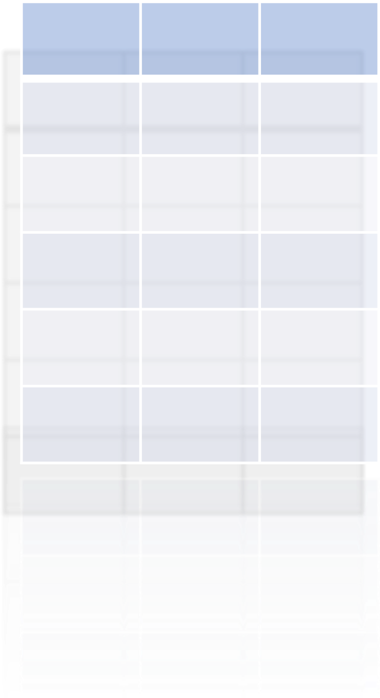
INF115 Lecture 5: *The Relational Model*

Adriaan Ludl
Department of Informatics
University of Bergen

Spring Semester 2021

Learning goals

- ❖ Motivation of the **relational model**, the theoretical foundation of relational datbases
- ❖ Tables can be modelled as **mathematical relations**
- ❖ How the requirements of primary and foreign keys are formulated in the **relational model**
- ❖ **Use relational algebra to write queries !**



The Relational Model

The theoretical framework on which all relational databases are based. Reference:

E. F. Codd (1970) **A Relational Model of Data for Large Shared Data Banks**.
Communication of the ACM, **13** (6).

The Relational Model has three parts:

- **Data structure**: What is a table **?**
  - Tables are relations !
- **Integrity rules**: Which data are permitted in this framework **?**
  - Primary and foreign keys
- **Data manipulation**: How to interact with data **?**
  - **Relational algebra** (*can do the same as SQL*)

Motivation: **Independence of Representations**

- *When programs are independent of the physical representation of data, then the **structure can be changed** and **the code still works !***

The Relational Model

The theoretical framework on which all relational databases are based. Reference:

E. F. Codd (1970) **A Relational Model of Data for Large Shared Data Banks**.
Communication of the ACM, **13** (6).

The Relational Model has three parts:

- ❖ **Data structure**: What is a table **?**
  - Tables are relations !
- ❖ **Integrity rules**: Which data are permitted in this framework **?**
  - Primary and foreign keys
- ❖ **Data manipulation**: How to interact with data **?**
  - **Relational algebra** (*can do the same as SQL*)

Motivation: **Independence of Representations**

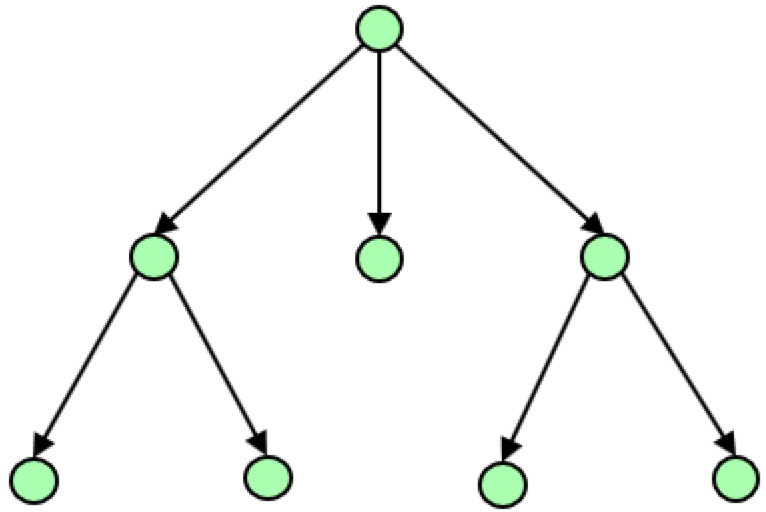
➢ *When programs are independent of the physical representation of data, then the **structure can be changed** and **the code still works !***

Independence of Representations

Motivation:

- *Hierarchical databases* and *network databases* are based on structures that aid navigation (pointers ... i.e. one given representation of the data).
- So **programs must *follow pointers*** and are <u>dependent on this organisation</u>.

➢ Is there a more general and abstract data structure ?



