# Eksamen i 2DAN101-1 20H Databaser og nettverk

I dette eksamens-settet finner du først 3 oppgaver med **MongoDB**

1. Forklar strukturen i en **MongoDB**-data med utgangspunkt i data gitt under. Sammenlikne med en relasjonsdatabase.  
    [  
   {**"AnsNr": 1, "Fornavn": "Per", "Etternavn": "Hansen", "Telefon": [{"Tlf1":35950001, "Tlf2": 35951151}], "Gate": "Byvegen 12", "PostNr": 2300, "Poststed": "Hamar"},**{**"AnsNr": 2, "Fornavn": "Lise", "Etternavn": "Jensen", "Telefon": [{"Tlf1":35950002, "Tlf2": 91523344}], "Gate": "Liavegen 125", "PostNr": 2312, "Poststed": "Tangen"},**{**"AnsNr": 3, "Fornavn": "Anders", "Etternavn": "Lie", "Telefon": [{"Tlf1":35950003, "Tlf2": 35952249}], "Gate": "Stangevegen 17", "PostNr": 2310, "Poststed": "Stange"},**{**"AnsNr": 4, "Fornavn": "Johanne", "Etternavn": "Amundsen", "Telefon": [{"Tlf1":35950001, "Tlf2": 35951317, "Tlf3": 44827364}], "Gate": "Fossbakka 38", "Postnr": 2330, "Poststed": "Vallset"},**{**"AnsNr": 5, "Fornavn": "Arne", "Etternavn": "Lie", "Telefon": [{"Tlf1":35950004}],   
   "Gate": "Grønnegata 100", "Postnr": 2300, "Poststed": "Hamar"}**]  
   En **MongoDB**-database består av dokumentsamlinger ("collection", svarer til "tabell"). I eksemplet over er det én dokumentsamling som betår av 5 dokumenter.  
   Et dokument er et **JSON**-objekt: **[{ [{}],...},{...},...]**,og svarer til en rad i en tabell.Et **JSON**-objekt er en array **[...]** av objekter **{...}**. Disse objektene kan igjen inneholde arrayer med elementer eller nye objekter. Nesting av objekter/arrayer kan gjøres vilkårlig dypt.   
   Primærnøkkel genereres automatisk av systemet.
2. Åpne **MongoDB** i kommandovinduet. (Eller du kan bruke **Compass**.) Opprett en database **Eksamen2020** med en dokumentsamling **ansatt.** Legg inn dataene gitt foran.  
   >**use Eksamen2020;**  
   >**db.createCollection("ansatt");**>**db.ansatt.insert([...]);** der **[...]** er hele uttrykket for dokumentsamlingen gitt foran.
3. Skriv ut dataene på en oversiktig måte, og vis utskriftkommando.   
   Legg ved en skjermdump som viser noe av utskriften.  
   >**db.ansatt.find().pretty();**

De neste spørsmålene omhandler relasjonsdatabaser. Du står fritt til å benytte **MySQL Workbench** hvis du synes det er hensiktsmessig.

1. Oversett **MongoDB**-databasen foran til tabeller i en relasjonsdatabase slik at tabellene oppfyller **3NF**.  
   Forklar skrittene du tar underveis.  
   Med bare én tabell **Ansatt** får vi

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **AnsNr** | **Fornavn** | **Etternavn** | **Telefon** | **Gate** | **PostNr** | **Poststed** |
| 1 | Per | Hansen | 35950001 35951151 | Byvegen 12 | 2300 | Hamar |

Brudd med **1NF** for **Telefon**.   
Løsning: lager så mange rader som ansatt har telefonnumre. Nå er ikke **AnsNr** lenger kandidatnøkkel. Legger til surrogatnøkkel i ny kolonne.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PNkl** | **AnsNr** | **Fornavn** | **Etternavn** | **Telefon** | **Gate** | **PostNr** | **Poststed** |
| 1 | 1 | Per | Hansen | 35950001 | Byvegen 12 | 2300 | Hamar |
| 2 | 1 | Per | Hansen | 35951151 | Byvegen 12 | 2300 | Hamar |

