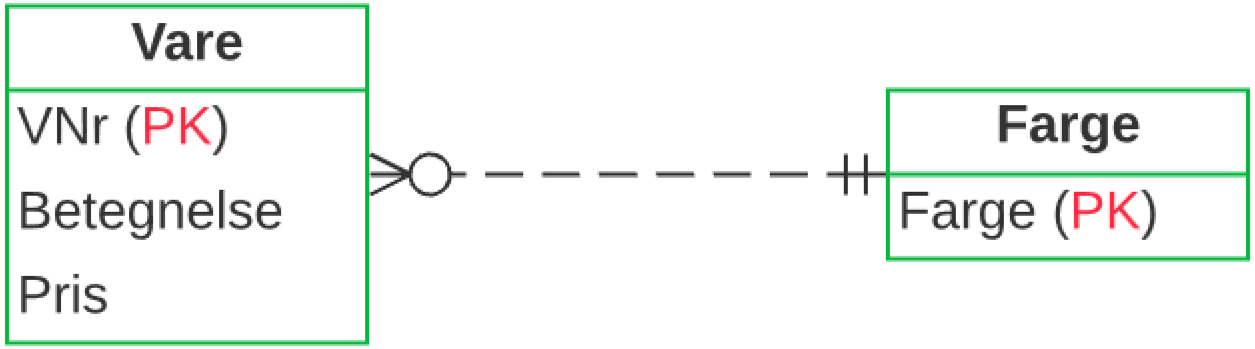
Betegnelse

Pris

Farge

- Introduce additional «**code tables**»

  to <u>control which values are **permitted**</u> (using foreign keys).



Vare

VNr (PK)

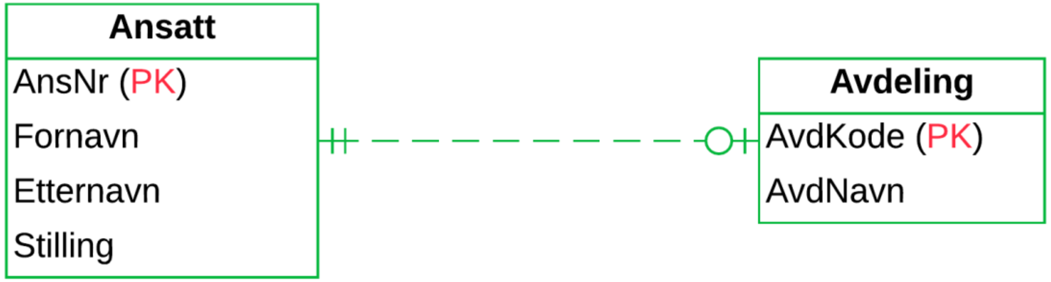
Betegnelse

Pris

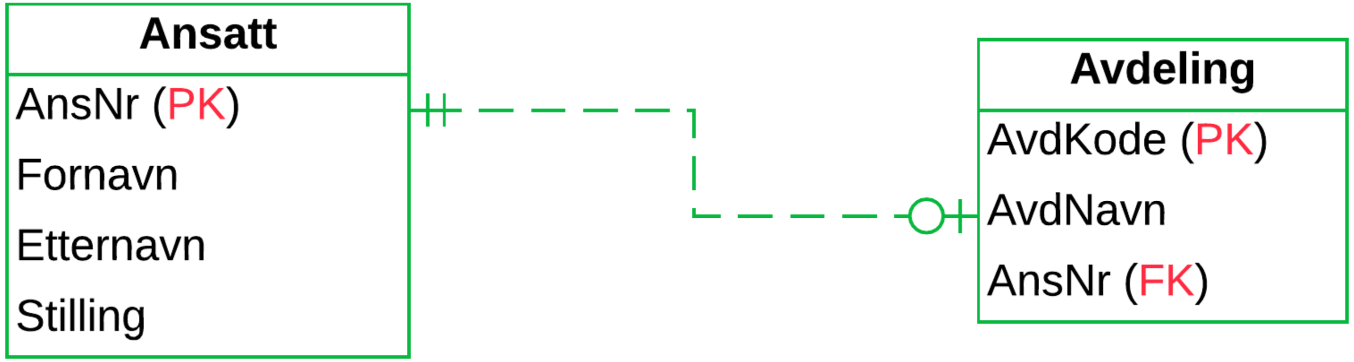
Farge

Farge (PK)

# One-to-One Relationships



- The **identifier** for entity *A* becomes a **foreign key** in *B* or vice-versa.
- Which solution yields less null values ?
- Alternatively: Can the entities be combined ?



