Mark Tzvetoslavov Dochev - marktz

IN2090 – Obligatorisk Oppgave 3
Normalformer

Oppgave 1 – Lage databaser

Tog

Nå som vi har fått i oppgave å lage vårt eget database, vil det være lurt å starte den med følgende SQL uttrykk, som vil sørge for selve opprettelsen til denne:

```
DROP SCHEMA IF EXISTS TogDatabase CASCADE;
BEGIN; -- Begin transaction
CREATE SCHEMA TogDatabase;
```

Nå som vi har opprettet skjemaet for vår database, kan vi begynne å opprette første tabell, som i dette tilfelle, må være opprettet for at resten av tabellene skal kunne referere til de ulike togene:

Tog (togNr, startStasjon, endeStasjon, ankomstTid)

Tog-relasjonen inneholder et tog nummer, som er satt til å være primærnøkkelen i tabellen. Vi har i tillegg en togrute, som består av en start stasjon og en ende stasjon. Sist, har vi ankomsttiden som er et klokkeslett som toget ankommer endestasjonen. Siden det er spesifisert i oppgaven at en togrute (startStasjon – endeStasjon) går maks en gang i døgnet, har jeg bestemt at det lureste i dette tilfelle er å sette en UNIQUE skranke som vil da sørge for at en togrute ikke gjentar seg.

```
CREATE TABLE Tog(
  togNr INT PRIMARY KEY, -- auto_increment eller serial
  startStasjon TEXT NOT NULL, -- varchar(50)
  endeStasjon TEXT NOT NULL, -- varchar(50)
  ankomstTid TIME NOT NULL,
  CONSTRAINT togRute UNIQUE (startStasjon, endeStasjon)
);
```

For eksempelets skyld, har jeg valgt å lage sette inn litt informasjon inn i tabellen for å få litt innsikt på hva vi driver med. Dette eksemplet vil da inneholde unike togruter som ikke gjentar seg, selv om for eksempel ankomst tiden er likt for noen togruter. Det er verdt å nevne at dette ikke vil påvirke databasen, siden to ulike togruter kan ha samme ankomsttid.

```
-- Følgende utrykk vil tillates å sette inn i vårt tabell, siden hver togrute
-- som settes inn er unikt.
INSERT INTO Tog
VALUES (1, 'Oslo Lufthavn', 'Basel SBB', '14:00:00'),
       (2, 'Alicante Railway Station', 'Viana do Castelo', '09:00:00'),
       (3, 'Düsseldorf Hauptbahnhof', 'Kraków Central', '16:00:00'),
       (4, 'Leningradskiy Railway Station', 'Palaeofarsalos', '16:00:00');
```

Derimot, hvis vi prøver å legge et tog (med en unik primærnøkkel) men som inneholder en allerede eksisterende togrute, vil vi få en feilmelding. Grunnen til det er fordi vi har satt en UNIQUE-skranke som sjekker slike tilfeller. Dette oppfyller da kravet om at en togrute kan gå maks en gang i døgnet.

```
33  -- Følgende uttrykk vil gi feil, siden vi har definert at start stasjonen og
34  -- endestasjonen (altså et bestemt togrute) skal være unikt.
35  INSERT INTO Tog
36  VALUES (5, 'Oslo Lufthavn', 'Basel SBB', '18:00:00'),
37
```

TogTabell

Neste tabell i vår database er **TogTabell**. Denne inneholder informasjon om avgangstiden på et tog nummer, samt hvilken stasjon den kjører fra (startstasjon):

TogTabell (togNr, avgangsTid, stasjon)

I begynnelsen av oppgaven er det spesifisert at togNr i TogTabell og Plass er fremmednøkkel til Tog. Derfor, har jeg satt togNr i TogTabell til å referere til togNr i Tog. Dette kan også settes som en egen constraint for å sette den som en FOREIGN KEY. Det er verdt å nevne at tabellen har to understreket attributter, nemlig togNr og avgangsTid. Dette indikerer at kombinasjonen av de to utgjør en unik verdi. Vi kan tenke oss at for eksempel ved ulike, vi legger inn to ulike avgangstider for samme tog. Derfor, i tillegg til å ha legget til togNr og avgangsTid som primærnøkkel, har jeg også laget en UNIQUE constraint, som sørger for at et togNr ikke kan settes to ganger. Dette vil da oppfylle alle krav som er satt.

