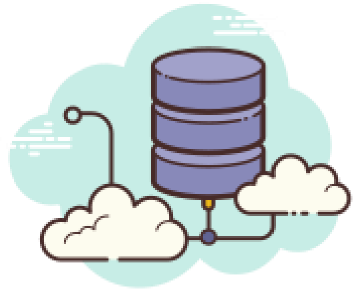
INF115 Lecture 4: *Queries on Several Tables*

Adriaan Ludl
Department of Informatics
University of Bergen

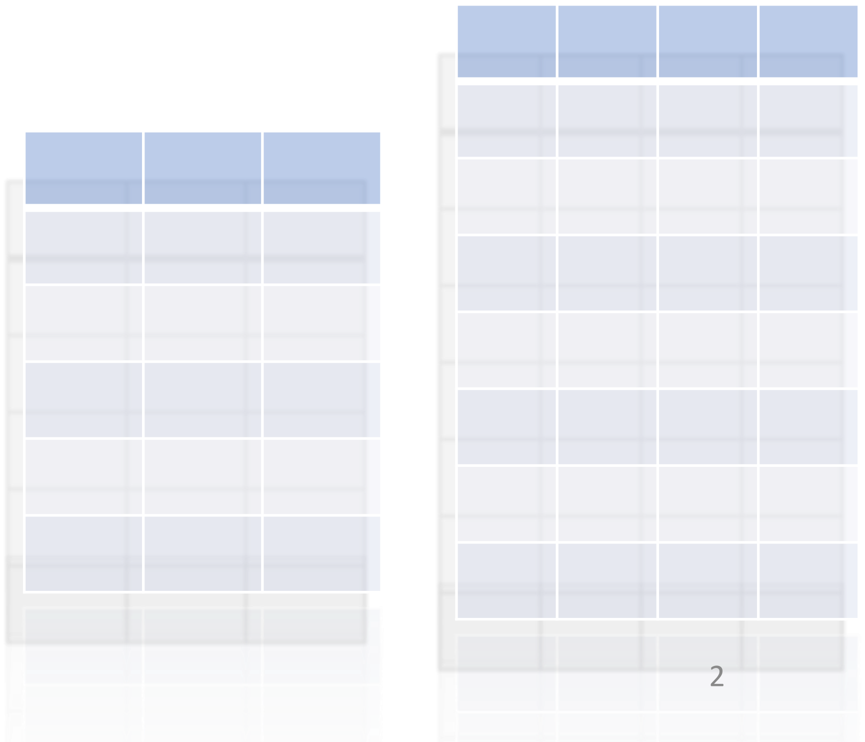
Spring Semester 2021

Chapter 4: *Queries on Several Tables*



Learning Goals:

- **Understand** *why we need queries on several tables*
- Understand the difference between **various ways to join tables**
  - such as **cross product** and **inner join**
- Use SQL to write:
  - **Inner Joins** and **Outer Joins**,
  - **Self Joins**,
  - *General* Joins and *Natural* Joins.
- Use SQL **set operators**



