# **Oblig 3 Databaser og nettverk 18.11.2021 kl. 12:00**

24.11.2021

1. **MongoDB**   
   I oppgave 1 trenger du data fra oppgave 1 i Oblig1.
   1. Opprett en MongoDB database **Oblig3** med en dokumentsamling (**Collection**) **Film**.

use oblig3;  
db.createCollection('film');

* 1. Lag **JSON**-objekter og sett inn data med følgende nøkler:  
     **FilmNr, Tittel, År, Land, Sjanger, Alder, Tid, Pris, Antall**Selve datene tar dere fra **Oblig1**, **oppgave 1**. Ingen **Pris** er uten verdi, **Antall** angir filmer på lager og kan godt ha verdien **0**.

db.film.remove({});

db.film.**insertMany**([  
{FilmNr: 1, Tittel: 'Casablanca', År: 1942, Land: 'USA', Sjanger: 'Drama',  
 Alder: 15, Tid: 102, Pris: 149.00, Antall: 3},  
{FilmNr: 2, Tittel: 'Fort Apache', År: 1948, Land: 'USA', Sjanger: 'Western',  
 Alder: 15, Tid: 127, Pris: 144.00, Antall: 0},  
{FilmNr: 3, Tittel: 'Apocalypse Now', År: 1979, Land: 'USA', Sjanger: 'Action',  
 Alder: 18, Tid: 155, Pris: 123.00, Antall: 2},  
{FilmNr: 4, Tittel: 'Streets of Fire', År: 1984, Land: 'USA', Sjanger: 'Action',  
 Alder: 15, Tid: 93, Pris: 124.00, Antall: 0},  
{FilmNr: 5, Tittel: 'High Noon', År: 1952, Land: 'USA', Sjanger: 'Western',  
 Alder: 15, Tid: 85, Pris: 123.00, Antall: 2},  
{FilmNr: 6, Tittel: 'Cinema Paradiso', År: 1988, Land: 'Italia', Sjanger: 'Komedie',  
 Alder: 11, Tid: 111, Pris: 123.00, Antall: 2},  
{FilmNr: 7, Tittel: 'Asterix hos britene', År: 1988, Land: 'Frankrike', Sjanger: 'Tegnefilm',  
 Alder: 7, Tid: 78, Pris: 149.00, Antall: 4},  
{FilmNr: 8, Tittel: "Veiviseren", År: 1987, Land: "Norge", Sjanger: "Action",  
 Alder: 15, Tid: 96, Pris: 87.00, Antall: 2},  
{FilmNr: 9, Tittel: 'Salmer fra kjøkkenet', År: 2001, Land: 'Norge', Sjanger: 'Komedie',  
 Alder: 7, Tid: 80, Pris: 149.00, Antall: 1},  
{FilmNr: 10, Tittel: 'Anastasia', År: 1997, Land: 'USA', Sjanger: 'Tegnefilm',  
 Alder: 7, Tid: 94, Pris: 123.00, Antall: 3},  
{FilmNr: 11, Tittel: 'La Grande bouffe', År: 1973, Land: 'Frankrike', Sjanger: 'Drama',  
 Alder: 15, Tid: 129, Pris: 87, Antall: 1},  
{FilmNr: 12, Tittel: 'Blues Brothers 2000', År: 1998, Land: 'USA', Sjanger: 'Komedie',  
 Alder: 11, Tid: 124, Pris: 135.00, Antall: 3},  
{FilmNr: 13, Tittel: 'Beatles: Help', År: 1965, Land: 'Storbritannia', Sjanger: 'Musikk',  
 Alder: 11, Tid: 144, Pris: 153.00, Antall: 0}  
]);

* 1. Henter Tittel, Sjanger og Pris for filmer som er produsert i 1988 eller seinere, sortert synkende på pris (=dyrest først).  
     db.film.find({År:{$gte:1988}}, { \_id:0, Tittel:1,Sjanger:1,Pris:1}). sort({Pris:-1}).pretty();
  2. Henter alle kolonner for filmer som ikke er til salgs (tomt på lager), sortert på alder og sjanger.  
     db.film.find({Antall: 0}).sort({Alder: 1},{Sjanger: 1}).pretty();

db.film.find({Antall: 0}, { \_id:0}).sort({Alder: 1},{Sjanger: 1}).pretty();

* 1. Finner antall filmer som er til salgs i hver sjanger, og summen av prisen

db.film.aggregate(

{ $group: {

\_id: "$Sjanger",

TotalPris: { $sum: '$Pris' }

}

});

* 1. Setter inn en ny rad. Finn på data selv.

