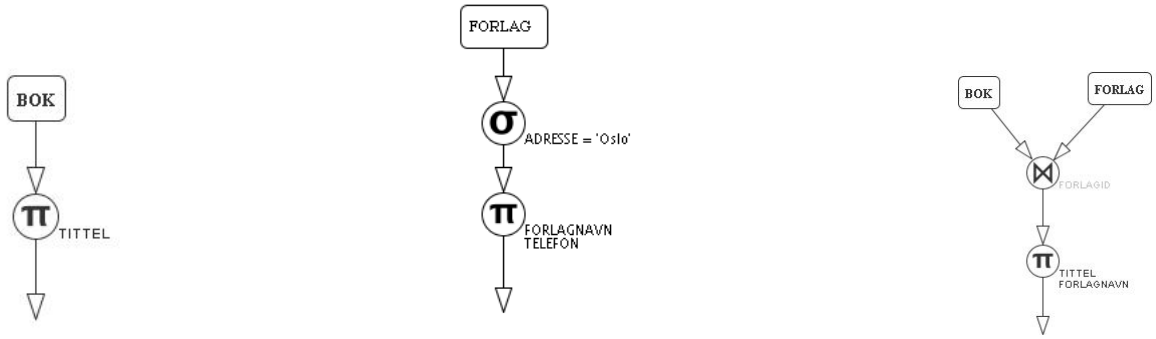
TDT4145 - Øving 3

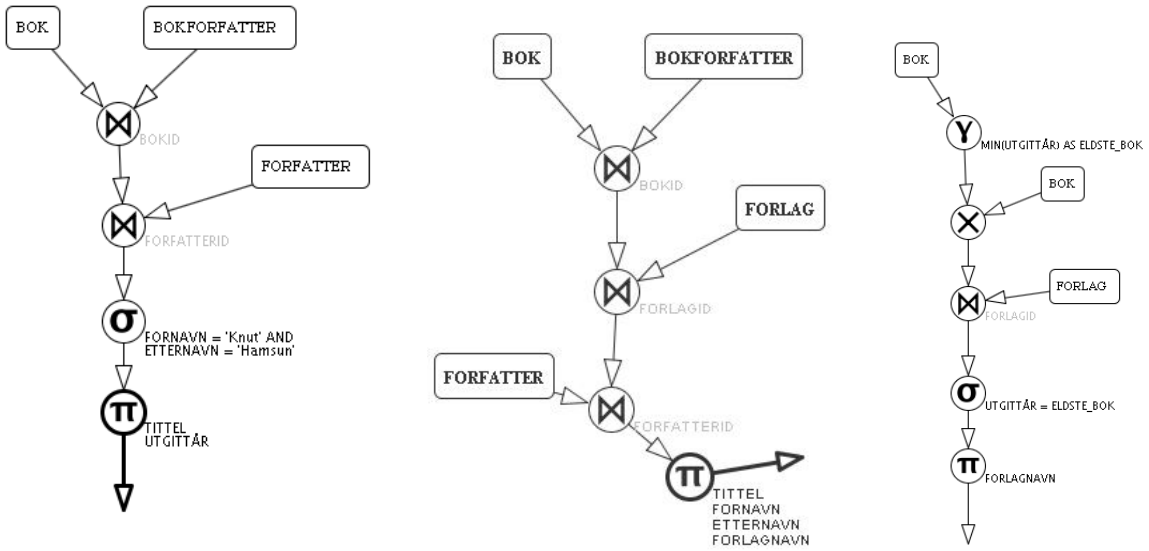
Håkon Ødegård Løvdal og Fredrik Christoffer Berg

