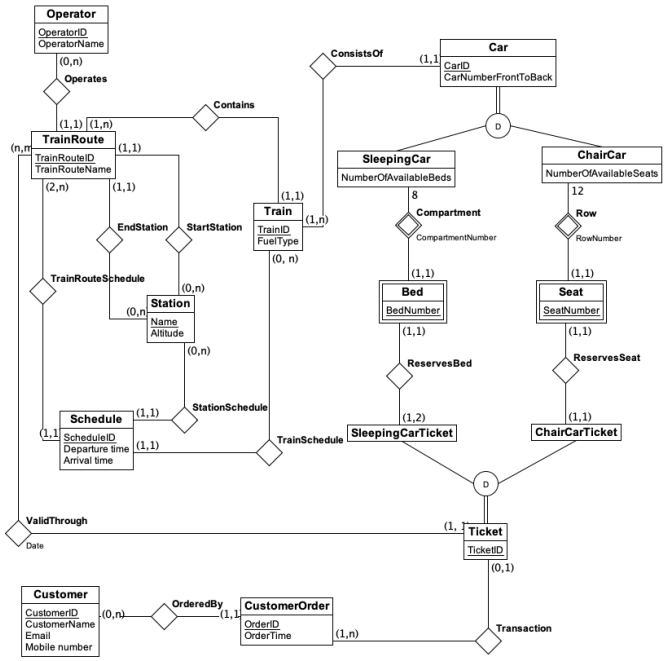
Miniprosjekt 2023

Oppgave A) ER-modell



Figur 1: ER-modell som beskriver en vilkårlig togbane i Norge.

Antagelser ER-modell:

En operatør trenger ikke nødvendigvis eie en togrute for å være en operatør (0, n), og en togrute hører til kun én operatør. En togrute må bestå av kun én start- og én slutt stasjon. En stasjon kan tilhøre flere togruter, men trenger ikke nødvendigvis å gjøre det. Dette forutsetter at det kan eksistere null planlagte togruter i databasen. I tillegg kreves det at en togrute inneholder minst ett tog. Dette mener vi først og fremst gir mest logisk flyt, men forhindrer også kjøp av billetter på togruter der det ikke kjører tog – noe kunden åpenbart ikke ønsker.

Vi har i denne modelleringen valgt å implementere en Schedule-entitetsklasse, med relasjoner til stasjon, tog og togrute. Det følger at én instans av Schedule inneholder flere tog, stasjoner og togruter. Et tog må inneholde minst én vogn, og kan inneholde flere. En vogn må tilhøre ett tog - vi tar for eksempel ikke hensyn til vogner som er på verksted.

I modelleringsprosessen ble det også konkludert med å benytte total disjunkt spesialisering for å skille mellom vogntyper. Dette kan i prinsippet forenkles ved å modulere CarType som et attributt, men vi begrunner med at spesialiseringen gir bedre oversikt over egenskapene til de ulike vogntypene (f.eks. hvorvidt det er nattog eller ikke).

I oppgavebeskrivelsen virker det som man kan ta høyde for 8 senger og 12 seter, selv om det ikke konkretiseres. For enkelhetens skyld, så antar vi at alle vogner av hver type har like mange plasser. I tillegg antar at vi at man kan ha null eller flere kunder.

Total, disjunkt spesialisering av Billett-entiteten skiller mellom hvilke vogntyper den gjelder. Merk at «SleepingCarTicket» kan tilhøre én eller (maks) to senger. Dette er nyttig for kundene som ikke ønsker å dele kupé med fremmede. I kontrast, kan man maks reservere ett «ChairCar»-sete per billett. En «CustomerOrder» eksisterer ikke før en kunde har opprettet den, så den må tilhøre minst én kunde. En «CustomerOrder» må tilhøre minst én billett, men en billett må ikke nødvendigvis tilhøre en «CustomerOrder» (ergo, billetten er ikke kjøpt enda, men den eksisterer likevel). Merk at en billett reserverer et sete/seng på en togruteinstans. Vi antar at en togrute bestandig har mange tilhørende billetter.

Kommentar

Vi ønsker å presisere at denne modellering først tok høyde for prisforskjeller mellom vogntypene (noe som ifølge oppgavebeskrivelsen kan ignoreres). Vår implementasjon gjør det lett å modellere pris, som for eksempel attributter på SleepingCarTicket og ChairCarTicket.