Eller vi kan lage en ny tabell **Telefon** med referanse til **AnsNr**.   
Bruker **AnsNr** som primærnøkkel i **Ansatt**-tabell. I **Telefon**-tabell blir **AnsNr+Telefon** primærnøkkel.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AnsNr** | **Fornavn** | **Etternavn** | **Gate** | **PostNr** | **Poststed** |
| 1 | Per | Hansen | Byvegen 12 | 2300 | Hamar |

|  |  |
| --- | --- |
| **AnsNr** | **Telefon** |
| 1 | 35950001 |
| 1 | 35951151 |

|  |  |
| --- | --- |
| **AnsNr** | **Telefon** |
| 1 | 35950001 |
| 1 | 35951151 |

|  |  |
| --- | --- |
| **PostNr** | **Poststed** |
| 2300 | Hamar |

Bruker **AnsNr** som primærnøkkel i **Ansatt**-tabell. Da er **2NF** oppfylt. Men **3NF** er ikke oppfylt idet det er avhengighet mellom **PostNr** og **Poststed**. Løsning er en ny tabell **Poststed** der **PostNr** refererer til **PostNr** i **Ansatt**-tabell.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AnsNr** | **Fornavn** | **Etternavn** | **Gate** | **PostNr** |
| 1 | Per | Hansen | Byvegen 12 | 2300 |

Det er en triviell avhengighet mellom **Fornavn** og **Etternavn**. Gjør ikke noe med det.

1. Skriv **SQL**-kode som oppretter en database (schema) **Eksamen2020.** Gjer til gjeldende database**.**  
   Skriv **SQL**-kode som oppretter tabellene du lagde foran.   
   Gjengi koden i besvarelsen med alle skranker (Constraints) du finner naturlig/nødvendig. Du skal la de ansattes **Fornavn**+**Etternavn** være entydig.  
   **CREATE DATABASE Eksamen2020;  
   USE Eksamen2020;  
   CREATE TABLE Poststed (  
    PostNr INT,  
    Poststed VARCHAR(45) NOT NULL,  
    CONSTRAINT PRIMARY KEY(PostNr)**

**) Engine=InnoDB;**

**CREATE TABLE Ansatt (  
 AnsNr INT,  
 Fornavn VARCHAR(45) NOT NULL,  
 Etternavn VARCHAR(45) NOT NULL,**

**Gate VARCHAR(45) NOT NULL,  
 PostNr INT NOT NULL,  
 UNIQUE (Fornavn, Etternavn),   
 CONSTRAINT PRIMARY KEY(AnsNr),  
 CONSTRAINT FOREIGN KEY(PostNr) REFERENCES Poststed(PostNr)**

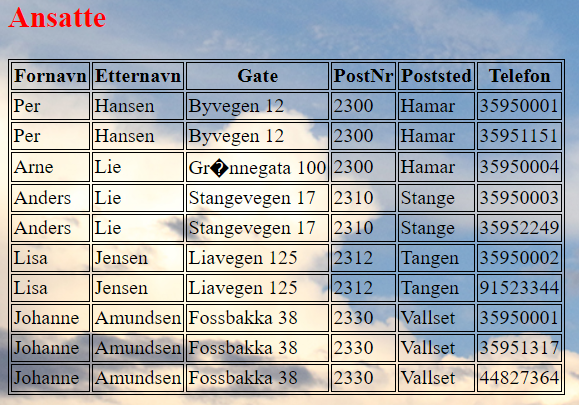
**) Engine=InnoDB;  
CREATE TABLE Telefon (  
 AnsNr INT NOT NULL,   
 TelefonNr INT,  
 CONSTRAINT PRIMARY KEY(AnsNr, TelefonNr),  
 CONSTRAINT FOREIGN KEY(AnsNr) REFERENCES Ansatt(AnsNr)**

**) Engine=InnoDB;**

Under finner du data tatt fra **MongoDB**-databasen:   
**Per, Hansen, 35950001, 35951151, Byvegen 12, 2300, Hamar,  
Lise, Jensen, 35950002, 91523344, Liavegen 125, 2312, Tangen  
Anders, Lie, 35950003, 35952249, Stangevegen 17, 2310, Stange  
Johanne, Amundsen, 35950001, 35951317, 44827364, Fossbakka 38, 2330, Vallset  
Arne, Lie, 35950004, Grønnegata 100, 2300, Hamar**

1. Skriv **SQL**-setninger som legger disse dataene inn i tabellene du har laget.  
   **INSERT INTO Poststed VALUES   
    (2300,'Hamar'), (2312,'Tangen'), (2310,'Stange'), (2330,'Vallset');  
   INSERT INTO Ansatt (AnsNr,Fornavn,Etternavn,Gate,PostNr)   
    VALUES (1, 'Per', 'Hansen', 'Byvegen 12', 2300),   
    (2, 'Lise', 'Jensen', 'Liavegen 125', 2312),   
    (3, 'Anders', 'Lie', 'Stangevegen 17', 2310),   
    (4, 'Johanne', 'Amundsen', 'Fossbakka 38', 2330),   
    (5, 'Arne', 'Lie', 'Grønnegata 100', 2300);  
   INSERT INTO Telefon VALUES (1, 35950001),(1,35951151),  
    (2, 35950002),(2, 91523344), (3, 35950003),(3,35952249),  
    (4, 35950001),(4, 35951317),(4, 44827364), (5, 35950004);**

