Avansert bruk av SQL

- Avanserte spørringer
  - Spørring på spørring
  - Unionspørringer
  - Delspørringer
- Begrensninger ved SQL
- Relasjonsmodellen
  - Relasjonsalgebra
- Bruksområder for SQL
  - Systemarkitekturer
  - Programmering
  - Verktøy

Dagens sitat (anonym kilde):

- Give me a fish and I eat for a day. Teach me to fish and I eat for a lifetime.

Pensum: Kapittel 5 og 6

Eksempel på likekobling

- Vis alle ordrer behandlet av hver ansatt. Sorter utskriften på etternavn og ordredato.

```
SELECT Etternavn, OrdreNr, Ordredato
FROM Ansatt, Ordre
WHERE Ansatt.AnsNr = Ordre.AnsNr
ORDER BY Etternavn, Ordredato
```

- Likheten i **WHERE** er en koblingsbetingelse.
- Spørringen kalles en (indre) likekobling (join).
- AnsNr må prefikses med tabellnavn fordi kolonnenavnet forekommer i begge tabeller.
- Etternavn, OrdreNr og Ordredato kan prefikses.

Syntaks for likekobling

- Likekoblinger forekommer så hyppig at det er innført en spesiell skrivemåte:

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

- Generelt:

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

- Rekkefølgen av tabellene spiller ingen rolle.

.

Egenkoblinger

- Tabeller kan kobles med ”seg selv”.
- Finn alle kombinasjoner av varer med samme pris:

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

- Tenk slik: DBHS ”lager” 2 kopier av tabellen Vare, og kobler disse på vanlig måte.
- Tabeller spiller av og til flere ”roller”, som også kan medføre behov for å bruke flere ”kopier” av samme tabell i en spørring. Eksempel: Avgangsflyplass og ankomstflyplass i en flyavgang, se oppg 4 i kap. 4.

Egenkobling fra oblig

- Vi har tabellene
- EMP(EMPNO,ENAME,SAL,COMM,MNGR,DEPTNO) og
- DEPT(DEPTNO,LOC,DNAME)
- Finn navnet på alle ansatte som tjener mer enn JONES v.h.a egenkobling. Feltet DEPTNO er felles for de to tabellene og er primærnøkkel i DEPT og fremmednøkkel i EMP-tabellen.

Løsning

- SELECT E1.ENAME,E1.SAL,E2.ENAME,E2.SAL FROM EMP AS E1, EMP AS E2 WHERE E2.ENAME='JONES' AND E1.SAL>E2.SAL;

- Ny: Finn navn på de ansatte som tjener mer enn sjefen sin ved hjelp av egenkobling

- E1

JACKSON	100000
JONES	50000
ADAMS	75000

E2

JACKSON	100000
JONES	50000
ADAMS	75000

Løsning

```
SELECT E1.ENAME,E1.SAL,E2.ENAME.E2.SAL
FROM EMP AS E1, EMP AS E2
WHERE E1.MGR=E2.EMPNO AND E1.SAL>E2.SAL
```

NY:

FINN DE SOM HAR SAMME JOBB SOM JONES ELLER HAR HØYERE LØNN ENN FORD

E1

JACKSON	133
JONES	122
ADAMS	113

E2

124	ADAMS
133	JONES
144	MARPLE

Løsning

```
SELECT E3.ENAME,E3.SAL,E3.JOB
FROM EMP AS E1, EMP AS E2, EMP AS E3
WHERE E1.ENAME='JONES' AND E3.JOB=E1.JOB OR E2.NAME='FORD' AND
    E3.SAL>E2.SAL;
```

