INF115 Lecture 10: *Normalisation*

Adriaan Ludl
Department of Informatics
University of Bergen

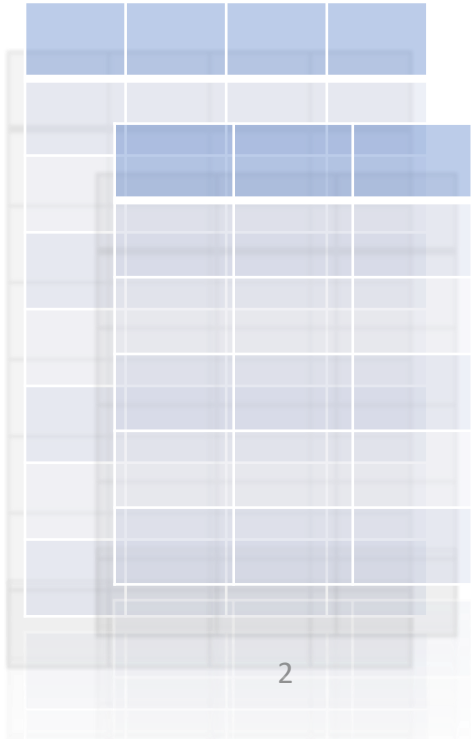
Spring Semester 2021

Chapter 8: *Normalisation*

Learning Goals:

- Be able to **normalize tables** to **avoid redundance** in the database.
  - How to identify **functional dependencies**.
  - How to **decompose** tables.
  - **Normal forms**
  - **Denormalisation**

- Understand how **views** can be used in **database design**.



Normalisation removes redundance

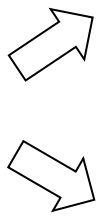
❖ **Redundance** (dobbeltlagring) creates problems:

- Wasting disk space unnecessarily.
- Increases the risk of **inconsistencies.**
- Certain information will not be represented.

➢ Tables should be **split** into two smaller tables

- Without loosing information.

AnsNr	Etternavn	PostNr	Poststed
1	Hansen	**3800**	**Bø**
2	Nordvik	3200	Sandefjord
3	Moen	3200	Sandefjord
4	Nilsen	**3800**	**Bø**



PostNr	Poststed
3200	Sandefjord
3800	Bø

AnsNr	Etternavn	PostNr
1	Hansen	3800
2	Nordvik	3200
3	Moen	3200
4	Nilsen	3800

Functional dependence (1)

Normalisation builds on the concept of **functional dependence.**

There is a **functional dependence** from a colum X to a column Y,

if X determines Y.

Examples:

- *AnsNr* determines *Etternavn*
- *AnsNr* determines *PostNr*
- *PostNr* determines *Poststed*

➢ What does it mean for one column to determine another ?

Functional dependence (2)

A **functional dependence** $X \rightarrow Y$ expresses the following relation:

- If two rows have the same value in X, then they must also have the same value in Y.

	X		Y
	17		**21**
	17		**21**

❖ We say that X **determines** (bestemmer) Y, and call X the **determinant**.

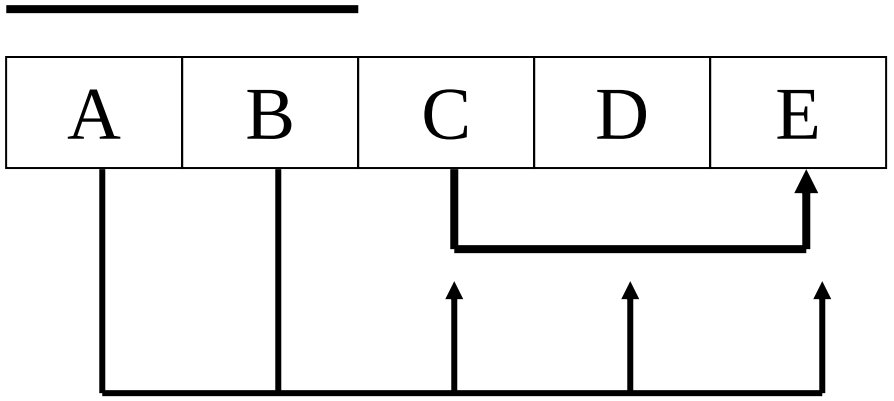
- Both X and Y can in general be ***collections*** *of several columns*.

➢ To decide whether $X \rightarrow Y$, we must understand the **meaning** of X and Y.