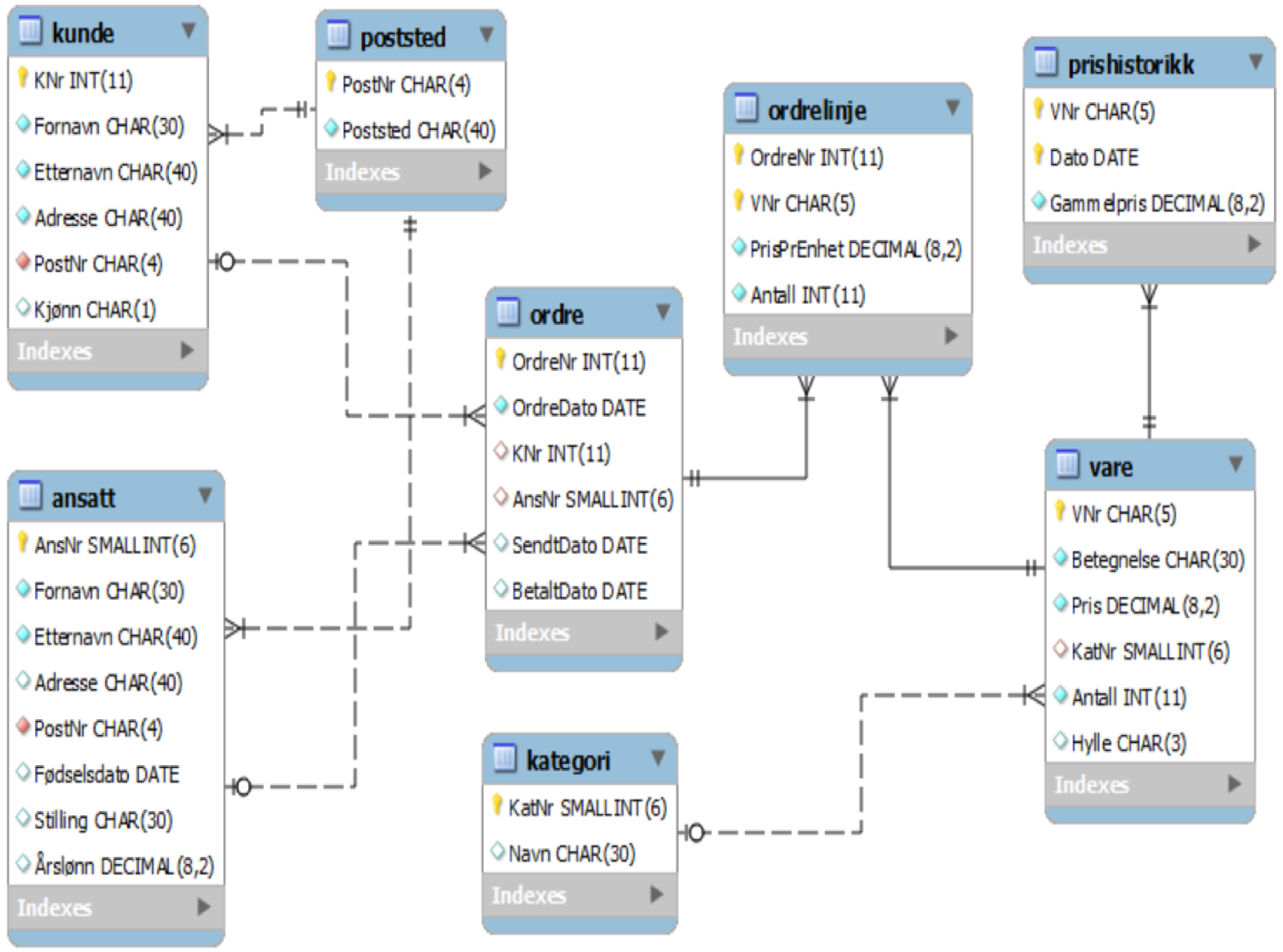
Motivation

- Saving data in multiple tables allows to avoid **redundance** («dobbeltlagring»).
- Relational databases are made up of **many** tables.

- Various tables are connected by **logical relations** (sammenhenger):
  - Need to «**join**» («**koble**») data from several tables.
  - Joins are often based on foreign keys (but not always).

Which tables have to be used to get … ?

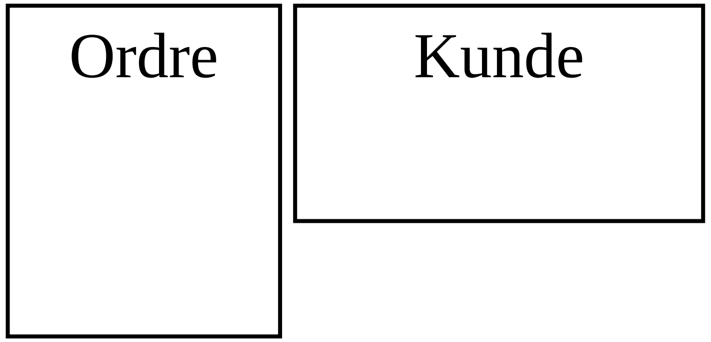
1. The list of clients sorted by postcode (poststed).
2. The total price per order. Showing the order number and price.
3. Sales per client in every category. Sorted by client name.
4. etc. …



Multiple tables in the FROM section

What will be the result of:

```
SELECT *
FROM Ordre, Kunde
```



- Which rows in *Ordre* shall be joined with which rows in *Kunde* ?
  - Ordre contains a column KNr …

- What will be the number of rows and columns in the result ?

