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

29.10.2021 kl. 15

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

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

oblig3 0.000GB

> use oblig3

switched to db oblig3

* 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.insert([ {filmNr: 2, Tiitel: 'Fort Apechea', År: 1942, Land: 'USA', Sjanger: 'Drama', Alder: 15, Tid: 102, Pris: 149, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.Film.insert([ {filmNr: 3, Tiitel: 'Apocolypse Now', År: 1979, Land: 'USA', Sjanger: 'Action', Alder: 18, Tid: 155, Pris: 155, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.Film.insert([ {filmNr: 4, Tiitel: 'Streets of Fire', År: 1984, Land: 'USA', Sjanger: 'Action', Alder: 18, Tid: 93, Pris: 0, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.Film.insert([ {filmNr: 5, Tiitel: 'High Noon', År: 1952, Land: 'USA', Sjanger: 'Western', Alder: 15, Tid: 85, Pris:123, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.Film.insert([ {filmNr: 6, Tiitel: 'Cinema Parediso', År: 1988, Land: 'Italy', Sjanger: 'Comedy', Alder: 11, Tid:123, Pris: 0, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.Film.insert([ {filmNr: 7, Tiitel: 'Asterix hos Brittene', År: 1988, Land: 'France', Sjanger: 'Animation', Alder: 7, Tid: 78, Pris: 149, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.Film.insert([ {filmNr: 8, Tiitel: 'Veiviseren', År: 1987, Land: 'Norway', Sjanger: 'Action', Alder: 15, Tid: 96, Pris: 87, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.Film.insert([ {filmNr: 9, Tiitel: 'Salmer fra kjøkkenet', År: 2002, Land: 'Norway', Sjanger: 'Comedy', Alder: 7, Tid: 80, Pris: 149, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.Film.insert([ {filmNr: 10, Tiitel: 'Anastasia', År: 1997, Land: 'USA', Sjanger: 'Animation', Alder: 7, Tid: 94, Pris: 123, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.Film.insert([ {filmNr: 11, Tiitel: 'La grande bouffe', År: 1973, Land: 'France', Sjanger: 'Drama', Alder: 15, Tid: 129, Pris: 135, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.Film.insert([ {filmNr: 12, Tiitel: 'The Blues Brothers', År: 1998, Land: 'USA', Sjanger: 'Comedy', Alder: 11, Tid: 124, Pris: 135, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

> db.Film.insert([ {filmNr: 13, Tiitel: 'Beatles: Help', År: 1965, Land: 'Great Britain', Sjanger: 'Music', Alder: 11, Tid: 144, Pris: 0, Antall: 0}]);

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 1,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

"upserted" : [ ]

Grunnen til at jeg ikke tok insert many her var fordi det ble syntax feil som mongodb er dårlig på å finne hvor er hen.

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, Tiitel:1,Sjanger:1}). sort({Pris:-1}).pretty();

* 1. Henter alle kolonner for filmer som ikke er til salgs (tomt på lager), sortert på alder og sjanger.  
     > db.Film.find({Antall:{$eq:0}}, {\_id:0, Alder:1,Sjanger:1}).pretty();
  2. Finner antall filmer som er til salgs i hver sjanger, og summen av prisen

>> db.Film.aggregate( [ {$group: {\_id:"$Sjanger", sum : {$sum : "$Pris"}}}]);

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

db.Film.insert([ {filmNr: 14, Tiitel: 'BingoBanden', År: 2019, Land: 'Norway', Sjanger: 'Comedy', Alder: 11, Tid: 15, Pris: 0, Antall: 0}]);

Korrigerer tittelen på filmen "High Noon". Den heter egentlig "High Moon".  
  
db.Film.update({Tiitel: 'High Noon'}, {$set : {Tittel : 'High Moon'}});

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

db.film.updateMany({ "Sjanger": "Action", "Pris": {$ne: null}}, { $mul: { Pris: 1.1 }});

* 1. Sletter filmen Anastasia.

db.film.remove({Tittel: 'Anastasia'});

1. **Mere MongoDB**

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 : 155, Antall : 1}, {FilmNr:5, Tittel: "High Moon", Pris : 149, Antall : 1}], FakturaNr : 1, Dato : Date()});

Legg til fakturaer for de andre kundene du hadde i Oblig 1.  
  
> db.Faktura.insert({Fornavn : "Arthur", Etternavn : "Morgan", Adresse: "Veigata 13", PostNr : 1153, Poststed : "Hamar", Filmer: [{FilmNr: 3, Tiitel: " Apocolypse Now", Pris : 155, Antall: 1}, {FilmNr: 5, Tiitel: "High Moon", Pris: 149, Antall : 1}], FakturaNr: 1, Dato : Date()});

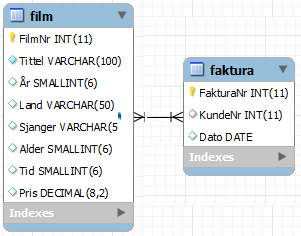
WriteResult({ "nInserted" : 1 })

> db.Faktura.insert({Fornavn : "Fidel", Etternavn : "Castro", Adresse: "CubaParken", PostNr : 1953, Poststed : "Oslo", Filmer: [{FilmNr: 3, Tiitel: " Apocolypse Now", Pris : 155, Antall: 1}, {FilmNr: 5, Tiitel: "High Moon", Pris: 149, Antall : 1}], FakturaNr: 1, Dato : Date()});

WriteResult({ "nInserted" : 1 })

1. **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) |  |

Hva er feil med ER-diagrammet til høyre?  
  