➢ The relation must be true for **all possible** table contents.

Functional dependence (3)

Graphical notation:



- If X is a **candidate key**: $X \rightarrow Y$ for all Y

- Some dependencies can remain implied.

Deduction rules for functional dependencies

- **If** $X \supseteq Y$, **then** $X \rightarrow Y$
- **If** $X \rightarrow Y$ **and** $Y \rightarrow Z$, **then** $X \rightarrow Z$
- **If** $X \rightarrow Y$, **then** $X, Z \rightarrow Y, Z$

- Some functional dependencies can **be deduced** from others:

  If AnsattNr $\rightarrow$ PostNr **and** PostNr $\rightarrow$ PostSted,

  then AnsattNr $\rightarrow$ PostSted.

- Normalisation starts by searching for functional dependencies, except the trivial ones:

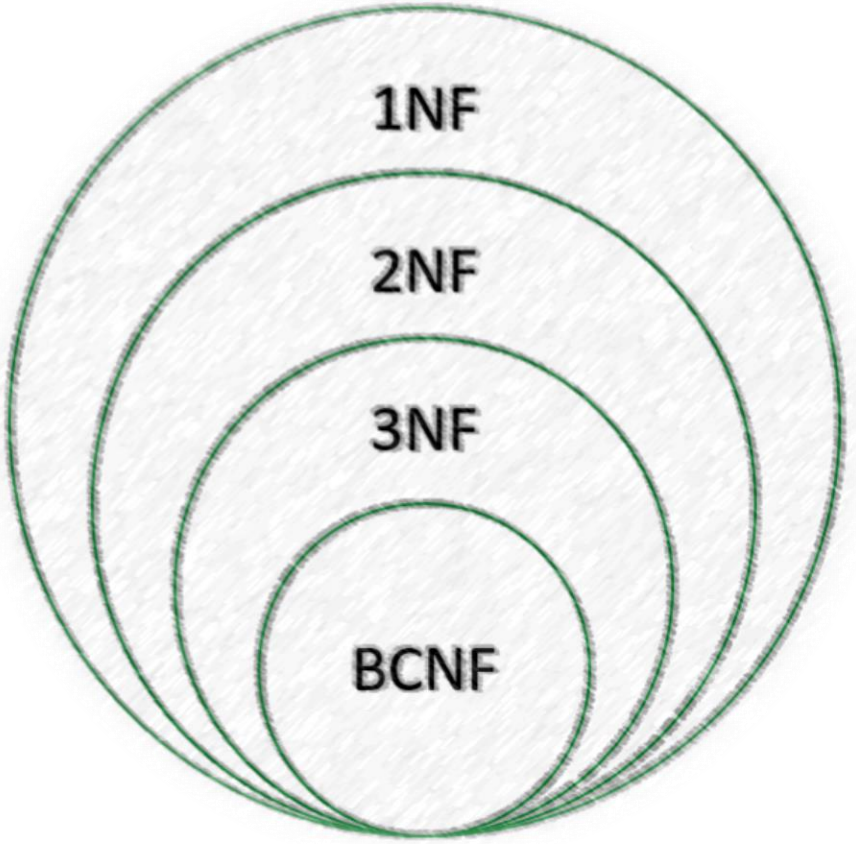
  Etternavn $\rightarrow$ Etternavn

- Some dependencies can be **simplified**:

  AnsattNr, Fornavn $\rightarrow$ Etternavn

The Hierarchy of Normal Forms

Every table in BCNF is in 3NF, etc.



Normalisation

Normalisation means that we **decompose** (split up) tables into several simpler tables such that:

- **Redundance is avoided,**
- **Information content is conserved.**

Normalisation theory gives us:

- **Precise rules** to decide **whether and how** tables shall be decomposed based on understanding of functional dependencies.