db.film.**insert**(  
{FilmNr: 14, Tittel: 'De syv samuraier', År: 1955, Land: 'Japan', Sjanger: 'Action',  
 Alder: 15, Tid: 150, Pris: 149.00, Antall: 1});

db.film.**insert**(  
{FilmNr: 15, Tittel: 'Jorden rundt på 80 dager', År: 2021, Land: 'USA', Sjanger: 'Animasjon',  
 Alder: 7, Tid: 82, Pris: 120.00, Antall: 4});

* 1. Korrigerer tittelen på filmen "High Noon". Den heter egentlig "High Moon".  
     To versjoner:

db.film.update({Tittel: 'High Noon'},   
 { FilmNr: 5, Tittel: 'High Moon', År: 1952, Land: 'USA', Sjanger: 'Western',  
 Alder: 15, Tid: 85, Pris: 123.00, Antall: 2}); hele posten oppdateres  
db.film.update({Tittel: 'High Noon'}, { $set: {Tittel: 'High Moon'}}); bare aktuelt felt

* 1. Legger på 10% på prisen på alle Action filmer

db.film.updateMany(

{Sjanger: 'Action'},

[{ $set: { Pris: { $multiply: [ 1.1, "$Pris" ] } } }],

);  
db.film.update(

{Sjanger: 'Action'},

[{ $set: { Pris: { $multiply: [ 1.1, "$Pris" ] } } }],

{ multi: true }

);

* 1. Sletter filmen Anastasia.  
     db.film.remove({Tittel: 'Anastasia'});

1. **Mere MongoDB**
2. En kunde med navn **Kari Mo** med adresse **Esperantogata 22** i **Hamar** (postnummer **2300**) har kjøpt to filmer, nemlig **Apocalypse Now** og **High Moon**.  
   Opprett en dokumentsamling **faktura**  og sett inn post med følgende nøkler:  
    {**Fornavn, Etternavn, Adresse, PostNr, Poststed,   
    Filmer: [{FilmNr, Tittel, Pris, Antall}]** (array med ett element for hver kjøpt film)   
    **FakturaNr**, **Dato** (fakturadato)}  
   db.faktura.insert({  
    Fornavn: 'Kari', Etternavn: 'Mo', Adresse: 'Esperantogata 22', PostNr: 2300, Poststed: Hamar',  
    Filmer: [{FilmNr: 3,Tittel: 'Apocalypse Now', Pris: 123, Antall: 1},   
    { FilmNr: 5, Tittel: 'High Moon', Pris: 123, Antall: 2}],  
    FakturaNr: 1, Dato: Date()});
3. Legg til fakturaer for de andre kundene du hadde i Oblig 1.
4. **SQL database**

|  |  |
| --- | --- |
| En relasjonsdatabase for de samme dataene som i oppgave 2 har (bare delvis) det ugyldige ER-diagrammet vist til høyre (se på det som et konseptuelt ER-diagram) |  |

1. Hva er feil med ER-diagrammet til høyre?

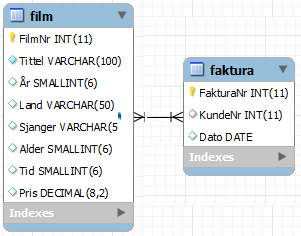
Vi kan ikke lage ER-diagrammer med mange-til-mange-relasjoner. (kan se på det som et konseptuelt ER-diagram, men vi får ikke til et ER-diagram med direkte mange-til-mange kobling ikke i Workbench. For i databasen MÅ vi da ha en koblingstabell)

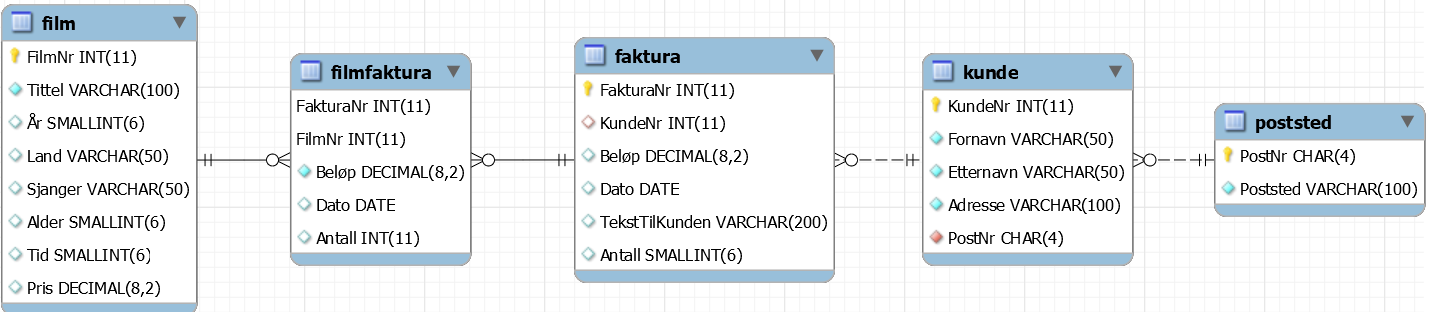
1. Angi avhengighetene i nøklene gitt over. Eller med andre ord: hvilke brudd på 1., 2. og 3. normalform finner du?

