

05 – Enkel SQL

Leif Harald Karlsen
leifhka@ifi.uio.no



Universitetet i Oslo

1 / 43

SQL: Structured Query Language

- ◆ SQL er et spørrespråk for relasjonelle databaser
- ◆ Det mest brukte slike spørrespråker
- ◆ Brukt 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

2 / 43

Imperativ vs. deklarativ

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

- ◆ 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?"

3 / 43

Python/Java vs. SQL

- ◆ Programmeringsspråk (f.eks. Python og Java) er imperative språk, altså presise språk for å uttrykke *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 deklarativt, 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?"

4 / 43

- ◆ Python-programmer forteller datamaskinen *hvordan den skal beregne* svaret man vil ha
- ◆ En SQL-spørring forteller datamaskinen *hva den skal beregne*,
- ◆ og det er opp til databasen of finne ut *hvordan* svaret skal finnes

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**.

- ◆ (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>

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 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

9 / 43

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

10 / 43

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

11 / 43

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

12 / 43

Velge over et intervall av verdier

Spørring som finner navnet på alle kunder som har kjøpt mer enn 10 produkter

```
SELECT Name
FROM Customer
WHERE NrProducts > 10
```

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

13 / 43

Kombinere betingelser

Spørring som finner fødselsdatoen og navnet til kunder som kjøpte mellom 4 og 10 produkter

```
SELECT Birthdate, Name
FROM Customer
WHERE NrProducts > 4 AND
      NrProducts < 10
```

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

14 / 43

Kombinere betingelser med OR

Spørring som finner navnet til kunder som har kjøpt færre enn 5 produkter eller fler enn 15 produkter

```
SELECT Name
FROM Customer
WHERE NrProducts < 5 OR
      NrProducts > 15
```

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

15 / 43

Bruke både AND og OR

Spørring som finner navn på kunder som har kjøpt mindre enn 5 eller mer enn 15 produkter og er født etter '2000-01-01'

```
SELECT Name FROM Customer
WHERE (NrProducts < 5 OR
      NrProducts > 15) AND
      Birthdate > '2000-01-01'
```

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

16 / 43

Velge TVer

Spørring som henter navnet, merket og pris på 48 og 50 tommer TVer

```
SELECT Name, Brand, Price
FROM Product
WHERE Name = 'TV 50 inch' OR
      Name = 'TV 48 inch'
```

Resultat

ProductID	Name	Brand	Price	Stock
0	TV 50 inch	Sony	8999	29
1	Laptop 2.5GHz	Lenovo	7499	12
2	Laptop 8GB RAM	HP	6999	80
3	Speaker 500	Bose	4999	42
4	TV 48 inch	Panasonic	11999	31
5	Phone S6	IPhone	5195	65

17 / 43

Søke i tekst

- ◆ Med det vi har lært hittil har vi ingen måte å spørre etter alle TVer
 - ◆ (altså alle produkter som har navn som starter med 'TV')
- ◆ Vi kan kun bruke likhet, ingen måte å søke i tekst
- ◆ Dette kan gjøres med SQLs [LIKE](#)
- ◆ Kan så bruke '%' som "wildcard" som matcher alt

18 / 43

LIKE

For eksempel:

- ◆ Name [LIKE](#) 'TV%'
 - ◆ Sant for alle Name-verdier som starter med 'TV'
 - ◆ f.eks. 'TV 50 inch' og 'TVSHOW'
 - ◆ men ikke f.eks. 'hello' eller 'MTV'
- ◆ Name [LIKE](#) '%TV'
 - ◆ sant for alle Name-verdier som slutter med 'TV'
 - ◆ f.eks. '50 inch TV' og 'MTV'
 - ◆ men ikke f.eks. 'TV2' eller 'Fun TV program'
- ◆ Name [LIKE](#) '%TV%'
 - ◆ sant for alle Name-verdier som inneholder 'TV' (hvor som helst)
 - ◆ f.eks. '50 inch TV' og 'Fun TV program'
 - ◆ men ikke f.eks. 'T2V' eller 'hello'
- ◆ Name [LIKE](#) '%TV%inch'
 - ◆ sant for alle Name-verdier som inneholder 'TV' og slutter med 'inch'
 - ◆ f.eks. 'TV 50 inch' og 'Fun TV program pinch'
 - ◆ men ikke f.eks. 'TV 50 inches' eller '50 inch TV'

19 / 43

Velge TVer med LIKE

Spørring som finner navn, pris og merke på alle TVer

```
SELECT Name, Brand, Price
FROM Product
WHERE Name LIKE 'TV%'
```

Resultat

ProductID	Name	Brand	Price	Stock
0	TV 50 inch	Sony	8999	29
1	Laptop 2.5GHz	Lenovo	7499	12
2	Laptop 8GB RAM	HP	6999	80
3	Speaker 500	Bose	4999	42
4	TV 48 inch	Panasonic	11999	31
5	Phone S6	IPhone	5195	65

