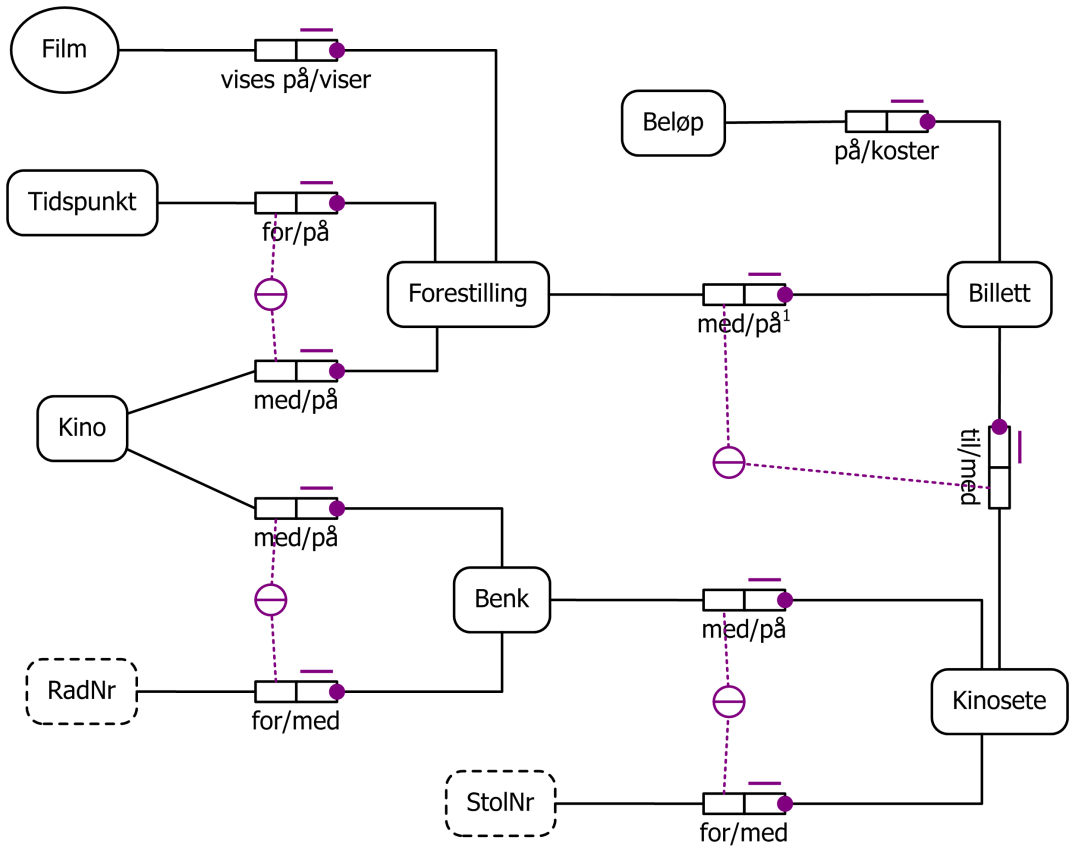
Ekvivalente stier (Equivalence of Path, EOP) i stORM