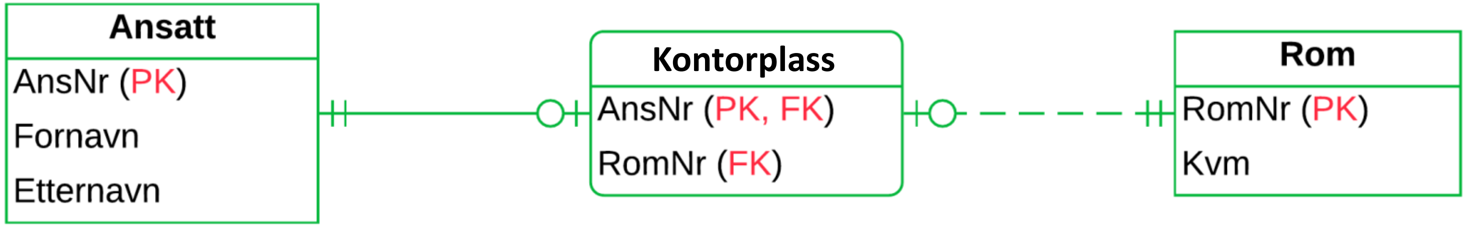
# Coupling tables for one-to-one relationships

We can also set up **coupling tables** for one-to-one relationships

- Useful if only few instances from both sides participate.

➢ Example: one-to-one relationships «office desk» (kontorplass) between *Employee* and *OfficeRoom (Rom)*.

- The building has many rooms, a few are offices.
- The company has many employees, only a few have an office (and none share offices).



❖ The relationship will be treated in the same way as a many-to-many relationship.

# Quizz on *From Models to Databases* (part 1)

Please answer the practice quizz on mitt.uib now ☺

(you can take it again later if you want)

**Link:**

➢ https://mitt.uib.no/courses/27455/quizzes

Break !
Lecture resumes in 15 minutes

# Using subtypes (UML)

**Substypes are a part of the modelling language**,
but traditional *relational databases* <u>do not support them.</u>

Thus, we have to «<u>simulate</u>» subtypes.

