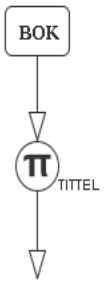
NTNU
Norges teknisk-naturvitenskapelige Universitet

Institutt for datateknikk og informasjonsvitenskap

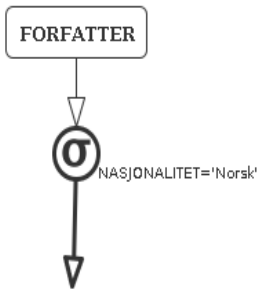
Fag TDT4145 Datamodellering og databasesystemer
Løsningsforslag til øving 3: Algebra og SQL

Oppgave 1

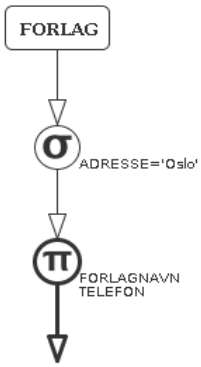
1)



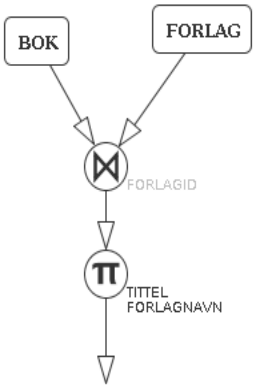
2)



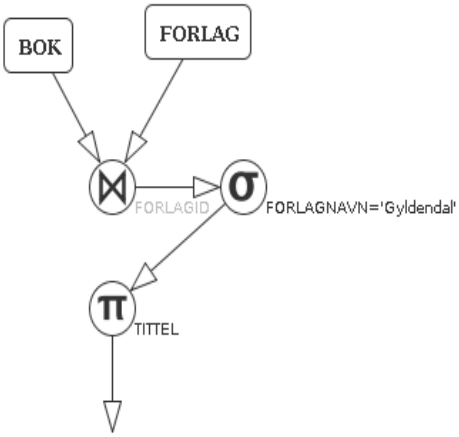
3)



4)

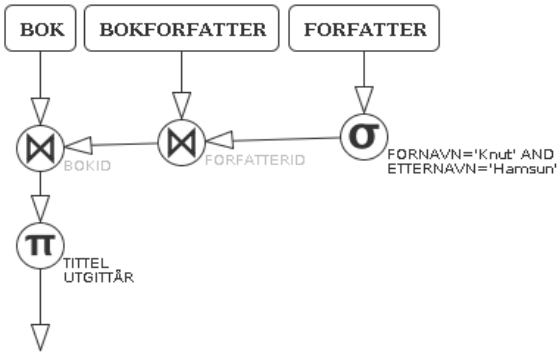


5)

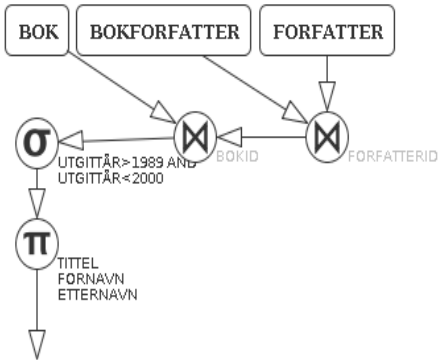


'

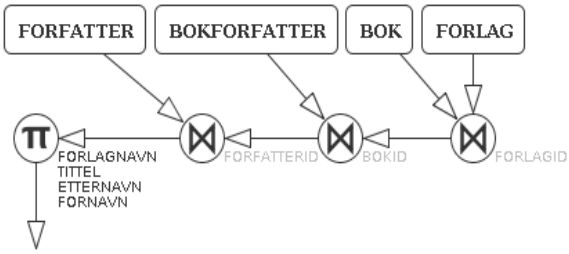
6)



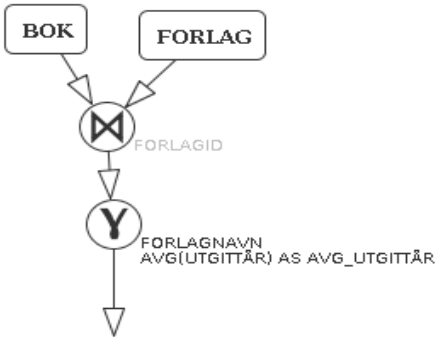
7)



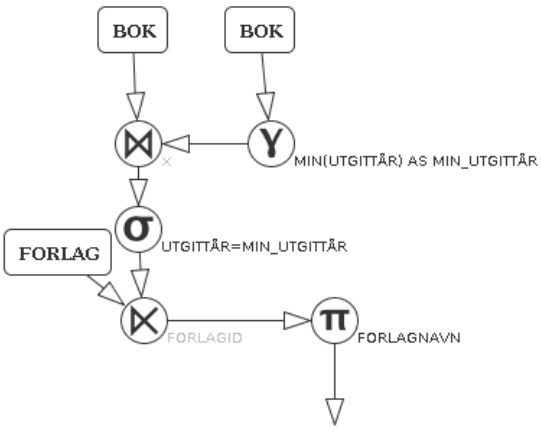
8)



