INF130

Kapittel 4: Spørringer mot flere tabeller

Themes for today

- Chapter 4: Queries to several tables
- Monday: Go through mandatory exercise

Ikke i presentasjonen

4

Spørringer mot flere tabeller

Learning goals

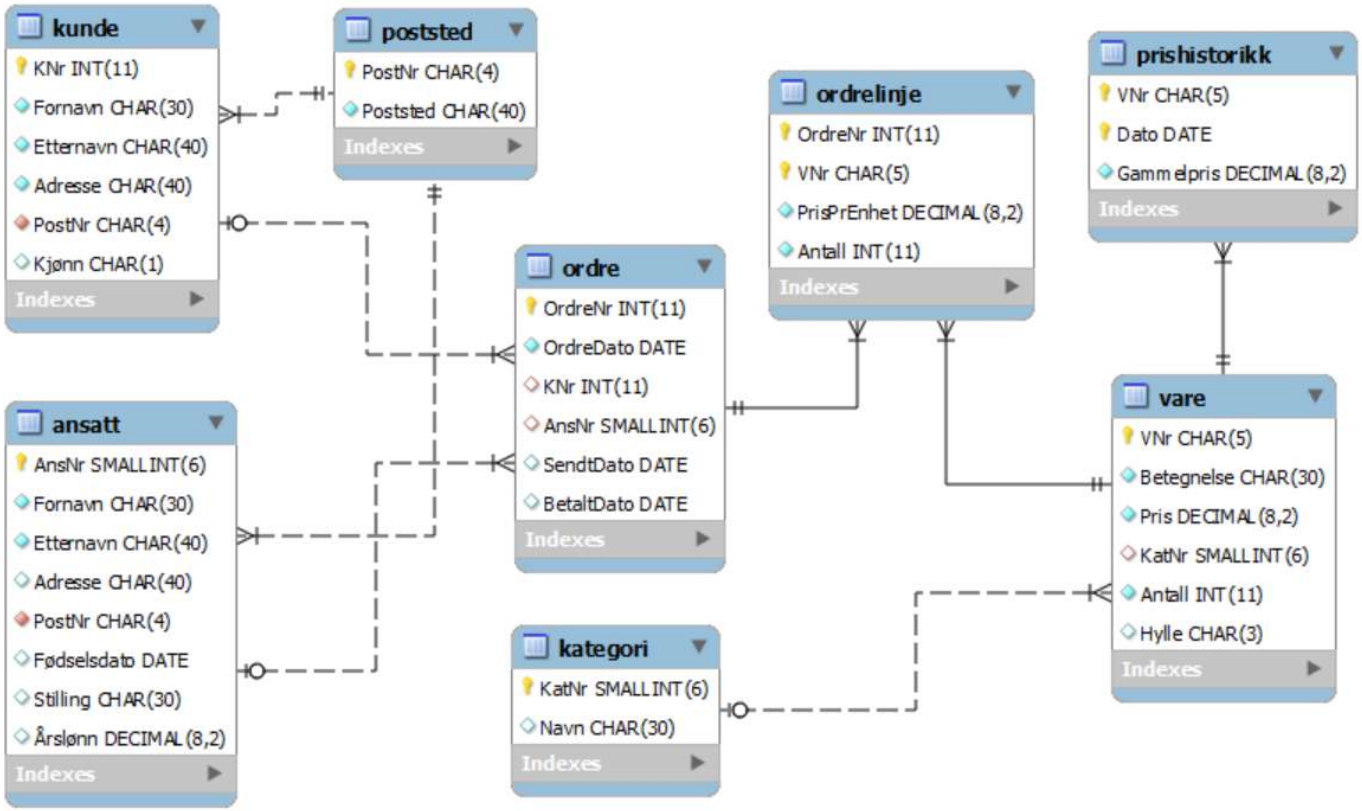
- Understand the need for queries to several tables
- Understand the effect of the different ways of connecting tables
- Use SQL to write inner joins, outer joins, self joins, general and natural joins
- Use SQL operators for quantity

Motivation

- We store data in several tables to avoid **redundance**
- Relational databases consist of many tables
- There are logic connections between data in different tables
  - Need to connect data from several tables
  - Same type of value must exist in several tables
  - Connections are often based on foreign keys, but not always

Which tables to use ?

