IN2090 – Databaser og datamodellering

03 – `FROM`-klausulen og joins

Leif Harald Karlsen
leifhka@ifi.uio.no



Universitetet i Oslo

Kombinere informasjon fra flere tabeller

- (Enkle) `SELECT`-spørringer har formen:

```
SELECT <kolonner>
  FROM <tabeller>
 WHERE <uttrykk>
```

Kombinere informasjon fra flere tabeller

- (Enkle) `SELECT`-spørringer har formen:

```
SELECT <kolonner>
  FROM <tabeller>
 WHERE <uttrykk>
```

Kombinere informasjon fra flere tabeller

- (Enkle) `SELECT`-spørringer har formen:

```
SELECT <kolonner>
  FROM <tabeller>
 WHERE <uttrykk>
```

- Frem til nå har vi bare sett på spørringer over én og én tabell

Kombinere informasjon fra flere tabeller

- (Enkle) `SELECT`-spørringer har formen:

```
SELECT <kolonner>
  FROM <tabeller>
 WHERE <uttrykk>
```

- Frem til nå har vi bare sett på spørringer over én og én tabell
- Ofte ønsker vi å kombinere informasjon fra ulike tabeller

Kombinere informasjon fra flere tabeller

- (Enkle) `SELECT`-spørringer har formen:

```
SELECT <kolonner>
  FROM <tabeller>
 WHERE <uttrykk>
```

- Frem til nå har vi bare sett på spørringer over én og én tabell
- Ofte ønsker vi å kombinere informasjon fra ulike tabeller
- Dette kan gjøres ved å legge til flere tabeller i `FROM`-klausulen

Spørringer over flere tabeller

Hva skjer dersom vi putter flere tabeller i `FROM`?

To tabeller i `FROM`

```
SELECT *
  FROM products, orders
```

Resultat

Spørringer over flere tabeller

Hva skjer dersom vi putter flere tabeller i `FROM`?

To tabeller i `FROM`

```
SELECT *
  FROM products, orders
```

Resultat

products

ProductID	ProductName	Price
0	TV 50 inch	8999
1	Laptop 2.5GHz	7499

orders

OrderID	OrderedProduct	Customer
0	1	John Mill
1	1	Peter Smith
2	0	Anna Consuma
3	1	Yvonne Potter

Spørringer over flere tabeller

Hva skjer dersom vi putter flere tabeller i `FROM`?

To tabeller i `FROM`

```
SELECT *
  FROM products, orders
```

Resultat

ProductID	ProductName	Price	OrderID	OrderedProduct	Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter

Spørringer over flere tabeller

Hva skjer dersom vi putter flere tabeller i `FROM`?

To tabeller i `FROM`

```
SELECT *
  FROM products, orders
```

Resultat – Fargekodet

products

ProductID	ProductName	Price
0	TV 50 inch	8999
1	Laptop 2.5GHz	7499

orders

OrderID	OrderedProduct	Customer
0	1	John Mill
1	1	Peter Smith
2	0	Anna Consuma
3	1	Yvonne Potter

Spørringer over flere tabeller

Hva skjer dersom vi putter flere tabeller i `FROM`?

To tabeller i `FROM`

```
SELECT *
  FROM products, orders
```

Resultat – Fargekodet

ProductID	ProductName	Price	OrderID	OrderedProduct	Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter

Kryssprodukt

- Med flere tabeller i `FROM`-klausulen får vi alle mulige kombinasjoner av radene fra hver tabell

Kryssprodukt

- Med flere tabeller i `FROM`-klausulen får vi alle mulige kombinasjoner av radene fra hver tabell
- Dette kalles *kryssproduct* eller *Kartesisk produkt*

Kryssprodukt

- Med flere tabeller i `FROM`-klausulen får vi alle mulige kombinasjoner av radene fra hver tabell
- Dette kalles *kryssproduct* eller *Kartesisk produkt*
- Altså, det som var $\times$ i relasjonsalgebraen

Kryssprodukt av to tabeller

Med to tabeller:

T1

C1	C2	C3
x1	x2	x3
y1	y2	y3

T2

D1	D2
a1	a2
b1	b2
c1	c2
d1	d2



```
SELECT * FROM T1, T2
```

C1	C2	C3	D1	D2
x1	x2	x3	a1	a1
x1	x2	x3	b1	b2
x1	x2	x3	c1	c2
x1	x2	x3	d1	d2
y1	y2	y3	a1	a2
y1	y2	y3	b1	b2
y1	y2	y3	c1	c2
y1	y2	y3	d1	d2

Kryssproduktet av tre tabeller

Med tre tabeller:

T1

C1	C2	C3
x1	x2	x3
y1	y2	y3

T2

D1	D2
a1	a2
b1	b2
c1	c2
d1	d2

T3

E1	E2
n1	n2
m1	m2



SELECT * FROM T1, T2, T3

C1	C2	C3	D1	D2	E1	E2
x1	x2	x3	a1	a1	n1	n2
x1	x2	x3	a1	a1	m1	m2
x1	x2	x3	b1	b2	n1	n2
x1	x2	x3	b1	b2	m1	m2
x1	x2	x3	c1	c2	n1	n2
x1	x2	x3	c1	c2	m1	m2
x1	x2	x3	d1	d2	n1	n2
x1	x2	x3	d1	d2	m1	m2
y1	y2	y3	a1	a2	n1	n2
y1	y2	y3	a1	a2	m1	m2
y1	y2	y3	b1	b2	n1	n2
y1	y2	y3	b1	b2	m1	m2
y1	y2	y3	c1	c2	n1	n2
y1	y2	y3	c1	c2	m1	m2
y1	y2	y3	d1	d2	n1	n2
y1	y2	y3	d1	d2	m1	m2

Hvorfor er dette nyttig?

- Kryssproduktet lar oss relatere en hvilken som helst verdi i en kolonne i en tabell til en hvilken som helst verdi i en kolonne i en annen tabell

Hvorfor er dette nyttig?

- Kryssproduktet lar oss relatere en hvilken som helst verdi i en kolonne i en tabell til en hvilken som helst verdi i en kolonne i en annen tabell
- Ved å bruke `WHERE` -og `SELECT`-klausulene kan vi velge ut hva vi ønsker fra denne tabellen av alle mulige kombinasjoner

Eksempel spørring med flere tabeller

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
  FROM products, orders
 WHERE ProductID = OrderedProduct
```

Resultat

Eksempel spørring med flere tabeller

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
  FROM products, orders
 WHERE ProductID = OrderedProduct
```

Resultat

products

ProductID	Name	Price
0	TV 50 inch	8999
1	Laptop 2.5GHz	7499

orders

OrderID	OrderedProduct	Customer
0	1	John Mill
1	1	Peter Smith
2	0	Anna Consuma
3	1	Yvonne Potter

Eksempel spørring med flere tabeller

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
  FROM products, orders
 WHERE ProductID = OrderedProduct
```

Resultat

ProductID	ProductName	Price	OrderID	OrderedProduct	Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter

Eksempel spørring med flere tabeller

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
  FROM products, orders
 WHERE ProductID = OrderedProduct
```

Resultat

ProductID	ProductName	Price	OrderID	OrderedProduct	Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter

Eksempel spørring med flere tabeller

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
  FROM products, orders
 WHERE ProductID = OrderedProduct
```

Resultat

ProductID	ProductName	Price	OrderID	OrderedProduct	Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter

Eksempel spørring med flere tabeller

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
  FROM products, orders
 WHERE ProductID = OrderedProduct
```

Resultat

ProductName	Customer
TV 50 inch	Anna Consuma
Laptop 2.5GHz	John Mill
Laptop 2.5GHz	Peter Smith
Laptop 2.5GHz	Yvonne Potter

Joins

- Spørringer over flere tabeller kalles *joins*,

Joins

- Spørringer over flere tabeller kalles *joins*,
- Mange måter å relatere tabeller på, altså mange mulige joins, f.eks.