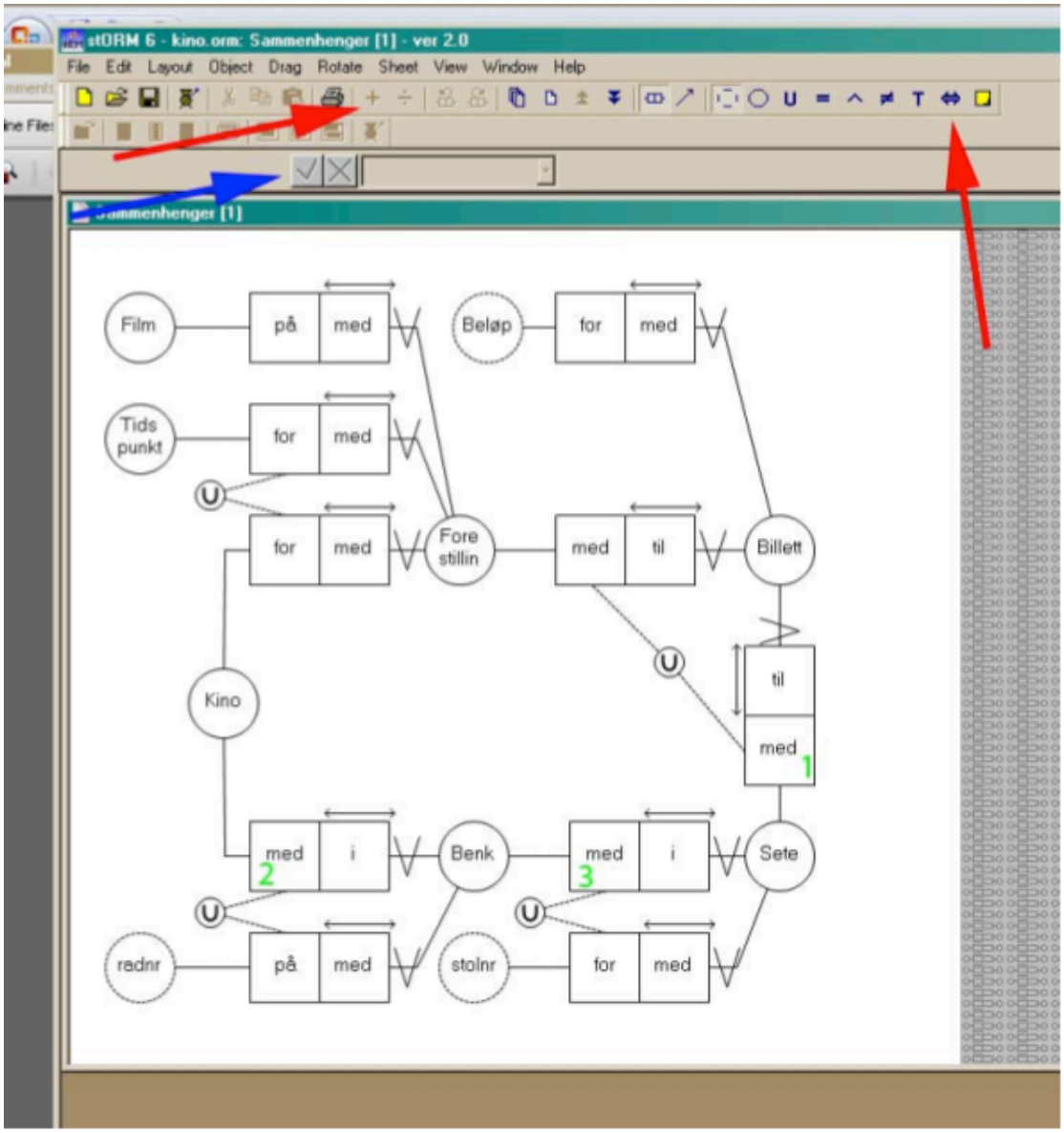
Dette er ikke rett fram, derfor denne beskrivelsen. Vi tar utgangspunkt i følgende modell for kinoer og kinoforestillinger:



[1]en Billett *på* en Forestilling *på* en bestemt Kino
er *til* et Kinosete *på* en Benk *på* den samme Kinoen

Bilde 1 - ORM2-modell

I bildet under ser du modellen og litt av skjermbildet. Merk spesielt de to røde pilene, som peker ut verktøy som skal brukes i prosessen å definere en EOP, samt den blå pila som viser statusfeltet og et verktøy.



Bilde 2 - Modellen og skjermbildet

Vi ser av modellen at referansen til Kino (kinonavn, definert ved perfekt bro på annet ark) inngår i referansen til både Benk og Forestilling. Videre inngår referansen til Benk i referansen til Sete, og til slutt fremkommer Billett som en begrepsdannelse over Sete og Forestilling. Det betyr at referansen til Kino blir