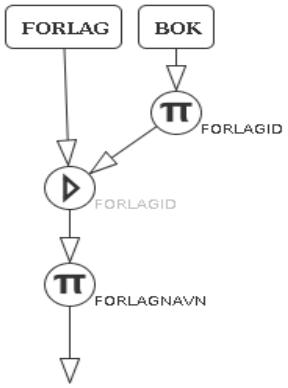
9)



10)



11)



Oppgave 2

α)
```
CREATE TABLE KUNDE
    (KUNDENR                INT              NOT NULL,
     NAVN                   VARCHAR(20)      NOT NULL,
     KREDITTGRENSE          INT              NOT NULL,
     POSTNR                 CHAR(4),
     PRIMARY KEY (KUNDENR),
     FOREIGN KEY  (POSTNR) REFERENCES POSTSTED(POSTNR)
    );

 CREATE TABLE POSTSTED
    (POSTNR                 CHAR(4)          NOT NULL,
     POSTSTED               VARCHAR(20)      NOT NULL,
     PRIMARY KEY (POSTNR)
    );

CREATE TABLE BESTILLING
    (ARTNR                  CHAR(1)          NOT NULL,
     KUNDENR                INT              NOT NULL,
     ANT                    INT              NOT NULL,
     PRIMARY KEY (ARTNR, KUNDENR),
     FOREIGN KEY (ARTNR) REFERENCES ARTIKKEL(ARTNR),
     FOREIGN KEY (KUNDENR) REFERENCES KUNDE(KUNDENR)
    );

CREATE TABLE ARTIKKEL
    (ARTNR                  CHAR(1)          NOT NULL,
     NAVN                   VARCHAR(20)      NOT NULL,
     ANT                    INT              NOT NULL,
     PRIS                   DECIMAL(5,2)     NOT NULL,
     PRIMARY KEY (ARTNR),
     UNIQUE (NAVN)
    );
```

β) For å forplante endringer i KUNDE eller ARTIKKEL til tabellen BESTILLING, kan vi endre deklarasjonene av fremmednøklene slik at vi oppnår ønsket effekt:

```
CREATE TABLE BESTILLING
   (  …
    FOREIGN KEY (ARTNR) REFERENCES ARTIKKEL(ARTNR)
       ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (KUNDENR) REFERENCES KUNDE(KUNDENR)
       ON DELETE CASCADE ON UPDATE CASCADE
    );
```

For postnumre virker det mest naturlig å la endringer forplantes til KUNDE, mens vi lar være å gjøre noe hvis et poststed blir nedlagt (slettet) slik at brukerne kan håndtere situasjonen:

```
CREATE TABLE KUNDE
   ( …
```

```
    FOREIGN KEY (POSTNR) REFERENCES POSTSTED(POSTNR)
       ON UPDATE CASCADE
   );
```

χ) For å undersøke om kravet er oppfylt må vi gjøre en spørring som involverer aggregering og som trenger data fra flere tabeller. Vi kan ikke få spesifisert kravet som en del av tabelldefinisjonene i **a)** og vi må lage en generell regel, en såkalt ”assertion”:

```
CREATE ASSERTION KREDITT_BEGRENSNING
CHECK ( NOT EXIST (
                SELECT   K.KUNDENR, K.KREDITTGRENSE,
                         SUM(B.KVANTUM * A.PRIS)
                FROM     KUNDE K, BESTILLING B, ARTIKKEL A
                WHERE    B.ARTNR = A.ARTNR
                  AND    B.KUNDENR = K.KUNDENR
                GROUP BY K.KUNDENR, K.KREDITTGRENSE
                HAVING   SUM(B.KVANTUM * A.PRIS) > KREDITTGRENSE));
```

Legg merke til at vi må ta med KREDITTGRENSE i SELECT-linja for å kunne bruke den i HAVING-betingelsen.

Oppgave 3

a)
```
SELECT TITTEL FROM BOK;
```

b)
```
SELECT * FROM FORFATTER
   WHERE NASJONALITET='Norsk';
```

c)
```
SELECT FORLAGID, FORLAGNAVN, TELEFON FROM FORLAG
   WHERE ADRESSE='Oslo'
   ORDER BY FORLAGNAVN;
```

d)
```
SELECT TITTEL, FORLAGNAVN FROM BOK, FORLAG
   WHERE BOK.FORLAGID=FORLAG.FORLAGID;
```

e)
```
SELECT TITTEL, UTGITTÅR FROM BOK, BOKFORFATTER, FORFATTER
   WHERE BOK.BOKID=BOKFORFATTER.BOKID AND
   BOKFORFATTER.FORFATTERID=FORFATTER.FORFATTERID AND
   FORNAVN='Knut' AND ETTERNAVN='Hamsun';
```

