IN2090 – Databaser og datamodellering

03 – Intro til SQL

Leif Harald Karlsen
leifhka@ifi.uio.no



Universitetet i Oslo

SQL: Structured Query Language

- SQL er et spørrespråk for relasjonelle databaser

SQL: Structured Query Language

- SQL er et spørrespråk for relasjonelle databaser
- Det mest brukte spørrespråket for slike databaser

SQL: Structured Query Language

- SQL er et spørrespråk for relasjonelle databaser
- Det mest brukte spørrespråket for slike databaser
- Brukes for å formulere spørringer, altså spørsmål, til en database

SQL: Structured Query Language

- SQL er et spørrespråk for relasjonelle databaser
- Det mest brukte spørrespråket for slike databaser
- Brukes for å formulere spørringer, altså spørsmål, til en database
- SQL kan også brukes for å manipulere en database

SQL: Structured Query Language

- SQL er et spørrespråk for relasjonelle databaser
- Det mest brukte spørrespråket for slike databaser
- Brukes for å formulere spørringer, altså spørsmål, til en database
- SQL kan også brukes for å manipulere en database
  - Lage tabeller

SQL: Structured Query Language

- SQL er et spørrespråk for relasjonelle databaser
- Det mest brukte spørrespråket for slike databaser
- Brukes for å formulere spørringer, altså spørsmål, til en database
- SQL kan også brukes for å manipulere en database
  - Lage tabeller
  - sette inn data

SQL: Structured Query Language

- SQL er et spørrespråk for relasjonelle databaser
- Det mest brukte spørrespråket for slike databaser
- Brukes for å formulere spørringer, altså spørsmål, til en database
- SQL kan også brukes for å manipulere en database
  - Lage tabeller
  - sette inn data
  - slette data

SQL: Structured Query Language

- SQL er et spørrespråk for relasjonelle databaser
- Det mest brukte spørrespråket for slike databaser
- Brukes for å formulere spørringer, altså spørsmål, til en database
- SQL kan også brukes for å manipulere en database
  - Lage tabeller
  - sette inn data
  - slette data
  - ...

SQL: Structured Query Language

- SQL er et spørrespråk for relasjonelle databaser
- Det mest brukte spørrespråket for slike databaser
- Brukes for å formulere spørringer, altså spørsmål, til en database
- SQL kan også brukes for å manipulere en database
  - Lage tabeller
  - sette inn data
  - slette data
  - ...
- Ble laget i 1974, men ble først standardisert i 1986

Imperativ vs. deklarativ

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma,

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma, kan du gå 2 meter til venstre,

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma, kan du gå 2 meter til venstre, strekke ut armen din,

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma, kan du gå 2 meter til venstre, strekke ut armen din, trekke dørhåndtaket ned og mot deg.

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma, kan du gå 2 meter til venstre, strekke ut armen din, trekke dørhåndtaket ned og mot deg. Så gå gjennom døren,

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma, kan du gå 2 meter til venstre, strekke ut armen din, trekke dørhåndtaket ned og mot deg. Så gå gjennom døren, snu deg til venstre, gå 4 meter frem, snu deg til høyre, ...,

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma, kan du gå 2 meter til venstre, strekke ut armen din, trekke dørhåndtaket ned og mot deg. Så gå gjennom døren, snu deg til venstre, gå 4 meter frem, snu deg til høyre, ..., og sette glasset ned på bordet og slippe det.”

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma, kan du gå 2 meter til venstre, strekke ut armen din, trekke dørhåndtaket ned og mot deg. Så gå gjennom døren, snu deg til venstre, gå 4 meter frem, snu deg til høyre, ..., og sette glasset ned på bordet og slippe det.”
- Deklarativt:

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma, kan du gå 2 meter til venstre, strekke ut armen din, trekke dørhåndtaket ned og mot deg. Så gå gjennom døren, snu deg til venstre, gå 4 meter frem, snu deg til høyre, ..., og sette glasset ned på bordet og slippe det.”
- Deklarativt:
  - “Hei mamma,