20 / 43

Regulære uttrykk

- ◆ `LIKE` støtter kun `%` (og `_` for wildcard enkelt karakter)
- ◆ Ønsker man komplisert matching kan man bruke `SIMILAR TO` eller `~`
- ◆ `SIMILAR TO` bruker litt rar miks av `LIKE`-syntaks (`%`) og vanlige regulære uttrykk
- ◆ F.eks. er `Name = 'abc'` et mulig svar for

```
SELECT Name
FROM Products
WHERE Name SIMILAR TO '%(b|d)%'
```

- ◆ Man kan også bruke `~` for vanlige (POSIX) regulære uttrykk
- ◆ F.eks.

```
Name ~ '.*(b|d).*'
```

er samme som over

- ◆ `LIKE` finnes fordi den er sikrere mhp. ytelse (kan alltid eksekveres raskt)

21 / 43

Negasjon

- ◆ Av og til vil vi bare ha svar som *ikke* tilfredstiller et uttrykk
- ◆ Bruker da `NOT`-nøkkelordet
- ◆ For eksempel:

```
SELECT Name
FROM Products
WHERE NOT Description LIKE '%simple%'
```

er sant for alle rader som ikke har order 'simple' i sin Description

- ◆ Merk at
 - ◆ `NOT (E1 AND E2)` er ekvivalent med `(NOT E1) OR (NOT E2)`
 - ◆ `NOT (E1 OR E2)` er ekvivalent med `(NOT E1) AND (NOT E2)`

22 / 43

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.

23 / 43

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
```

24 / 43

SQL og skjema

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

```
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

25 / 43

Null

- ◆ Når vi setter inn data vil vi av og til mangle en verdi (f.eks. fordi den er ukjent eller ikke finnes)
- ◆ For eksempel, kan det være vi ikke vet fødselsdatoen til en bestemt student
- ◆ Likevel ønsker vi å legge studenten inn i databasen slik at vi kan lagre informasjon om studenten
- ◆ Men hva skal vi sette inn?
 - ◆ Den tomme teksten? Feil type!
 - ◆ År 0? Ikke korrekt!
- ◆ For ukjente og manglende verdier har SQL `NULL`
- ◆ Så, for å sette inn studenten `Sam Penny` med ukjent fødselsdato, bruker vi `NULL`

Students		
SID	StdName	StdBirthdate
0	Anna Consuma	1978-10-09
1	Anna Consuma	1978-10-09
2	Peter Young	2009-03-01
3	Carla Smith	1986-06-14
4	Sam Penny	?

26 / 43

SQL og null

- ◆ Hvordan sjekker vi om en verdi er `NULL`?
- ◆ Dersom vi prøver

```
SELECT StdName
FROM Students
WHERE StdBirthdate = NULL
```

får vi ingen svar!

- ◆ Faktisk så er `NULL = NULL` ikke sant
- ◆ og heller ikke `NOT (NULL = NULL)`!
- ◆ Grunnen til dette er at `NULL` representerer en manglende eller ukjent verdi
- ◆ Så `NULL` kan potensielt representere en hvilken som helst verdi
- ◆ Så `StdBirthdate = NULL` og `NULL = NULL` er begge ukjente, altså `NULL`
- ◆ Og `NULL` er ikke `TRUE` (sant) så det tilfredstiller ikke `WHERE`-klausulen

27 / 43

Sjekke for NULLs

- ◆ For å sjekke om en verdi er `NULL` må vi bruke `IS NULL`.
- ◆ For eksempel:

```
SELECT StdName
FROM Students
WHERE StdBirthdate IS NULL
```

så får vi `Sam Penny` som svar

- ◆ Vi kan også bruke `IS NOT NULL` for å sjekke at en verdi ikke er `NULL`

28 / 43

NULLs oppførelse

- ◆ Merk at **NULL** oppfører seg som *ukjent*:
 - ◆ **NULL AND TRUE** resulterer i **NULL**
 - ◆ **NULL OR FALSE** resulterer i **NULL**
 - ◆ **NULL AND FALSE** resulterer i **FALSE**
 - ◆ **NULL OR TRUE** resulterer i **TRUE**
 - ◆ **10 + NULL** resulterer i **NULL**
 - ◆ (Prøv å lese hver setning over med *ukjent* i stedet for **NULL**)
- ◆ Så resultatet av et uttrykk med **NULL** er **NULL** dersom svaret avhenger av hva **NULL** kan være

29 / 43

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 <columns>
  FROM <table>
 WHERE <condition>      ⇒      π<columns>(σ<condition>(<table>))
```

30 / 43

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

π _{Alder} (Person)
Alder
13
24
25

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

```
SELECT Alder
  FROM Person
```



Alder
13
24
13
25
25

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

31 / 43

Dupliserte svar

- ◆ Svarene fra en spørring kan altså inneholde duplikater
- ◆ F.eks. dersom vi kjører