En ganske omfattende svakhet i modellen er mangelen på konsistente togruteinstanser over flere dager. Selv om Ticket-klassen har en relasjon med dato-attributt, som gir har vi ingen måte å holde oversikt på hvilken dag en togrute tilhører. Vi skal vurdere muligheter for å implementere dette senere i realiseringen av databasen i Python – eog ventuelt småjustere modellen.

Oppgave B) Skjema

Operator(OperatorID, OperatorName)
- OperatorID er primærnøkkel.

TrainRoute(TrainRouteID, TrainRouteName, OperatorID)
- TrainRouteID er primærnøkkel.
- OperatorID er fremmednøkkel for Operator.

Train(TrainID, FuelType)
- TrainID er primærnøkkel.

Schedule(ScheduleID, DepartureTime, ArrivalTime)
- ScheduleID er primærnøkkel.

TrainSchedule(ScheduleID, TrainID)
- ScheduleID er fremmednøkkel for Schedule
- TrainID er fremmednøkkel for Train

TrainRouteSchedule(ScheduleID, TrainRouteID)
- ScheduleID er fremmednøkkel for Schedule
- TrainRouteID er fremmednøkkel for Train

StationSchedule(ScheduleID, Name)
- ScheduleID er fremmednøkkel for Schedule
- Name er fremmednøkkel for Train

Station(Name, Altitude)
- Name er primærnøkkel.
- TrainRouteID er fremmednøkkel.

Car(CarID, CarNumberFrontToBack, TrainID)
- CarID er primærnøkkel.
- TrainID er fremmednøkkel for Train.

SleepingCar(NumberOfAvailableBeds, CarID)
- CarID er fremmednøkkel for Car.

ChairCar(NumberOfAvailableSeats, CarID)
- CarID er fremmednøkkel for Car.

Bed(BedNumber, CarID)
- BedNumber er svak primærnøkkel, en unik identifikator i en Car.
- CarID er fremmednøkkel for Car.

Seat(SeatNumber, CarID)
- SeatNumber er svak primærnøkkel, en unik identifikator i en Car.
- CarID er fremmednøkkel for Car.

ChairCarTicket(TicketID, SeatNumber, CarID)
- TicketID er fremmednøkkel.
- SeatNumber er en svak fremmednøkkel.

SleepingCarTicket(TicketID, BedNumber, CarID)
- TicketID er fremmednøkkel.
- BedNumber er en svak fremmednøkkel.

Ticket(TicketID, Date, TrainRouteID, OrderID)
- TicketID er primærnøkkel
- TrainRouteID er fremmednøkkel for TrainRoute.
- OrderID er fremmednøkkel for CustomerOrder.

CustomerOrder(OrderID, OrderTime, CustomerID)
- OrderID er primærnøkkel.
- CustomerID er fremmednøkkel for Customer.

Customer(CustomerID, CustomerName, Email, MobileNumber)
- CustomerID er primærnøkkel.

Alle tabeller er på 4NF fordi det ikke finnes FA-er eller MVD-er i tabellene som bryter 4NF og gir grunnlag for uønsket redundans.

Vi valgte å splitte opp Schedule tabellen for å unngå å ha TrainID, TrainRouteID og Name i samme tabell, da de sammen er en nøkkel. Vi ville da hatt flerverdiavhengigheter i tillegg til å unngå unødvendig redundans. Fra problembeskrivelsen nummereres vognene forfra og bakover i toget, vi har i tillegg til dette innført CarID for å kunne flytte vogner mellom tog.

Vi valgte også å legge Date til Ticket-tabellen fordi en billett må ha en relasjon til en togrute. Det er unødvendig å ha denne informasjonen i en egen tabell fordi det ikke ut over normalformen til Ticket-tabellen, og man eliminere en tabell fra skjemaet. Dette gjør det hele mer oversiktlig.

Oppgave C) SQL

Se DB1.sql for SQL-script

Noen restriksjoner som ikke uttrykkes i relasjonsdatabaseskjemaet og derfor må håndteres i applikasjonsprogrammene inkluderer:

Det er ikke spesifisert noen restriksjoner for antall billetter som kan kjøpes per ordre eller per kunde. Dette må håndteres i applikasjonsprogrammene for å sikre at kunder ikke kjøper for mange billetter på en gang.