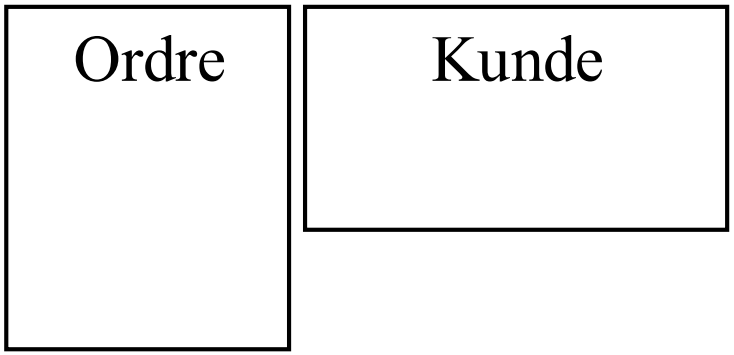
- Customer list sorted on postal area
- Total price pr. order. Enough to show order number and price
- Sale per customer in each category. Sort on customer name



Several tables in the FROM part

- What is the result ?

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



- Which rows in Ordre shall be connected with which rows in Kunde?
  - Ordre contains KNr …

- Number of **rows** and **culomns** in the result ?
- Generally we can fetch data from more than one table

- **Note! The Query above is meaningless !**

Cartesian product (cross product)

KNr	Navn
1	Per
2	Ola

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

2 × 3 = 6 !

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

Inner join

KNr	Navn
1	Per
2	Ola

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

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

- Joins columns from one of the tables, containing the same values.

Composed names

- Often we have columns with the same name in several tables
- In queries to several tables it isn't always unambigous what is meant with a column name

- We use composed names to tell which columns we mean (on the form **tabell.kolonne**), f.example:
  - Kunde.KNr
  - Kunde.Navn
  - Ordre.KNr

Examples of inner join

- Customers with their orders

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

- The equality in **WHERE** is a join condition
- The query is called an inner join

- KNr has to be prefixed with a table name because the columns name exist in both tables
- Etternavn and OrdreNr can be prefixed, but doesn't have to

Foreign keys and joins

- Often we join two tables with respect to foreign keys
- It is possible to join columns that are not foreign nor primary keys
- Combination of employees and customers living the same place (same postnr)

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

- Sometimes it is not right to join with regards to all the foreign keys

Inner join expressions

- Inner join occurs so often that a special expression exist:

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

- General:

```
T1 INNER JOIN T2 ON T1.kol1 = T2.kol2
```

- The order of the tables is irrelevant

Synonyms

- To reduce the work of writing one may introduce synonymes (short name) for the tables

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

- If you introduce synonyms they have to be used !
  - The synonyms O and K have to be used in **SELECT**, even if they ar introduced in **FROM**