1. normalform: flere verdier av film

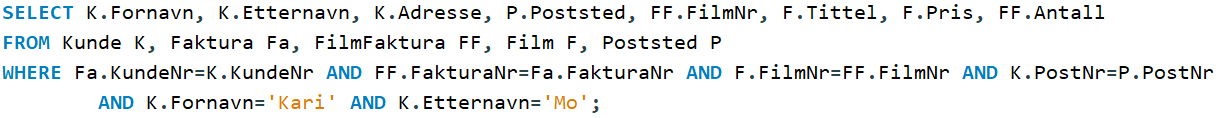
2. normalform: mange-til-mange-relasjonen mellom **film**-tabell og **faktura**-tabell. Velger **FakturaNr** og **FilmNr** som primærnøkkel. Det er delvise avhengigheter til **FakturaNr** og **FilmNr** slik at en ny tabell **filmfaktura** som vist på **ER**-diagrammet under løser problemet.

3. normalform er brutt siden **Poststed**  avhenger av **PostNr.**

1. Kompletter ER-diagrammet til høyre slik at databasen   
    oppfyller 1., 2. og 3. normalform.  
   Legger til **filmfaktura**-tabellen "mellom" **film**- og **faktura**-tabellene, med   
   (**FakturaNr,** **FilmNr**) som todelt primærnøkkel. Hver av dem opptrer også som fremmednøkler inn i de to tabellene "de kommer fra". I tillegg kan vi opprette en **Kunde**-tabell siden vi ser **KundeNr** opptrer som nøkkel i oppgavediagrammet, og forstås som fremmed nøkkel inn i en slik **Kunde**-tabell.  
   Et "standardbrudd" på 3NF er **Poststed**  og **PostNr**, og disse har også fått en egen tabell der **PostNr** er fremmed nøkkel inn i **Kunde**-tabell.



1. Skriv for Kari Mo: navn, adresse (med poststed), sammen med de filmer hun har kjøpt.

LØSN:

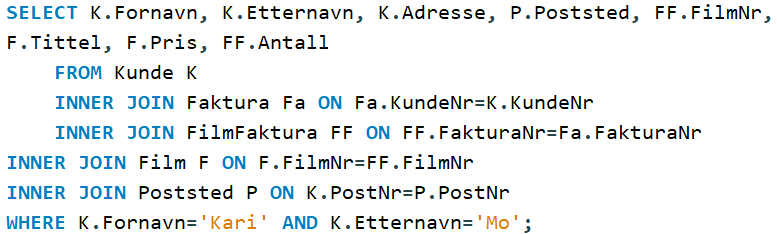
SELECT K.Fornavn, K.Etternavn, K.Adresse, P.Poststed, FF.FilmNr, F.Tittel, F.Pris, FF.Antall

FROM Kunde K, Faktura Fa, FilmFaktura FF, Film F, Poststed P

WHERE Fa.KundeNr=K.KundeNr AND FF.FakturaNr=Fa.FakturaNr AND F.FilmNr=FF.FilmNr AND K.PostNr=P.PostNr

AND K.Fornavn='Kari' AND K.Etternavn='Mo';

Eller:



SELECT K.Fornavn, K.Etternavn, K.Adresse, P.Poststed, FF.FilmNr, F.Tittel, F.Pris, FF.Antall

FROM Kunde K

INNER JOIN Faktura Fa ON Fa.KundeNr=K.KundeNr

INNER JOIN FilmFaktura FF ON FF.FakturaNr=Fa.FakturaNr

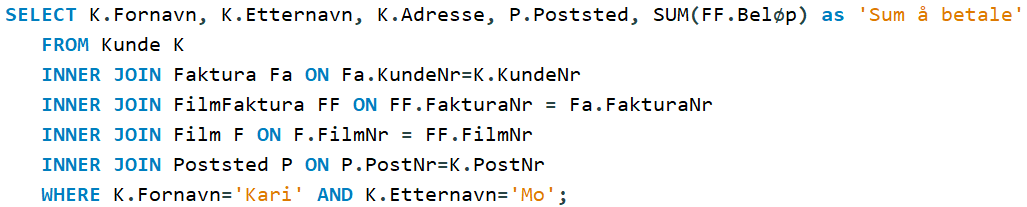
INNER JOIN Film F ON F.FilmNr=FF.FilmNr

INNER JOIN Poststed P ON K.PostNr=P.PostNr

WHERE K.Fornavn='Kari' AND K.Etternavn='Mo';

