FlyDB

Jasper Steinberg

March 20, 2025

Del 2: Realisert databasesystem

ZIP-filen jeg har levert har tre filer. Den første er setupFlyDB.sql. Den inneholder all SQL kode som oppretter tabeller og setter inn data. Denne filen dekker altså brukstilfelle 1, 2, 3, 4 og 7. For at sensor skal slippe å grave i denne fila har jeg lagt ved den relevante SQL koden til hver oppgave i dette dokumentet. Den andre fila heter FlyDB.sqlite, denne er tom ved nedlastning. Den siste fila heter main.py, og inneholder kode for å lese inn innholdet av setupFlyDB.sql inn i FlyDB.sqlite. Den innholder også pytonprogammet i brukstilfelle 6 og en funksjon som reproduserer spørringen i brukstilfelle 5. Det som kjøres når man kjører main.py er:

```
if __name__ == "__main__":

    database_setup()

    print("\nResulatet av spørringen i brukstilfelle 5: \n")
    test_brukstilfelle5()


    brukstilfelle6_userinput()
```

database_setup() oppretter databasen og leser inn tabeller og data. test_brukstilfelle5() reproduserer spørringen i brukstilfelle 5, dette er også beskrevet nærmere i dette dokumentet. brukstilfelle6_userinput() lar brukeren skrive inn input til funskjonen i brukstilfelle 7, dette er også beskrevet nærere i dokumentet. Jeg anbefaler og lese dokumentet, for så å kjøre main.py. Koden kan kjøres så mange ganger man ønsker. Første avsnitt i dette dokumentet er endringer fra del 1 av prosjektet. Etter det er beskrivelsen av alle brukstilfellene og hvordan de er løst. Til slutt har jeg del 1 av prosjektet slik at det er lett å slå opp informasjon om ER-modellen.

Endringer

Noen endringer er gjort i del 1 for å tilplasses del 2. Nasjonaliteten til en flyprodusent er en flerverdiattributt og har nå fått sin egen tabell. Vi har nå også gjort det slik at vi tillater at en delreise entitet representerer en full flyrute, og markerer dette ved å gi SekvensNummer 999. Dette gjør endel av brukstilfellene lettere, og vi burde strengt tatt bare ha kalt denne entiteten "Reise". Det er også nødvendig for brukstillfelle 4 at en gitt flyvning kan ha NULL-verdi for FlyID, altså at flyvningen er satt opp uten å ha fått et fly enda.

Brukstilfelle 1

For å legge til flyplassene kjører vi INSERT INTO setninger for de fem gitte flyplassene i vedlegget. Et eksempel på dette er gitt under.

```
INSERT INTO Flyplass(Flyplasskode, Flyplassnavn)
VALUES ("BGO", "Bergen lufthavn, Flesland");
```

Brukstilfelle 2

De tre gitte flyselskapene legges til databsen ved INSERT INTO setninger gitt ved et eksempel under.

```
INSERT INTO Flyselskap(Flyselskapskode, Navn)
VALUES ("DY", "Norwegian");
```

Før vi legger til flytyper, oppretter vi de gitte flyprodusentene. Vi søker opp og finner at The Boeing company ble stiftet i 1916. Airbus group ble stiftet i 1970, og De Havilland Canada i 1928. Deres respective nasjonaliteter blir så lagt til i Nasjonalitet tabellen, siden dette er en flerverdi attributt. Airbus group blir for eksempel lagt til slik:

```
INSERT INTO Flyprodusent(Navn, Stiftelsesår)
VALUES ("Airbus Group", 1970);

INSERT INTO Nasjonalitet (FlyprodusentNavn, Land)
VALUES ("Airbus Group", "Frankrike");

INSERT INTO Nasjonalitet (FlyprodusentNavn, Land)
VALUES ("Airbus Group", "Tyskland");

INSERT INTO Nasjonalitet (FlyprodusentNavn, Land)
VALUES ("Airbus Group", "Spania");

INSERT INTO Nasjonalitet (FlyprodusentNavn, Land)
VALUES ("Airbus Group", "Storbritannia");
```

Nå kan vi legge til flytypene, dette gjør vi slik:

```
INSERT INTO Flytype(Navn, Produsent, Produksjonsstart, Produksjonsslutt, RadAntall,
    SeteAntall)
VALUES ("Boeing 737 800", "The Boeing Company", 1997, 2020, 31, 31*6);

INSERT INTO Flytype(Navn, Produsent, Produksjonsstart, Produksjonsslutt, RadAntall,
    SeteAntall)
VALUES ("Airbus a320neo", "Airbus Group", 2016, NULL, 30, 30*6);

INSERT INTO Flytype(Navn, Produsent, Produksjonsstart, Produksjonsslutt, RadAntall,
    SeteAntall)
VALUES ("Dash-8 100", "De Havilland Canada", 1984, 2005, 10, 2+9*4);
```