➢ In general we can get data from more than 2 tables !

NB! The query above is not useful ! Explanation follows …

Cartesian product (kryssprodukt) of two tables

Table *Kunde*

KNr	Navn
1	Per
2	Ola

Table *Order*

OrdreNr	KNr	AnsNr
1	1	21
2	2	21
3	2	28

Number of rows: $\boxed{2 \times 3 = 6}$

Output Table
(cart. product)

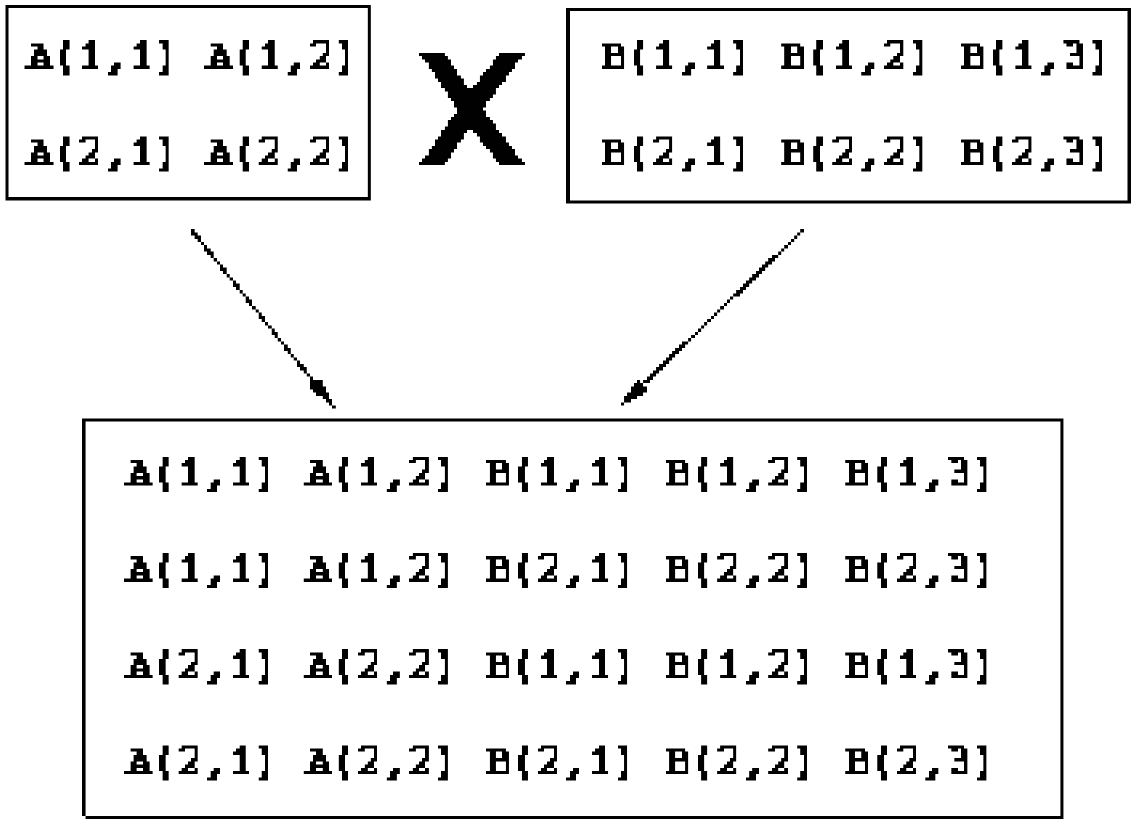
KNr	Navn	OrdreNr	KNr	AnsNr
1	Per	1	1	21
1	Per	2	2	21
1	Per	3	2	28
2	Ola	1	1	21
2	Ola	2	2	21
2	Ola	3	2	28

For large tables

this becomes

*an **even larger table** !*

Cartesian Product example



A(1,1)	A(1,2)
A(2,1)	A(2,2)