E1

JACKSON	MANAGER
JONES	SALESMAN
ADAMS	PRESIDENT

E2

JACKSON	75000
FORD	50000
ADAMS	45000

E3

BROWN	75000	MANAGER
JONES	20000	SALESMAN
ADAMS	30000	WORKER

Mer om JOIN

Gitt SQL setningene:

```
mysql> CREATE TABLE PERSON (NAVN VARCHAR(10) PRIMARY
    KEY, AVD INT);
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO PERSON VALUES ('OLE','1'),('ANNE','2'),
    ('PER',NULL);
Query OK, 3 rows affected (0.16 sec)
```

```
mysql> CREATE TABLE AVDELING (AVDNR INT PRIMARY
    KEY,AVDNAVN VARCHAR(10));
Query OK, 0 rows affected (0.19 sec)

mysql> INSERT INTO AVDELING VALUES ('1','LAGER'),
    ('2','KJOKKEN'),('3','SALG'),('
4','OSLO');
```

Nå har vi:

Navn,avdeling
Ole,1
Anne,2
Per,NULL

Avdnr,Avdnavn
1,Lager
2,Kjøkken
3,Salg
4,Oslo

INNER JOIN:

SELECT navn,avdnavn from person,avdeling WHERE
person.avdnr=avdeling.avdnr;

Gir resultatet?

LEFT JOIN

mysql> SELECT NAVN,AVDNAVN FROM PERSON **LEFT JOIN** AVDELING ON PERSON.AVD=
AVDELING.AVDNR;

```
+------+---------+
| NAVN | AVDNAVN |
+------+---------+
| OLE  | LAGER   |
| ANNE | KJOKKEN |
| PER  | NULL    |
+------+---------+
```

RIGHT JOIN

RIGHT JOIN

```
mysql> SELECT NAVN,AVDNAVN FROM PERSON
    AVDELING ON
          PERSON.AVD=AVDELING.AVDNR;
+------+---------+
| NAVN | AVDNAVN |
+------+---------+
| OLE  | LAGER   |
| ANNE | KJOKKEN |
| NULL | SALG    |
| NULL | OSLO    |
+------+---------+
```

Oppgaver

- Finn de avdelingene i DEPT som ikke har noen ansatte i EMP

Løsning

```
SELECT DNAME FROM DEPT
    LEFT JOIN EMP
ON DEPT.DEPTNO=EMP.DEPTNO
WHERE JOB IS NULL;
```

DEPTNO	DNAME	ENAME	JOB	SAL
114	CHICAGO	ADAMS	MANAGER	10000
120	NEW YORK	BRIAN	BOSS	30000
145	ALASKA			

Spørringer på spørringer

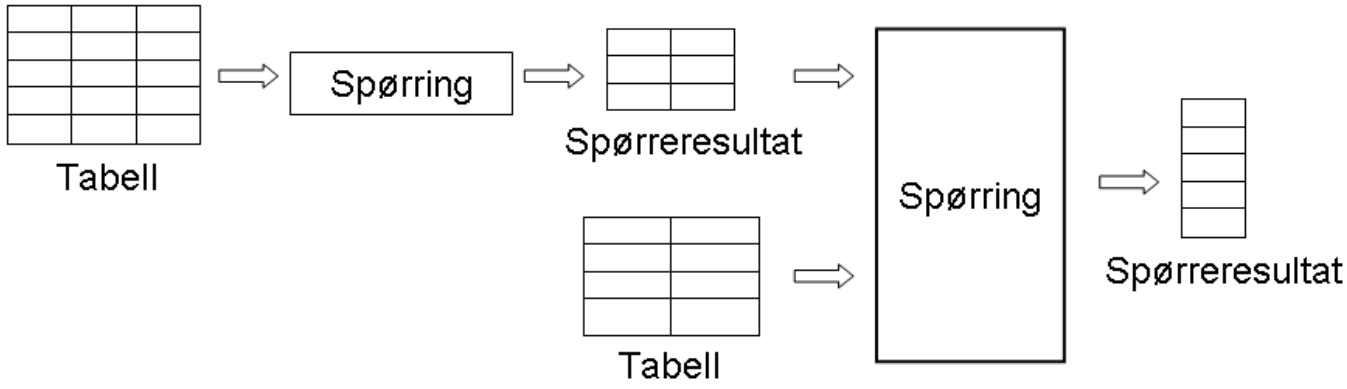
- Problem: Finn antall stillingsbetegnelser.
  - Delproblem 1: Finn stillingsbetegnelsene

```
SELECT DISTINCT Stilling
FROM Ansatt
```

  - Delproblem 2: Beregn antallet

```
SELECT COUNT(*) AS Antall
FROM StillingsBetegnelser
```

- StillingsBetegnelser er navnet på den første spørringen.
- Vi bryter ned et sammensatt problem i to enklere delproblemer!