Joins

- Spørringer over flere tabeller kalles *joins*,
- Mange måter å relatere tabeller på, altså mange mulige joins, f.eks.
  - equi-join
  - theta-join
  - inner join
  - self join
  - anti join
  - semi join
  - outer join
  - natural join
  - cross join

Joins

- Spørringer over flere tabeller kalles *joins*,
- Mange måter å relatere tabeller på, altså mange mulige joins, f.eks.
  - equi-join
  - theta-join
  - inner join
  - self join
  - anti join
  - semi join
  - outer join
  - natural join
  - cross join
- De er alle bare forskjellige måter å kombinere informasjon fra to eller flere tabeller

Joins

- Spørringer over flere tabeller kalles *joins*,
- Mange måter å relatere tabeller på, altså mange mulige joins, f.eks.
  - equi-join
  - theta-join
  - inner join
  - self join
  - anti join
  - semi join
  - outer join
  - natural join
  - cross join
- De er alle bare forskjellige måter å kombinere informasjon fra to eller flere tabeller
- Oftest (men ikke alltid) interesert i å “joine” på nøkler

Navn på joins

- *Cross join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
```

Navn på joins

- *Cross join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
```

- *Equi-join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
WHERE t1.a = t2.b
```

Navn på joins

- *Cross join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
```

- *Equi-join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
WHERE t1.a = t2.b
```

- *Theta-join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
WHERE <theta>(t1.a,t2.b)
```

  hvor `<theta>` er en eller annen relasjon (f.eks. `<`, `=`, `!=`, `LIKE`) eller mer komplisert uttrykk

Navn på joins

- *Cross join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
```

- *Equi-join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
WHERE t1.a = t2.b
```

- *Theta-join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
WHERE <theta>(t1.a,t2.b)
```

  hvor `<theta>` er en eller annen relasjon (f.eks. `<`, `=`, `!=`, `LIKE`) eller mer komplisert uttrykk
- *Equi-join* er en spesiell type *Theta-join*

Navn på joins

- *Cross join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
```

- *Equi-join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
WHERE t1.a = t2.b
```

- *Theta-join* mellom `t1` og `t2`

```
SELECT * FROM t1, t2
WHERE <theta>(t1.a,t2.b)
```

  hvor `<theta>` er en eller annen relasjon (f.eks. `<`, `=`, `!=`, `LIKE`) eller mer komplisert uttrykk
- *Equi-join* er en spesiell type *Theta-join*
- Alle disse formene for join (og et par til vi skal se etterpå) kalles *indre joins* (eng.: *inner joins*)

Problemer med spørring over flere tabeller

Hvilken kunde har kjøpt hvilket produkt?

Resultat

products

ProductID	Name	Price
0	TV 50 inch	8999
1	Laptop 2.5GHz	7499

orders

OrderID	ProductID	Customer
0	1	John Mill
1	1	Peter Smith
2	0	Anna Consuma
3	1	Yvonne Potter

Problemer med spørring over flere tabeller

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
  FROM products, orders
 WHERE ProductID = ProductID -- ERROR!
```

Resultat

products

ProductID	Name	Price
0	TV 50 inch	8999
1	Laptop 2.5GHz	7499

orders

OrderID	ProductID	Customer
0	1	John Mill
1	1	Peter Smith
2	0	Anna Consuma
3	1	Yvonne Potter

Problemer med spørring over flere tabeller

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
  FROM products, orders
 WHERE ProductID = ProductID -- ERROR!
```

Resultat

ProductID	ProductName	Price	OrderID	ProductID	Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter

Problemer med spørring over flere tabeller

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
  FROM products, orders
 WHERE ProductID = ProductID -- ERROR!
```

Resultat

ProductID	ProductName	Price	OrderID	ProductID	Customer
0	TV 50 inch	8999	0	1	John Mill
0	TV 50 inch	8999	1	1	Peter Smith
0	TV 50 inch	8999	2	0	Anna Consuma
0	TV 50 inch	8999	3	1	Yvonne Potter
1	Laptop 2.5GHz	7499	0	1	John Mill
1	Laptop 2.5GHz	7499	1	1	Peter Smith
1	Laptop 2.5GHz	7499	2	0	Anna Consuma
1	Laptop 2.5GHz	7499	3	1	Yvonne Potter

Like kolonnenavn

- Når vi har flere tabeller i samme spørring kan vi få flere kolonner med likt navn

Like kolonnenavn

- Når vi har flere tabeller i samme spørring kan vi få flere kolonner med likt navn
- For å fikse dette kan vi bruke tabellnavnet som prefiks

Like kolonnenavn

- Når vi har flere tabeller i samme spørring kan vi få flere kolonner med likt navn
- For å fikse dette kan vi bruke tabellnavnet som prefiks
- F.eks. `products.ProductID` og `orders.OrderID`

Like kolonnenavn

- Når vi har flere tabeller i samme spørring kan vi få flere kolonner med likt navn
- For å fikse dette kan vi bruke tabellnavnet som prefiks
- F.eks. `products.ProductID` og `orders.OrderID`

Hvilken kunde har kjøpt hvilket produkt?