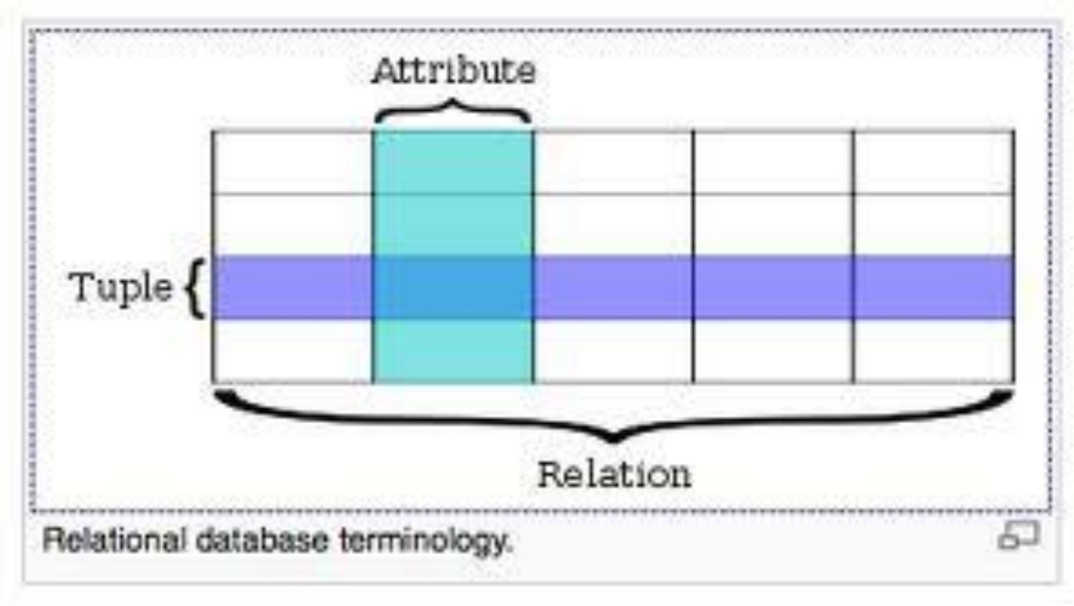
Independence of Representations

Motivation:

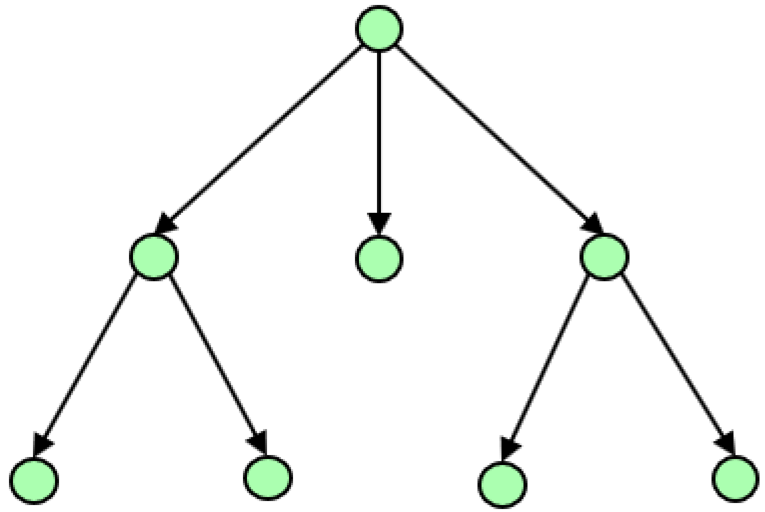
- *Hierarchical databases* and *network databases* are based on structures that aid navigation (pointers ... i.e. one given representation of the data).
- So **programs must *follow pointers*** and are <u>dependent on this organisation</u>.

➢ Is there a more general and abstract data structure ?

❖ **Relational Model**



❖ **Hierarchical Model**



Unique columns without ordering = a set of columns

AnsattNr	Etternavn	Fornavn	AnsattDato	Stilling	Lønn
1	Veum	Varg	01.01.1996	Løpegutt	383 000.00
2	Stein	Trude	10.10.2004	DBA	470 700.00

Unique columns without ordering

- Column names must be unique in a given table.

AnsattNr	Etternavn	Fornavn	AnsattDato	Stilling	Lønn
1	Veum	Varg	01.01.1996	Løpegutt	383 000.00
2	Stein	Trude	10.10.2004	DBA	470 700.00

- Their ordering is irrelevant.

