Databaser Aflevering 2

E/R diagram lavet i samarbejde med Anders Halager & Troels Hansen.

De Danske Rustne Baner (DRB) vil have lavet en database som skal køre deres E-ticket system.
Af de forskellige ting som eventuelt kan komme med inkluderer:

Trains, wagons, seats, connections, passengers, E-tickets, seat reservations & railway stations.

Trains & wagons:

Efter lidt drøftelse fandt vi hurtigt ud af at der ingen behov var for en ”Trains” entity. Når folk køber billetter køber de kun rettigheden til at rejse på givne strækninger og ikke til bestemte tog. Vognene hænger heller ikke altid sammen da de kan deles og samles på vilkårlige stationer så ordet ”tog” er ikke helt så brugbart. Når de rejsende ikke har pladsbillet er stort set hverken tog, vogn eller sæde relevant for dem ud over at de måske vil vide hvilke vogne der kører på hvilke strækninger.

Runs-On relationen beskriver hvilke vogne der kan køre på hvilke strækninger. Mange vogne kan køre på én connection og én connection kan godt have mange vogne kørende på sig.

- **Wagons(ID)**
    - **Single-value constraint: Ikke to vogne må have samme ID.**

- **Runs-On(WagonID, Depart, Arrive, StartName, EndName)**

Seats:

Vi skelner ikke mellem sæder i togene så både 1st class, economic og senge i sovevognene da de jo alle har næsten samme egenskaber. Hvis vi f.eks. havde skelnet mellem pladserne fra sove-vognene og pladserne fra de normale vogne (1st class og ecomonic) som helt forskellige entities ville vi få en del redundancy. Det eneste som sove-vognene har til forskel er en ”level” attribut som fortæller om sengen er øverst eller nederst. Vi valgte at give ”Seat” en ’type’ attribut som kan repræsentere alle typer sæder.
Værdierne for ’type’ kan være: **{’Economic’, ’1stClass’, ’BedLvl1’, ’BedLvl2’}**
Hvor vidt om sæderne skulle have været i et ”is-a” hierarki i stedet er svært at sige, da ”Economy” og ”1stClass” attributmæssigt ikke er forskellige. Så skulle de to samles i ”Seat” attributten og skelnes imellem med en ”type” attribut og så vil det kun være ”Bed” der er en specialudgave af ”Seat”. Men i sådan et tilfælde vil ”Bed” også have en ”type” attribut som ikke er meningen at den skal have da der f.eks. ikke findes 1st class Beds. Vi kunne sagtens have lavet en constraint der forbød ”Seat” i at have en type attribut (type skal være null) hvis den er en bed, men der er derved et unødigt tomt felt og laver lidt redundancy. Vi følte at en ren ”type” metode var den bedste løsning.
Seat er en weak-entity set da et sæde med et bogstav og række nr. ikke er unik. Der kan findes mange vogne som indeholder et sæde som har samme bogstav og række nr., så et sæde kan kun gøres unik når den henter en key fra den vogn den befinder sig i.

For at modellere ”neighbourhood” sammenhængen mellem sæderne har de et ”relationship” til hinanden som vi kalder ”Close-to”. Hvert sæde kan have mange naboer så vi har lavet et many-many relationship mellem dem. Da Seat er ”weak” skal ”WagonID” også med for hvert sæde for at kunne identificere det rigtige sæde.

- **Seats(Row, Letter, WagonID, type)**
  - **Domain constraint: Type skal være enten ‘Economic’, ‘1stClass’, ’BedLvl1’ eller ‘BedLvl2’.**

- **Close-To(Row1, Letter1, WagonID1, Row2, Letter2, WagonID2)**
  - **General constraint: Et sæde må ikke være “nabo” til sig selv.**

Connections & E-tickets:

Mellem hver station er der en forbindelse som en billet kan give tilladelse til at køre på. Hver enkelt billet kan give tilladelse til at køre på en mængde af ”connections” så vi bruger en many-one relation som vi kalder ”Spans” da en billet ”strækker over” en mængde forbindelser. Price 1,2 og 3 beskriver prisen på 1st class, economy class og sovepladser.