The Normalisation Process

1. Remove **non-atomic** columns**.**

Tabeller på 1NF

2. Remove **partial** dependencies.

Tabeller på 2NF

3. Remove **transitive** dependencies.

Tabeller på 3NF

4. Remove **remaining** redundance.

Tabeller på BCNF

Quizz on Normalisation (part 1)

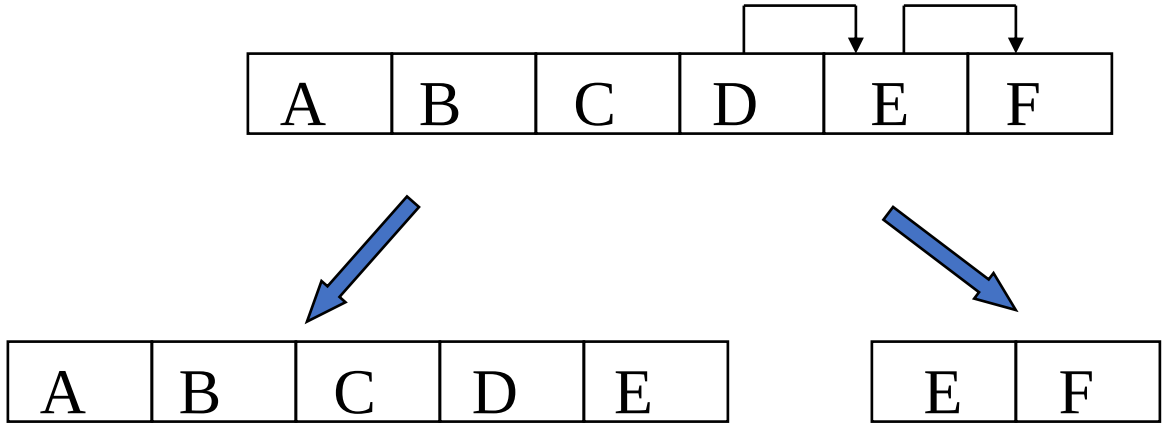
Please answer the practice quizz on mitt.uib now ☺

(you can take it again later if you want)

Link:

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

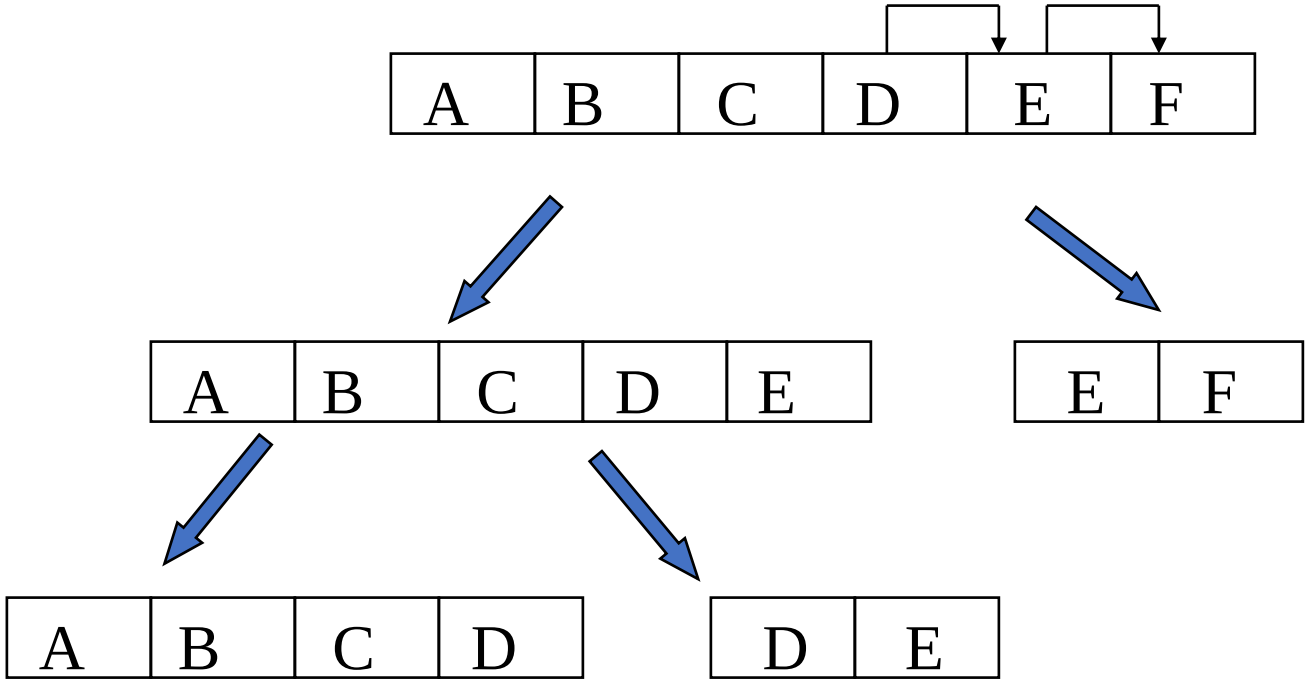
Normalisation Steps



For functional dependence E→F in table T:

- E and F are put into a **new** table.
- **Remove** F from T.