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma, kan du gå 2 meter til venstre, strekke ut armen din, trekke dørhåndtaket ned og mot deg. Så gå gjennom døren, snu deg til venstre, gå 4 meter frem, snu deg til høyre, ..., og sette glasset ned på bordet og slippe det.”
- Deklarativt:
  - “Hei mamma, vann er flytende H_2O

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma, kan du gå 2 meter til venstre, strekke ut armen din, trekke dørhåndtaket ned og mot deg. Så gå gjennom døren, snu deg til venstre, gå 4 meter frem, snu deg til høyre, ..., og sette glasset ned på bordet og slippe det.”
- Deklarativt:
  - “Hei mamma, vann er flytende H_2O og glass er smeltet sand formet på en slik måte at dets innhold ikke renner ut.

Imperativ vs. deklarativ

La oss si at du er en tørst og din mor er i nærheten. To måter å få henne til å hente vann på:

- Imperativ:
  - “Hei mamma, kan du gå 2 meter til venstre, strekke ut armen din, trekke dørhåndtaket ned og mot deg. Så gå gjennom døren, snu deg til venstre, gå 4 meter frem, snu deg til høyre, ..., og sette glasset ned på bordet og slippe det.”
- Deklarativt:
  - “Hei mamma, vann er flytende H_2O og glass er smeltet sand formet på en slik måte at dets innhold ikke renner ut. Kan du hente meg et glass med vann, er du snill?

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*
- F.eks.:

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*
- F.eks.:
  - "Sett verdien av *x* til 2" ($\texttt{x = 2}$)

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*
- F.eks.:
  - "Sett verdien av *x* til 2" (`x = 2`)
  - "Legg tallet 5 til listen *lst*" (`lst.add(5)`)

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*
- F.eks.:
  - "Sett verdien av *x* til 2" (`x = 2`)
  - "Legg tallet 5 til listen *lst*" (`lst.add(5)`)
  - "For hvert element i listen *L* print verdien av elementet" (`for e in L: print(e)`)

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*
- F.eks.:
  - "Sett verdien av *x* til 2" (`x = 2`)
  - "Legg tallet 5 til listen *lst*" (`lst.add(5)`)
  - "For hvert element i listen *L* print verdien av elementet" (`for e in L: print(e)`)

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*
- F.eks.:
  - "Sett verdien av *x* til 2" (`x = 2`)
  - "Legg tallet 5 til listen *lst*" (`lst.add(5)`)
  - "For hvert element i listen *L* print verdien av elementet" (`for e in L: print(e)`)

- Et spørrespråk er et presist språk for å uttrykke *spørsmål til en database*

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*
- F.eks.:
  - "Sett verdien av *x* til 2" (`x = 2`)
  - "Legg tallet 5 til listen *lst*" (`lst.add(5)`)
  - "For hvert element i listen *L* print verdien av elementet" (`for e in L: print(e)`)

- Et spørrespråk er et presist språk for å uttrykke *spørsmål til en database*
- Slike spørsmål kalles ofte en *spørring* (eng.: *query*)

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*
- F.eks.:
  - "Sett verdien av *x* til 2" (`x = 2`)
  - "Legg tallet 5 til listen *lst*" (`lst.add(5)`)
  - "For hvert element i listen *L* print verdien av elementet" (`for e in L: print(e)`)

- Et spørrespråk er et presist språk for å uttrykke *spørsmål til en database*
- Slike spørsmål kalles ofte en *spørring* (eng.: *query*)
- SQL er deklartivt, f.eks.:

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*
- F.eks.:
  - "Sett verdien av *x* til 2" (`x = 2`)
  - "Legg tallet 5 til listen *lst*" (`lst.add(5)`)
  - "For hvert element i listen *L* print verdien av elementet" (`for e in L: print(e)`)

- Et spørrespråk er et presist språk for å uttrykke *spørsmål til en database*
- Slike spørsmål kalles ofte en *spørring* (eng.: *query*)
- SQL er deklartivt, f.eks.:
  - "Finn alle elementer som har et navn som starter på 'P'?"