NB: Jeg fikk vite at databasen ikke trenger å være så avansert, men hadde vært fint (du som retter ☺) om du kunne vise meg hvordan vi kan sjekke at avgangstiden er mindre enn ankomst tiden til det toget eller om startstasjonen stemmer med det vi satt i Tog-tabellen. Jeg har prøvd å hente informasjon fra Tog-tabellen, men synes det var litt vanskelig.

```
CREATE TABLE TogTabell(
  togNr INT REFERENCES Tog(togNr),
  avgangsTid TIME NOT NULL,
  stasjon TEXT NOT NULL, -- varchar(50)
  CONSTRAINT togTid PRIMARY KEY (togNr, avgangsTid),
  CONSTRAINT unikTogNr UNIQUE(togNr)
);
```

Igjen, har jeg valgt å sette inn litt data for å sjekke om alle skranker fungerer som de skal.

```
INSERT INTO TogTabell
VALUES (1, '09:00:00','Oslo Lufthavn'),
       (2, '06:00:00', 'Alicante Railway Station'),
       (3, '13:00:00', 'Düsseldorf Hauptbahnhof'),
       (4, '05:00:00', 'Leningradskiy Railway Station');

-- Følgende utrykk vil gi oss en feilmelding siden togNr og avgangsTid skal
-- være ment til å være unikt.

INSERT INTO TogTabell
VALUES (1, '08:00:00', 'Oslo Lufthavn');
```

Plass

Sist, men ikke minst, har vi Plass-tabellen. Plass består av ett sette (gitt ved vognNr og plassNr) på et bestemt tog og på en bestemt dato, samt hvorvidt plassen er et vindus sette og om det er ledig.

En ting jeg vil gjerne påpeke er at for å gjøre databasen min mer originalt, har jeg istedenfor ha en boolean datatype både i «vindu» og «ledig», har jeg opprettet dem til å være en TEXT med en CHECK skranke. Begge alternativer vil da være riktig, siden PostgreSQL tillater boolean verdier.

Et annet ting er at jeg lagde en CHECK -constraint som sørger for at togplasset ikke gjelder for datoer som er allerede gått. Dette vil gjelde nåværende datoer eller fremtidige datoer. Til slutt har jeg opprettet en primær nøkkel som vil bestå av dato, togNr, vognNr og plassNr.

```
CREATE TABLE Plass(
  dato DATE NOT NULL
  CHECK (dato >= CURRENT_DATE),
  togNr INT NOT NULL REFERENCES Tog(togNr),
  vognNr int NOT NULL,
  plassNr int NOT NULL,
  vindu TEXT NOT NULL
  CHECK (vindu = 'vindussette' OR vindu = 'ikke vindussette'),
  ledig TEXT NOT NULL
  CHECK (ledig = 'ledig' OR ledig = 'ikke ledig'),
  PRIMARY KEY(dato, togNr, vognNr, plassNr)
);
```

Til slutt, har jeg valgt igjen å sette litt informasjon for å sjekke at alle skranker vi har opprettet funker som de skal og at de oppfyller alle krav som er gitt av oppgaven:

```
90  INSERT INTO Plass
91  VALUES ('2020-11-19', 1, 2, 3, 'vindussette', 'ledig'),
92         ('2020-11-19', 1, 2, 4, 'vindussette', 'ledig'),
93         ('2020-11-19', 1, 3, 1, 'ikke vindussette', 'ikke ledig'),
94         ('2020-11-19', 1, 3, 2, 'ikke vindussette', 'ikke ledig'),
95         ('2020-11-19', 2, 1, 3, 'vindussette', 'ikke ledig');
96
97  -- Følgende utrykk vil gi oss en feilmelding siden dato, togNr, vognNr og plassNr
98  -- skal være ment til å være unikt.
99
100 INSERT INTO Plass
101 VALUES ('2020-11-19', 1, 2, 3, 'ikke vindussette', 'ikke ledig');
102
103 COMMIT;
```

I neste side finnes det hele SQL-scriptet for denne databasen!