Normalisation Steps



For functional dependence E→F in table T:

- E and F are put into a **new** table.
- **Remove** F from T.

First Normal Form (1NF)

- A table satisfies the condition of **1NF** if all values are **atomic**.
  - Prohibit «tables in tables» and «lists».
  - Example: The column TlfNr in the *Student* table contains a list of telephone numbers for every row. (We saw how to avoid this in in the last lecture.)

StudNr	…	TlfNr	…
1		12345678, 87654321	
2		22334455, 77668822, 99223344	

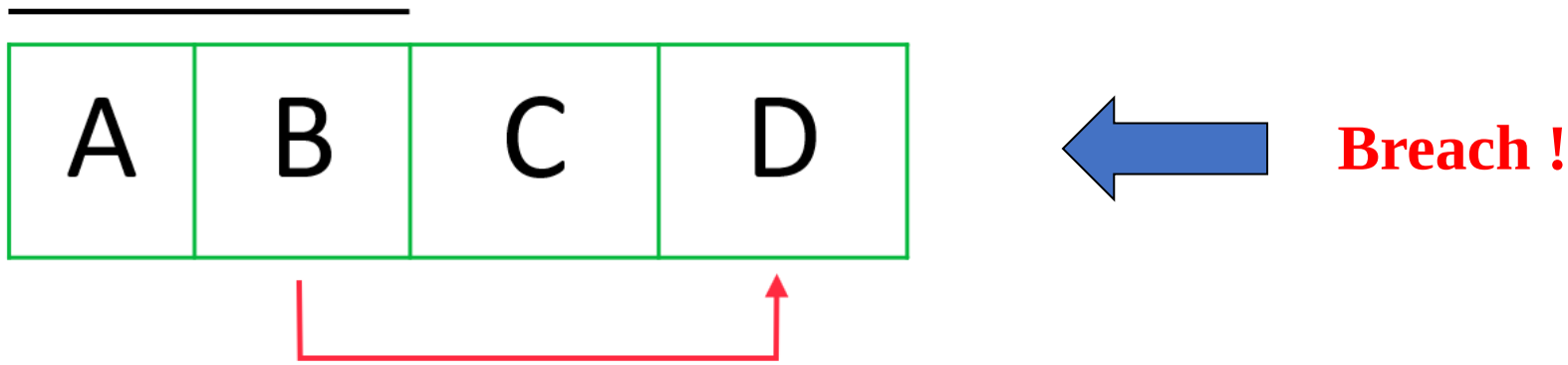
  - **Composite values** must be **decomposed !**
  - Example: Store people's names in two columns: Fornavn and Etternavn.

Second NormalForm (2NF)

- A table satisfies 2NF

  if it is in 1NF

  and in addition it does **not** contain attributes that are **partially dependent** on the primary key.



Example: Breach of 2NF

The number of hours that employees have worked on projects + first name:

ProsjektDeltakelse(AnsNr, ProsjNr, AntTimer, Fornavn)

Functional dependencies:

AnsNr, ProsjNr → AntTimer

AnsNr, ProsjNr → Fornavn

AnsNr → Fornavn

Candidate key: AnsNr + ProsjNr

- AnsNr → Fornavn is a **partial** dependence,

  because Fornavn depends only on a part of the candidate key.

- Split the table into two:

  ProsjektDeltakelse(AnsNr*, ProsjNr, AntTimer)