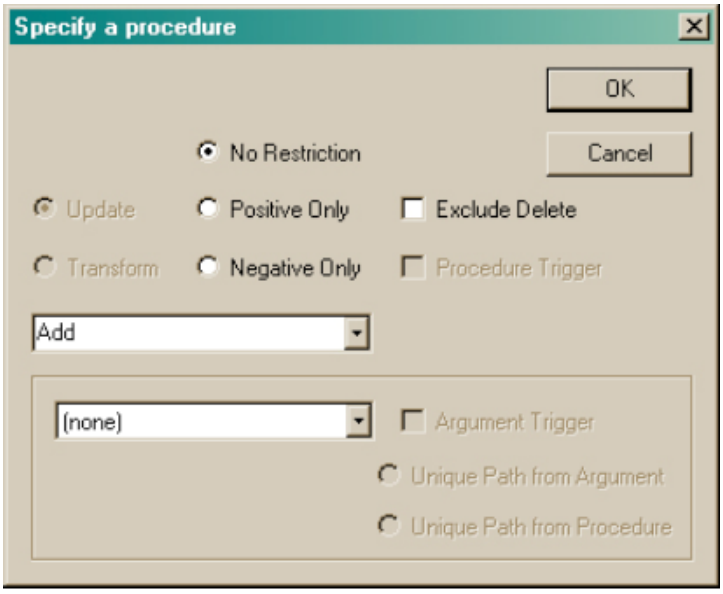
gruppert inn to ganger til Billett. I og med at det er et stort poeng at disse to feltene har samme verdi i hver enkelt post i tabellen, må vi sikre denne likheten. De to begrepene Tidspunkt og Film er også definert ved perfekte broer (som hhv. ÅÅÅÅMMDDHHMM og filmnavn) på et annet ark, og etter å kjørt hele produksjonslinja gjennom grupperer og SQL-generator, ser tabellen Billett slik ut (automatikken bytter ut Å med Aa):

```
CREATE TABLE Billett (
   AaAaAaAaMMDDHHMM_med        VARCHAR(20) NOT NULL,
   Kinonavn_med                VARCHAR(20) NOT NULL,
   stolnr_med                  VARCHAR(20) NOT NULL,
   Kinonavn_med_B              VARCHAR(20) NOT NULL,
   radnr_med                   VARCHAR(20) NOT NULL,
   Beloep_for                  VARCHAR(20) NOT NULL
);
...
ALTER TABLE Billett ADD PRIMARY KEY
(AaAaAaAaMMDDHHMM_med,Kinonavn_med,stolnr_med,Kinonavn_med_B,radnr_med);
```

Vi ser her de to forekomstene av Kinonavn i primærnøkkelen til Billett; Kinonavn_med og Kinonavn_med_B. Det er disse to forekomstene som må være like i hver enkelt post i tabellen. Er de ikke det, så har du ikke gyldig billett til den filmen du ønsker å se: forestillingen går på Saga, men setet befinner seg på Colosseum. Upraktisk!

Vi må definere disse to stiene som ekvivalente. Før vi beskriver fremgangsmåten repeterer vi kjapt at grupperende rolle er den rollen som står under entydighetspila, refererende rolle er den som ikke står under entydighetspil. Fremgangsmåten er som følger, og det er her viktig at du gjør eksakt det som står beskrevet, ikke mer og ikke mindre:

1. Identifiser de to stiene. Start i Billett og klikk på den refererende rollen i en av stiene (i bilde 2 markert med et grønt 1-tall), vi starter f.eks. med den nederste. Klikk på den grønne +'en ved den røde pila i bilde 2. Du får da opp et dialogvindu:



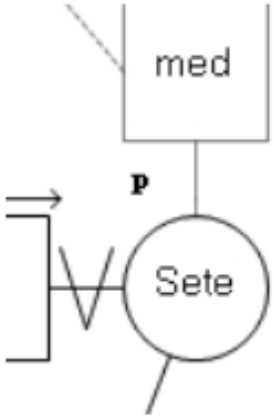
Bilde 3 - Prosedyrer

2. Velg Projection fra den nedtrekssmenyen der det nå står Add. Klikk OK.

3. Uten å klikke noe annet sted går du til den andre enden av stien (Kino) og klikker på den refererende rollen (i vårt tilfelle ”med”, markert med grønt 2-tall). Merk deg at nå kommer det opp en parameterliste i det feltet som er markert med blå pil i bilde 2

Bilde 4 - Parameterliste

og det kommer opp en ”P” ved den rollen der du definerte en projeksjon:



Bilde 5 - Projeksjon

4. Fortsatt uten å klikke noe sted går du ett hopp i retning av Billett og klikker på refererende rolle (den heter også ”med”, markert med et grønt 3-tall). Vi merker oss at Parameterlista da blir utvidet med ett nytt attributt:

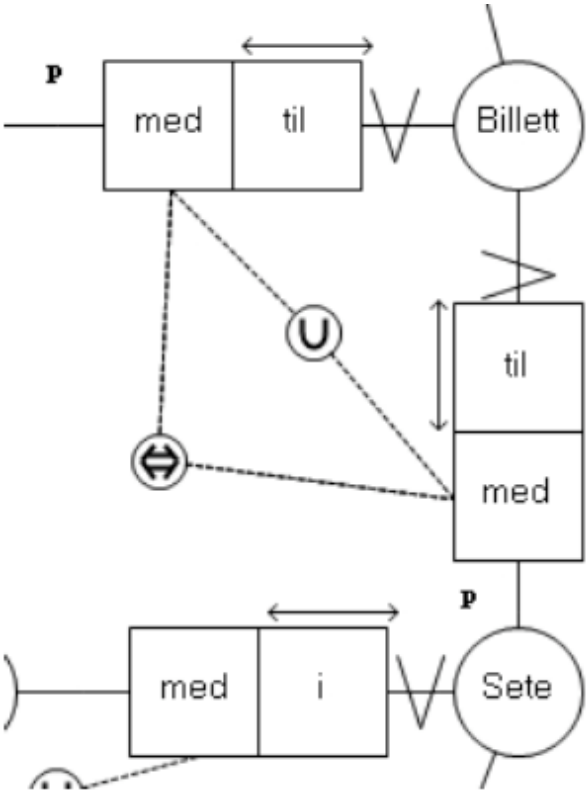
Bilde 6 - Lengre parameterliste

5. Klikk så på den grønne haken til venstre for parameterlista. Du har nå definert en sti.

6. Gjør akkurat det samme langs den andre stien. På samme måte som i den første stien vil du få en ”P” ved den rollen du definerer en projeksjon for.

7. Velg ekvivalens-verktøyet fra verktøylinja (rød pil i bilde 2) og sett inn beskrankningen på et passende sted i modellen.

8. Trekk ut bena til de to rollene som nå er markert med en P, og du er ferdig. Den viktigste biten av modellen ser nå slik ut:



Bilde 7 - Ekvivalente stier er definert

Når du nå kjører automatisk gruppering og SQL-generering, vil du se at definisjonen for tabellen Billett kun inneholder én forekomst av attributtet Kinonavn:

```
CREATE TABLE Billett (
   AaAaAaAaMMDDHHMM_med        VARCHAR(20) NOT NULL,
   Kinonavn_med                VARCHAR(20) NOT NULL,
   stolnr_med                  VARCHAR(20) NOT NULL,
   radnr_med                   VARCHAR(20) NOT NULL,
   Beloep_for                  VARCHAR(20) NOT NULL
);
...
ALTER TABLE Billett ADD PRIMARY KEY
(AaAaAaAaMMDDHHMM_med,Kinonavn_med,stolnr_med,radnr_med);
```