X

B(1,1)	B(1,2)	B(1,3)
B(2,1)	B(2,2)	B(2,3)

A(1,1)	A(1,2)	B(1,1)	B(1,2)	B(1,3)
A(1,1)	A(1,2)	B(2,1)	B(2,2)	B(2,3)
A(2,1)	A(2,2)	B(1,1)	B(1,2)	B(1,3)
A(2,1)	A(2,2)	B(2,1)	B(2,2)	B(2,3)

Inner Join (Likekobling): a subset of the cartesian product

Table *Kunde*

KNr	Navn
1	Per
2	Ola

OrdreNr	KNr	AnsNr
1	1	21
2	2	21
3	2	28

Table *Order*

Output Table
(inner join)

KNr	Navn	OrdreNr	KNr	AnsNr
1	Per	1	1	21
2	Ola	2	2	21
2	Ola	3	2	28

➢ The **column**(s) on **which the join is performed**,
will be copied from only one of the original tables.

▪ All contents are copied from the tables to be joined.

Compound names (Sammensatte navn)

Columns with identical names can exist in multiple tables.

In **queries on *several tables*** it **might not always be clear** which column is meant.

➢ Use *compound names* (such as **table.column**) to indicate which column is meant. Such as:

- *Kunde.KNr*
- *Kunde.Navn*
- *Ordre.KNr*

Inner Join Example in SQL (Likekobling)

List the clients with their order:

```
SELECT Kunde.KNr, Etternavn, OrdreNr
FROM Kunde, Ordre
WHERE Ordre.KNr = Kunde.KNr
```

- ➢ The equality in WHERE is a **join condition**.

  (Likheten i **WHERE** er en **koblingsbetingelse**.)
- ➢ This query is called an **inner join** (**likekobling**).

- ❖ Here *KNr* **must be <u>prefixed</u> with the <u>table name</u>,**

  because this column name appears in both tables!
- ❖ For *Etternavn* and *OrdreNr* **prefixes can be used**, but it is not strictly required

  because there is no ambiguity.

Foreign Keys and Joins

- Tables are *often* joined on *foreign keys*.
- It is possible to join on columns that are neither *foreign* nor *primary keys*.

Example: Find the combinations of employees and clients who live in the same locality (poststed):

```
SELECT *
FROM Ansatt, Kunde
WHERE Ansatt.PostNr = Kunde.PostNr
```

Syntax of Inner Joins

Inner joins are so common that there is a special way to write them in SQL:

```
SELECT *
FROM Ordre INNER JOIN Kunde
     ON Ordre.KNr = Kunde.KNr
```

Generally: `T1 INNER JOIN T2 ON T1.col1 = T2.col2`

- The order of tables in the query does not matter.
- INNER JOIN is also the recommended form, but at the start it is perhaps easier to write join conditions with WHERE statements …

Quizz on Multiple Tables (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

Synonyms

We can use synonyms (shortened names) for tables:

```
SELECT O.OrdreNr, K.KNr
FROM Ordre AS O INNER JOIN Kunde AS K
     ON O.KNr = K.KNr
```

We must use *the synonyms* if we define them !

❖ The synonyms O and K must also be used in the **SELECT** section,
even though they are defined in the **FROM** section.

❖ In Oracle one cannot use **AS** for tables. In MySQL one can choose.

Inner Joins with additional conditions

We can use general conditions in addition to join conditions:

```
SELECT V.VNr, K.Navn
FROM Vare AS V, Kategori AS K
WHERE V.KatNr = K.KatNr
AND V.Pris > 100
AND K.Navn = 'Fiske'
```

- ➢ How should the DBMS execute such a query ?
  - Join conditions first?
  - Other conditions first ?

Inner Joins with additional conditions

We can use general conditions in addition to join conditions:

```
SELECT V.VNr, K.Navn
FROM Vare AS V, Kategori AS K
WHERE V.KatNr = K.KatNr
AND V.Pris > 100
AND K.Navn = 'Fiske’
```

- How should the DBMS execute such a query ?
  - Join conditions first?
  - **Other conditions first ?**

15 minute break!