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*
- F.eks.:
  - "Sett verdien av *x* til 2" (`x = 2`)
  - "Legg tallet 5 til listen *lst*" (`lst.add(5)`)
  - "For hvert element i listen *L* print verdien av elementet" (`for e in L: print(e)`)

- Et spørrespråk er et presist språk for å uttrykke *spørsmål til en database*
- Slike spørsmål kalles ofte en *spørring* (eng.: *query*)
- SQL er deklartivt, f.eks.:
  - "Finn alle elementer som har et navn som starter på 'P'?"
  - "La 'Forelder' være alle elementer som har en 'harBarn'-relasjon til et element"

Python/Java vs. SQL

- Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å utrrykke *sekvenser av instruksjoner for en datamaskin*
- F.eks.:
  - "Sett verdien av *x* til 2" (`x = 2`)
  - "Legg tallet 5 til listen *lst*" (`lst.add(5)`)
  - "For hvert element i listen *L* print verdien av elementet" (`for e in L: print(e)`)

- Et spørrespråk er et presist språk for å uttrykke *spørsmål til en database*
- Slike spørsmål kalles ofte en *spørring* (eng.: *query*)
- SQL er deklartivt, f.eks.:
  - "Finn alle elementer som har et navn som starter på 'P'?"
  - "La 'Forelder' være alle elementer som har en 'harBarn'-relasjon til et element"
  - "Finn antall ansatte som har en sjef som tjener mer enn 1000000 KR?"

Typer SQL-spørringer

Det første ordet i en spørring sier hva spørringen gjør:

Typer SQL-spørringer

Det første ordet i en spørring sier hva spørringen gjør:

SELECT henter informasjon (svarer på et spørsmål)

Typer SQL-spørringer

Det første ordet i en spørring sier hva spørringen gjør:

SELECT henter informasjon (svarer på et spørsmål)

CREATE lager noe (f.eks. en ny tabell)

Typer SQL-spørringer

Det første ordet i en spørring sier hva spørringen gjør:

- **SELECT** henter informasjon (svarer på et spørsmål)
- **CREATE** lager noe (f.eks. en ny tabell)
- **INSERT** setter inn rader i en tabell

Typer SQL-spørringer

Det første ordet i en spørring sier hva spørringen gjør:

SELECT henter informasjon (svarer på et spørsmål)

CREATE lager noe (f.eks. en ny tabell)

INSERT setter inn rader i en tabell

UPDATE oppdaterer data i en tabell

Typer SQL-spørringer

Det første ordet i en spørring sier hva spørringen gjør:

- **SELECT** henter informasjon (svarer på et spørsmål)
- **CREATE** lager noe (f.eks. en ny tabell)
- **INSERT** setter inn rader i en tabell
- **UPDATE** oppdaterer data i en tabell
- **DELETE** sletter rader fra en tabell

Typer SQL-spørringer

Det første ordet i en spørring sier hva spørringen gjør:

SELECT	henter informasjon (svarer på et spørsmål)
CREATE	lager noe (f.eks. en ny tabell)
INSERT	setter inn rader i en tabell
UPDATE	oppdaterer data i en tabell
DELETE	sletter rader fra en tabell
DROP	sletter en hel ting (f.eks. en hel tabell)

Typer SQL-spørringer

Det første ordet i en spørring sier hva spørringen gjør:

SELECT	henter informasjon (svarer på et spørsmål)
CREATE	lager noe (f.eks. en ny tabell)
INSERT	setter inn rader i en tabell
UPDATE	oppdaterer data i en tabell
DELETE	sletter rader fra en tabell
DROP	sletter en hel ting (f.eks. en hel tabell)

De første SQL-forelesningene omhandler kun `SELECT`.

SELECT-spørringer

- (Enkle) SELECT-spørringer har formen:

```
SELECT <kolonner>
  FROM <tabeller>
```

SELECT-spørringer

- (Enkle) SELECT-spørringer har formen:

```
SELECT <kolonner>
  FROM <tabeller>
```

- hvor <kolonner> er en liste med kolonne-navn,
- og <tabeller> er en liste med tabell-navn

SELECT-spørringer

- (Enkle) SELECT-spørringer har formen:

```
SELECT <kolonner>
  FROM <tabeller>
```

- hvor <kolonner> er en liste med kolonne-navn,
- og <tabeller> er en liste med tabell-navn