Oppgave

- FINN NAVNET PÅ DEN SOM TJENER MEST OG LØNNEN HANS

Løsning

```
SELECT ENAME FROM EMP
WHERE SAL=(SELECT MAX(SAL) FROM EMP);
```

Oppgave

- FINN NAVN OG LØNN PÅ DE SOM TJENER MER ENN GJENNOMSNITTET FOR SIN AVDELING

Løsning

- select E1.* From EMP as E1 where sal>(select avg(SAL) from EMP as E2 where E1.DEPTNO=E2.DEPTNO);

Delspørringer

- Finn alle som tjener mer enn gjennomsnittet:

```
SELECT *
FROM Ansatt
WHERE Lønn > ( SELECT AVG(Lønn) FROM Ansatt )
```

- Delspørringen må returnere en tabell med nøyaktig èn rad og èn kolonne (for å kunne brukes på høyresiden av >).
- Tenk at resultatet av delspørringen ”limes inn” i hovedspørringen.
  - Med gjennomsnittslønn 220.000 får vi Lønn>220.000.
- Hvor mange ganger må DBHS gjennomløpe tabellen Ansatt?

Vekselvirkende delspørringer

- Finn de som tjener mer enn gjennomsnittslønnen innenfor sin stillingskategori:

```
SELECT A1.*
FROM Ansatt AS A1
WHERE Lønn >
      ( SELECT AVG(Lønn)
        FROM Ansatt AS A2
        WHERE A1.Stilling = A2.Stilling )
```

- Sammenlign med spørringen «Finn alle som tjener mer enn gjennomsnittet».
- Hvor mange ganger må DBHS nå gjennomløpe tabellen Ansatt?

ALL og SOME

- Finn de som tjener mer enn alle sekretærer:

```
SELECT *
FROM Ansatt
WHERE Lønn > ALL
      (SELECT Lønn
       FROM Ansatt
       WHERE Stilling='Sekretær')
```

- Hva om vi skriver SOME i stedet for ALL?
- Kan vi klare oss uten ALL? Tips: Bruk MAX.
- Delspørringen må returnere èn kolonne (som her må inneholde beløp).

EXISTS

- Finn ansatte som ikke har deltatt på noen prosjekter:

```
SELECT A.*
FROM Ansatt AS A
WHERE NOT EXISTS
      (SELECT *
       FROM ProsjektDeltakelse AS PD
       WHERE PD.AnsNr = A.AnsNr)
```

- EXISTS kan av og til erstattes av IN.
  - La delspørringen returnere en liste med ansattnumre.

- Finn ansatte som har vært med på alle prosjekter.
  - Tips: For ansatt X må det ikke finnes prosjekter som X ikke har deltatt i.

Oppgave

- List navn og lønn til den som tjener mest, navn og lønn til den som tjener minst og differansen med en spørring

Løsning

```
SELECT E1.ENAME, E1.SAL,E2.ENAME,E2.SAL,E1.SAL-E2.SAL
FROM EMP AS E1, EMP AS E2
WHERE E1.SAL=(SELECT MAX(SAL) FROM EMP)
AND (E2.SAL=(SELECT MIN(SAL) FROM EMP);
```

FINN GJENNOMSNITTSLØNN FOR HVER AVDELING.