- Oracle doesn't use **AS**. In MySQL you can choose

Inner join with extra conditions

- Can have 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 = 'Keramikk'
```

- How should DBHS carry out this kind of query ?
  - Join conditions first ?
  - Other conditions first ?

Data types and comparisons

- This is **meaningless**:

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

- Joined columns must have the same data type and also contain the **same values**
- Joint columns do **not** need to have the same **name**
- Will DBHS accept the query above

Join more tables than 2

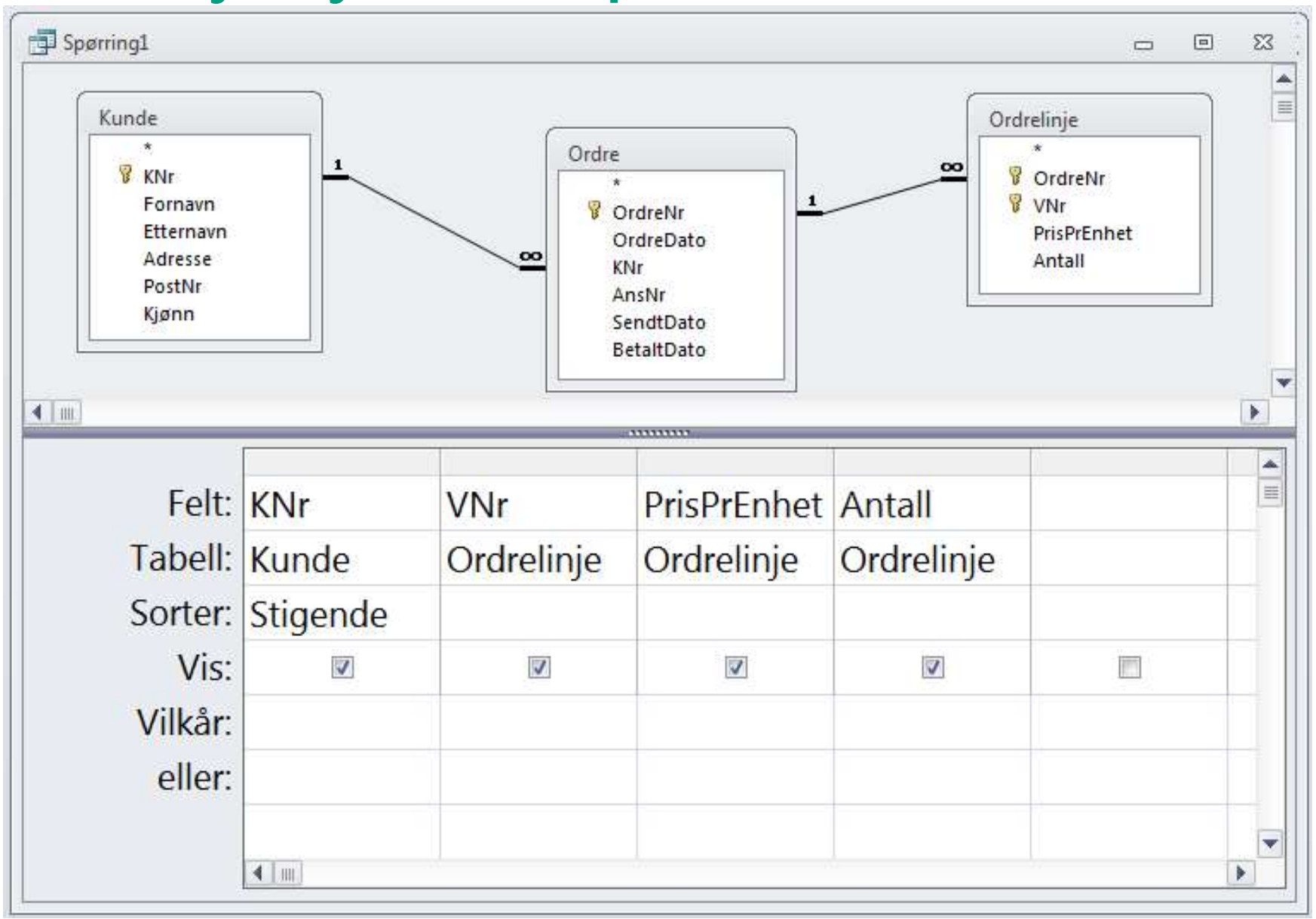
- Which customers bought vare nr 32067 ?

```
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 = 32067
```

- Can also be written like this (simpler):

```
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 = 32067
```

Query By Example



Inner join with grouping

- Find the number of orders pr. customer:

```
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 customers with more than 10 orders ?
- Total orders per customer:
  - Which tables need to joined ?
  - What to group ?
  - Which quantity function to use ?

https://www.w3schools.com/sql/sql_having.asp

Outer joins (right and left)

- Inner joins take only values that exist in both tables. This is not always what we need

- Show customers with respective order. All customers shall be included

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

- «Left» and «right» refers to the order in FROM.

- Number of rows in the result ?

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			

- Outer join = inner join + 1 row for hver each witout «match».

General joins

- It is possible to join with other operators than equal (=).