A. Film tabbellen peker med 1 eller flere peker til faktura og fakura peker med 1 eller flere peker mot film(kråkefot med strek).

dette er en ønedvenig kobling der film ikke trenger noen referanser fra fakura. den kan derfor byttes med 0 eller flere(sirkel og kråkefot)

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

1. NF er ikke brutt der alle kollonene ikke har noen nested records og har alle verdiene er enkelt verdier

2. NF er heller ikke brutt der ingen av parameterne påvirker hverandre i forhold til hverandre. ( dette kunne vært annerledes om man f eks hadde hadde hatt valget om om format, digital eller DVD som kunne påvirket pris.)

3 .NF er ikke oppfylt der Alder er avhengig av år, man kan fikse dette ved å enten fjerne år eller separere den til en annen tabell.

1. Kompletter ER-diagrammet til høyre slik at databasen   
    oppfyller 1., 2. og 3. normalform

Skriv for Kari Mo: navn, adresse (med poststed), sammen med de filmer hun har kjøpt.   
Film(FilmNR, Tittel, År, Land, Sjanger, Tid, Pris)

Faktura(FakturaNr, KundeNr, Dato, FilmNR)

Kunde(KundeNR, Fornavn, Etternavn, Postnummer\*)

Poststed(Postnummer, Poststed)

F. create database if not exists Oblig3;

USE Oblig3;

CREATE TABLE IF NOT EXISTS Film(

FNr INT(11),

Title VARCHAR(100) NOT NULL,

År SMALLINT(6),

Land VARCHAR(50) NOT NULL,

Sjanger VARCHAR(50) NOT NULL,

Alder SMALLINT(6),

Tid SMALLINT(6),

Pris DECIMAL(8,2),

CONSTRAINT PKFilm PRIMARY KEY(FNr)

)Engine = InnoDB;

CREATE TABLE IF NOT EXISTS Kunde(

Kundenummer INT(20) PRIMARY KEY,

Fornavn VARCHAR(20),

Etternavn VARCHAR(20),

Adresse VARCHAR(50),

Postnummer INT

)Engine = InnoDB;

CREATE TABLE IF NOT EXISTS Faktura(

FakturaNr INT(11) NOT NULL PRIMARY KEY,

KundeNr INT(20),

Dato DATE,

FilmNr INT(11),

FOREIGN KEY(KundeNr) references Kunde(Kundenummer),

FOREIGN KEY(FilmNr) REFERENCES Film(FNr)

)ENgine = INNODB;

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

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

Skriv ut antall som er solgt av hver sjanger i filmtabellen.  
INSERT INTO FAKTURA VALUES

(1, 3, CURDATE(), 5),

(2, 3, CURDATE(), 11),

(3, 3, CURDATE(), 7);

(1, 3, CURDATE(), 8),

(2, 3, CURDATE(), 9),

(3, 3, CURDATE(), 10);

ALTER TABLE Film

ADD COLUMN Solgt INT(5) AFTER Pris;

DELIMITER $$

CREATE TRIGGER LeggTilSalg

AFTER INSERT

ON FAKTURA FOR EACH ROW

BEGIN

UPDATE Film

set Solgt = Solgt +1

where Fnr = new.filmnr;

END$$

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

UPDATE Film

set Pris = Pris/2

where Solgt = 0;

Forklar hva ei MAC adresse er, og hvordan denne brukes.  
Forklar hva ei MAc Adresse er, Hvordan kan denne brukes:

en mac adresse står for Media Acces Controll, den består av 6 hexadesimaler og blir brukt til å sende data.

eksempel på dette er når man klikker på en link vet serveren at den skal sende akkurat den datapakken til din maskin.

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  
a, c, d.

Forkort denne IPv6 adressen 20a1: 00b8: 0bbf: 000e:0000:0000:0000:0d00  
for å forkorte ipv6 adresser er det bare å elimnere ledende nuller i hexadesimalene .

20a1: 00b8: 0bbf: 000e:0000:0000:0000:0d00

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

2001:0DB8:0:CD3/60

a.2001:0DB8::CD30:0:0:0:0/60

siden nuller kan fjernes er de bare første null som er vist her, den er gykdig

b.2001:0DB8:0:CD3/60

ddet samme gjelder her, er det også fjernet nuller. denne er også gyldig.

1. Adressetype Link-Local unicast er definert slik: FE80::/10. Hvilke av adressene nedenfor er en gyldig Link-Local unicast adresse
   1. FF80::/64
   2. FE91::7/48

FE80:FE80:0000/64   
Hvilke av adressene nedenfor er en gyldig Link-Local unicast adresse

a.FF80::/64 denne

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

LYKKE TIL!