- select DEPTNO,AVG(SAL) from EMP GROUP BY DEPTNO;

```
+--------+-----------+
| DEPTNO | AVG(SAL)  |
+--------+-----------+
|     10 | 2916.6667 |
|     20 | 2175.0000 |
|     30 | 1566.6667 |
+--------+-----------+
3 rows in set (0.00 sec)
```

Begrensninger ved SQL

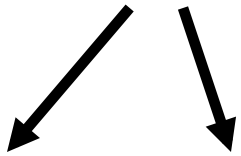
- La Slekt være en tabell med kolonner Person, Mor og Far (som alle inneholder personnr).
  - Lag en spørring som finner mor og far for en bestemt person.
  - Hva med bestemødre og bestefedre?
  - Hva med samtlige forfedre for en gitt person?
  - Hvor mange gjennomløp av tabellen medfører spørringene (hvis personnr ikke er ordnet på noe systematisk vis) ?

- Det er vanskelig/umulig å løse denne oppgaven med SQL (avhenger av om DBHS støtter rekursjon).

- Vi bør kjenne "grensene" for SQL så vi ikke kaster bort mye tid med å løse en umulig oppgave. Kan være bedre å kombinere SQL med "generelle" programmeringsspråk.

Unike kolonner uten rekkefølge

- Kolonnenavn må være unike innen en tabell.



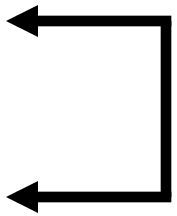
AnsattNr	Etternavn	Fornavn	AnsattDato	Stilling	Lønn
1	Veum	Varg	01.01.1991	Løpegutt	kr 123 000,00
2	Stein	Trude	10.10.1994	DBA	kr 270 000,00

- Rekkefølge er uten betydning.

En mengde av rader

ProsjektNr	Budsjett	Leder	Start	Slutt
1001	kr 15 000,00	20	12.01.2002	12.03.2002
1002	kr 750 000,00	8	23.06.2002	23.07.2002
1007	kr 125 000,00	2	12.06.2003	
1009	kr 500 000,00	20	01.01.2003	
1012	kr 10 000,00	4	10.07.2003	
1020	kr 900 000,00	8	23.07.2002	01.09.2003

- Rekkefølge er uten betydning.



- Ingen rader skal være like. Det betyr at enhver rad har en entydig identifikator = primærnøkkel.

Atomære verdier

ProsjektNr	Ansatte
1001	1
1002	4, 8, 13, 20

➢ Ikke tillatt!

- Verdier er «enkle» (tall, tekster, datoer).
- Vi tillater ikke ”repeterende” verdier.
- Vi tillater ikke ”tabeller i tabeller”.

- Nyere objektrelasjonelle databaser tillater mer kompliserte verdier.

Forhold/relasjoner som tabeller

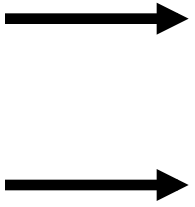
AnsattNr	ProsjektNr	AntTimer
1	1	12
1	7	20
1	9	7
1	12	14
2	7	3
4	1	1
4	7	10
4	9	120
4	12	75
5	2	100
5	12	94
8	2	10
8	12	20
8	20	20
11	7	50
11	9	20
13	2	3
20	9	4
20	20	20

- En ansatt kan jobbe på mange prosjekter.
- Et prosjekt kan ha mange prosjektmedarbeidere.
- Det er et mange-til-mange forhold (relasjon) mellom prosjekter og ansatte.
- For hver kombinasjon av prosjekt og ansatt lagrer vi antall nedlagte arbeidstimer.
- En tabell kan altså representere et forhold/relasjon.

Funksjonelle avhengigheter

- Vi sier det er en funksjonell avhengighet fra A til B, skrevet A → B, hvis to rader med samme verdi i kolonne A må ha samme verdi i kolonne B.

- Vi kan også snakke om funksjonelle avhengigheter fra en samling kolonner X til en kolonne A: X → A.



ANr	Navn	PNr
12	Lise	7
17	Ola	8
12	Lise	23
21	Kari	26

Her gjelder ANr→Navn.