I det følgende vil du bli bedt om å skrive noen **SQL**-spørringer fra **Eksamen2020**-databasen du har laget. Legg ved en skjermdump som viser resultatet fra spørringen.  
En **MongoDB**-spørring er føyet til noen av oppgavene, og er ment som en hjelp ved at de kan eksekveres mot **MongoDB**-databasen. Du får se en utskrift som kan gi et hint om utskriften du skal lage.

1. Skriv alle Fornavn+Etternavn sortert avtakende etter etternavn**.**  
   **>db.ansatt.find({},{\_id:0, Fornavn:1,Etternavn:1}).sort({Etternavn:-1});**  
   NB: får ikke alltid skilt nøkler fra verdier i MongoDB.   
   **SELECT Fornavn, Etternavn FROM ansatt ORDER BY Etternavn DESC;**
2. Skriv alle navn, adresser og alle telefonnumre.  
   **>db.ansatt.aggregate({$project:{\_id:0,"Fornavn":1,"Etternavn":1,"Gate":1,"PostNr":1,  
    "Poststed": 1,"Telefon ":1}}).pretty();  
   SELECT A.Fornavn, A.Etternavn, A.Gate, A.PostNr, P.Poststed,T.TelefonNr   
   FROM (Ansatt AS A JOIN Poststed AS P ON A.PostNr=P.PostNr)   
   JOIN Telefon AS T ON A.AnsNr=T.AnsNr;**
3. Skriv hvor mange som bor i hvert poststed. **db.ansatt.aggregate({$group:{"\_id":"$Poststed", antall:{$sum:1}}});  
   SELECT P.Poststed, COUNT(A.Postnr) AS antall FROM Poststed AS P JOIN Ansatt AS A ON P.PostNr=A.PostNr GROUP BY A.PostNr;**
4. Hva er det som karakteriserer en primærnøkkel i en tabell?  
   Primærnøkkelen er karakterisert ved **NOT NULL, UNIQUE** og **INDEX**.  
   Det betyr at alle felter må gis verdi, alle verdier må være forskjellige. Og kolonnen(e) holdes hele tiden sortert.
5. Lag en enkel web-side som presenterer data fra databasen gitt i begynnelsen av oppgaven.  
    For eksempel som i figuren under.  
   
6. Endre tabell med ansatte slik den får en ny kolonne med den ansattes kjønn.  
   Alle ansatte skal ha angitt kjønn i den nye tabellen, og det skal bare være mulig å gi verdiene **M** og **K**.  
   **ALTER TABLE Ansatt ADD COLUMN Kjonn CHAR(1) NOT NULL;  
   ALTER TABLE Ansatt ADD CONSTRAINT CHECK (Kjonn IN ('M', 'K'));**Oppdater tabellen slik at kjønn blir registrert for Anders Lie og Johanne Amundsen.  
   **UPDATE Ansatt SET Kjonn='M' WHERE Fornavn='Anders' AND Etternavn='Lie';**  
   **UPDATE Ansatt SET Kjonn='K' WHERE Fornavn='Johanne' AND Etternavn='Amundsen';**
7. Du kan bruke betinget **INSERT** ved (for eksempel) **SQL**-spørringen  
   I**NSERT INTO Poststed (PostNr,Poststed)**

**SELECT 2330, 'Vallset' FROM DUAL WHERE**

**NOT EXISTS (SELECT postNr FROM Poststed WHERE postNr=2330);  
DUAL**-tabellen er en systemtabell du kan hente vilkårlige data fra.  
Forklar hva spørringen foran (farget grønn) gjør.  
**Hvis poststed Vallset med postnummer 2330 ikke finnes i tabellen Poststed skal dette poststedet settes inn.**

1. Legg inn data gitt ved  
   **Ole, Olsen, 35950010, 95959595, 474747474, Fossvegen 112, 2330, Vallset**Dataene skal legges inn i databasen med en eller flere innsettinger som alle må utføres.  
   Du kan få bruk for betingete innsettinger slik det er angitt i forrige punkt.