A set of rows

ProsjektNr	Budsjett	Leder	Start	Slutt
1001	kr 15 000.00	20	12.01.2019	12.03.2019
1002	kr 750 000.00	8	23.06.2019	23.07.2019
1007	kr 125 000.00	2	12.06.2020	
1009	kr 500 000.00	20	01.01.2020	
1012	kr 10 000.00	4	10.07.2020	
1020	kr 900 000.00	8	23.07.2019	01.09.2019

A set of rows

ProsjektNr	Budsjett	Leder	Start	Slutt
1001	kr 15 000.00	20	12.01.2019	12.03.2019
1002	kr 750 000.00	8	23.06.2019	23.07.2019
1007	kr 125 000.00	2	12.06.2020	
1009	kr 500 000.00	20	01.01.2020	
1012	kr 10 000.00	4	10.07.2020	
1020	kr 900 000.00	8	23.07.2019	01.09.2019

- Ordering is irrelevant.

- No two rows can be the same. This means that **every row has a unique identifier** = its **primary key**.

Atomic values

ProsjektNr	Ansatte
1001	1
1002	4, 8, 13, 20

➢ **Not permitted !**

Values are « simple » (numbers, text, dates ...).

- « Repeating » values are not allowed.
- « Tables within tables » are not allowed.

Note: *Object* relational databases do allow more complex values.

Two tables with the same contents

AnsNr	Fornavn	Etternavn
1	Per	Hansen
2	Lise	Jensen
3	Anders	Lie
4	Johanne	Amundsen

AnsNr	Etternavn	Fornavn
3	Lie	Anders
2	Jensen	Lise
4	Amundsen	Johanne
1	Hansen	Per

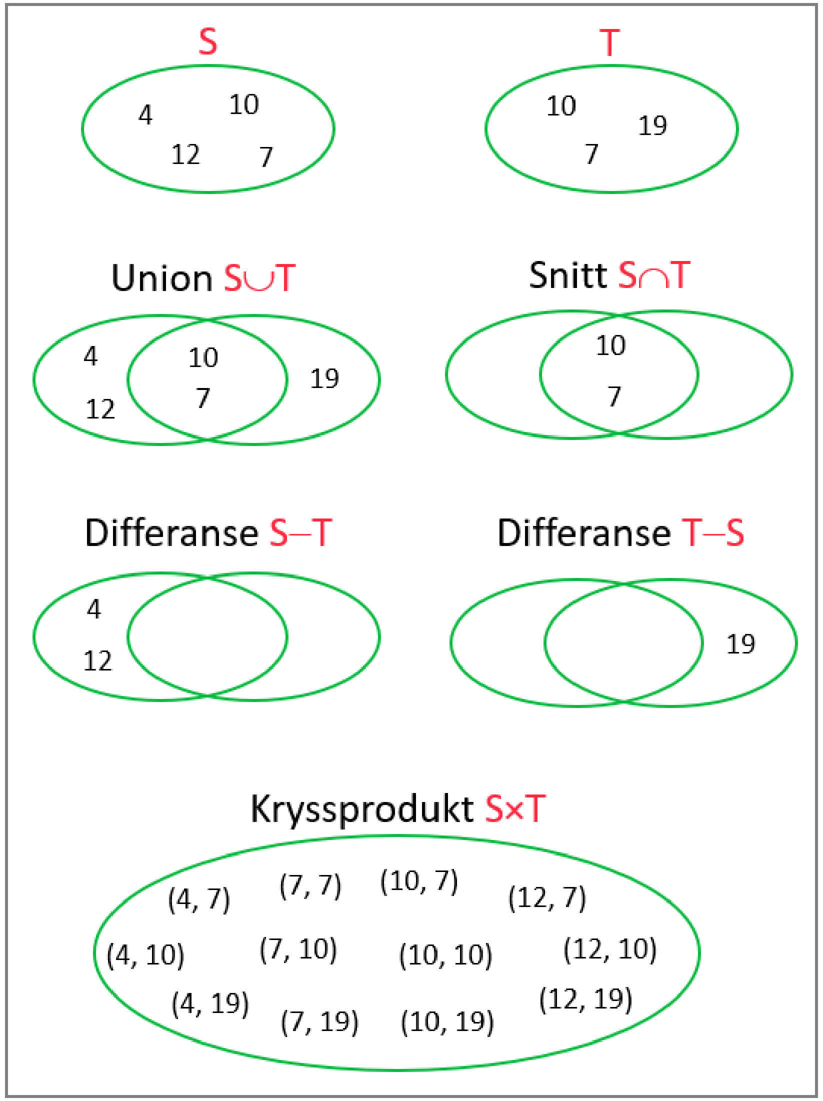
Note the permutations of rows and columns !