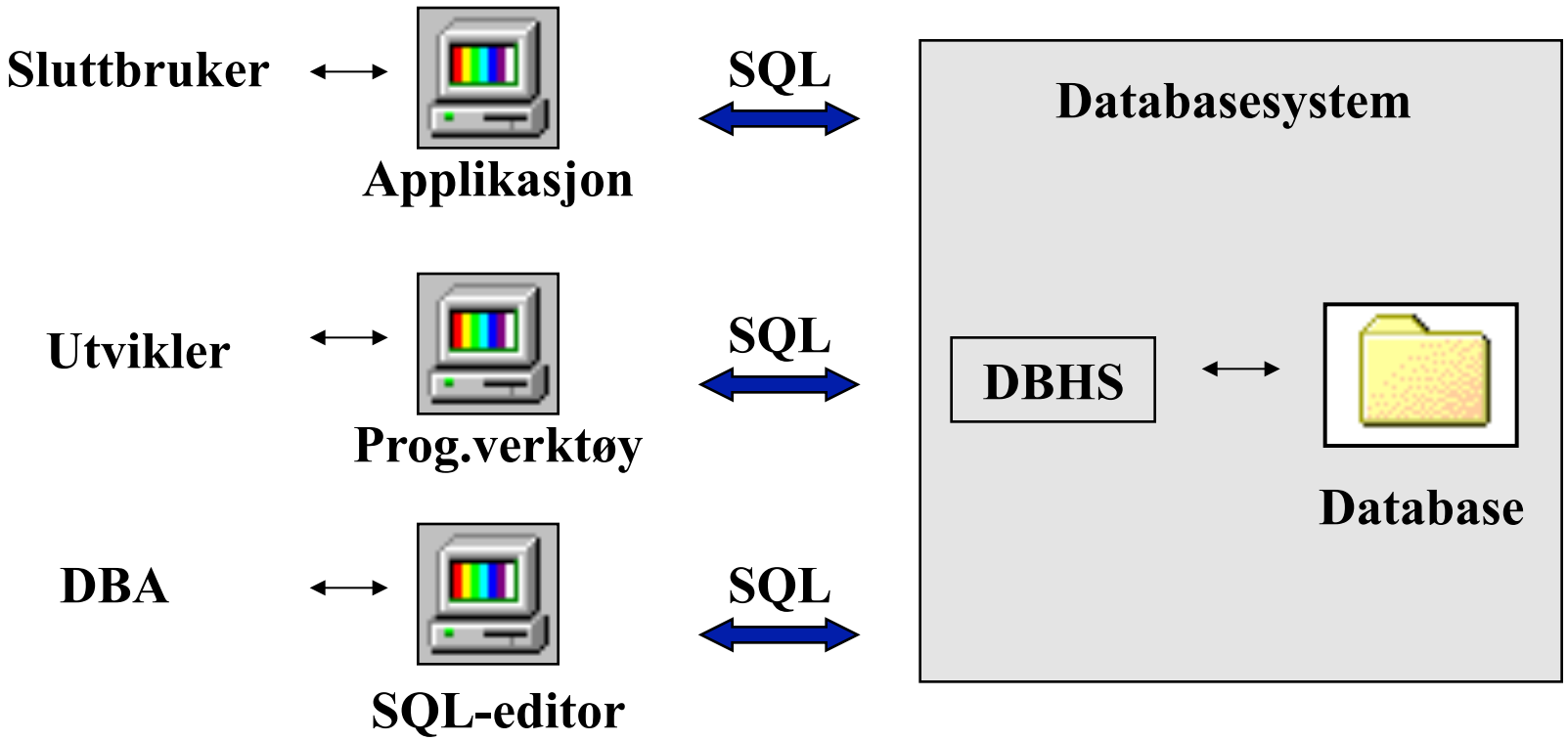
Primærnøkler

- Gitt en tabell T. En supernøkkel for T er en samling kolonner X slik at X → A gjelder for alle kolonner A.
- En kandidatnøkkel er en minimal supernøkkel.
- Database-designeren velger en av kandidatnøklene som primærnøkkel.
- Alle tabeller skal ha nøyaktig 1 primærnøkkel.
- Bestem kandidatnøkler og primærnøkkel for disse tabellene:

  Ansatt(AnsNr, PersonNr, Fornavn, Etternavn, Stilling)