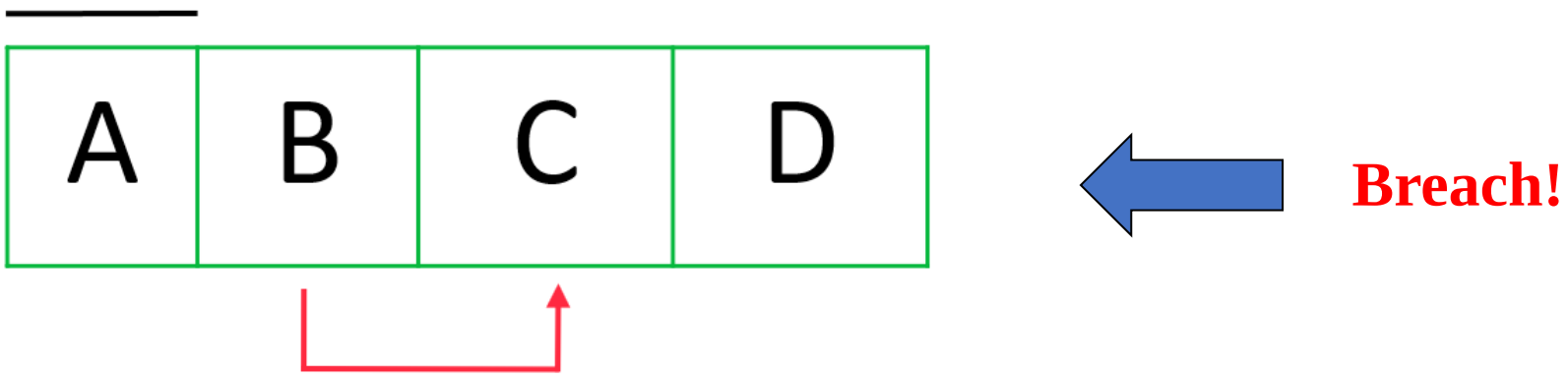
  Ansatt(AnsNr, Fornavn)

3. NormalForm (3NF)

- A table satisfies 3NF

  if it is 2NF

  and in addition it does **not** contain **transitive functional dependencies.**



Example: Breach of 3NF

Products and product categories:

Vare(VNr, Betegnelse, KatNr, KatNavn)

Functional dependencies:

VNr → Betegnelse, KatNr, KatNavn

KatNr → KatNavn

Candidate key : VNr

- KatNr → KatNavn is a **transitive** dependence.
- Split the table into two:

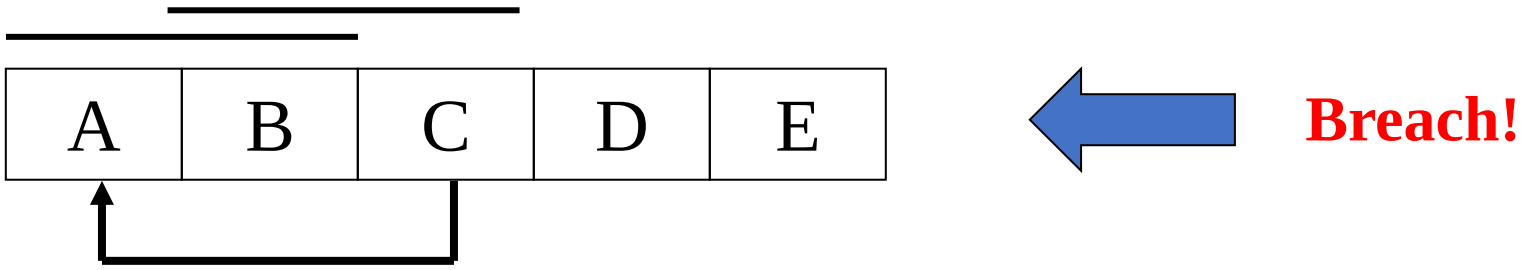
  Vare(<u>VNr</u>, Betegnelse, KatNr*)

  Kategori(<u>KatNr</u>, KatNavn)

Boyce-Codd NormalForm (BCNF)

❖ A table satisfies **BCNF** if any **minimal determinant** is a candidate key.

- Example of 3NF that breaches BCNF, assume two composite and overlapping candidate keys:



Several dependencies on the same column

- People and information on the municipality (kommune) where they live:

  Person(FNr, Enavn, KoNr, KoNavn, KoAreal)

- Functional dependencies show that FNr is a candidate key:

  FNr $\rightarrow$ Enavn, KoNr, KoNavn, KoAreal

  KoNr $\rightarrow$ KoNavn

  KoNr $\rightarrow$ KoAreal

- If we treat the two breaches **each individually**, we get:

  Person(FNr, Enavn, KoNr*)

  Kommune1(KoNr, KoNavn)

  Kommune2(KoNr, KoAreal)

- We end up with **two tables with the same primary key.**
- So we can **merge these two tables**, since it is easier to combine the two dependencies on KoNr:

  Person(FNr, Enavn, KoNr*)

  Kommune(KoNr, KoNavn, KoAreal)

Exercise 1

StudNr	Snavn	KursKode	KursNavn	StPoeng	EksDato	Kar
191234	Hansen	DAT1000	Databaser	7.5	12.12.2019	C

1. List the functional dependencies.
2. Find candidate and primary keys.
3. Decide which is the current normal form.
4. Carry out normalisation to BCNF.