Like kolonnenavn

- Når vi har flere tabeller i samme spørring kan vi få flere kolonner med likt navn
- For å fikse dette kan vi bruke tabellnavnet som prefiks
- F.eks. `products.ProductID` og `orders.OrderID`

Hvilken kunde har kjøpt hvilket produkt?

```
SELECT ProductName, Customer
  FROM products, orders
 WHERE products.ProductID = orders.ProductID
```

Navngi tabeller

- Det er ofte nyttig å kunne gi en tabell et nytt navn

Navngi tabeller

- Det er ofte nyttig å kunne gi en tabell et nytt navn
- F.eks. dersom tabellnavnet er langt og gjentas ofte i `WHERE`-klausulen

Navngi tabeller

- Det er ofte nyttig å kunne gi en tabell et nytt navn
- F.eks. dersom tabellnavnet er langt og gjentas ofte i `WHERE`-klausulen
- Eller dersom vi ønsker å gjøre en self-join (mer om dette om litt)

Navngi tabeller

- Det er ofte nyttig å kunne gi en tabell et nytt navn
- F.eks. dersom tabellnavnet er langt og gjentas ofte i `WHERE`-klausulen
- Eller dersom vi ønsker å gjøre en self-join (mer om dette om litt)
- Tabeller kan navngis med `AS`-nøkkelordet

Eksempel: Navngi tabeller

Finn produktnavnet og prisen til hver bestilling (2155 rader)

```
SELECT p.product_name, o.unit_price
  FROM products AS p, order_details AS o
 WHERE p.product_id = o.product_id;
```

Eksempel: Navngi tabeller

Finn produktnavnet og prisen til hver bestilling (2155 rader)

```
SELECT p.product_name, o.unit_price
  FROM products AS p, order_details AS o
 WHERE p.product_id = o.product_id;
```

Kan også droppe `AS`-nøkkelordet, og f.eks. kun skrive

```
FROM products p, order_details o
```

Eksempeler på joins: `Northwind`-databasen

Finn alle unike par av (fulle) navn på kunde og ansatte som har inngått en handel med last (eng.: *freight*) over 500kg(13 rader)

Eksempeler på joins: `Northwind`-databasen

Finn alle unike par av (fulle) navn på kunde og ansatte som har inngått en handel med last (eng.: *freight*) over 500kg(13 rader)

```
SELECT DISTINCT
       c.company_name as kunde,
       e.first_name || ' ' || e.last_name AS ansatt
  FROM orders AS o, customers AS c, employees AS e
 WHERE o.customer_id = c.customer_id AND
       o.employee_id = e.employee_id AND
       o.freight > 500;
```

Relasjonell algebra og SQL

- SQL-spørringene med joins kan også oversettes til relasjonsalgebra

Relasjonell algebra og SQL

- SQL-spørringene med joins kan også oversettes til relasjonsalgebra
- For eksempel kan de enkle SQL-spørringene vi nå har sett oversettes slik:

```
SELECT <columns>
  FROM <t1>, <t2>, ..., <tN>
 WHERE <condition>
```



$$\pi_{\texttt{<columns>}}(\sigma_{\texttt{<condition>}}(\texttt{<t1>} \times \texttt{<t2>} \times \cdots \times \texttt{<tN>}))$$

Egen notasjon for joins

- SQL har en egen notasjon for joins

Egen notasjon for joins

- SQL har en egen notasjon for joins
- For den typen joins vi har gjort hittil har man `INNER JOIN` -og `ON`-nøkkelordene

Egen notasjon for joins

- SQL har en egen notasjon for joins
- For den typen joins vi har gjort hittil har man `INNER JOIN` -og `ON`-nøkkelordene
- Fremfor å skrive:

```
SELECT product_name
  FROM products AS p, orders AS o
 WHERE p.product_id = o.product_id AND
       o.unit_price > 7000
```