1. Skriv ut totalsum Kari Mo får på sin faktura.

LØSN:



SELECT K.Fornavn, K.Etternavn, K.Adresse, P.Poststed, SUM(FF.Beløp) as 'Sum å betale'

FROM Kunde K

INNER JOIN Faktura Fa ON Fa.KundeNr=K.KundeNr

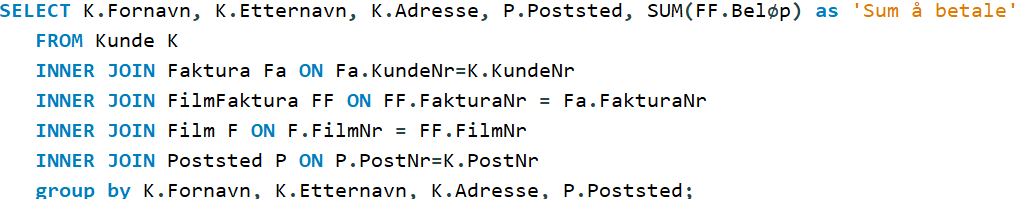
INNER JOIN FilmFaktura FF ON FF.FakturaNr = Fa.FakturaNr

INNER JOIN Film F ON F.FilmNr = FF.FilmNr

INNER JOIN Poststed P ON P.PostNr=K.PostNr

WHERE K.Fornavn='Kari' AND K.Etternavn='Mo';

Dersom en skal skrive ut fakturabeløp for flere personer (her skrives alle i basen ut), må en bruke goup by:



SELECT K.Fornavn, K.Etternavn, K.Adresse, P.Poststed, SUM(FF.Beløp) as 'Sum å betale'

FROM Kunde K

INNER JOIN Faktura Fa ON Fa.KundeNr=K.KundeNr

INNER JOIN FilmFaktura FF ON FF.FakturaNr = Fa.FakturaNr

INNER JOIN Film F ON F.FilmNr = FF.FilmNr

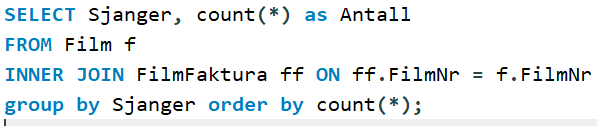
INNER JOIN Poststed P ON P.PostNr=K.PostNr

group by K.Fornavn, K.Etternavn, K.Adresse, P.Poststed;

Fyll inn noe mere data i basen så du får testet f. og g., bruk f.eks. data fra oblig 1 pluss noen ekstra fakturaer

1. Skriv ut antall som er solgt av hver sjanger i filmtabellen.

LØSN:



SELECT Sjanger, count(\*) as Antall

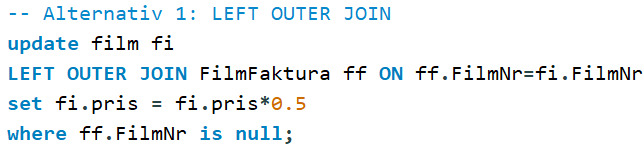
FROM Film f

INNER JOIN FilmFaktura ff ON ff.FilmNr = f.FilmNr

group by Sjanger order by count(\*);

1. Lag en en SQL som reduserer prisen med 50 %, men bare på de filmer som det ikke er solgt noen av.

LØSN: Vi må se på tabellen FilmFaktura for å se om det er solgt noe av en film.