Lecture resumes at 15:00

Datatypes and comparisons

The query below is **meaningless**:

```
SELECT *
FROM Ansatt AS A INNER JOIN Ordre AS O
ON A.AnsNr = O.OrdreNr
```

- ❖ **Columns** used in together in a join **must have the same datatype.**
- ❖ They must also contain **values which describe comparable quantities**.
- ❖ Nevertheless, they can have different names.
- ➢ Will the DBMS accept the query above ?

Joining more than 2 tables:

Which clients have bought product 1014 ?

```
SELECT K.*
FROM
  Kunde AS K INNER JOIN
  ( Ordre AS O INNER JOIN Ordrelinje AS OL
     ON O.OrdreNr = OL.OrdreNr )
  ON K.KNr = O.KNr
WHERE OL.VNr = 1014
```

This can also be written as:

```
SELECT K.*
FROM Kunde AS K, Ordre AS O, Ordrelinje AS OL
WHERE O.OrdreNr = OL.OrdreNr
AND K.KNr = O.KNr
AND OL.VNr = 1014
```

Inner Join with Grouping

a) Find the number of orders per client:

```
SELECT K.KNr, Etternavn,
       COUNT(*) AS AntallOrdrer
FROM Kunde AS K, Ordre AS O
WHERE K.KNr = O.KNr
GROUP BY K.KNr, Etternavn
```

➢ What if we only want to show clients with more than 10 orders ?

Inner Join with Grouping

a) Find the number of orders per client:

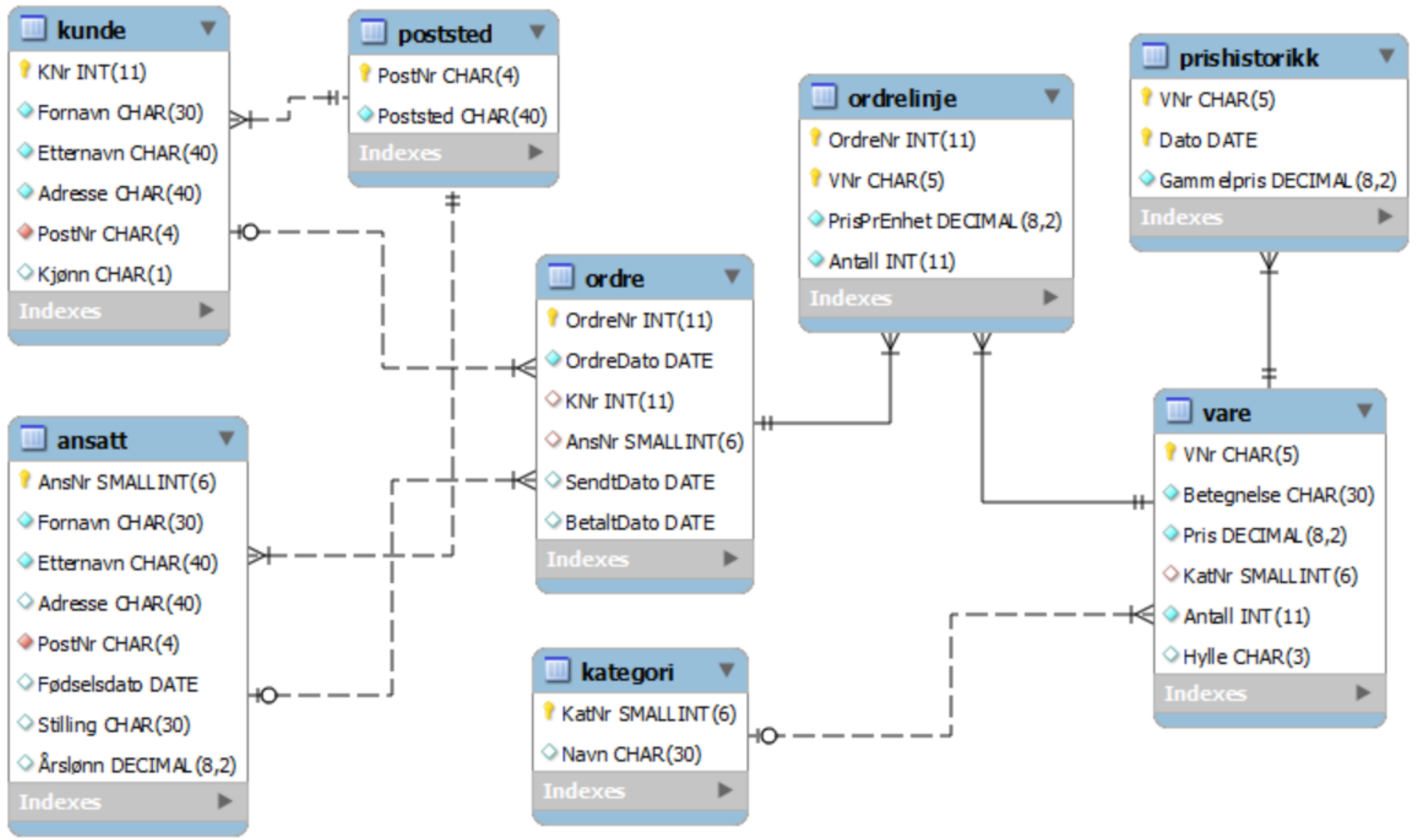
```
SELECT K.KNr, Etternavn,
       COUNT(*) AS AntallOrdrer
FROM Kunde AS K, Ordre AS O
WHERE K.KNr = O.KNr
GROUP BY K.KNr, Etternavn
HAVING AntallOrdrer > 10
```