- Repeat steps 2 to 4 for every intermediary result.

Exercise 2

- Given a table T(A, B, C, D). There are functional dependencies from A to B, from B to D and from C to A
  - Find candidate keys.
  - Which is the current normal form of this table?
  - Carry out normalisation to BCNF.

- Is it surprising that we can normalise a table that contains data which we cannot see ?
  - How is this possible ?

Solution to Exercise 1

StudNr	Snavn	KursKode	KursNavn	StPoeng	EksDato	Kar
191234	Hansen	DAT1000	Databaser	7.5	12.12.2019	C

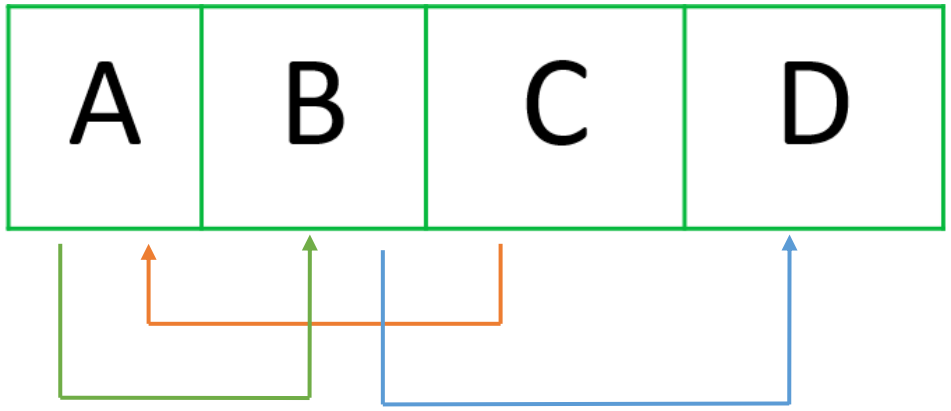
- Functional dependencies:
  StudNr $\rightarrow$ Snavn
  KursKode $\rightarrow$ KursNavn, StPoeng
  StudNr, KursKode, EksDato $\rightarrow$ Kar

- Find candidate and primary keys: <u>StudNr KursKode EksDato</u>

- Current normal form: 1NF

- Carry out normalisation to BCNF:
  - T1(<u>StudNr</u>, SNavn),
  - T2(<u>KursKode</u>, KursNavn, StPoeng),
  - T3(<u>StudNr, KursKode, EksDato</u>, Kar).

Solution to Exercise 2



1. Candidate keys: C
2. The current normal form is: 2NF
3. Normalisation to BCNF: T1(A, B), T2(B,D), T3(C,A).

❖ The information about **functional dependencies** tells us all we need to know in order to normalise the table.

ER and Normalisation

- Is normalisation an alternative to ER ? A possible strategy:
  1. Begin with the « **universal relation** » (all data in one table).
  2. Find functional dependencies.
  3. Carry out normalisation.
- Entities with attributes are a « *natural* » or convenient step in this thought process.
  - ER is useful to **communicate** with the users.
- Normalisation can be used to **check** a data model.
  - This is should be done by the database designer.

Denormalisation

Normalisation gives many tables.

- Many tables give many joins.
- Verifying certain validation rules requires that the DBMS looks into many tables.

Due to ***efficiency*** concerns it can be justified in certain cases to combine normalised tables.

- This is called **denormalisation** and gives a form of «controlled redundance».
- Especially, normalising from 3NF to BCNF gives validation rules that are difficult to check for the DBMS. Therefore it is common to stop normalisation at 3NF.