Resultatet av en SELECT-spørring er alltid en ny tabell, som består av

- kolonnene i <kolonner>
- basert på radene i tabellene i <tabeller>

Velge en enkelt kolonne

Spørring som henter ut alle navn i `Customer`-tabellen

```
SELECT Name
  FROM Customer
```

Resultat

Velge en enkelt kolonne

Spørring som henter ut alle navn i `Customer`-tabellen

```
SELECT Name
  FROM Customer
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

Velge en enkelt kolonne

Spørring som henter ut alle navn i `Customer`-tabellen

```
SELECT Name
  FROM Customer
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

Velge en enkelt kolonne

Spørring som henter ut alle navn i `Customer`-tabellen

```
SELECT Name
  FROM Customer
```

Resultat

Name
Anna Consuma
Peter Young
Carla Smith
Sam Penny
John Mill
Yvonne Potter

Velge flere kolonner

Spørring som henter alle navn -og fødselsdato-par i `Customer`-tabellen

```
SELECT Name, Birthdate
  FROM Customer
```

Resultat

Velge flere kolonner

Spørring som henter alle navn -og fødselsdato-par i `Customer`-tabellen

```
SELECT Name, Birthdate
  FROM Customer
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

Velge flere kolonner

Spørring som henter alle navn -og fødselsdato-par i `Customer`-tabellen

```
SELECT Name, Birthdate
  FROM Customer
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

Velge flere kolonner

Spørring som henter alle navn -og fødselsdato-par i `Customer`-tabellen

```
SELECT Name, Birthdate
  FROM Customer
```

Resultat

Name	Birthdate
Anna Consuma	1978-10-09
Peter Young	2009-03-01
Carla Smith	1986-06-14
Sam Penny	1961-01-09
John Mill	1989-11-16
Yvonne Potter	1971-04-12

Velge alle kolonner

Spørring som henter alle kolonnene i `Customer`-tabellen

```
SELECT *
  FROM Customer
```

Resultat

Velge alle kolonner

Spørring som henter alle kolonnene i `Customer`-tabellen

```
SELECT *
  FROM Customer
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

Velge alle kolonner

Spørring som henter alle kolonnene i `Customer`-tabellen

```
SELECT *
  FROM Customer
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

Legge inn filter via `WHERE`

- Ofte er vi kun interessert i spesifikke rader

Legge inn filter via `WHERE`

- Ofte er vi kun interessert i spesifikke rader
- Vi kan da bruke en `WHERE`-klausul for å velge ut de radene vi ønsker

Legge inn filter via `WHERE`

- Ofte er vi kun interessert i spesifikke rader
- Vi kan da bruke en `WHERE`-klausul for å velge ut de radene vi ønsker
- SQL-spørringer har da formen

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

Legge inn filter via `WHERE`

- Ofte er vi kun interessert i spesifikke rader
- Vi kan da bruke en `WHERE`-klausul for å velge ut de radene vi ønsker
- SQL-spørringer har da formen

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

- `<betingelse>` er et uttrykk over kolonnenavnene fra tabellene

Legge inn filter via `WHERE`

- Ofte er vi kun interessert i spesifikke rader
- Vi kan da bruke en `WHERE`-klausul for å velge ut de radene vi ønsker
- SQL-spørringer har da formen

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

- `<betingelse>` er et uttrykk over kolonnenavnene fra tabellene
- For hver rad evalueres dette uttrykket til sant eller usant

Legge inn filter via `WHERE`

- Ofte er vi kun interessert i spesifikke rader
- Vi kan da bruke en `WHERE`-klausul for å velge ut de radene vi ønsker
- SQL-spørringer har da formen

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

- `<betingelse>` er et uttrykk over kolonnenavnene fra tabellene
- For hver rad evalueres dette uttrykket til sant eller usant
- Resultatet er det samme som før, men begrenset til kun de radene som gjør `<betingelse>` sann

Velge ut spesifikke rader

Spørring som gir fødselsdatoen til kunden ved navn John Mill

```
SELECT Birthdate
  FROM Customer
 WHERE Name = 'John Mill'
```

Resultat

Velge ut spesifikke rader

Spørring som gir fødselsdatoen til kunden ved navn John Mill

```
SELECT Birthdate
  FROM Customer
 WHERE Name = 'John Mill'
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

Velge ut spesifikke rader

Spørring som gir fødselsdatoen til kunden ved navn John Mill