➢ What if we only want to show clients with more than 10 orders ?

Inner Join with Grouping

b) Total amount of purchases per product:

- Which tables to join?
- What shall we group by ?
- Which set function do we have to use ?



Inner Join with grouping

b) Total amount of purchases per product (vare):

Solution:

```
SELECT V.VNr,
    SUM(OL.Antall * OL.PrisPrEnhet) AS SamletSalg
FROM Vare AS V, Ordrelinje AS OL
WHERE V.VNr = OL.VNr
GROUP BY V.VNr
```

Which columns to use in groupings ?

The query below from the previous task (a) has `Etternavn` in the **SELECT** section, because we want this column in the output.

So we also add it in GROUP BY, although it does not affect the grouping.

❖ Some systems would give **error messages**, if it would not be included in GROUP BY.

```
SELECT K.KNr, Etternavn,
       COUNT(*) AS AntallOrdrer
FROM Kunde AS K, Ordre AS O
WHERE K.KNr = O.KNr
GROUP BY K.KNr, Etternavn
```

Which columns to use in groupings ?

```
SELECT KatNr, Betegnelse, AVG(Pris) AS Snitt
FROM Vare
GROUP BY KatNr
```

In this query it is a **mistake** to include *Betegnelse*

– because it is a property of ***products*** and **not** of ***categories***.

However, *Pris* can be included after SELECT – even though it does not appear in GROUP BY, because it is a parameter of AVG.

Outer Joins: Left and Right

- Inner joins will return values from both tables,

  but this is not always what we want.

- Show clients with their respective orders. All clients shall be shown.

```
SELECT K.KNr, O.OrdreNr
FROM Kunde AS K LEFT OUTER JOIN Ordre AS O
     ON K.KNr = O.KNr
```

- «**Left**» and «**right**» refer to their ***order of appearance*** in the FROM section.
- How many rows will there be in the results ?

Left Outer Join

KNr	Navn
1	Per
2	Ola
3	Lise

OrdreNr	KNr	AnsNr
1008	1	25
1009	2	25
1010	1	28

KNr	Navn	OrdreNr	KNr	AnsNr
1	Per	1008	1	25
2	Ola	1009	2	25
1	Per	1010	1	28
3	Lise			

Nb of rows in Left Outer Join **=** Nb of rows in Inner Join

+ 1 row for each row on the left without match on the right.