f)
```
SELECT FORNAVN, ETTERNAVN, FØDEÅR FROM FORFATTER
   WHERE ETTERNAVN LIKE 'H%';
```

g)
```
SELECT COUNT(*) FROM FORLAG;
```

h)
```
SELECT TITTEL, FORNAVN, ETTERNAVN, FORLAGNAVN
   FROM BOK, FORLAG, FORFATTER, BOKFORFATTER
   WHERE BOK.BOKID=BOKFORFATTER.BOKID AND
   BOK.FORLAGID=FORLAG.FORLAGID AND
   NASJONALITET='Britisk';
```

i)
```
SELECT FORFATTER.FORNAVN, FORFATTER.ETTERNAVN, COUNT(*)
   FROM BOK, FORFATTER, BOKFORFATTER
   WHERE BOK.BOKID=BOKFORFATTER.BOKID AND
   BOKFORFATTER.FORFATTERID=FORFATTER.FORFATTERID
   GROUP BY FORFATTER.FORNAVN, FORFATTER.ETTERNAVN
   ORDER BY COUNT(*) DESC;
```

j)
```
SELECT TITTEL, UTGITTÅR FROM BOK
   WHERE UTGITTÅR=(SELECT MIN(UTGITTÅR) FROM BOK);
```

k)
```
SELECT F.FORLAGNAVN, COUNT(*) FROM FORLAG F, BOK B
   WHERE B.FORLAGID=F.FORLAGID
   GROUP BY F.FORLAGNAVN
   HAVING COUNT(*)>2;
```

l)
```
SELECT F.FORLAGNAVN FROM FORLAG F
WHERE F.FORLAGID NOT IN(
   SELECT F.FORLAGID FROM FORLAG F, BOK B
   WHERE F.FORLAGID=B.FORLAGID);
```

Oppgave 4

a) Hensikten med virtuelle tabeller (views) er å definere nye abstraksjoner basert på eksisterende tabeller for å forenkle bruken av tabellene. Views kan også benyttes som en sikkerhetsmekanisme.

Problemet med views er rettet mot oppdaterbarhet. Det kan ofte være vanskelig eller umulig å definere hvordan en oppdatering gjennom et view skal avbildes på basistabellene. I det generelle tilfelle vil man derfor forby oppdateringer gjennom views. De samme problemene gjelder også for slettinger og innsettninger. I noen tilfeller kan imidlertid manipulering gjennom views gis en entydig tolkning overfor basis-tabellene.

b)
```
CREATE VIEW PROJECT2 (PNAME, DNAME, NO_OF_EMPLS, TOTAL_HOURS)
   AS SELECT PNAME, DNAME, COUNT(*), SUM(HOURS)
      FROM   PROJECT, DEPARTMENT,WORKS_ON
      WHERE  DNUM = DNUMBER
        AND  PNUMBER = PNO
      GROUP BY PNAME;
```

c) 1)
```
SELECT   DNO, COUNT(*), SUM(SALARY), AVG(SALARY)
FROM     EMPLOYEE
GROUP BY DNO;
```

2)
```
SELECT   DNO, COUNT(*)
FROM     EMPLOYEE
GROUP BY DNO
HAVING   SUM(SALARY) > 10000;
```

3) Viewet er definert over en tabell, men inneholder aggregater, og det er derfor ikke oppdatertbart, Slike som viewet er definert, er det en tabell med avdelingsnummer og aggregerte data om ansatte. Hvis vi ønsker å endre et avdelingsnummer, bør vi oppdatere DEPARTEMENT og sørge for at EMPLOYEE blir oppdatert etter hensikten.

4) Samme betraktninger om oppdateringer av vews med aggregater som i **c).** Skal vi slette avdelinger, må vi oppdatere DEPARTMENT-tabellen og deretter slette eller bytte avdeling for de ansatte.

Oppgave 5

a)
```
SELECT * FROM Supplier WHERE status>15;
```

b)
```
SELECT sname, s.city
FROM Supplier s, SuppliesPart sp, Part p
WHERE s.sno=sp.sno AND sp.pno=p.pno AND p.pname='Screw';
```

Evt. SELECT DISTINCT. Kan like gjerne ha et nøstet IN-query her:

```
SELECT  sname, city
FROM Supplier
WHERE sno IN
    (SELECT sno
    FROM SupplierPart sp, Part p
    WHERE sp.pno=p.pno AND pname='Screw');
```

c)
```
SELECT pno, pname
FROM Part
WHERE pno IN
   (SELECT pno
    FROM SuppliesPart
    GROUP BY pno
    HAVING COUNT(*)>1);
```

d)
```
SELECT COUNT(*)
```

```
   FROM Supplier
```

e)
```
   SELECT s.city
   FROM Supplier s, Part p, SuppliesPart sp
   WHERE s.sno=sp.sno AND sp.pno=p.pno AND p.weight>10;
```

f)
```
   SELECT DISTINCT sname
   WHERE sno NOT IN
       (SELECT sp.sno
       FROM SuppliesPart sp, Part p
       WHERE sp.pno=p.pno AND p.pname='Screw')
   ORDER BY sname;
```