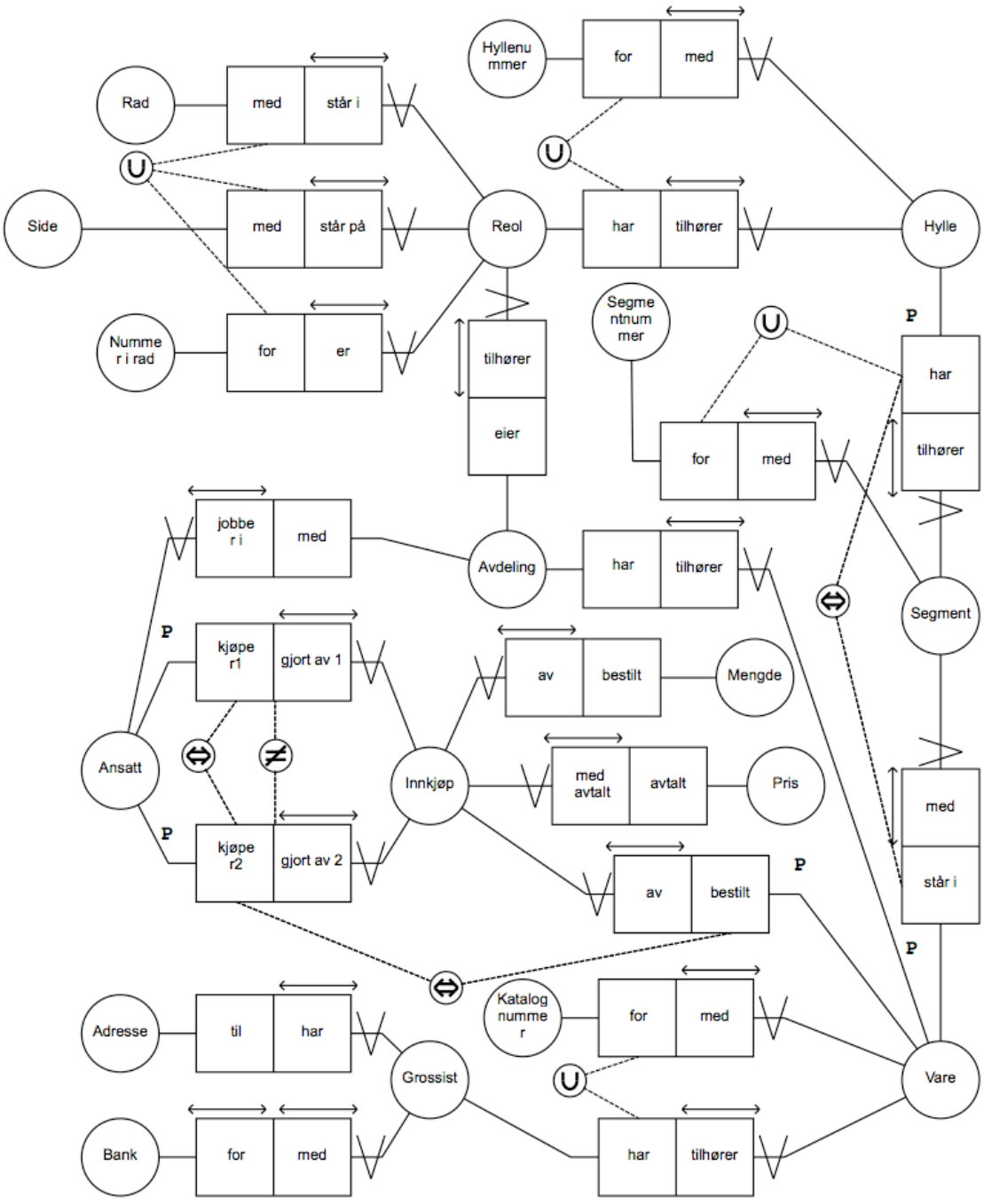
Du vil nå aldri få problemer med at forestillingen går på Saga, mens kinosetet er på Colosseum.

Så vil muligens noen stille spørsmålet ”men hvorfor alt dette, kan vi ikke bare slette og endre det som skal gjøres i SQL-fila?”. Mulig det, hvis du vet akkurat hva du skal gjøre. Og da sitter du med en SQL-fil som ikke stemmer overens med dokumentasjonen (datamodellen din). Opp med handa den som synes det høres ut som en god idé.

NB! *I vår versjon av stORM virker oppskriften over bare hvis EOPen kan håndheves ved å gjøre endringer i én tabell (tabellen til begrepet der de to stiene starter; i eksempelet over er dette Billett). Andre EOPer støttes ikke.*

I bildet under vises et annet eksempel (der stORM ikke støtter håndheving av EOPer): Det inneholder en bit av en modell for administrasjon av super-markeder. Her er det følgende EOPer:

1. Segment_tilhører→Hylle_tilhører→Reol_tilhører→Avdeling og
   Segment_med→Vare_tilhører→Avdeling:
   Et segment av en hylle i en reol inneholder én type varer. Alle varer av samme type tilhører én avdeling. Dette skal være samme avdeling som den som administrerer reolen der hyllesegmentet befinner seg.
2. Innkjøp_gjort_av_1→Ansatt_jobber_i→Avdeling og
   Innkjøp_gjort_av_2→Ansatt_jobber_i→Avdeling:
   De to ansatte som sammen må undertegne innkjøpet av en vare, må tilhøre samme avdeling.
3. Innkjøp_av→Vare_tilhører→Avdeling og
   Innkjøp_gjort_av_2→Ansatt_jobber_i→Avdeling:
   Innkjøp av en vare til en avdeling må gjøres av ansatte i avdelingen.