17. februar 2014

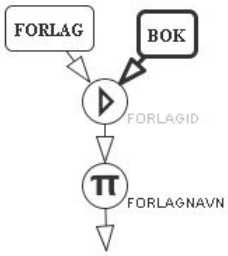
Oppgave 1



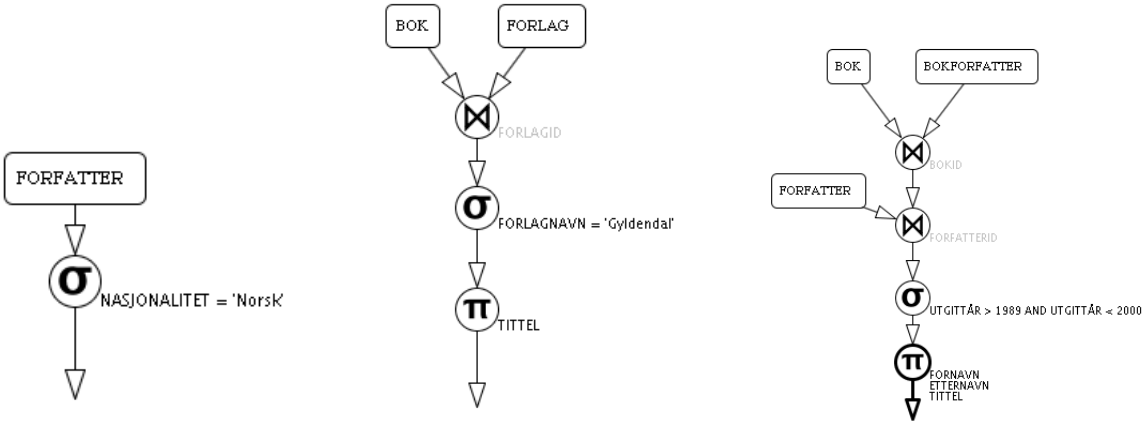
Figur 1: Relasjonsalgebra 1, 3 og 4



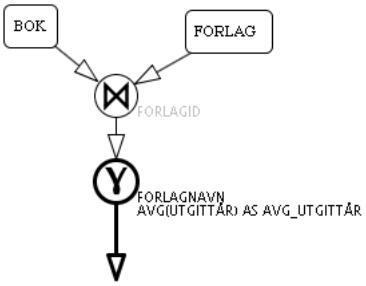
Figur 2: Relasjonsalgebra 6, 8 og 10



Figur 3: Relasjonsalgebra 11



Figur 4: Relasjonsalgebra 2, 5 og 7 (frivillig)



Figur 5: Relasjonsalgebra 9 (frivillig)

Oppgave 2

Deloppgave a

```
CREATE TABLE poststed (
        postnr char(4),
        poststed varchar(30) not null,
        CONSTRAINT poststed_pk PRIMARY KEY (postnr)
);

CREATE TABLE artikkel (
        artnr integer,
        navn varchar(30) not null,
        ant integer not null,
        pris integer not null,
        CONSTRAINT artikkel_pk PRIMARY KEY (artnr)
);

CREATE TABLE kunde (
        kundenr integer,
        navn varchar(30) not null,
        kredittgrense integer,
        postnr char(4) not null,
        CONSTRAINT kunde_pk PRIMARY KEY (kundenr),
        CONSTRAINT kunde_fk FOREIGN KEY (postnr) REFERENCES poststed(postnr)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);

CREATE TABLE bestilling (
        artnr integer,
        kundenr integer,
        kvantum integer not null,
        CONSTRAINT bestilling_pk PRIMARY KEY(artnr, kundenr),
        CONSTRAINT bestilling_fk1 FOREIGN KEY (artnr) REFERENCES artikkel(artnr
            )
                ON UPDATE CASCADE
                ON DELETE CASCADE,
        CONSTRAINT bestilling_fk2 FOREIGN KEY (kundenr) REFERENCES kunde(
            kundenr)
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
```

Deloppgave b

I SQL utrykker vi disse kravene ved følgende syntaks:
(Se oppgave a)

```
ON UPDATE CASCADE
ON DELETE CASCADE
```

Dersom et poststed blir fjernet, så er det naturlig at kunder med dette postnummeret, også får dette fjernet. Vi mener ikke det er naturlig at en kunde skal kunne definere et eget postnummer, oppgitte nummer må derfor eksistere i databasen.

Deloppgave c

I SQL er det mulig å lage assertions som sjekker en gitt betingelse som henter informasjon fra flere tabeller. Denne må være sann for at man kan sette inn verdiene. Dette er derimot ikke mulig i MySQL, så der kan man få det til med såkalte triggers isteden.

Vi benytter oss av en ASSERTION her, da det er det vi følger gir mest mening ifølge læreboka:

```
CREATE ASSERTION KREDITTSJEKK
CHECK ( NOT EXIST (
        SELECT k.kundenr, k.navn, k.kredittgrense
        FROM kunde k, bestilling b, artikkel a
        WHERE k.kundenr = b.kundenr
        AND a.artnr = b.artnr
        GROUP BY k.kundenr
        HAVING SUM(a.pris * b.kvantum) < k.kredittgrense));
```