Egen notasjon for joins

- SQL har en egen notasjon for joins
- For den typen joins vi har gjort hittil har man `INNER JOIN` -og `ON`-nøkkelordene
- Fremfor å skrive:

```
SELECT product_name
  FROM products AS p, orders AS o
 WHERE p.product_id = o.product_id AND
       o.unit_price > 7000
```

- kan man skrive

```
SELECT p.product_name
  FROM products AS p INNER JOIN order_details AS o
       ON (p.product_id = o.product_id)
 WHERE o.unit_price > 7000
```

Egen notasjon for joins

- SQL har en egen notasjon for joins
- For den typen joins vi har gjort hittil har man `INNER JOIN` -og `ON`-nøkkelordene
- Fremfor å skrive:

```
SELECT product_name
  FROM products AS p, orders AS o
 WHERE p.product_id = o.product_id AND
       o.unit_price > 7000
```

- kan man skrive

```
SELECT p.product_name
  FROM products AS p INNER JOIN order_details AS o
       ON (p.product_id = o.product_id)
 WHERE o.unit_price > 7000
```

- De to spørringene er ekvivalente

Egen notasjon for joins

- SQL har en egen notasjon for joins
- For den typen joins vi har gjort hittil har man `INNER JOIN` -og `ON`-nøkkelordene
- Fremfor å skrive:

```
SELECT product_name
  FROM products AS p, orders AS o
 WHERE p.product_id = o.product_id AND
       o.unit_price > 7000
```

- kan man skrive

```
SELECT p.product_name
  FROM products AS p INNER JOIN order_details AS o
       ON (p.product_id = o.product_id)
 WHERE o.unit_price > 7000
```

- De to spørringene er ekvivalente
- Øverste kalles implisitt join, nederste kalles eksplisitt join

Egen notasjon for joins

- SQL har en egen notasjon for joins
- For den typen joins vi har gjort hittil har man `INNER JOIN` -og `ON`-nøkkelordene
- Fremfor å skrive:

```
SELECT product_name
  FROM products AS p, orders AS o
 WHERE p.product_id = o.product_id AND
       o.unit_price > 7000
```

- kan man skrive

```
SELECT p.product_name
  FROM products AS p INNER JOIN order_details AS o
       ON (p.product_id = o.product_id)
 WHERE o.unit_price > 7000
```

- De to spørringene er ekvivalente
- Øverste kalles implisitt join, nederste kalles eksplisitt join
- Skal senere se at enkelte joins ikke kan skrives på den øverste formen

Egen notasjon for joins

- SQL har en egen notasjon for joins
- For den typen joins vi har gjort hittil har man `INNER JOIN` -og `ON`-nøkkelordene
- Fremfor å skrive:

```
SELECT product_name
  FROM products AS p, orders AS o
 WHERE p.product_id = o.product_id AND
       o.unit_price > 7000
```

- kan man skrive

```
SELECT p.product_name
  FROM products AS p INNER JOIN order_details AS o
       ON (p.product_id = o.product_id)
 WHERE o.unit_price > 7000
```

- De to spørringene er ekvivalente
- Øverste kalles implisitt join, nederste kalles eksplisitt join
- Skal senere se at enkelte joins ikke kan skrives på den øverste formen
- Den nederste formen gjør det lettere å se hvordan tabellene er “joinet”

Flere join-eksempler (`Northwind`-DB)

Finn ut hvilke drikkevarer som er kjøpt og av hvem [404 rader]

Flere join-eksempler (Northwind-DB)

Finn ut hvilke drikkevarer som er kjøpt og av hvem [404 rader]

```
SELECT p.product_name, u.company_name
FROM categories AS c
     INNER JOIN products AS p ON (c.category_id = p.category_id)
     INNER JOIN order_details AS d ON (p.product_id = d.product_id)
     INNER JOIN orders AS o ON (d.order_id = o.order_id)
     INNER JOIN customers AS u ON (u.customer_id = o.customer_id)
WHERE c.category_name = 'Beverages';
```