Videre kan vi legge til de 11 flyene i vår miniverden. Et par eksempler på dette er:

```
INSERT INTO Fly (Registreringsnummer, Serienummer, Navn, FørsteDriftsår, Flytype)
VALUES ("LN-ENU", "42069", NULL, 2015, "Boeing 737 800");

INSERT INTO Fly (Registreringsnummer, Serienummer, Navn, FørsteDriftsår, Flytype)
VALUES ("SE-RUB", "9518", "Birger Viking", 2020, "Airbus a320neo");

INSERT INTO Fly (Registreringsnummer, Serienummer, Navn, FørsteDriftsår, Flytype)
VALUES ("LN-WIH", "383", "Oslo", 1994, "Dash-8 100");
```

Brukstilfelle 3

Først oppretter vi flyrutene, så kan vi legge til alle delflyvningene som inngår i enhver flyrute. Det er ikke lagt til noen informasjon om når noen av flyrutene startet eller avsluttes, så vi setter NULL-verdi for Sluttdato og den fiktive datoen 2000-01-01 som Oppstartsdato. Den første flyvningen blir dermed lagret ved:

```
INSERT INTO Flyrute(Flyrutenummer, Ukedagskode, Oppstartdato, Sluttdato, Budsjett,
     konomi  , Premium)
VALUES ("WF1311", "12345", "2000-01-01", NULL, 599, 899, 2018);

INSERT INTO Delreise(Flyrutenummer, SekvensNummer, Avgangstid, Ankomsttid, Budsjett,
     konomi  , Premium, Startflyplass, Endeflyplass)
VALUES ("WF1311", 1, "15:15:00", "16:20:00", 599, 899, 2018, "TRD", "BOO");

INSERT INTO BetjenesAv(FlyruteID, FlytypeID, FlyselskapID)
VALUES ("WF1311", "Dash-8 100", "WF");
```

De resterende flyrutene blir lagt til på samme måte, noen med flere delreiser.

Brukstilfelle 4

Vi skal nå legge inn noen flyvninger. Vi antar at vi kan generere et løpenummer selv siden dette ikke er oppgitt, og gir flyvningene løpenummer i stigende rekkefølge, altså "1", "2" o.s.v. Vi setter

NULL-verdi for FlyID, og avangangs/ankomsttid siden disse ikke er kjent enda. Insert setningene blir dermed:

```
INSERT INTO Flyvning (Løpenummer, Dato, Status, FaktiskAvgangstid, FaktiskAnkomstid,
    FlyID, FlyruteID)
VALUES (1, "2025-04-01", "planned", NULL, NULL, NULL, "WF1302");

INSERT INTO Flyvning (Løpenummer, Dato, Status, FaktiskAvgangstid, FaktiskAnkomstid,
    FlyID, FlyruteID)
VALUES (2, "2025-04-01", "planned", NULL, NULL, NULL, "DY753");

INSERT INTO Flyvning (Løpenummer, Dato, Status, FaktiskAvgangstid, FaktiskAnkomstid,
    FlyID, FlyruteID)
VALUES (3, "2025-04-01", "planned", NULL, NULL, NULL, "SK888");
```

Brukstilfelle 5

For å gjøre spørringen så enkelt som mulig er det lurt å fylle inn Flåte tabellen først, et eksempel på en av insert setningene er gitt under

```
INSERT INTO Flåte (FlyID, FlytypeID, FlyselskapID)
VALUES ("LN-ENU", "Boeing 737 800", "DY");
```

Da blir spørringen vi søker

```
SELECT Flyselskap.Navn as Flyselskap, Flåte.FlytypeID as Flytype, COUNT(*) AS
    AntallFly
FROM Flåte
INNER JOIN Flyselskap ON (Flåte.FlyselskapID = Flyselskap.Flyselskapskode)
GROUP BY Flyselskap.Navn, Flåte.FlytypeID;
```

Vi joiner med flyselskap slik at vi kan bruket navnet på flyselskapet og ikke flyselskapskoden i sluttabellen. Vi grupperer først på flyselskapene, og så på flytypene innad i et flyselskap. Hvert flyselskap har bare en flytype i vår realisering, men det kan fort endre seg. Til slutt teller vi antall rader, altså antall fly hvert flyselskap har av hver flytype. Spørringen kjørt i DB Browser gir følgende output:

	Flyselskap	Flytype	AntallFly
1	Norwegian	Boeing 737 800	4
2	SAS	Airbus a320neo	4
3	Widerøe	Dash-8 100	3

Figure 1: Output av spørringen

For lett reprodukjon av resulatet har jeg laget en funksjon i pyton-fila kalt test_brukstilfelle5().