```
SELECT Birthdate
  FROM Customer
 WHERE Name = 'John Mill'
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

Velge ut spesifikke rader

Spørring som gir fødselsdatoen til kunden ved navn John Mill

```
SELECT Birthdate
  FROM Customer
 WHERE Name = 'John Mill'
```

Resultat

CustomerID	Name	Birthdate	NrProducts
0	Anna Consuma	1978-10-09	19
1	Peter Young	2009-03-01	1
2	Carla Smith	1986-06-14	8
3	Sam Penny	1961-01-09	14
4	John Mill	1989-11-16	8
5	Yvonne Potter	1971-04-12	6

Velge ut spesifikke rader

Spørring som gir fødselsdatoen til kunden ved navn John Mill

```
SELECT Birthdate
  FROM Customer
 WHERE Name = 'John Mill'
```

Resultat

Birthdate
1989-11-16

SQL og relasjonsalgebra: Oversettelse

- En SQL spørring og relasjonsalgebraen har mye til felles

SQL og relasjonsalgebra: Oversettelse

- En SQL spørring og relasjonsalgebraen har mye til felles
- En SQL-spørring kan oversettes til relasjonsalgebra

SQL og relasjonsalgebra: Oversettelse

- En SQL spørring og relasjonsalgebraen har mye til felles
- En SQL-spørring kan oversettes til relasjonsalgebra
- For eksempel kan de enkle SQL-spørringene vi nå har sett oversettes slik:

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

$\Rightarrow$

$\pi_{\text{<kolonner>}}(\sigma_{\text{<uttrykk>}}(\text{<tabeller>}))$

SQL og relasjonsalgebra: Forskjeller

- Men i den relasjonsmodellen er relasjonene mengder av tupler
- I en mengde kan et element kun forekomme én gang, f.eks.:

Person

Navn	Alder
Per	13
Ola	24
Mari	13
Karl	25
Ida	25

$\pi_{\text{Alder}}(\text{Person})$

Alder
13
24
25

SQL og relasjonsalgebra: Forskjeller

- Men i den relasjonsmodellen er relasjonene mengder av tupler
- I en mengde kan et element kun forekomme én gang, f.eks.:

Person

Navn	Alder
Per	13
Ola	24
Mari	13
Karl	25
Ida	25

$\pi_{\text{Alder}}(\text{Person})$

Alder
13
24
25

- I SQL har vi tabeller i stedet for relasjoner (multi-mengder av tupler):

```
SELECT Alder
  FROM Person
```

$\Rightarrow$

Alder
13
24
13
25
25

SQL og relasjonsalgebra: Forskjeller

- Men i den relasjonsmodellen er relasjonene mengder av tupler
- I en mengde kan et element kun forekomme én gang, f.eks.:

Person

Navn	Alder
Per	13
Ola	24
Mari	13
Karl	25
Ida	25

$\pi_{\text{Alder}}(\text{Person})$

Alder
13
24
25

- I SQL har vi tabeller i stedet for relasjoner (multi-mengder av tupler):

```
SELECT Alder
  FROM Person
```

$\Rightarrow$

Alder
13
24
13
25
25

- Dette trenger vi for aggregering (sum, gjennomsnitt, osv.) av kolonner

SQL og syntaks

SQL bryr seg ikke om indent og linjeskift (slik som f.eks. Python), så

SQL og syntaks

SQL bryr seg ikke om indent og linjeskift (slik som f.eks. Python), så

```
SELECT Birthdate
  FROM Customers
 WHERE NrProducts > 5
```

SQL og syntaks

SQL bryr seg ikke om indent og linjeskift (slik som f.eks. Python), så

```
SELECT Birthdate
  FROM Customers
 WHERE NrProducts > 5

SELECT Birthdate FROM Customers
 WHERE NrProducts > 5
```

SQL og syntaks

SQL bryr seg ikke om indent og linjeskift (slik som f.eks. Python), så

```
SELECT Birthdate
  FROM Customers
 WHERE NrProducts > 5

SELECT Birthdate FROM Customers
 WHERE NrProducts > 5

SELECT Birthdate
FROM Customers WHERE NrProducts > 5
```

SQL og syntaks

SQL bryr seg ikke om indent og linjeskift (slik som f.eks. Python), så

```
SELECT Birthdate
  FROM Customers
 WHERE NrProducts > 5

SELECT Birthdate FROM Customers
 WHERE NrProducts > 5

SELECT Birthdate
FROM Customers WHERE NrProducts > 5

SELECT Birthdate FROM Customers WHERE NrProducts > 5
```

er alle lov og representerer den samme spørringen.