Self-joins

- Av og til ønsker man å kombinere informasjon fra rader i samme tabell

Self-joins

- Av og til ønsker man å kombinere informasjon fra rader i samme tabell
- Dette kalles en *self-join*

Self-joins

- Av og til ønsker man å kombinere informasjon fra rader i samme tabell
- Dette kalles en *self-join*
- Dette gjøres ved å bruke den samme tabellen to eller flere ganger i `FROM`-klausulen

Self-joins

- Av og til ønsker man å kombinere informasjon fra rader i samme tabell
- Dette kalles en *self-join*
- Dette gjøres ved å bruke den samme tabellen to eller flere ganger i `FROM`-klausulen
- Må da gi dem forskjellige navn

Self-join-eksempel

Finn navn og pris på alle produkter som er dyrere enn produktet Laptop 2.5GHz?

```
SELECT P2.Name, P2.Price
  FROM Product AS P1, Product AS P2
 WHERE P1.Name = 'Laptop 2.5GHz' AND P1.Price < P2.Price
```

Resultat

Self-join-eksempel

Finn navn og pris på alle produkter som er dyrere enn produktet Laptop 2.5GHz?

```
SELECT P2.Name, P2.Price
  FROM Product AS P1, Product AS P2
 WHERE P1.Name = 'Laptop 2.5GHz' AND P1.Price < P2.Price
```

Resultat

P1

ProductID)	Name	Brand	Price
0	TV 50 inch	Sony	8999
1	Laptop 2.5GHz	Lenovo	7499
2	Laptop 8GB RAM	HP	6999
3	Speaker 500	Bose	4999
4	TV 48 inch	Panasonic	11999
5	Phone S6	IPhone	5195

P2

ProductID	Name	Brand	Price
0	TV 50 inch	Sony	8999
1	Laptop 2.5GHz	Lenovo	7499
2	Laptop 8GB RAM	HP	6999
3	Speaker 500	Bose	4999
4	TV 48 inch	Panasonic	11999
5	Phone S6	IPhone	5195

Self-join-eksempel

Finn navn og pris på alle produkter som er dyrere enn produktet Laptop 2.5GHz?

```
SELECT P2.Name, P2.Price
  FROM Product AS P1, Product AS P2
 WHERE P1.Name = 'Laptop 2.5GHz' AND P1.Price < P2.Price
```

Resultat

P1.ProductID	P1.Name	P1.Brand	P1.Price	P2.ProductID	P2.Name	P2.Brand	P2.Price
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	TV 50 inch	Sony	8999	5	Phone S6	IPhone	5195
1	Laptop 2.5GHz	Lenovo	7499	0	TV 50 inch	Sony	8999
1	Laptop 2.5GHz	Lenovo	7499	1	Laptop 2.5GHz	Lenovo	7499
1	Laptop 2.5GHz	Lenovo	7499	2	Laptop 8GB RAM	HP	6999
1	Laptop 2.5GHz	Lenovo	7499	3	Speaker 500	Bose	4999
1	Laptop 2.5GHz	Lenovo	7499	4	TV 48 inch	Panasonic	11999
1	Laptop 2.5GHz	Lenovo	7499	5	Phone S6	IPhone	5195
2	Laptop 8GB RAM	HP	6999	0	TV 50 inch	Sony	8999
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Self-join-eksempel

Finn navn og pris på alle produkter som er dyrere enn produktet Laptop 2.5GHz?

```
SELECT P2.Name, P2.Price
  FROM Product AS P1, Product AS P2
 WHERE P1.Name = 'Laptop 2.5GHz' AND P1.Price < P2.Price
```

Resultat

P1.ProductID	P1.Name	P1.Brand	P1.Price	P2.ProductID	P2.Name	P2.Brand	P2.Price
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
0	TV 50 inch	Sony	8999	5	Phone S6	IPhone	5195
1	**Laptop 2.5GHz**	**Lenovo**	**7499**	**0**	**TV 50 inch**	**Sony**	**8999**
1	Laptop 2.5GHz	Lenovo	7499	1	Laptop 2.5GHz	Lenovo	7499
1	Laptop 2.5GHz	Lenovo	7499	2	Laptop 8GB RAM	HP	6999
1	Laptop 2.5GHz	Lenovo	7499	3	Speaker 500	Bose	4999
1	**Laptop 2.5GHz**	**Lenovo**	**7499**	**4**	**TV 48 inch**	**Panasonic**	**11999**
1	Laptop 2.5GHz	Lenovo	7499	5	Phone S6	IPhone	5195
2	Laptop 8GB RAM	HP	6999	0	TV 50 inch	Sony	8999
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Self-join-eksempel