  Vielse(BrudPNr, BrudgomPNr, Kirkenavn, Dato)

Databasesystemer



Brukere og arbeidsoppgaver

- Sluttbruker
  - Ajourhold
  - Spørringer og rapporter
- Databaseadministrator
  - Brukeradministrasjon
  - Overvåke, optimalisere
- Utvikler
  - Definere tabellstruktur og forretningsregler
  - Utvikle applikasjoner (menyer, skjermbilder, rapporter)

- De aller fleste arbeidsoppgavene kan gjøres ved hjelp av SQL, men blir ofte utført med egne verktøy.
  - Rapporteringsverktøy, modelleringsverktøy, …

Verktøy som genererer SQL - I

- Datamodellering innebærer å planlegge tabellstrukturen i en database. Eksempel på datamodell i E/R:

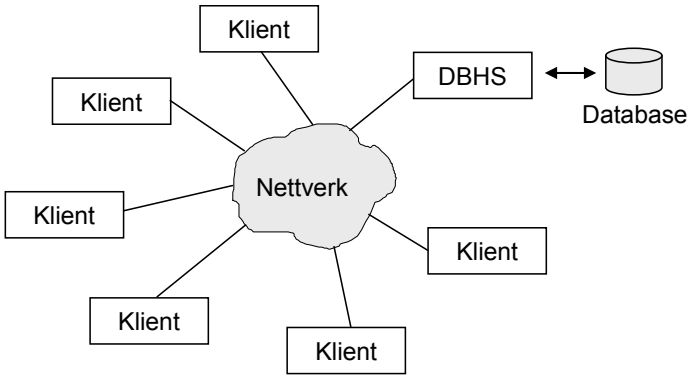
- Modelleringsverktøyet har en funksjon for å generere databaser (for ulike DBHS).
  - Verktøyet logger seg på databasesystemet.
  - Analyserer datamodellen.
  - Sender en serie med CREATE TABLE setninger til DBHS.
  - Tolker eventuelle feilmeldinger fra databasesystemet og viser dem for brukeren av modelleringsverktøyet.

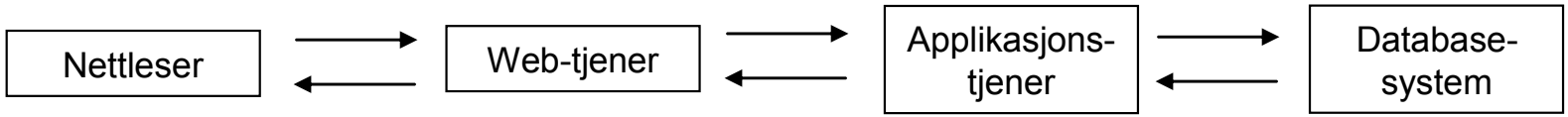
SQL-standarden

- Kan deles inn i tre deler:
  - Data Definition Language (DDL): definere tabeller, indekser og valideringsregler
  - Data Manipulation Language (DML): legge inn nye data, endre data, slette data, hente ut data
  - Data Control Language (DCL): brukeradministrasjon
- Standarder: SQL:86, SQL:89, SQL:92, SQL:1999, SQL:2003,…
- Databaseprogrammering
  - Ikke alle programmeringsoppgaver lar seg løse med SQL.
  - Databaseapplikasjoner består både av SQL og kode skrevet i et generelt programmeringsspråk.
  - Programmeringsgrensesnitt mot SQL er standardisert.

Systemarktitekturer

- Sentralisert databasesystem
- Personlige databaser
- Database for en avdeling på et lokalnett
- Virksomhetsdatabase
  - Sentral database med terminaler
  - Klient/tjener-arkitektur med arbeidsstasjoner
  - Distribuert database (flere databasetjenere med data)
- Web-database
  - Web-server og databaseserver på 1 maskin
  - Flerlagsarkitektur





Slutt på del I

- Dermed setter vi strek for del I om **SQL og tabeller**.
- Neste tema er **databasedesign** og ER-modellering