Forklar hvordan du vil sikre at databasen ikke blir inkonsistent

**START TRANSACTION;**

**INSERT INTO Poststed (PostNr,Poststed)**

**SELECT 2330, 'Vallset' FROM DUAL WHERE NOT EXISTS**

**(SELECT postNr FROM Poststed WHERE postNr=2330);**

**INSERT INTO Ansatt (AnsNr,Fornavn,Etternavn,Gate,PostNr,Kjonn)**

**SELECT 7,'Ole','Olsen','Fossvegen 112',2330,'M' FROM DUAL WHERE NOT EXISTS**

**(SELECT Fornavn, Etternavn FROM Ansatt WHERE Fornavn='Ole' AND  
 Etternavn='Olsen');**

**INSERT INTO Telefon (AnsNr,TelefonNr)**

**SELECT 7,35950010 FROM DUAL WHERE NOT EXISTS**

**(SELECT AnsNr, TelefonNr FROM Telefon WHERE AnsNr=7 AND   
 TelefonNr=35950010);**

**INSERT INTO Telefon (AnsNr,TelefonNr)**

**SELECT 7,95959595 FROM DUAL WHERE NOT EXISTS**

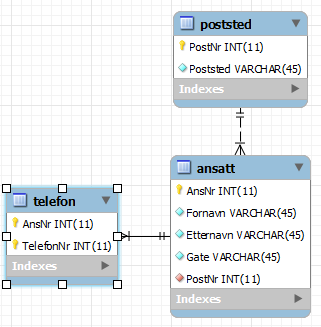
**(SELECT AnsNr, TelefonNr FROM Telefon WHERE AnsNr=7 AND   
 TelefonNr=95959595);**

**INSERT INTO Telefon (AnsNr,TelefonNr)**

**SELECT 7,474747474 FROM DUAL WHERE NOT EXISTS**

**(SELECT AnsNr, TelefonNr FROM Telefon WHERE AnsNr=7 AND   
 TelefonNr=474747474);**

**COMMIT;**

1. Lag et ER-diagram for databasen din, og lim inn et bilde av ER-diagrammet i besvarelsen  
   
2. Det er satt av forskjellige adresseområder i IPv6, for forskjellige typer adresser. En av dem er «Global Unicast», som har fått adresseområde definert ved 2000::/3
3. Hva betyr «Global Unicast» adresse (hva brukes den til)?

LØSN:

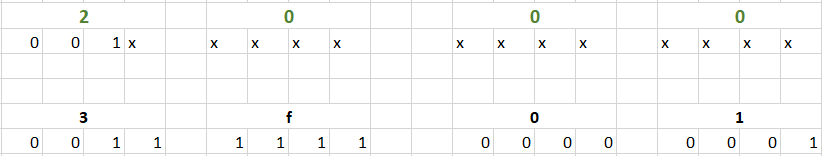
Det er en global entydig IPv6 adresse, som brukes for en til en kommunikasjon. Den er beskrevet i RFC 3587.

1. Er denne IPv6 adressa av type «Global Unicast»

3f01: 0db8: b000: 0001:0000:0000:0000:0200

LØSN: Ja

2000::/3 betyr at hvis de 3 første bitene må stemme overens med de 3 første bitene i 2000, så er det en «Global Unicast» adresse.



1. Skriv denne IPv6 adressen så kort som mulig

LØSN:

3f01: db8: b000:1::200

1. I dette nettverket er det 8 byte som gir adressen til nettverket. Hvordan vil du angi det for adressen ovenfor?

LØSN:

Med slash 64 etter ipv6 adressen: 3f01: db8: b000:1::200/64

1. NAT
2. Hvordan virker NATing

LØSN: NAT = Network Adress Transfer.

NATing gjør at selv om du er koblet til et privat nettverk – slik at ip-adressen din ikke routes på internett – kan du allikevel «snakke» med datamaskiner via internett. Når du prøver å nå ei ip adresse utenfor det private nettverket du er i, vil routeren din endre avsenderadresse på ip-pakka før den sendes videre. Den endres til routerens ip-adresse (som er public, og kan routes på internett). I tillegg legger den på et unikt portnr, for å skille mellom de forskjellige PCer og programmer som svaret skal routes tibake til. Routeren holder rede på hvor svaret skal sendes tilbake, for konvertering av ip-adresse og portnr når svaret kommer.

1. Hvorfor er det mindre behov for NATing i IPv6 nettverk

LØSN: Hovedårsaken til bruk av private IPv4 nettverk, er at det er mangel på adresser. Fordi det ikke er mangel på IPv6 adresser, blir det dermed mindre bruk av private nettverk, og mindre behov for NATing.