Finn navn og pris på alle produkter som er dyrere enn produktet Laptop 2.5GHz?

```
SELECT P2.Name, P2.Price
  FROM Product AS P1, Product AS P2
 WHERE P1.Name = 'Laptop 2.5GHz' AND P1.Price < P2.Price
```

Resultat

P1.ProductID	P1.Name	P1.Brand	P1.Price	P2.ProductID	**P2.Name**	P2.Brand	**P2.Price**
0	TV 50 inch	Sony	8999	5	Phone S6	IPhone	5195
1	Laptop 2.5GHz	Lenovo	7499	0	**TV 50 inch**	Sony	**8999**
1	Laptop 2.5GHz	Lenovo	7499	1	Laptop 2.5GHz	Lenovo	7499
1	Laptop 2.5GHz	Lenovo	7499	2	Laptop 8GB RAM	HP	6999
1	Laptop 2.5GHz	Lenovo	7499	3	Speaker 500	Bose	4999
1	Laptop 2.5GHz	Lenovo	7499	4	**TV 48 inch**	Panasonic	**11999**
1	Laptop 2.5GHz	Lenovo	7499	5	Phone S6	IPhone	5195
2	Laptop 8GB RAM	HP	6999	0	TV 50 inch	Sony	8999

Self-join-eksempel

Finn navn og pris på alle produkter som er dyrere enn produktet Laptop 2.5GHz?

```
SELECT P2.Name, P2.Price
  FROM Product AS P1, Product AS P2
 WHERE P1.Name = 'Laptop 2.5GHz' AND P1.Price < P2.Price
```

Resultat

P2.Name	P2.Price
TV 50 inch	8999
TV 48 inch	11999

Naturlig Join

- Vi joiner ofte på de kolonnene som har likt navn

Naturlig Join

- Vi joiner ofte på de kolonnene som har likt navn
- F.eks. `categories.category_id` med `products.category_id`

Naturlig Join

- Vi joiner ofte på de kolonnene som har likt navn
- F.eks. `categories.category_id` med `products.category_id`
- Dette kan gjøres enklere med *naturlig join*

Naturlig Join

- Vi joiner ofte på de kolonnene som har likt navn
- F.eks. `categories.category_id` med `products.category_id`
- Dette kan gjøres enklere med *naturlig join*
- Naturlig join joiner (med likhet) automatisk på alle kolonner med likt navn

Naturlig Join

- Vi joiner ofte på de kolonnene som har likt navn
- F.eks. `categories.category_id` med `products.category_id`
- Dette kan gjøres enklere med *naturlig join*
- Naturlig join joiner (med likhet) automatisk på alle kolonner med likt navn
- I tillegg projiserer den vekk de dupliserte kolonnene

Naturlig Join

- Vi joiner ofte på de kolonnene som har likt navn
- F.eks. `categories.category_id` med `products.category_id`
- Dette kan gjøres enklere med *naturlig join*
- Naturlig join joiner (med likhet) automatisk på alle kolonner med likt navn
- I tillegg projiserer den vekk de dupliserte kolonnene
- Trenger derfor aldri gi tabellene navn (i resultatet av en naturlig join vil det aldri finnes kolonner med likt navn)

Naturlig Join

- Vi joiner ofte på de kolonnene som har likt navn
- F.eks. `categories.category_id` med `products.category_id`
- Dette kan gjøres enklere med *naturlig join*
- Naturlig join joiner (med likhet) automatisk på alle kolonner med likt navn
- I tillegg projiserer den vekk de dupliserte kolonnene
- Trenger derfor aldri gi tabellene navn (i resultatet av en naturlig join vil det aldri finnes kolonner med likt navn)
- Merk: Må være sikker på at vi ønsker å joine på ALLE kolonnene med likt navn!