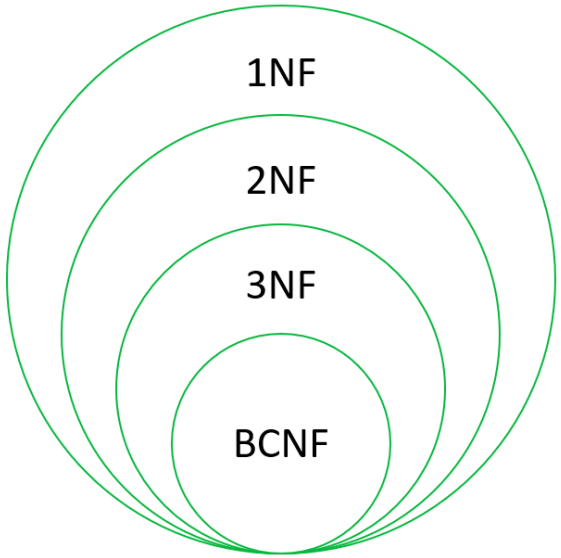
Key Concepts of Normalisation

- Redundance
- Functional dependence
- Determinant

- Partial dependence
- Transitive dependence

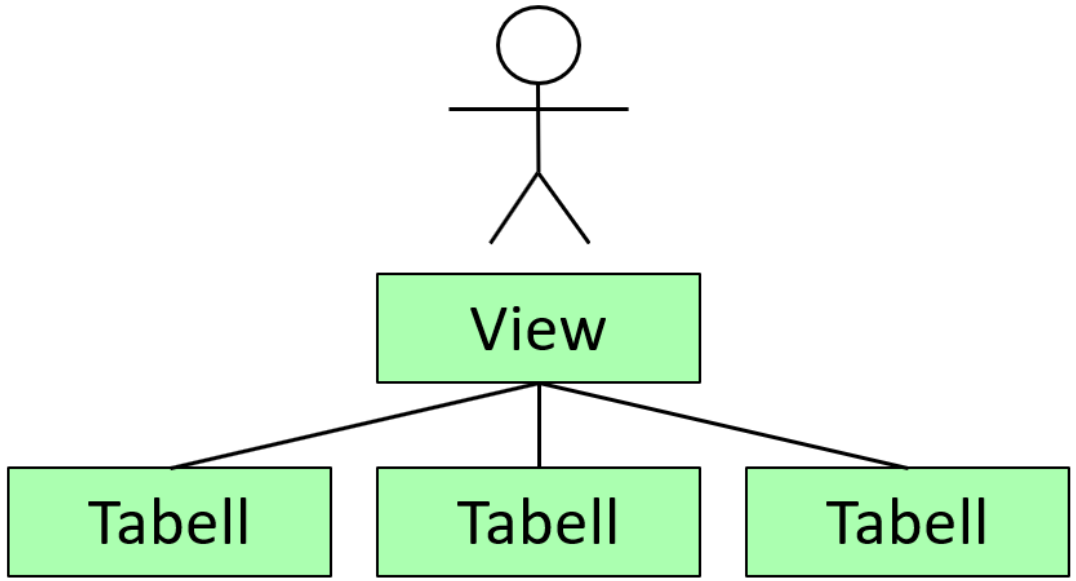
- Normal forms: 1NF, 2NF, 3NF, BCNF
- Normalisation Steps

- Denormalisation



Views and Database Design

- **Security**: Allow fine grained management of ***access rights***.
- **Presentation**: Hide the table structure and provide a ***simplified representation*** of the database.
- **Representational Independence**: Give the DBA the freedom to change the table structure without requiring to recode everything.



Representational Independence – Example

You want to **restructure the tables** in a database, for example:

- Split a table into two,
- Add or remove columns,
- Change the datatype in one or several columns.

But – many programs use the «old» database.

➢ So we make **views** which

- Get data from the *new* database,
- And have the **same name** and **structure** as the tables in the old database.