Bilde 8 - Administrasjon av supermarkeder

Disse EOPene vil involvere to eller flere tabeller. F.eks. innebærer den første EOPen at hvis vi tar en join mellom tabellene Segment, Hylle, Reol og Vare (tabellene er gjengitt under, fremmednøklene indikerer hvilke attributter som det skal joines på), så skal vi for hvert tuppel i resultattabellen ha at `Avdeling#_eier` (fra Reol) og `Avdeling#_har` (fra Vare) alltid har samme verdi. Selv om stORM godtar at vi definerer projeksjoner og ekvivalenser mellom projeksjonene i designfasen, kommer den med feilmeldinger knyttet til EOPene om vi forsøker å gruppere diagrammet.

```
CREATE TABLE Segment (
   Segment#_for               VARCHAR(20) NOT NULL,
   Rad#_har                   VARCHAR(20) NOT NULL,
   Side#_har                  VARCHAR(20) NOT NULL,
   Nr_i_rad_har               VARCHAR(20) NOT NULL,
   Hylle#_har                 VARCHAR(20) NOT NULL,
   Katalog#_staar_i           VARCHAR(20) NOT NULL,
   Grossistnavn_staar_i       VARCHAR(20) NOT NULL
);
ALTER TABLE Segment ADD PRIMARY KEY
   (Segment#_for,Rad#_har,Side#_har,Nr_i_rad_har,Hylle#_har);
ALTER TABLE Segment ADD CONSTRAINT Hylle_Segment
   FOREIGN KEY (Rad#_har,Side#_har,Nr_i_rad_har,Hylle#_har)
   REFERENCES Hylle (Rad#_har,Side#_har,Nr_i_rad_har,Hylle#_for);
ALTER TABLE Segment ADD CONSTRAINT Vare_Segment
   FOREIGN KEY (Katalog#_staar_i,Grossistnavn_staar_i)
   REFERENCES Vare (Katalog#_for,Grossistnavn_har);

CREATE TABLE Hylle (
   Rad#_har                   VARCHAR(20) NOT NULL,
   Side#_har                  VARCHAR(20) NOT NULL,
   Nr_i_rad_har               VARCHAR(20) NOT NULL,
   Hylle#_for                 VARCHAR(20) NOT NULL
);
ALTER TABLE Hylle ADD PRIMARY KEY
   (Rad#_har,Side#_har,Nr_i_rad_har,Hylle#_for);
ALTER TABLE Hylle ADD CONSTRAINT Reol_Hylle
   FOREIGN KEY (Rad#_har,Side#_har,Nr_i_rad_har)
   REFERENCES Reol (Rad#_med,Side#_med,Nr_i_rad_for);

CREATE TABLE Reol (
   Rad#_med                   VARCHAR(20) NOT NULL,
   Side#_med                  VARCHAR(20) NOT NULL,
   Nr_i_rad_for               VARCHAR(20) NOT NULL,
   Avdeling#_eier             VARCHAR(20) NOT NULL
);
ALTER TABLE Reol ADD PRIMARY KEY (Rad#_med,Side#_med,Nr_i_rad_for);
ALTER TABLE Reol ADD CONSTRAINT Avdeling_Reol
   FOREIGN KEY (Avdeling#_eier) REFERENCES Avdeling (Avdeling#_paa);

CREATE TABLE Vare (
   Katalog#_for               VARCHAR(20) NOT NULL,
   Grossistnavn_har           VARCHAR(20) NOT NULL,
   Avdeling#_har              VARCHAR(20) NOT NULL
);
ALTER TABLE Vare ADD PRIMARY KEY (Katalog#_for,Grossistnavn_har);
```