Oppgave 1 – Lage databaser

```sql
DROP SCHEMA IF EXISTS TogDatabase CASCADE;

BEGIN; -- Begin transaction
CREATE SCHEMA TogDatabase;

CREATE TABLE Tog(
  togNr INT PRIMARY KEY, -- auto_increment eller serial
  startStasjon TEXT NOT NULL, -- varchar(50)
  endeStasjon TEXT NOT NULL, -- varchar(50)
  ankomstTid TIME NOT NULL,
  CONSTRAINT togRute UNIQUE (startStasjon, endeStasjon)
);

INSERT INTO Tog
VALUES (1, 'Oslo Lufthavn', 'Basel SBB', '14:00:00'),
       (2, 'Alicante Railway Station', 'Viana do Castelo', '09:00:00'),
       (3, 'Düsseldorf Hauptbahnhof', 'Kraków Central', '16:00:00'),
       (4, 'Leningradskiy Railway Station', 'Palaeofarsalos', '16:00:00');

CREATE TABLE TogTabell(
  togNr INT REFERENCES Tog(togNr),
  avgangsTid TIME NOT NULL,
  stasjon TEXT NOT NULL, -- varchar(50)
  CONSTRAINT togTid PRIMARY KEY (togNr, avgangsTid),
  CONSTRAINT unikTogNr UNIQUE(togNr)
);

INSERT INTO TogTabell
VALUES (1, '09:00:00','Oslo Lufthavn'),
       (2, '06:00:00', 'Alicante Railway Station'),
       (3, '13:00:00', 'Düsseldorf Hauptbahnhof'),
       (4, '05:00:00', 'Leningradskiy Railway Station');
```

```sql
CREATE TABLE Plass(
  dato DATE NOT NULL
  CHECK (dato >= CURRENT_DATE),
  togNr INT NOT NULL REFERENCES Tog(togNr),
  vognNr int NOT NULL,
  plassNr int NOT NULL,
  vindu TEXT NOT NULL
  CHECK (vindu = 'vindussette' OR vindu = 'ikke vindussette'),
  ledig TEXT NOT NULL
  CHECK (ledig = 'ledig' OR ledig = 'ikke ledig'),
  PRIMARY KEY(dato, togNr, vognNr, plassNr)
);

INSERT INTO Plass
VALUES ('2020-11-19', 1, 2, 3, 'vindussette', 'ledig'),
       ('2020-11-19', 1, 2, 4, 'vindussette', 'ledig'),
       ('2020-11-19', 1, 3, 1, 'ikke vindussette', 'ikke ledig'),
       ('2020-11-19', 1, 3, 2, 'ikke vindussette', 'ikke ledig'),
       ('2020-11-19', 2, 1, 3, 'vindussette', 'ikke ledig');

COMMIT;
```

Oppgave 2 – FDer og Normalformer

a) Hvilke kandidatnøkler har R?

For å finne hvilke kandidatnøkler R har, kan vi se tilbake på relasjonen og hvilke FDer R har:

- **R** (A, B, C, D, E, F, G)

Funksjonelle avhengigheter:

- C D E --- > B
- A F ------ > B
- B ---------> A
- B C F ----> D E
- D -------- > G

Konklusjoner så langt:

- Ikke på høyresider: C, F
- Kun på høyresider: G
- Forsøk å utvide med: A, B, D, E

Nå som vi har følgende informasjon, kan vi konkludere at kandidatnøkkelen vår vil inneholde **C** og **F**. Vår oppgave nå er å forsøke å utvide med alle andre bokstaver som ligger både på høyreside og venstre-side. Kandidatnøkkelen vår vil ikke inneholde bokstaven **G**.

1. X = C F A
   {C, F, A}$^{+}$ = C, F, A, B, D, E, G
   Alle elementer er med, noe som vil si at {C, F, A} er en kandidatnøkkel.

2. X = C F B
   {C, F, B}$^{+}$ = C, F, B, D, E, G, A
   Alle elementer er med, noe som vil si at {C, F, B} er en kandidatnøkkel.

3. X = C F D
   {C, F, D}$^{+}$ = C, F, D, G
   Vi mangler noen elementer, noe som vil si at {C, F, D} ikke er en kandidatnøkkel.

4. X = C F E
   {C, F, E}$^{+}$ = C, F, E
   Vi mangler noen elementer, noe som vil si at {C, F, E} ikke er en kandidatnøkkel.