Oppgave 3

```
3a)
SELECT tittel
FROM bok;

3b)
SELECT *
FROM forfatter
WHERE nasjonalitet = 'Norsk';

3c)
SELECT forlag, navn, telefon
FROM forlag
WHERE adresse = 'Oslo'
ORDER BY forlagnavn;

3e)
SELECT b.tittel, b.utgittår
FROM bok b, bokforfatter bf, forfatter f
WHERE b.bokid = bf.bokid
AND f.forfatterid = bf.forfatterid
AND f.fornavn = 'Knut'
AND f.etternavn = 'Hamsun';

3f)
SELECT fornavn, etternavn, fødeår
FROM forfatter
WHERE etternavn LIKE "H%";
```

```
3i)
SELECT f.fornavn, f.etternavn,
        (SELECT count(*)
         FROM bokforfatter bf
         WHERE bf.forfatterid = f.forfatterid)
AS "AntBok"
FROM forfatter f
ORDER BY AntBøker DESC;

3j)
SELECT tittel, utgittår
FROM bok b
ORDER BY utgittår ASC LIMIT 1;

3k)
SELECT f.forlagnavn,
        (SELECT count(*)
         FROM bok b
         WHERE f.forlagid = b.forlagid)
AS "AntBok"
FROM forlag f HAVING antbok > 2;

3l)
SELECT f.forlagnavn
FROM forlag f
WHERE f.forlagid NOT IN
        (SELECT b.forlagid
         FROM bok b);

######################
Frivillige oppgaver:
######################

3d)
SELECT b.tittel, f.forlagnavn
FROM bok b, forlag f
WHERE b.forlagid = f.forlagid;

3g)
SELECT count(forlagid) AS "AntForlag"
FROM forlag;


3h)
SELECT b.tittel, f.fornavn, f.etternavn, fl.forlagnavn
FROM bokforfatter bf, bok b, forfatter f, forlag fl
WHERE bf.bokid = b.bokid
AND bf.forfatterid = f.forfatterid
AND b.forlagid = fl.forlagid
AND f.nasjonalitet = 'Britisk';
```

Oppgave 4

Deloppgave a

Hensikten til et view er å kunne spesifisere virtuelle tabeller. Tuplene i slike tabeller er ikke nødvendigvis fysisk lagret i databasen. Ved å bruke views, kan vi hente ut tupler vi ofte trenger, men som ikke er direke lagret som en tabell i databasen. Samtidig kan vi ved å bruke views, begrense hvilke data ulike brukergrupper kan se.

Oppdatering av virtuelle tabeller kan føre til problemer med tanke på hvordan de underliggende basetabellene skal oppdateres. F.eks kan en view update som involverer sammenslåtte tabeller, føre til usikkerhet rundt hvilke av feltene som skal oppdateres. Derfor kan det ofte være lurt å forby oppdatering gjennom view, og kun tillate updates på basetabeller.

Deloppgave b

```
CREATE VIEW DEPTSUMMARY(PNAME, DEPTNUM, TOTEMP, TOTHOURS)
AS SELECT p.pname, p.dnum, count(*), sum(w.hours)
        FROM project p, works_on w
        WHERE p.pnumber = w.pno
        GROUP BY p.pnumber
```

Deloppgave c

1

Lovlig: her kjøres hele select-setningen fra viewet

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

2

Lovlig: her kjøres følgende spørring på basistabellene

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

3

Ulovlig: Views definert med gruppering- og aggreingsfunksjoner er ikke oppdaterbare

4

Ulovlig: Views definert med gruppering- og aggreingsfunksjoner er ikke slettbare

Oppgave 5

```
a)

SELECT sno, sname, status, city
FROM Supplier
HAVING status > 15;

b)

SELECT s.sname, s.city
FROM Supplier s
JOIN SuppliesPart sp ON s.sno = sp.sno
JOIN Part p ON sp.pno = p.pno
WHERE p.pname = 'Screw';

c)

SELECT p.pno, p.pname
FROM Part p, SuppliesPart sp
WHERE sp.pno = p.pno
GROUP BY pname
HAVING COUNT(*) > 1;

d)

SELECT count(*) AS "AntLev"
FROM Supplier;

e)

SELECT DISTINCT s.city
FROM Supplier s
JOIN SuppliesPart sp ON s.sno = sp.sno
JOIN Part p ON p.pno = sp.pno
WHERE p.weight > 10.0;

Alternativt:
SELECT DISTINCT s.city
FROM Supplier s, Part p, SuppliesPart sp
WHERE s.sno = sp.sno
AND p.pno = sp.pno
AND p.weight > 10.0;

(Disse returnerer et tomt sett, da ingen del i databasen
 veier mer enn 2.0)
(Legger også merke til at det er deler som forsvinner fordi
 delene ikke eksisterer SuppliesPart, så man kan ikke få ut leverandørby)

f)

SELECT s.sname
```

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