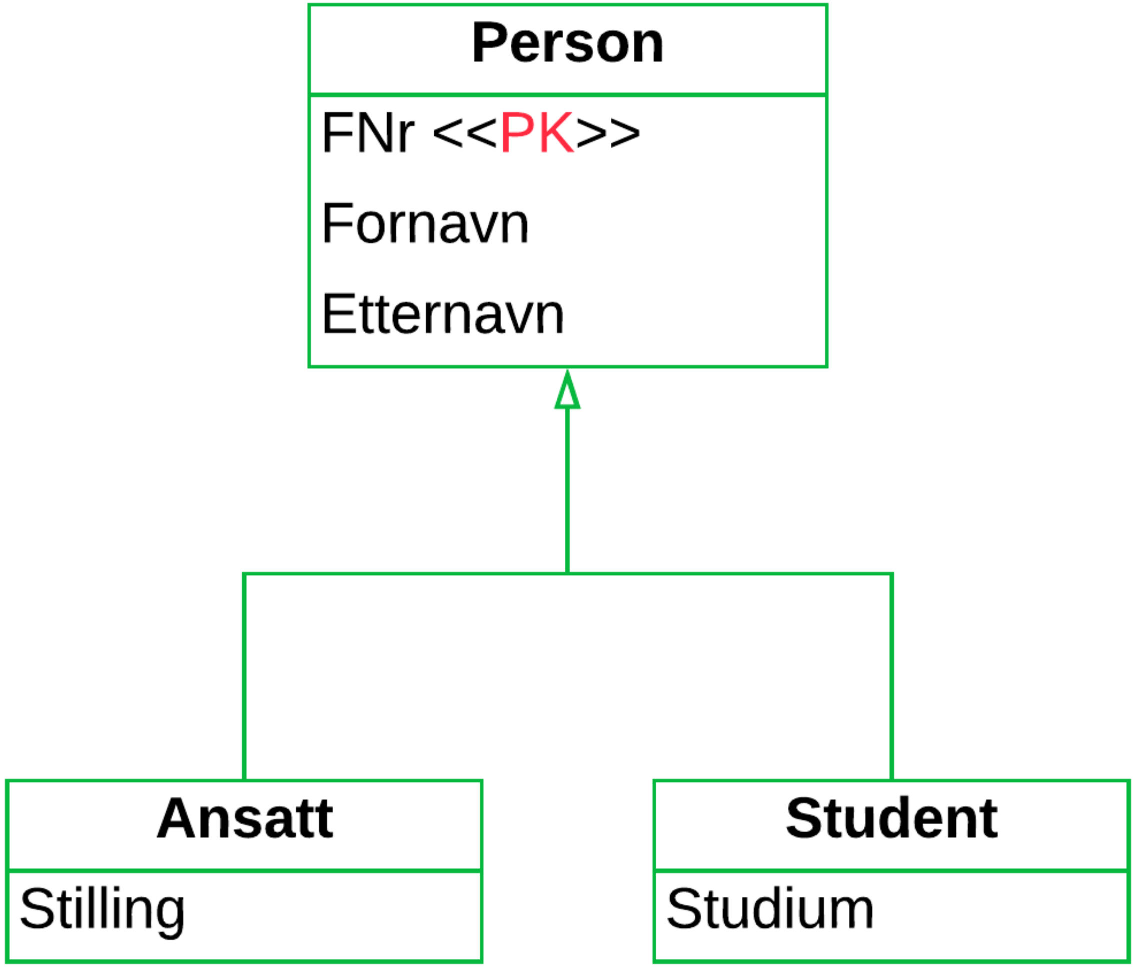
**The following questions will arise:**

- One or several tables ?
- What shall be inherited ?
- Null values ?
- How to represent membership of a subtype ?

***Object relational*** databases implement subtypes.

- In this case the ***modelling language*** and the ***implementation language*** build on the same principles.

# Example



**Person**

FNr <<PK>>

Fornavn

Etternavn

**Ansatt**

Stilling

**Student**

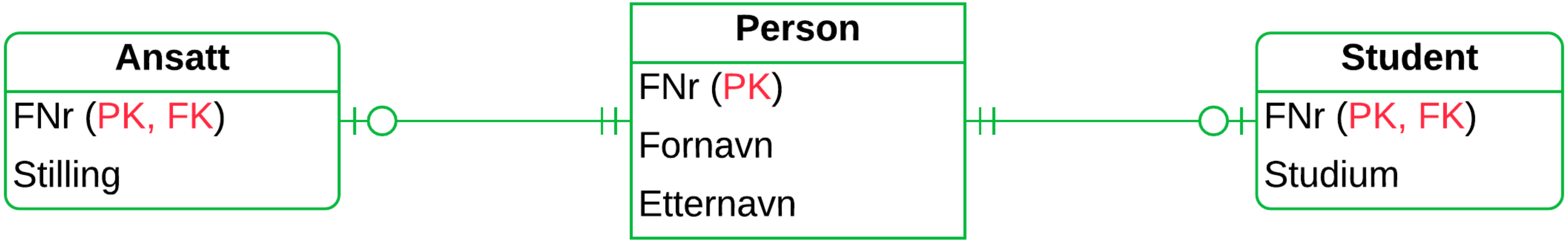
Studium

# Implementing subtypes– variant 1

- Create tables for **both for the supertype and the subtypes**.

- Subtypes inherit the **identifier** of the *supertype*.

- Instances will be part of both a « subtable » and the « supertable ».



# Implementing subtypes – variant 2

- Create a table **only for the supertype**.

- The **supertype** table must have *all attributes from the subtypes* **and** an additional **attribute** indicating the **subtype**.

- Can lead to **many null values**.

| **Person** |
|---|
| FNr (PK) |
| Fornavn |
| Etternavn |
| PersonType |
| Stilling |
| Studium |

# Implementing subtypes – variant 3

- Create tables **only for the subtypes**.

- The tables for the ***subtypes inherit all attributes*** from the supertype.
- To list all instances of the supertype we have to **join tables** !
- It can be difficult to define and use a serial number.
- *Every instances must be part of a subtype !*

| **Ansatt** |
|---|
| FNr (PK) |
| Fornavn |
| Etternavn |
| Stilling |

| **Student** |
|---|
| FNr (PK) |
| Fornavn |
| Etternavn |
| Studium |

# MySQL Workbench does not support subtypes

Instead one has to implement one of the three «table solutions» :

1) **Create entities for both the supertype and the subtype** which are joined in a 1:1 relationship.