Naturlig Join: Eksempel

Finn navnet på alle drikkevarer [12 rader]

Naturlig Join: Eksempel

Finn navnet på alle drikkevarer [12 rader]

```
SELECT product_name
  FROM categories NATURAL JOIN products
 WHERE category_name = 'Beverages';
```

Enkle `SELECT`-spørringer i et nøtteskall

- `FROM`-klausulen sier hvilke tabell(er) som skal brukes for å besvare spørringen og joiner dem sammen

Enkle `SELECT`-spørringer i et nøtteskall

- `FROM`-klausulen sier hvilke tabell(er) som skal brukes for å besvare spørringen og joiner dem sammen
- `WHERE`-klausulen velger ut hvilke rader som skal være med i svaret
  - Kolonnenavn brukes som variable som instansieres med radenes verdier
  - Kan sammenlikne kolonner og verdier med f.eks. `=`, `!=`, `<`, `<=`, `LIKE`
  - Bruker `AND`, `OR` og `NOT` på uttrykk
  - Evaluerer til enten `TRUE`, `FALSE` eller `NULL` for hver rad
  - Kun de som evaluerer til `TRUE` blir med i svaret

Enkle `SELECT`-spørringer i et nøtteskall

- `FROM`-klausulen sier hvilke tabell(er) som skal brukes for å besvare spørringen og joiner dem sammen
- `WHERE`-klausulen velger ut hvilke rader som skal være med i svaret
  - Kolonnenavn brukes som variable som instansieres med radenes verdier
  - Kan sammenlikne kolonner og verdier med f.eks. `=`, `!=`, `<`, `<=`, `LIKE`
  - Bruker `AND`, `OR` og `NOT` på uttrykk
  - Evaluerer til enten `TRUE`, `FALSE` eller `NULL` for hver rad
  - Kun de som evaluerer til `TRUE` blir med i svaret
- `SELECT`-klausulen velger hvilke verdier/kolonner som skal være med i svaret
  - Kan også endre rekkefølgen på kolonner, bruke dem i uttrykk, osv.
  - Bruk `*` for å velge alle kolonnene

Enkle `SELECT`-spørringer i et nøtteskall

- `FROM`-klausulen sier hvilke tabell(er) som skal brukes for å besvare spørringen og joiner dem sammen
- `WHERE`-klausulen velger ut hvilke rader som skal være med i svaret
  - Kolonnenavn brukes som variable som instansieres med radenes verdier
  - Kan sammenlikne kolonner og verdier med f.eks. `=`, `!=`, `<`, `<=`, `LIKE`
  - Bruker `AND`, `OR` og `NOT` på uttrykk
  - Evaluerer til enten `TRUE`, `FALSE` eller `NULL` for hver rad
  - Kun de som evaluerer til `TRUE` blir med i svaret
- `SELECT`-klausulen velger hvilke verdier/kolonner som skal være med i svaret
  - Kan også endre rekkefølgen på kolonner, bruke dem i uttrykk, osv.
  - Bruk `*` for å velge alle kolonnene
- SQL bryr seg ikke om mellomrom og linjeskift, eller store og små bokstaver

Enkle `SELECT`-spørringer i et nøtteskall

- `FROM`-klausulen sier hvilke tabell(er) som skal brukes for å besvare spørringen og joiner dem sammen
- `WHERE`-klausulen velger ut hvilke rader som skal være med i svaret
  - Kolonnenavn brukes som variable som instansieres med radenes verdier
  - Kan sammenlikne kolonner og verdier med f.eks. `=`, `!=`, `<`, `<=`, `LIKE`
  - Bruker `AND`, `OR` og `NOT` på uttrykk
  - Evaluerer til enten `TRUE`, `FALSE` eller `NULL` for hver rad
  - Kun de som evaluerer til `TRUE` blir med i svaret
- `SELECT`-klausulen velger hvilke verdier/kolonner som skal være med i svaret
  - Kan også endre rekkefølgen på kolonner, bruke dem i uttrykk, osv.
  - Bruk `*` for å velge alle kolonnene
- SQL bryr seg ikke om mellomrom og linjeskift, eller store og små bokstaver

Mye mer kan gjøres i hver klausul og det finnes flere klausuler, mer om dette senere i kurset!

Takk for nå!

Neste video vil se på nestede spørringer.