General Joins

- It is possible to join using **other operators**, not only equality (=).
- Find all products (vare) that used to be more expensive:

```
SELECT DISTINCT V.VNr
FROM Vare AS V, Prishistorikk AS H
WHERE V.VNr = H.VNR
AND V.Pris < H.Gammelpris
```

- Some GIS-examples with «geographical operators»:
  - Find cities in Telemark (i.e. a point INSIDE a polygon)
  - Find roads that cross Mjøsa (line CROSSING a line)
  - Find owners who will be affected by roadworks
    (a polygon OVERLAPPING another polygon)

Natural Joins

A Natural join compares all columns whose names occur in both tables:

```
SELECT *
FROM Ansatt NATURAL JOIN Poststed
```

- This feature is not present in all systems.
- It is equivalent to an inner join:

```
SELECT Ansatt.*, Poststed.Poststed
FROM Ansatt INNER JOIN Poststed
ON Ansatt.PostNr = Poststed.PostNr
```

Self Joins

- Tables can be joined with « themselves ».
- Find all combinations of products with the same price:

```
SELECT V1.VareID, V2.VareID, V1.Pris
FROM Vare AS V1, Vare AS V2
WHERE V1.VareID <> V2.VareID
AND V1.Pris = V2.Pris
```

As if the DBMS would **make two copies** of the table *Vare*

and joins them in the usual way.

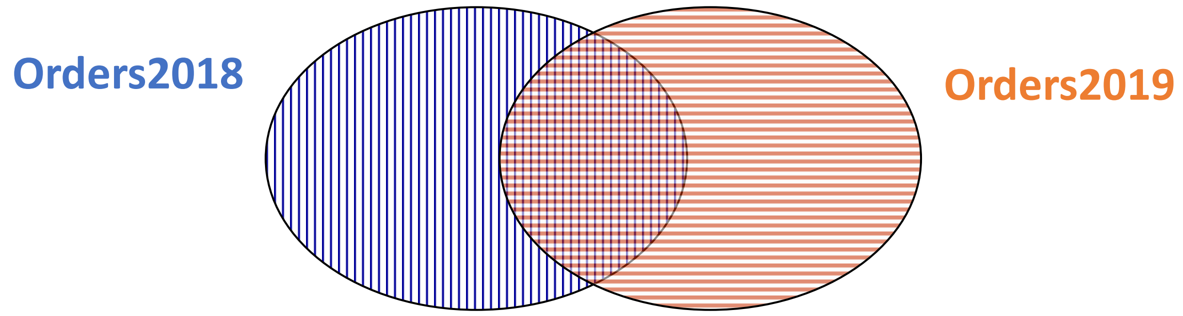
Self Joins

- **Self joins must be used when** we have a **foreign key** that **points to a primary key in the same table.**
- Example: Show employees and the name of their leader.

AnsNr	Fornavn	Etternavn	Leder
1	Georg	Barth	2
2	Gunnlaug	Angeltveit	
3	Morgan	Dalland	7
6	Vilde	Aksnes	8
7	Henriette	Brobakken	2
8	Synøve	Bakketun	2
9	Ragnvald	Allum	7
11	Oliver	Abrahamsen	7
13	Oda	Cappelen	8
16	Andrine	Ebbesen	7