2) **Collect all attributes in the supertype**.

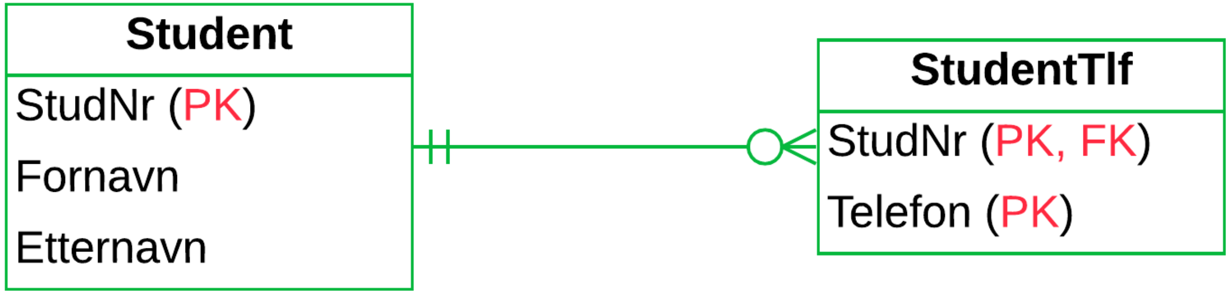
3) **Only create entities for the subtypes** and they must have all attributes of the supertype.

# Non-atomic attributes

- *Composite attributes* are **replaced** by their **components**.
- *Derived attributes* **are ignored**. They will not be stored explicitely, but can be ***generated*** from the other data *when needed*.
- *Multivalued attributes* give an **additional table** that **inherits *primary keys*** from the ***main table*** (such as in Telefon[] below).

**Student**

StudNr (PK)
Navn
Fornavn
Etternavn
/Initialer
Telefon [ ]



**Student**

StudNr (PK)
Fornavn
Etternavn

**StudentTlf**

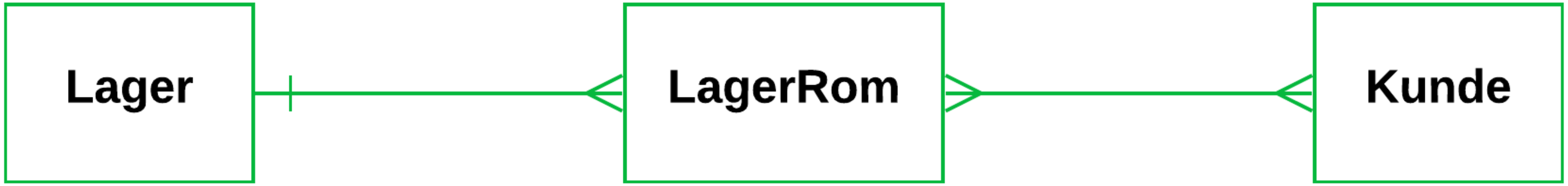
StudNr (PK, FK)
Telefon (PK)

# Exercise: Storage Rental (Lagerutleie)

- A **storage facility** or **warehouse** can have many **storerooms,** while one storeroom is part of a given warehouse.
- One client can **rent** many **storerooms**.
- One storeroom can be rented out many times, but only to one client at a time.