SQL og bokstavering

- For SQL-nøkkelord og navn på tabeller og kolonner er SQL versalinsensitiv

SQL og bokstavering

- For SQL-nøkkelord og navn på tabeller og kolonner er SQL versalinsensitiv (eng.: *case-insensitive*)

SQL og bokstavering

- For SQL-nøkkelord og navn på tabeller og kolonner er SQL versalinsensitiv (eng.: *case-insensitive*)
- Altså, SQL skiller ikke mellom store og små bokstaver

SQL og bokstavering

- For SQL-nøkkelord og navn på tabeller og kolonner er SQL versalinsensitiv (eng.: *case-insensitive*)
- Altså, SQL skiller ikke mellom store og små bokstaver
- Så
  - `SELECT Name FROM Customers`
  - `select name from customers`

  er ekvivalente spørringer

SQL og bokstavering

- For SQL-nøkkelord og navn på tabeller og kolonner er SQL versalinsensitiv (eng.: *case-insensitive*)
- Altså, SQL skiller ikke mellom store og små bokstaver
- Så
  - `SELECT Name FROM Customers`
  - `select name from customers`

  er ekvivalente spørringer
- Men, SQL skiller på store og små bokstaver på verdier
  - så `'London'` og `'london'` er to forskjellige verdier

SQL og bokstavering

- For SQL-nøkkelord og navn på tabeller og kolonner er SQL versalinsensitiv (eng.: *case-insensitive*)
- Altså, SQL skiller ikke mellom store og små bokstaver
- Så
  - `SELECT Name FROM Customers`
  - `select name from customers`

  er ekvivalente spørringer
- Men, SQL skiller på store og små bokstaver på verdier
  - så `'London'` og `'london'` er to forskjellige verdier
- Bruk `--` (to bindestreker) for kommentarer (blir ignorert av databasen), f.eks.

```
SELECT Name --Dette er en kommentar
  FROM Customers
```

SQL og skjema

- Tabellnavn kan i `FROM`-klausulen bli prefiksert med et skjemanavn, for eksempel:

SQL og skjema

- Tabellnavn kan i `FROM`-klausulen bli prefiksert med et skjemanavn, for eksempel:
- gitt et skjema `UiO` som inneholder tabell `Students`,

SQL og skjema

- Tabellnavn kan i `FROM`-klausulen bli prefiksert med et skjemanavn, for eksempel:
- gitt et skjema `UiO` som inneholder tabell `Students`,
- så vil vi skrive `UiO.Students` i SQL

```
SELECT Name
  FROM UiO.Students
```

SQL og skjema

- Tabellnavn kan i `FROM`-klausulen bli prefiksert med et skjemanavn, for eksempel:
- gitt et skjema `UiO` som inneholder tabell `Students`,
- så vil vi skrive `UiO.Students` i SQL

```
SELECT Name
  FROM UiO.Students
```

- Skjemaet `public` finnes automatisk i alle databaser og er standard skjemaet

SQL og skjema

- Tabellnavn kan i `FROM`-klausulen bli prefiksert med et skjemanavn, for eksempel:
- gitt et skjema `UiO` som inneholder tabell `Students`,
- så vil vi skrive `UiO.Students` i SQL

```
SELECT Name
  FROM UiO.Students
```

- Skjemaet `public` finnes automatisk i alle databaser og er standard skjemaet
- Om man ikke spesifiserer et skjema er det dette som brukes, så

```
SELECT Name FROM Person
```

blir

```
SELECT Name FROM public.Person
```

Takk for nå!

Neste video vil se på litt mer avanserte `WHERE`-klausuler.