Set Union, Intersection and Difference

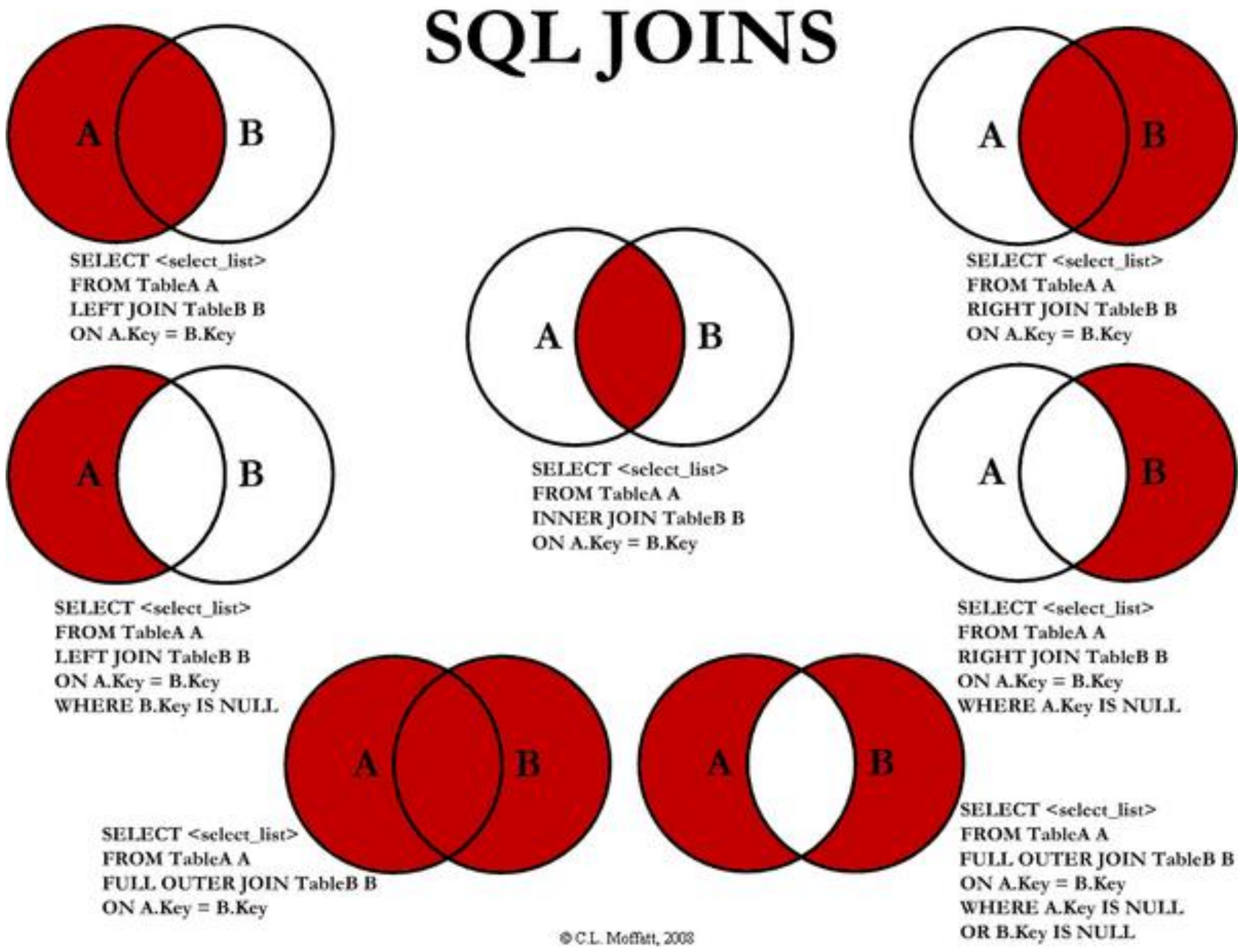
A table is a set of rows.
Thus, we can use traditional set operations:



- Suppose Orders2018 and Orders2019 contain historical data.
- Who has shopped in 2018 and/or 2019?

```
SELECT KNr FROM Orders2018
UNION
SELECT KNr FROM Orders2019
```

➢ Some systems (Oracle, PostgreSQL) also feature **INTERSECT** (snitt) and **MINUS/EXCEPT** (differanse).



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Quizz on Multiple Tables (part 2)

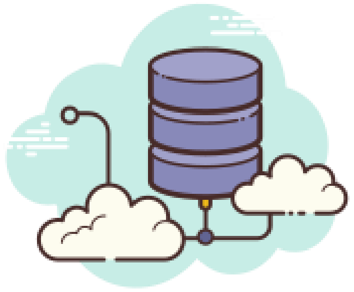
Please answer the practice quizz on mitt.uib now ☺

(you can take it again later if you want)

Link:

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

Summary: *Queries on Several Tables*



Use SQL to write:

- ❖ **Inner joins**: using WHERE or INNER JOIN.
- ❖ **Outer joins**: LEFT or RIGHT OUTER JOIN.
- ❖ **General joins:** using other operators: >, <, =>, =<, <> …
- ❖ **Natural joins:** on all columns shared between two tables.
- ❖ **Self joins:** e.g. when using a foreign key that is a primary **key** where both keys occur in the same table.

Use SQL **set operators:** UNION, INTERSECT, (MINUS)