”Connection” er weak da en enkelt forbindelse der kun har ’depart’ og ’arrive’ tidspunkter ikke kan ses som en unik afgang da der godt kan findes andre forbindelser der har samme depart og arrive tidspunkter. Derfor har ”Connections” også to stationer som derved gør afgangen unik.

- **Connections(Depart, Arrive, StartName, EndName, EcoPrice, 1stPrice, BedPrice)**
  - **General constraint: StartName og EndName må ikke være ens.**

- **E-Tickets(ID)**
  - **Single-value constraint: Ikke to billetter må have same ID.**

- **Spans(Depart, Arrive, StartName, EndName, E-TicketID)**

Passengers:

Vi diskuterede en del om dette emne da passagerers navne ikke kan gemmes ved f.eks. køb af billetter ved billetautomater. Det eneste der kunne gemmes var om passageren valgte en børne-, voksen- eller senior-billet. Grunden til at vi i vores model ikke har tilladt børn og pensionister at have specielle priser er at vi enten skulle begrænse en rabat-ordning til kun at kunne bruge procenter. Ellers skulle hver forbindelse have 9 forskellige attributter til priser:
"pris1barn", "pris1voksen", "pris1senior", "pris2barn… osv. osv. hvilket ikke er særligt smart da hvis der over hele systemet skulle ændres rabat-priser så skulle samtlige poster i databasen opdateres og endda måske til den samme værdi hvilket giver en hel masse redundancy hvis det var tilfældet. Derfor har vi i første omgang ikke kunne få ”Passengers” med i vores design.

Reservations:

Vi har modelleret reservationer som en relation der er mellem E-tickets, connections og seats. Man kan f.eks. med en billet og en forbindelse finde det matchende sæde og andre lignende kombinationer. En reservation er altid bundet til én billet, én forbindelse og et sæde. Dvs. at der kan tilhøre mange reservationer til samme billet da en reservation kun tilhører én strækning. Vi er dog ikke begrænset til at der kun kan tilhøre én reservation pr. forbindelse pr. billet. Det åbner også op for at man kan bestille en ekstra pladsbillet til f.eks. sin taske som det er muligt i virkeligheden.

- **Reservations(E-TicketID, Depart, Arrive, StartName, EndName, Row, Letter, SeatID)**

Railway stations:

Hver forbindelse indeholder både en start og en slut tog-station (pilene viser at hver forbindelse har to stationer). Pilenes retning læses som ”flere forbindelser kan have én togstation” eller ”én togstation kan være del af flere forbindelser”.
Vi har lavet den constraint at stationerne ikke må have samme navne af den grund at vi så kun har brug for én key. Dette formindsker endda også antal data der skal gemmes i ”Connections”.

- **Stations(Name, Location)**
  - **Single-value constraint: Ikke to stationer må have samme navn.**

Opsummering af Schemas og Constraints:

- **Wagons(ID)**
  - **Single-value constraint: Ikke to vogne må have samme ID.**

- **Runs-On(WagonID, Depart, Arrive, StartName, EndName)**

- **Seats(Row, Letter, WagonID, type)**
  - **Domain constraint: Type skal være enten ‘Economic’, ‘1stClass’, ’BedLvl1’ eller ‘BedLvl2’.**

- **Close-To(Row1, Letter1, WagonID1, Row2, Letter2, WagonID2)**
  - **General constraint: Et sæde må ikke være “nabo” til sig selv.**

- **Connections(Depart, Arrive, StartName, EndName, EcoPrice, 1stPrice, BedPrice)**
  - **General constraint: StartName og EndName må ikke være ens.**

- **E-Tickets(ID)**
  - **Single-value constraint: Ikke to billetter må have same ID.**