Tables As Relations

AnsattNr	ProsjektNr	AntTimer
1	1	12
1	7	20
1	9	7
1	12	14
2	7	3
4	1	1
4	7	10
4	9	120
4	12	75
5	2	100
5	12	94
8	2	10
8	12	20
8	20	20
11	7	50
11	9	20
13	2	3
20	9	4
20	20	20

- An employee can work on many projects.
- A project can have many contributors (use git :-).
- This is a **many to many relation** between employees and projects.
- For every **combination** of project and employee we store the number of work hours spent.
- Thus, a **table** <u>can *represent* </u>a **relation.**

The relational model builds on set theory (mengdelære)



Intersection

Differences:

Cartesian product:

Tables = Relations

Let S_1 = { a,b,c } and S_2 = { 1,2 } be two sets.

Then the **cartesian product** of S_1 and S_2 (kryssproduktet) is defined as the set of the following ordered pairs:

$$S_1 \times S_2 = \{ (a,1), (a,2), (b,1), (b,2), (c,1), (c,2) \}$$

- A **relation** over sets S_1 and S_2 is a *subset* of $S_1 \times S_2$,

  for example: { (a,2), (b,1), (b,2) }.

Visualisation of a (finite) relation as table:

S_1	S_2
a	2
b	1
b	2

- In general, $S_1 \times \ldots \times S_n$ is composed of

  the **tuples:** ($x_1, \ldots, x_n$).

Quizz on *The Relational Model* (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 !

Functions as tables

Let S_1 ={ a,b,c } and S_2 = { 1,2 } be two sets.

- A **function** from S_1 to S_2 is a *relation* over S_1 and S_2 which does **not** contain any pairs with the same S_1-values and different S_2-values.

S_1	S_2
a	2
b	2
c	2
c	3

Breach of the function requirement !

Functional dependence

- We say that there is a **functional dependence** from A to B, written as $A \rightarrow B$, if two rows with the same value in column A must always have the same value in column B.

- We can also consider **functional dependence** between a collection of columns X to some column C: $X \rightarrow C$.

ANr	Navn	PNr
12	Lise	7
17	Ola	8
12	Lise	23
21	Kari	26

Functional dependence

- We say that there is a **functional dependence** from A to B, written as A $\rightarrow$ B, if two rows with the same value in column A must always have the same value in column B.

- We can also consider **functional dependence** between a collection of columns X to some column C: X $\rightarrow$ C.

ANr	Navn	PNr
12	Lise	7
17	Ola	8
12	Lise	23
21	Kari	26

Here:
ANr$\rightarrow$Navn

Primary keys

- ❖ Given a table T: A **super key** for T is a collection of columns X such that X $\rightarrow$ A holds for all columns A.
- ❖ A **candidate key** is a *minimal* super key.
- ➢ The database designers **choose** one of the possible candidate keys as **primary key.**
- ➢ *Every table shall have exactly one primary key.*

Exercise: Find candidate keys and primary key for the tables:

Ansatt(**AnsNr**, **PersonNr**, Fornavn, Etternavn, Stilling)

Vielse(BrudPNr, BrudgomPNr, Kirkenavn, Dato)

Primary Keys and Entity Integrity

- ❖ Every table shall have exactly one primary key.
- ❖ A primary key <u>cannot</u> contain Null values.
- ❖ A primary key <u>cannot</u> contain two equal values.
- ➢ This is called **entity integrity.**
- ➢ **The DBMS must guarantee *entity integrity* at all times.**

Entity integrity is checked when:

- ✓ A new row is inserted.
- ✓ A primary key value is updated.

Foreign Keys and Reference Integrity

Two attributes are **union compatible** if they belong to the same domain.

- A **foreign key** is an attribute A that is union compatible with a primary key B.
- The values in A are a subset of the values in B.