Med tanke på at vi har 2 kandidatnøkler, har vi da funnet svaret vi trenger. Men, det er et siste steg vi må gjennom for å se om vi har en kandidatnøkkel til. Det vi kan gjøre nå er å sla sammen **D** og **E** for å se om vi har en kandidatnøkkel til.

5. X = C F D E
   {C, F, D, E}$^{+}$ = C, F, D, E, B, A, G
   Alle elementer er med, noe som vil si at {C, F, D, E} er en kandidatnøkkel.

Kandidatnøklene til R er {A, C, F}, {B, C, F} og {C, D, E, F}

b) Finn den høyeste normalformen som R tilfredsstiller

Kandidatnøklene til R er {A, C, F}, {B, C, F} og {C, D, E, F}

FD 1: C D E --- > B 3NF

1. Er X en supernøkkel? Nei, X er ikke en supernøkkel. Gå til spørsmål 2.
2. Er A et nøkkelattributt? Ja, A er et nøkkelattributt. Gå til neste FD.

FD 2: A F ------ > B 3NF

1. Er X en supernøkkel? Nei, X er ikke en supernøkkel. Gå til spørsmål 2.
2. Er A et nøkkelattributt? Ja, A er et nøkkelattributt. Gå til neste FD.

FD 3: B ---------> A 3NF

1. Er X en supernøkkel? Nei, X er ikke en supernøkkel. Gå til spørsmål 2.
2. Er A et nøkkelattributt? Ja, A er et nøkkelattributt. Gå til neste FD.

FD 4: B C F ----> D E BCNF

1. Er X en supernøkkel? Ja, X er en supernøkkel

FD 5: D --------- > G 1NF

1. Er X en supernøkkel? Nei, X er ikke en supernøkkel. Gå til spørsmål 2.
2. Er A et nøkkelattributt? Ja, A er et nøkkelattributt. Gå til neste FD.
3. Er X en del av supernøkkel? Ja, X er en del av kandidatnøkkelet. Brudd på 2NF.

Den høyeste normalformen som R tilfredsstiller er **1NF**

c) Dekomponer R tapsfritt til BCNF.

Start dekomposisjonen ved å ta utgangspunkt i FDen CDE --- > B:

Vi kan tenke oss at kandidatnøklene våre for denne oppgaven er som følgende:
{A, C, F}, {B, C, F} og {C, D, E, F}

Siden CDE bryter med BCNF, kan vi nå utvide denne.
{C, D, E}$^{+}$ = C, D, E, B, A, G

Vi får følgende relasjon:

- S1 (C, D, E, B, A, G)
- S2 (C, D, E, F)

  Vi kan tenke oss at S1 har følgende FDer: FD1, FD2 og FD5. En ting som vi fort legger merke til er at S2 bryter ikke med BCNF, så denne kan vi legge til side i våre resultater. S1 bryter med BCNF og derfor må vi fortsette å utvide den.

Vi kan nå fortsette å utvide S1. Hvis vi da bruker FD2 denne gangen, vil dette endte opp som følgende.
B --- > A bryter med BCNF, derfor må vi utledde.
$\{B\}^+$ = B A
Vi får følgende relasjon:

- S11(A, B)
- S12 (B, C, D, E, G)

I dette tilfelle kan vi se at S11 bryter ikke lenger med BCNF (sjekker den opp mot FD2). Da kan vi legge det til siden, og fortsette å utvide S12, siden det bryter med BCNF.

Neste FD vi må sjekke opp er FD5, nemlig D -- > G. Vi kan se at den også bryter med BCNF, så da kan vi ta utgangspunkt i forrige relasjon og utvide den.
D --- > G bryter med BCNF, derfor må vi utledde.
$\{D\}^+$ = D, G
Vi får følgende relasjon:

- S121 (D, G)
- S122 (B, C, D, E)

Til slutt, kan vi merke at S122 ikke kan utleddes videre, så da kan vi konkludere at denne også er med dekomponeringen. Vi kan konkludere at følgende relasjoner er med en tapsfri dekomponering av R:

- S11(A, B), S121 (D, G), S122 (B, C, D, E) og S2 (C, D, E, F)