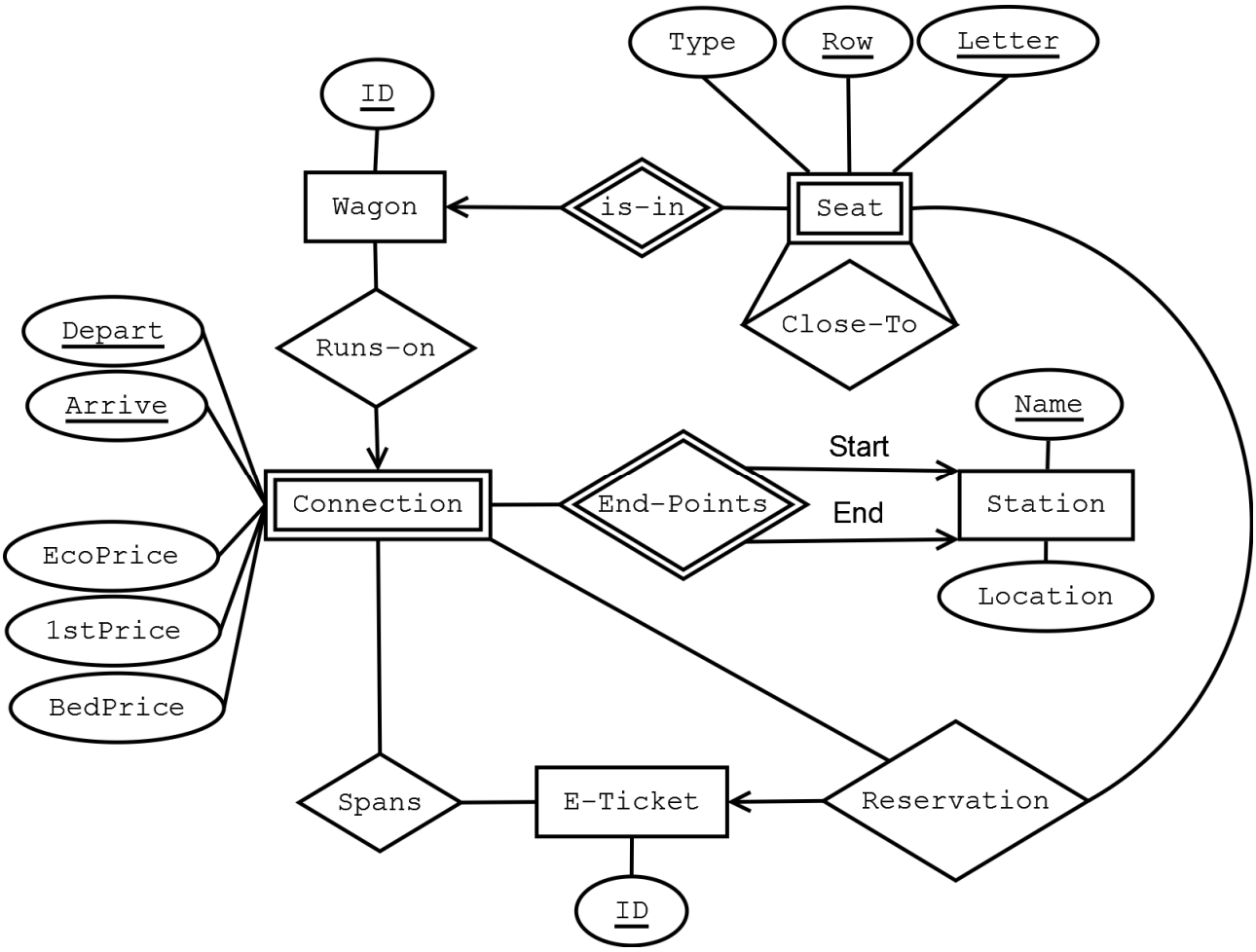
- **Spans(Depart, Arrive, StartName, EndName, E-TicketID)**

- **Reservations(E-TicketID, Depart, Arrive, StartName, EndName, Row, Letter, SeatID)**

- **Stations(Name, Location)**
  - **Single-value constraint: Ikke to stationer må have samme navn.**

Referential Integrity constraints kan ses på E/R diagrammet.

E/R Diagram



Note: Vi bruger Louis' notation for Referential Integrity frem for bogens.

→ = 1

→ = 0 eller 1 - (bruges ikke i vores design)