**Your task is to elaborate a logical data model in your group.**

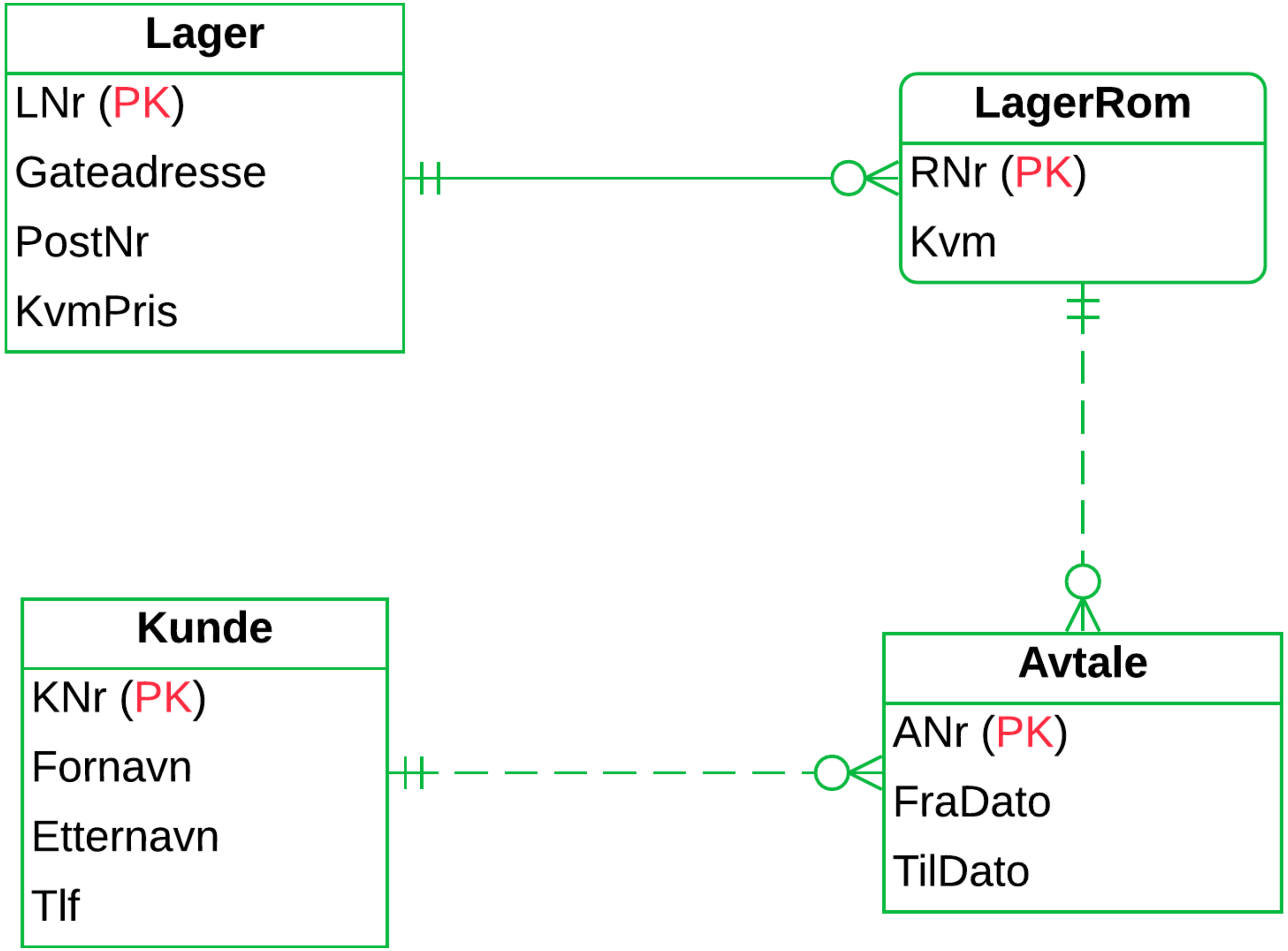
First sketch:



