dDB Excercise Week 3

Functional Dependancies

Ved at bestemme functional dependancy (FD) i en relation, kan denne optimeres og redundans kan minimeres. FD er når en eller flere attributter funktionelt definerer en eller flere andre attributter. Det kan udtrykkes som: $\{A_1, A_2, \ldots A_n\} \rightarrow \{B_1, B_2, \ldots B_n\}$

Ikke-trivielle FD's er alle entiteters keys og deres dertilhørende attributter. Som f.eks.:

Stations: { Name } → { Location }
Da ingen stationer har det samme navn, bestemmes Name til vores key.

Seats: { Row, Letter, WagonID } → { SeatClass }
Row og Letter definere ikke alene hvilken type sæde der er tale om. Men så snart at WagonID kommer med, så bliver SeatClass entydigt bestemt derudfra. Dette er da definitionen på en komplet-ikke-triviel FD.

Connections: { Departure, Arrival, StartPoint, EndPoint } → { ChildRebate, OldRebate, Std.Rebate}
For hver strækning og tidspunkt vil der være en unik pris for alle sæderne, så der vil ikke kunne findes flere entries i databasen med de tidspunkter og start/slut stationer som vil give forskellige priser. Derfor er det en FD.

Passengers: { ID } → { Name, Age, OrderNo }
ID vil som key funktionelt definere de øvrige attibutter i Passengers-entiteten.

Boyce-Codd Normal Form

Ved en gennemgang af mine skemaer mener jeg ikke at kunne finde nogen som bryder med BCNF formen. De FD'er som er listet oven over har alle en ikke-triviel venstreside som er en superkey. Det vil sige, at venstresiden funktionelt bestemmer højresiden.

Gruppearbejde i relationel algebra

Jeg kunne ikke få det LaTeX til at virke ordentligt. Men jeg fik at vide at pseudo SQL også ville blive godkendt. Så Anders Riggelsen og mig fik flg. opstillet:

1) *How many seats are still available (i.e. not booked) on today's 10am direct connection between Vejle and Fredericia?*

```
A(WagonID) :=
SELECT wagon_id
FROM Connection
WHERE
      depart LIKE '%10am' AND
      start = "Vejle" AND
      end = "Fredececia"

      B(row, letter, id) := natural join (Seats, A)

      C(row, letter, id) := natural join (Reservations, A)

Answer(trains available) := count(B - C)
```

   De ledige sæder er derfor fundet som B – C

2) *"Find all ways together with their price to reach Aalborg from Frederikshavn today with at most one stop in between."*

```
SELECT  (A.price + B.price), A.start, B.start, B.end
FROM Connections AS A, Connections AS B
WHERE
      A.start = "Frederikshavn" AND
      A.end = B.start AND
      B.end = "Aalborg" AND
      A.depart = $today AND
      B.arrive = $today
```

3) *When should I leave Aarhus the latest if I want to be in Randers before 2PM today with at most one stop?*

```
SELECT max(A.depart)
FROM Connection AS A, Connection AS B
WHERE
      A.start = ″Aarhus″ AND
      A.end = B.start AND
      B.end = ″Randers″ AND
      B.arrive < 2pm today
```

4) *”Find 2 neighbouring seats (if possible) all the way along the following route: Aarhus-Vejle-Kolding anytime today provided the waiting time in Vejle is less than 1 hour.”*

```
W(id) :=
    SELECT WagonID
    FROM Runs-On as A,Runs-On AS B
    WHERE
    A.start = ″Århus″ AND
    A.slut = B.start = ″Vejle″ AND
    B.slut = ″Kolding″ AND
    (B.depart - A.arrive) < 1 hour

      Seats(row1, letter1, id1, row2, letter2, id2)
          := natural join (W, Close-To)

      Reserved(letter, row, id) :=
            SELECT (letter, row, id) FROM Reservations
            WHERE
            `start` IN {″Århus″, ″Vejle″} AND
            `end` IN {″Vejle″, ″Kolding″}

      Answer(row1, letter1, id1, row2, letter2, id2) :=
            SELECT * FROM Pairs, Reserved
            WHERE
            Reserved <> (Pairs.row1, Pairs.letter1, Pairs.id1) AND
            Reserved <> (Pairs.row2, Pairs.letter2, Pairs.id2)
```

DRBs E/R Diagram