- Find products that had a higher price previosly:

```
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 town in Telemark (point withen a polygon)
  - Find roads that cross Mjøsa (line crossing line)
  - Find properties touched by an extension of the road (polygon Overlapping polygon)

Self joins

- Tables can be coupled with ”themselves”

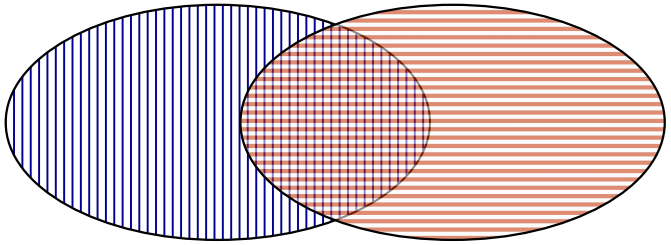
- Find all combinations of products with the same price:

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

- Think like this: DBHS «**makes 2 copies**» of the table Vare, and joins these

Section, union and difference

- A table consists of a number of rows. Vi can use standard operations on quantitites

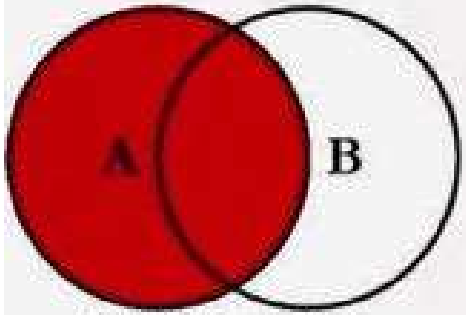


- Suppose Ordre2010 and Ordre2011 contain historical data
- Who ordered something in 2010 and/or 2011?

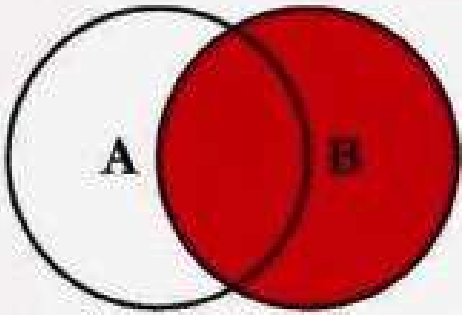
```
SELECT KNr FROM Ordre2010
UNION
SELECT KNr FROM Ordre2011
```

- Some systems support **INTERSECT** (section) and **MINUS/EXCEPT** (differencee).

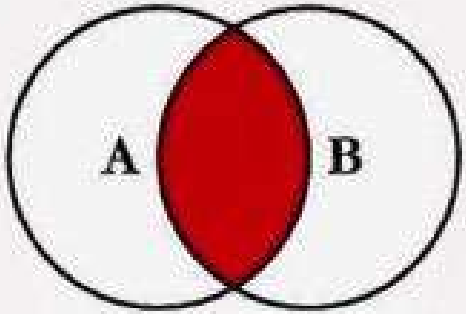
SQL JOINS



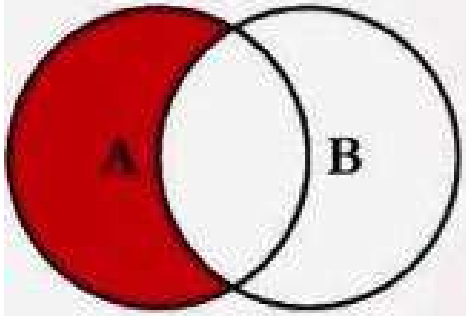

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



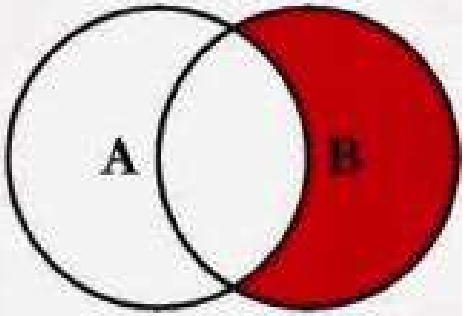

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



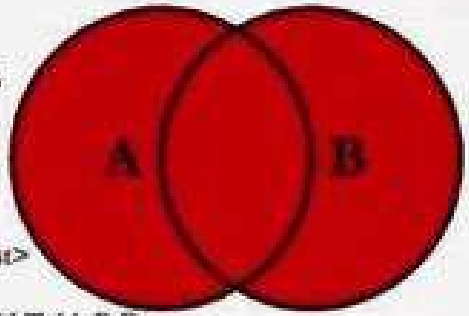

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



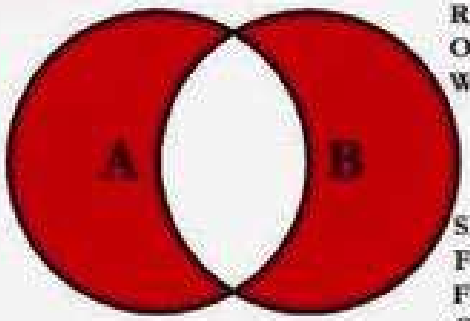

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
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
```

Øvinger

- Oppgave 4
- Oppgave 5