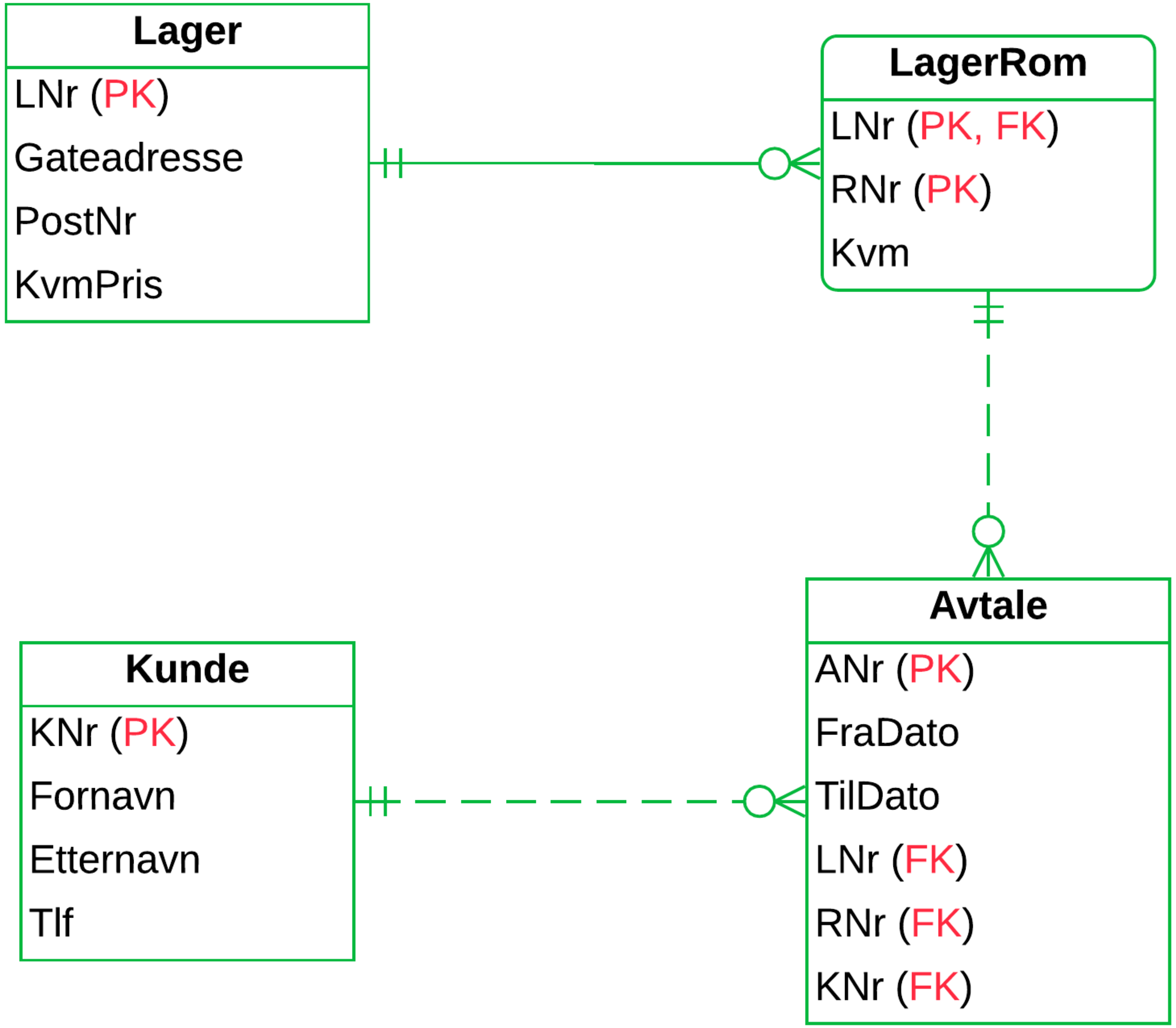
# *Lagerutleie*: Database System **Requirements**

- For every **warehouse** store the **location**: address and postcode. It will be given a unique id-number (lagernummer).
- The **price** of **storerooms** depends on size, but the business operates with a fix price per square meter at each location.
- The rooms in every warehouse are **numbered** starting from 1.
- The **size** (in $m^2$) of every room must be measured and stored, since it matters for the rental price.
- All **clients** who want to rent a storeroom must register and will get a unique *clientnumber*. Their name and telephone number (and email) will also be stored.
- When a client wants to rent a room, a **contract** (avtale) is made for a given time. The contract is always made for an integer number of days.

# Lagerutleie: **Conceptual Data Model**



# Lagerutleie: **Logical Data Model**



**Lager**

LNr (PK)

Gateadresse

PostNr

KvmPris

**LagerRom**

LNr (PK, FK)

RNr (PK)

Kvm

**Kunde**

KNr (PK)

Fornavn

Etternavn

Tlf

**Avtale**

ANr (PK)

FraDato

TilDato

LNr (FK)

RNr (FK)

KNr (FK)

# Quizz on *From Models to Databases* (part 2)

Please answer the practice quizz on mitt.uib now ☺

(you can take it again later if you want)

**Link:**

- [https://mitt.uib.no/courses/27455/quizzes](https://mitt.uib.no/courses/27455/quizzes)

# Summary: *From Models to Databases*

Translate data models to logical table structures:

- **Implementing** data models
- **Weak entities** and **identifying relationships**
- Resolving relationships as **coupling tables**
- Implementing **subtypes**
- Modelling **non-atomic attributes**