```
SELECT contacttitle
  FROM customers
 WHERE contacttitle LIKE '%Manager%'
```

over northwind-databasen får vi 33 svar:

contacttitle
Marketing Manager
Accounting Manager
Marketing Manager
Sales Manager
Accounting Manager
Marketing Manager
Marketing Manager
:
:

32 / 43

Fjerning av duplikater

- ◆ Duplikater er av og til uønsket
- ◆ (Men ikke alltid, f.eks. for aggregering (kommer senere))
- ◆ Vi kan fjerne duplikater med `DISTINCT`-nøkkelordet i `SELECT`-klausulen
- ◆ F.eks.:

```
SELECT DISTINCT contacttitle
FROM customers
WHERE contacttitle LIKE '%Manager%'
```

gir kun 3 svar:

contacttitle
Sales Manager
Marketing Manager
Accounting Manager

33 / 43

Uttrykk i SELECT

- ◆ Hittil har vi bare hentet ut data direkte fra tabeller
- ◆ Ofte ønsker man å transformere dataene før vi returnerer svaret
- ◆ Dette kan gjøres med bruke uttrykk for å manipulere verdiene i `SELECT`-klausulen
- ◆ For eksempel, for å få alle priser i NOK fremfor USD (antar at 1 USD = 8 NOK) kan vi gjøre:

```
SELECT productname, retailprice * 8
FROM products
```

- ◆ Eller, for å returnere fullt navn og full adresse til alle kunder kan vi kjøre:

```
SELECT custfirstname || ' ' || custlastname,
       custaddress || ', ' || custcity || ', ' || custzipcode
FROM customers
```

- ◆ `||` konkatenerer strenger (f.eks. 'hel' || 'lo' = 'hello')

34 / 43

Gi navn til kolonner

- ◆ Når vi har et uttrykk i en `SELECT`-klausul får den resulterende kolonnen ingen navn
- ◆ Vi kan gi kolonner resultat-tabellen navn ved å bruke `AS`-nøkkelordet
- ◆ F.eks.:

```
SELECT productname, retailprice * 8 AS retailpricenok
FROM products
```

```
SELECT custfirstname || ' ' || custlastname AS fullname,
       custaddress || ', ' || custcity || ', ' || custzipcode AS address
FROM customers
```

35 / 43

Menti

Menti

36 / 43

Oppgave 1

Hvor mange svar gir følgende spørring?

```
SELECT *
  FROM person
 WHERE name = 'Per'
```

person		
navn	alder	hobby
Per	34	baking
Kari	23	ski
Mari	56	vannski

Riktig svar: 1
Nøyaktig én person med navn 'Per'.

37 / 43

Oppgave 2

Hvor mange svar gir følgende spørring?

```
SELECT navn, alder
  FROM person
 WHERE true
```

person		
navn	alder	hobby
Per	34	baking
Kari	23	ski
Mari	56	vannski

Riktig svar: 3
Alle rader blir med, siden **WHERE**-klausulen alltid er **true**.

38 / 43

Oppgave 3

Hvor mange svar gir følgende spørring?

```
SELECT navn, alder
  FROM person
 WHERE alder > 20 AND
       navn LIKE '%a%'
```

person		
navn	alder	hobby
Per	34	baking
Kari	23	ski
Mari	56	vannski

Riktig svar: 2
Alle rader har `alder > 20`, men kun navnene 'Kari' og 'Mari' inneholder a

39 / 43

Oppgave 4

Hvor mange svar gir følgende spørring?

```
SELECT 1
  FROM person
```

person		
navn	alder	hobby
Per	34	baking
Kari	23	ski
Mari	56	vannski

Riktig svar: 3
Ingen **WHERE**-klausul betyr at alle svar blir med. Resultatet blir

?column?
1
1
1

40 / 43

Oppgave 5

Hvor mange svar gir følgende spørring?

```
SELECT *
  FROM person
 WHERE NOT hobby = NULL
```

person		
navn	alder	hobby
Per	34	baking
Kari	23	ski
Mari	56	NULL

Riktig svar: 0

hobby = NULL blir NULL og NOT NULL er NULL, altså blir WHERE-klausulen alltid NULL

41 / 43

Eksempler: Nortwhind-DB

Finn produktnavn, total verdi og hvorvidt flere er bestilt for alle produkter som selges i flasker [11 rader]

```
SELECT product_name,
       unit_price * units_in_stock AS total,
       units_on_order > 0 AS ordered
  FROM products
 WHERE quantity_per_unit LIKE '%bottles%';
```

42 / 43

Eksempler: Nortwhind-DB

Finn fullt navn (med tittel) på kontaktpersonen og telefon- og faksnummer til alle kundefirmaer i Tyskland og Frankrike hvor enten telefon- eller faksnummer er tilgjengelig [22 rader]

```
SELECT contact_title || ' ' || contact_name AS person,
       phone, fax
  FROM customers
 WHERE (country = 'Germany' OR country = 'France')
       AND (phone IS NOT NULL OR fax IS NOT NULL);
```

43 / 43