

Anders Hoff

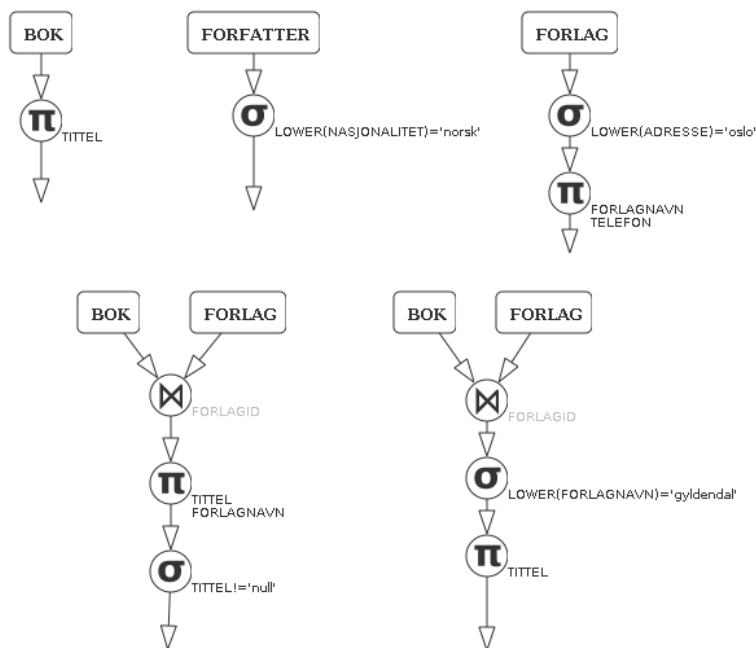
Øving 3

TDT4145 Datamodeller og databasesystemer

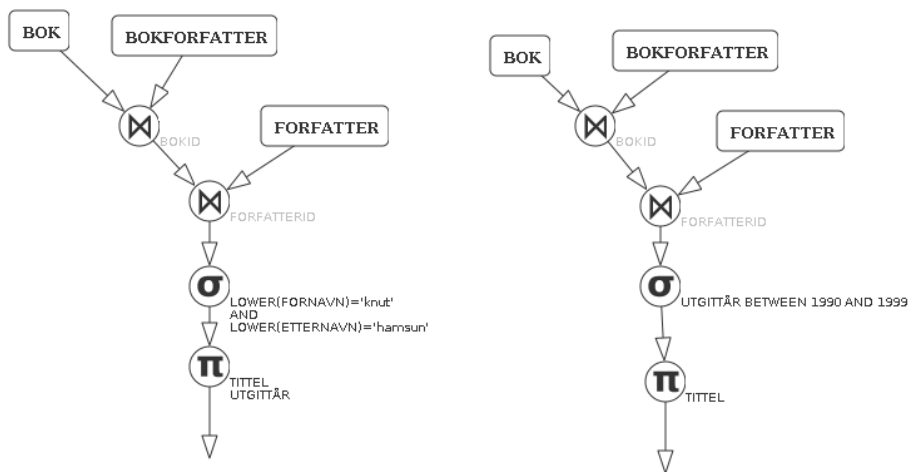
Februar 2010

Oppgave 1

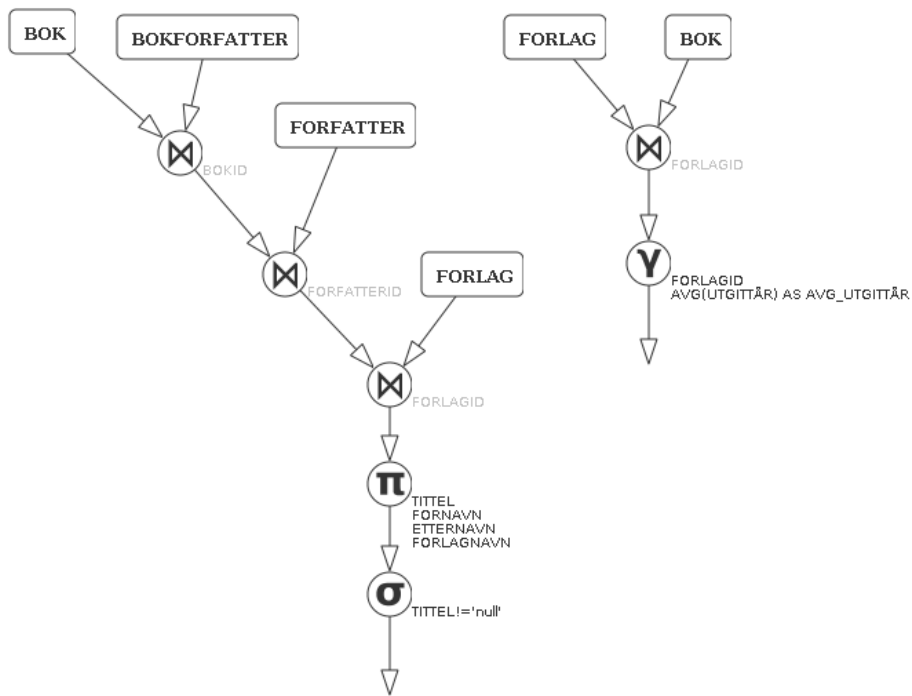
Oppgave 1.1 til 1-5 finnes i Figur 1. Oppgave 1.6 og 1.7 finnes u 2. Oppgave 1.8 og 1.9 finnes i Figur 3 og oppgave 1.10 til 1.11 finnes i Figur 4.



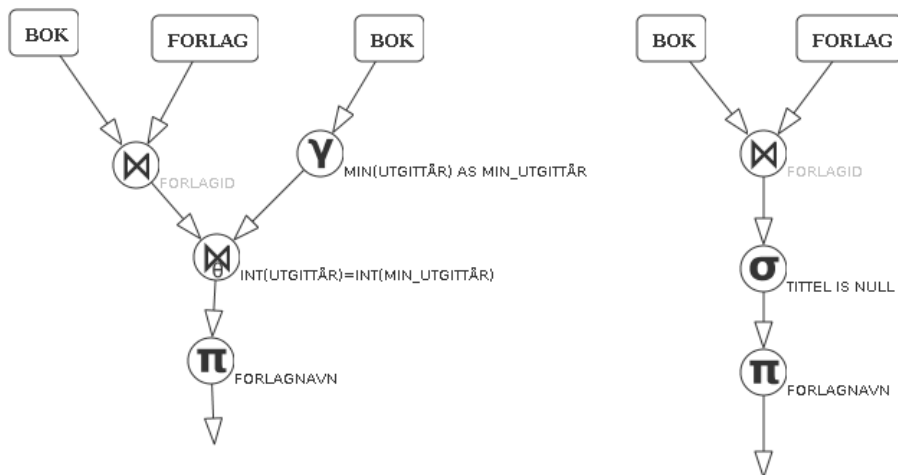
Figur 1: Oppgave 1.1 til 1.5 fra venstre mot høyre.



Figur 2: Oppgave 1.6 og 1.7 fra venstre mot høyre.



Figur 3: Oppgave 1.8 og 1.9 fra venstre mot høyre. I oppgave 1.8 har vi tolket det som at vi skulle ta gjennomsnittet av utgivelsesår for hvert enkelt forlag.



Figur 4: Oppgave 1.10 og 1.11 fra venstre mot høyre.

Oppgave 2

Sql finnes i Listing 1.

```

1 CREATE TABLE KUNDE(
2 KUNDENR INT UNSIGNED AUTOINCREMENT NOT NULL,
3 NAVN VARCHAR(255) NOT NULL,
4 KREDITTGRENSE DECIMAL(15,2) ,
5 POSTNR CHAR(4) NOT NULL,
6 PRIMARY KEY(KUNDENR) ,