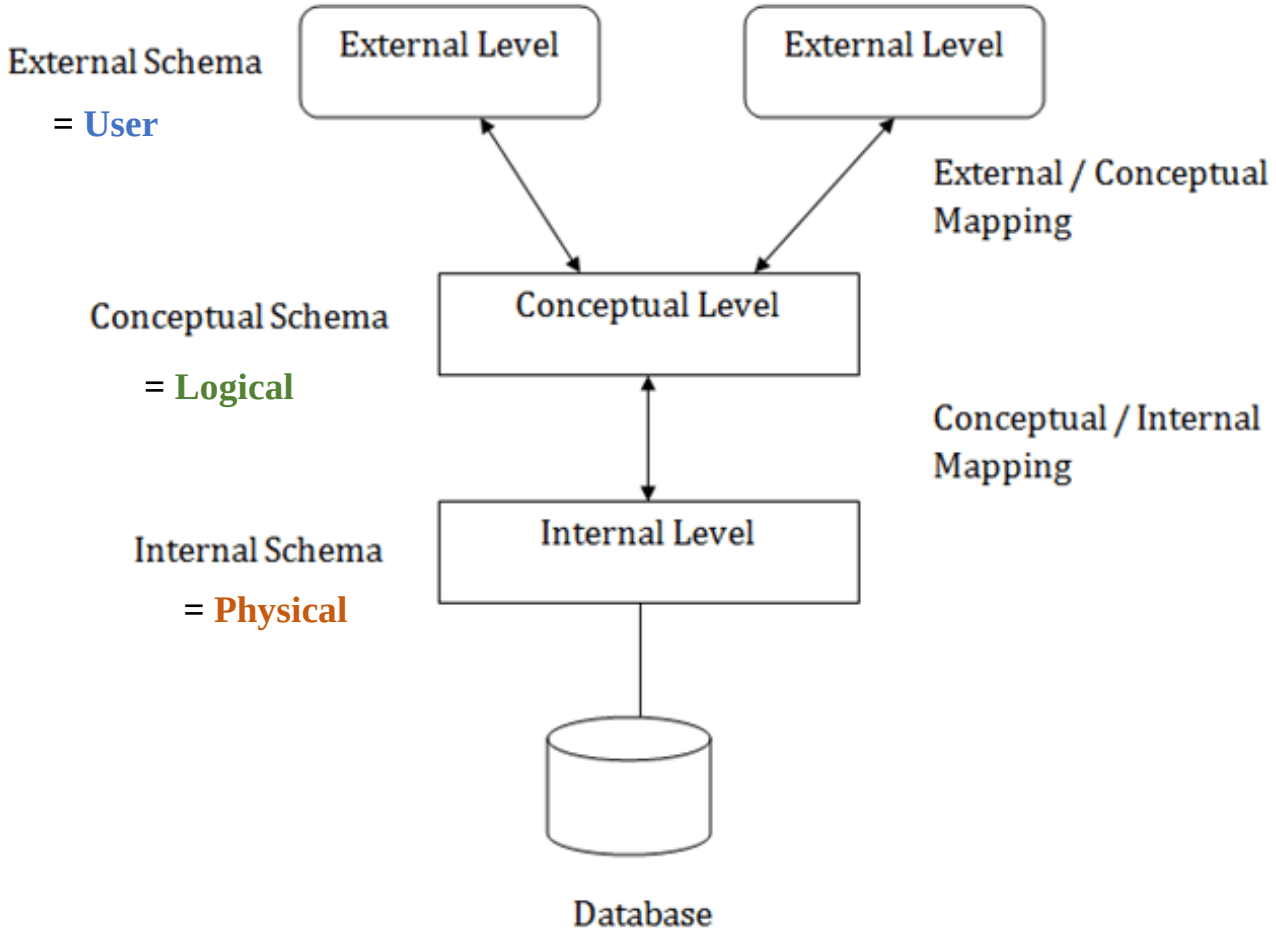
ANSI/SPARC Three Schema Architecture

Also called «Three Level Architecture», introduced in the 1970ies.

Reminds us that it is **useful** to

distinguish *technical details*
from the *conceptual design*
of the database

– i.e. distinguish «what» and «how».



https://www.javatpoint.com/dbms-three-schema-architecture

Quizz on Normalisation (part 2)

Please answer the practice quizz on mitt.uib now ☺

(you can take it again later if you want)

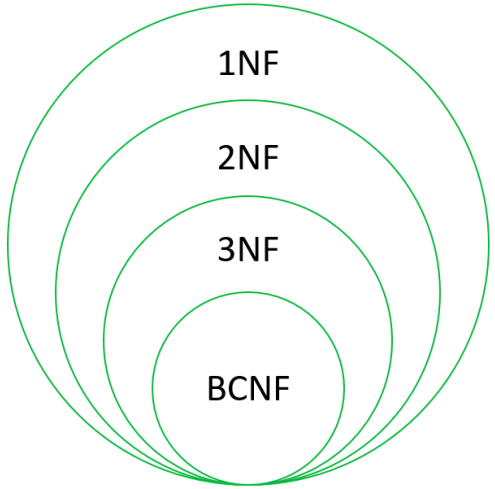
Link:

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

Chapter 8: *Normalisation*

Summary:

- How to **normalize tables** to **avoid redundance** in the database.
  - How to identify **functional dependencies**.
  - How to **decompose** tables.
  - **Normal forms**: 1NF, 2NF, 3NF, BCNF
  - **Denormalisation.**



- Understand how **views** can be used in **database design**.