Det er heller ikke spesifisert noen restriksjoner på billettpriser eller betalingsmåter. Dette må også håndteres i applikasjonsprogrammene for å sikre at betalingsinformasjon er riktig lagret og behandlet, og for å sikre at billettpriser er konsistente og riktige.

```sql
CREATE TABLE Operator (
    OperatorID INT PRIMARY KEY,
    OperatorName VARCHAR(255)
);

CREATE TABLE TrainRoute (
    TrainRouteID INT PRIMARY KEY,
    TrainRouteName VARCHAR(255),
    OperatorID INT,
    FOREIGN KEY (OperatorID) REFERENCES Operator(OperatorID)
);

CREATE TABLE Train (
    TrainID INT PRIMARY KEY,
    FuelType VARCHAR(255)
);

CREATE TABLE Schedule (
    ScheduleID INT PRIMARY KEY,
    DepartureTime DATETIME,
    ArrivalTime DATETIME
);

CREATE TABLE TrainSchedule (
    ScheduleID INT,
    TrainID INT,
    PRIMARY KEY (ScheduleID, TrainID),
    FOREIGN KEY (ScheduleID) REFERENCES Schedule(ScheduleID),
    FOREIGN KEY (TrainID) REFERENCES Train(TrainID)
);

CREATE TABLE TrainRouteSchedule (
    ScheduleID INT,
    TrainRouteID INT,
    PRIMARY KEY (ScheduleID, TrainRouteID),
```

```
    FOREIGN KEY (ScheduleID) REFERENCES Schedule(ScheduleID),
    FOREIGN KEY (TrainRouteID) REFERENCES TrainRoute(TrainRouteID)
);

CREATE TABLE StationSchedule (
    ScheduleID INT,
    Name VARCHAR(255),
    PRIMARY KEY (ScheduleID, Name),
    FOREIGN KEY (ScheduleID) REFERENCES Schedule(ScheduleID),
    FOREIGN KEY (Name) REFERENCES Station(Name)
);

CREATE TABLE Station (
    Name VARCHAR(255) PRIMARY KEY,
    Altitude INT,
    TrainRouteID INT,
    FOREIGN KEY (TrainRouteID) REFERENCES TrainRoute(TrainRouteID)
);

CREATE TABLE Car (
    CarID INT PRIMARY KEY,
    CarNumberFrontToBack INT,
    TrainID INT,
    FOREIGN KEY (TrainID) REFERENCES Train(TrainID)
);

CREATE TABLE SleepingCar (
    NumberOfAvailableBeds INT,
    CarID INT,
    FOREIGN KEY (CarID) REFERENCES Car(CarID)
);

CREATE TABLE ChairCar (
    NumberOfAvailableSeats INT,
    CarID INT,
    FOREIGN KEY (CarID) REFERENCES Car(CarID)
);

CREATE TABLE Bed (
    BedNumber INT,
    CarID INT,
    PRIMARY KEY (BedNumber, CarID),
    FOREIGN KEY (CarID) REFERENCES Car(CarID)
);

CREATE TABLE Seat (
    SeatNumber INT,
    CarID INT,
    PRIMARY KEY (SeatNumber, CarID),
    FOREIGN KEY (CarID) REFERENCES Car(CarID)
);
```

```
CREATE TABLE ChairCarTicket (
    TicketID INT,
    SeatNumber INT,
    CarID INT,
    PRIMARY KEY (TicketID),
    FOREIGN KEY (SeatNumber, CarID) REFERENCES Seat(SeatNumber, CarID)
);

CREATE TABLE SleepingCarTicket (
    TicketID INT,
    BedNumber INT,
    CarID INT,
    PRIMARY KEY (TicketID),
    FOREIGN KEY (BedNumber, CarID) REFERENCES Bed(BedNumber, CarID)
);

CREATE TABLE CustomerOrder (
    OrderID INT PRIMARY KEY,
    OrderTime DATETIME,
    CustomerID INT,
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);

CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(255),
    Email VARCHAR(255),
    MobileNumber VARCHAR(255)
);

CREATE TABLE Ticket (
    TicketID INT PRIMARY KEY,
    Date DATETIME,
    TrainRouteID INT,
    OrderID INT,
    FOREIGN KEY (TrainRouteID) REFERENCES TrainRoute(TrainRouteID),
    FOREIGN KEY (OrderID) REFERENCES CustomerOrder(OrderID)
);
```