```

```

7 FOREIGN KEY(POSTNR) REFERENCES POSTSTED(POSTNR) ON DELETE RESTRICT
8 );
9
10 CREATE TABLE POSTSTED(
11 POSTNR CHAR(4) NOT NULL,
12 POSTSTED VARCHAR(255) NOT NULL,
13 PRIMARY KEY(POSTNR) );
14
15 CREATE TABLE ARTIKKEL(
16 ARTNR INT UNSIGNED AUTOINCREMENT NOT NULL,
17 NAVN VARCHAR(255) NOT NULL,
18 ANT INT UNSIGNED,
19 PRIS DECIMAL(15,2) UNSIGNED,
20 PRIMARY KEY(ARTNR) );
21
22 CREATE TABLE BESTILLING(
23 ARTNR INT UNSIGNED NOT NULL,
24 KUNDENR INT UNSIGNED NOT NULL,
25 KVANTUM INT UNSIGNED NOT NULL,
26 FOREIGN KEY(ARTNR) REFERENCES ARTIKKEL(ARTNR) ON DELETE CASCADE,
27 FOREIGN KEY(KUNDENR) REFERENCES KUNDE(KUNDENR) ON DELETE CASCADE
28 );

```

Listing 1: Oppgave 2

Oppgave 3

Sql finnes i Listing 2.

```

1 #a
2 SELECT * TITTEL FROM BOK;
3
4 #b
5 SELECT * FROM FORFATTER WHERE LOWER(NASJONALITET)='norsk';
6
7 #c
8 SELECT FORLAGNAVN, TELEFON
9 FROM FORLAG
10 WHERE LOWER(ADRESSE)='oslo'
11 ORDER BY FORLAGNAVN ASC;
12
13 #d
14 SELECT TITTEL, FORLAGNAVN
15 FROM FORLAG, BOK
16 WHERE BOK.FORLAGID=FORLAG.FORLAGID;
17
18 #e
19 SELECT TITTEL, UTGITTAAR
20 FROM
21     BOKFORFATTER JOIN FORFATTER
22     ON BOKFORFATTER.FORFATTERID=FORFATTER.FORFATTERID
23     JOIN BOK
24     ON BOKFORFATTER.BOKID=BOK.BOKID
25 WHERE LOWER(FORNAVN)='knut' AND LOWER(ETTERNAVN)='hamsun';
26
27 #f
28 SELECT FORNAVN, ETTERNAVN, FOEDEAAR
29 FROM FORFATTER

```

```

30 WHERE ETTERNAVN LIKE 'H%' ;
31
32 #g
33 SELECT COUNT(*) FROM FORLAG;
34
35 #h
36 SELECT TITTEL, FORNAVN, ETTERNAVN, FORLAGNAVN
37 FROM FORLAG, FORFATTER, BOKFORFATTER, BOK
38 WHERE
39     BOK.FORLAGID=FORLAG.FORLAGID
40 AND BOKFORFATTER.FORFATTERID=FORFATTER.FORFATTERID
41 AND BOKFORFATTER.BOKID=BOK.BOKID
42 AND LOWER(NASJONALITET)='britisk' ;
43
44 #i
45 SELECT FORNAVN,ETTERNAVN,COUNT(BOKID)
46 FROM FORFATTER, BOKFORFATTER
47 WHERE BOKFORFATTER.FORFATTERID=FORFATTER.FORFATTERID
48 GROUP BY FORNAVN, ETTERNAVN
49 ORDER BY COUNT(BOKID) DESC;
50
51 #j
52 SELECT TITTEL FROM BOK
53 WHERE UTGITT R=
54     (SELECT MIN(UTGITT R)
55     FROM BOK) ;
56
57 #k
58 SELECT * FROM
59     (SELECT FORLAGNAVN,COUNT(BOKID) AS TOT
60     FROM BOK,FORLAG
61     WHERE FORLAG.FORLAGID=BOK.FORLAGID
62     GROUP BY FORLAGNAVN) AS A
63 WHERE TOT>2;