- This is called **reference integrity.**

- Foreign keys can be composed of multiple columns,
- and they can contain Null values.

Reference integrity is checked when:

- ✓ When inserting new values in a foreign key.
- ✓ When deleting values in a primary key.

The DBMS must guarantee *reference integrity* at all times.

Operators of relational algebra

Relational algebra is a mathematical language for queries.
Can do much the same as in SQL SELECT queries:

- Projection = choose columns: $\Pi_{\text{PNr, Tittel}}$(Prosjekt)
- Selection = choose rows: $\sigma_{\text{Budsjett}<100000}$(Prosjekt)
- Union: $\Pi_{\text{Startdato}}(\text{Prosjekt}) \cup \Pi_{\text{Sluttdato}}(\text{Prosjekt})$
- Difference: $\Pi_{\text{AnsattNr}}(\text{Ansatt}) - \Pi_{\text{AnsattNr}}(\text{Arbeid})$
- Intersection: $\Pi_{\text{AnsattNr}}(\text{Arbeid}) \cap \Pi_{\text{Leder}}(\text{Prosjekt})$
- Cartesian product (kryssprodukt): Prosjekt × Ansatt
- Inner join (Likekobling) : $\text{Prosjekt} \bowtie_{\text{Prosjekt.Leder=Ansatt.AnsNr}} \text{Ansatt}$

Operators of relational algebra

Relational algebra is a mathematical language for queries.
Can do much the same as in SQL SELECT queries:

- Projection = choose columns: $\Pi_{\text{PNr, Tittel}}(\text{ Prosjekt })$
- Selection = choose rows: $\sigma_{\text{Budsjett}<100000}(\text{ Prosjekt })$
- Union: $\Pi_{\text{Startdato}}(\text{Prosjekt}) \cup \Pi_{\text{Sluttdato}}(\text{Prosjekt})$
- Difference: $\Pi_{\text{AnsattNr}}(\text{Ansatt}) - \Pi_{\text{AnsattNr}}(\text{Arbeid})$
- Intersection: $\Pi_{\text{AnsattNr}}(\text{Arbeid}) \cap \Pi_{\text{Leder}}(\text{Prosjekt})$
- Cartesian product (kryssprodukt): $\text{Prosjekt} \times \text{Ansatt}$
- Inner join (Likekobling) : $\text{Prosjekt} \bowtie_{\text{Prosjekt.Leder=Ansatt.AnsNr}} \text{Ansatt}$

Operators of relational algebra

Relational algebra is a mathematical language for queries.
Can do much the same as in SQL SELECT queries:

- Projection = choose columns: $\Pi_{\text{PNr, Tittel}}(\text{ Prosjekt })$
- Selection = choose rows: $\sigma_{\text{Budsjett}<100000}(\text{ Prosjekt })$
- Union: $\Pi_{\text{Startdato}}(\text{Prosjekt}) \cup \Pi_{\text{Sluttdato}}(\text{Prosjekt})$
- Difference: $\Pi_{\text{AnsattNr}}(\text{Ansatt}) - \Pi_{\text{AnsattNr}}(\text{Arbeid})$
- Intersection: $\Pi_{\text{AnsattNr}}(\text{Arbeid}) \cap \Pi_{\text{Leder}}(\text{Prosjekt})$
- Cartesian product (kryssprodukt): $\text{Prosjekt} \times \text{Ansatt}$
- Inner join (Likekobling) : $\text{Prosjekt} \bowtie_{\text{Prosjekt.Leder=Ansatt.AnsNr}} \text{Ansatt}$

Relational algebra and SQL

- A query language is **relationally complete** if it is equal in expressive power to relational algebra.

➢ ***SQL is relationally complete .***

$\Pi_{A,\ B,\ C}\ (R \bowtie_{X=Y} S)$ **is the same as:**

SELECT A, B, C FROM R, S WHERE X=Y

- Relational algebra is therefore *the reference* for *concrete* query languages (e.g. SQL).

- **Relational database management systems** (RDBMS) are built on the relational model.

Quizz on *The Relational Model* (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

Summary: *The Relational Model*

- **Relational model** and independence of representation.
- Tables can be modelled as **mathematical relations**, i.e. subsets of cartesian products.
- Primary and foreign keys are formulated as **functional dependence** in the **relational model.**
- **Use the operators of relational algebra to write queries !**