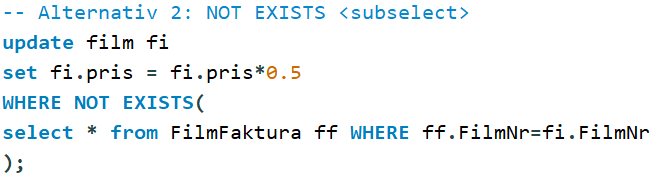


update film fi

LEFT OUTER JOIN FilmFaktura ff ON ff.FilmNr=fi.FilmNr

set fi.pris = fi.pris\*0.5

where ff.FilmNr is null;



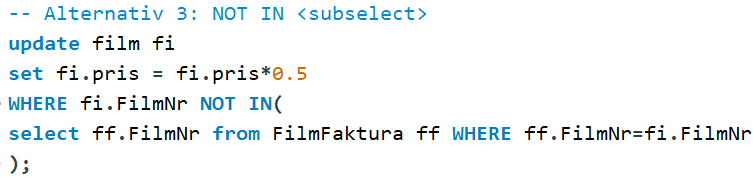
update film fi

set fi.pris = fi.pris\*0.5

WHERE NOT EXISTS(

select \* from FilmFaktura ff WHERE ff.FilmNr=fi.FilmNr

);



update film fi

set fi.pris = fi.pris\*0.5

WHERE fi.FilmNr NOT IN(

select ff.FilmNr from FilmFaktura ff WHERE ff.FilmNr=fi.FilmNr

);

I dette løsningsforslaget er det også et felt Film.Antall. Det kan enten vise antall på lager, eller antall solgt. Dersom det Film.Antall viser antall solgt, kan en oppdatere med en enklere UPDATE. Men da må en huske på å oppdatere dette feltet hver gang noe er solgt. Og en må ta høyde for at av og til tastes det feil film, som må rettes opp i ettertid. Applikasjonen som brukes til salg, eller databasen, bør rette opp Film.Antall automatisk, for det er redundans (dobbelt lagring av informasjon) og kan ellers fort bli feil.  
UPDATE Film SET Pris = Pris \* 0.5 WHERE Antall = 0;

1. Forklar hva ei MAC adresse er, og hvordan denne brukes.

LØSN: MAC står for Media Access Control. Alle nettverkskort o.l. (interface) har ei MAC-adresse, hardkodet fra fabrikken. Det kalles også ei fysisk adresse. Den brukes som adresse internt i et nettverk, av switchen. Lag 2 i OSI-modellen.

1. En PC har IPv4 adressen 132.16.254.1/24. Hvilke av Pc-ene a) til e) er i samme nettverk som denne?

**a)**132.16.254.254 **b)** 132.16.1.1 **c)**132.16.255.254 **d)**132.16.254.128 **e)** 32.16.254.1

LØSN: a) og d) fordi der er de 24 bitene i IPv4 adressen det samme som den oppgitte PCen i nettet.

1. Forkort denne IPv6 adressen 20a1: 00b8: 0bbf: 000e:0000:0000:0000:0d00

LØSN: 20a1:b8:bbf:e::d00

1. Gitt adresse 2001:0DB8:0000:CD30::/60

Hvilke av de to (ingen, en eller begge) skrivemåtene nedenfor er en gyldig skrivemåte for denne adressen. Begrunn svaret:

1. 2001:0DB8::CD30:0:0:0:0/60
2. 2001:0DB8:0:CD3/60

LØSN: a) er riktig. Det er litt merkelig å bruke :: der det er bare en 0’er gruppe, istedenfor til slutt der det er 4 slike, men det er lovlig.

b) er ikke en gyldig IPv6 adresse

1. Adressetype Link-Local unicast er definert slik: FE80::/10. Hvilke av adressene nedenfor er en gyldig Link-Local unicast adresse
   1. FF80::/64 = 1111 1111 1000 0000…. De 10 føsrste bitene stemmer ikke
   2. FE91::7/48 = 1111 1110 1001
   3. FE80:FE80:0000/64 = 1111 1110 1000… Ugyldig IPv6 adresse

LØSN: FE80::/10 = 1111 1110 1000…

b) er riktig. De 10 første bitene må stemme med FF80, men det gjør de ikke på a), mens c) ikke er en gyldig IPv6 adresse

1. Forklar prinsippet med asymmetrisk kryptering (kalles også offentlig nøkkel-kryptering).

LØSN: I asymmetrisk kryptering er det 2 nøkler, en offentlig og en privat. Den offentlige kan brukes til kryptering, men ikke til dekryptering. For å dekryptere meldinga må en ha den private nøkkelen. Asymmetrisk kryptering/dekryptering er mer ressurskrevende enn symmetrisk kryptering, der begge parter har samme nøkkel. Problemet er å få sendt en slik nøkkel (som må være unik for hver kunde) ut til kundene på en sikker måte.

Eksempel: HTTPS = http + TSL. TSL bruker en kombinasjon av de to. Asymmetrisk brukes til å utveksle en nøkkel, som så brukes til synkron kryptering. TSL = Transport Layer Security. Pr. i dag er det TSL 1.2 og TSL 1.3 som brukes. Eldre versjoner av TSL, samt den gamle SSL (Secure Socket Layer) er utdatert.