```

Listing 2: Oppgave 3

Oppgave 4

1. Hensikten med views er å forenkle komplekse søk, eller å begrense hva brukere kan se i en database. Et eksempel på forenkling er å lage et view som føyer sammen BOK og FORFATTER om BOKFORFATTER i Oppgave 3. På denne måte kan man enkelt hente ut informasjon om bok og forfatter via viewet.
vanskelighetene med å gjøre oppdateringer via views views kommer når man gjør endringer på data slik at det faller ut av viewet fordi det ikke lengre fyller kriteriene.
2. Viewet er definert i Listing 3.
3. Sql finnes i Listing 4. Nr 4 er ulovlig, de andre er lovlige.

```

1 CREATE VIEW MITTVIEW (PROSJNAVN,AVD,ANT_ANS,ANT_TIMER)
2 AS
3     SELECT PNAME,DNAME,COUNT(FNAME) ,SUM(HOURS)
4     FROM PROJECT, DEPARTMENT, EMPLOYEE, WORKS.ON
5     WHERE DEPARTMENT.DNUMBER=PROJECT.DNO
6     AND WORKS.ON.PNO=PROJECT.PNUMBER

```

```
7 | GROUP BY PNAME;
```

Listing 3: Oppgave 4.2

```
1 | #1 lovlig
2 | SELECT * FROM
3 |     (SELECT DNO, COUNT(*) , SUM(SALARY) , AVG(SALARY)
4 |     FROM EMPLOYEE
5 |     GROUP BY DNO) AS A;
6 |
7 | #2 lovlig
8 | SELECT A,B FROM
9 |     (SELECT DNO AS A, COUNT(*) AS B, SUM(SALARY) AS C
10 |     FROM EMPLOYEE
11 |     GROUP BY DNO) AS E
12 | WHERE C > 10000;
13 |
14 | #3
15 | UPDATE EMPLOYEE SET DNO=3 WHERE DNO=4;
16 |
17 | #4 ikke lovlig
18 | DELETE FROM EMPLOYEE
19 | WHERE DNO IN (SELECT DNO FROM
20 |     (SELECT DNO, COUNT(*) AS C
21 |     FROM EMPLOYEE GROUP BY DNO)
22 |     AS T1
23 | WHERE C > 4;
```

Listing 4: Oppgave 4.3

Oppgave 5

Indeksring kan brukes til å gjøre søk på spesifikke ting raskere. Et eksempel er å indeksere (“clustered”) på timestamp dersom man ofte skal gjøre spørringer som vil omhandle rader i tabellen som ligger nært hverandre i tid. Dette vil derimot skape tregghet dersom man er interessert i data som ikke har noen logisk fordeling i forbindelse med tid. Særlig ettersom “clustret” data blir lagret i den rekkefølgen som er logisk; i denne sammenhengen kronologisk.

Oppgave 6

Sql finnes i Listing 5

```
1 | #a)
2 | SELECT sno , sname FROM Supplier
3 | WHERE status > 15;
4 |
5 | #b)
6 | SELECT DISTINCT sname , Supplier.city
7 |     FROM Part , Supplier , SuppliesPart
8 | WHERE SuppliesPart.sno=Supplier.sno
9 | AND Part.pno=SuppliesPart.pno
10 | AND LOWER(Part.pname)='screw';
11 |
12 | #c)
13 | SELECT * FROM
14 |     (SELECT Part.pno , Part.pname , COUNT(SuppliesPart.sno) AS B
```

```

15         FROM Part ,SuppliesPart
16        WHERE Part.pno=SuppliesPart.pno
17        GROUP BY Part.pno) AS A
18 WHERE B > 1;
19
20 #d)
21 SELECT COUNT(*) FROM Supplier;
22
23 #e)
24
25 SELECT Supplier.city FROM
26        Supplier NATURAL JOIN SuppliesPart
27        NATURAL JOIN Part
28 WHERE weight >10;
29
30 #f)
31 SELECT DISTINCT sname
32        FROM Part ,Supplier ,SuppliesPart
33 WHERE SuppliesPart.sno=Supplier.sno
34 AND Part.pno=SuppliesPart.pno
35 AND LOWER(Part.pname)!='screw'
36 ORDER BY sname ASC;

```

Listing 5: Oppgave 6