Brukstilfelle 6

Løsningen er gitt i pyton-fila. Funksjonen heter brukstilfelle6_userinput(), og kjører når man kjører main. Brukeren skriver til terminal. Et eksempel på kjøring vil være

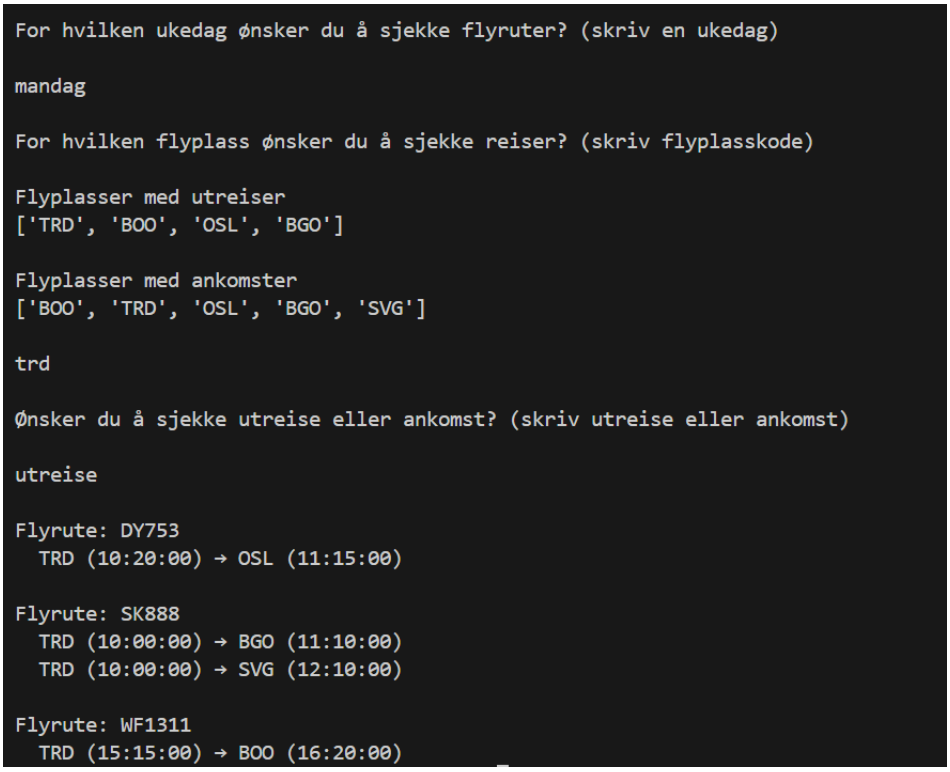


Figure 2: Eksempel på kjøring av brukstilfelle 6

Brukstilfelle 7

Først må vi opprett en fiktiv kunde, vi gir ham kundenummer 1;

```
INSERT INTO Kunde (KundeNr, Navn, Telefon, Epost, Nasjonalitet, Fordelsprogram)
VALUES (1, "Jasper Steinberg", 96922262, "jaspers@stud.ntnu.no", "Norsk", NULL);
```

Vi tenker oss Jasper reserverer første raden med 2 seter, som vi tenker oss er premium klasse. Han reserverer også all 4 setene på andre rad som vi tenker oss har klassen økonomi. Til slutt reserverer ham 4 seter på tredje rad som vi tenker oss er i budsjett klassen. Den avledete prisen til BillettKjøpet blir da $2 \times 2018 + 4 \times 899 + 4 \times 599 = 10028$. Vi oppretter et billettkjøp, med fremmednøkkel mot kunden Jasper. Vi lar Billettkjøpet har referansenummer "1":

```
INSERT INTO BillettKjøp (Referansenummer, Totalpris, Kjøper)
VALUES (1, 2*2018 + 4*899 + 4*599, 1);
```

Vi legger nå til billettene, med fremmednøkkel mot billettkjøpet med referansenummer 1. Her må vi huske på hvordan flyet ser ut. For eksempel har første rad bare de to setene 1C og 1D.

```
INSERT INTO Billett(BillettID, RadNummer, RadBokstav, Klasse, Pris, Innsjekk,
    BillettKjøpID)
VALUES (1, 1, "C", "premium", 2018, NULL, 1);

INSERT INTO Billett(BillettID, RadNummer, RadBokstav, Klasse, Pris, Innsjekk,
    BillettKjøpID)
VALUES (2, 1, "D", "premium", 2018, NULL, 1);

INSERT INTO Billett(BillettID, RadNummer, RadBokstav, Klasse, Pris, Innsjekk,
    BillettKjøpID)
VALUES (3, 2, "A", "economy", 899, NULL, 1);

INSERT INTO Billett(BillettID, RadNummer, RadBokstav, Klasse, Pris, Innsjekk,
    BillettKjøpID)
VALUES (4, 2, "B", "economy", 899, NULL, 1);

INSERT INTO Billett(BillettID, RadNummer, RadBokstav, Klasse, Pris, Innsjekk,
    BillettKjøpID)
```

```
VALUES (5, 2, "C", "economy", 899, NULL, 1);

INSERT INTO Billett(BillettID, RadNummer, RadBokstav, Klasse, Pris, Innsjekk,
    BillettKjøpID)
VALUES (6, 2, "D", "economy", 899, NULL, 1);

INSERT INTO Billett(BillettID, RadNummer, RadBokstav, Klasse, Pris, Innsjekk,
    BillettKjøpID)
VALUES (7, 3, "A", "budget", 599, NULL, 1);

INSERT INTO Billett(BillettID, RadNummer, RadBokstav, Klasse, Pris, Innsjekk,
    BillettKjøpID)
VALUES (8, 3, "B", "budget", 599, NULL, 1);

INSERT INTO Billett(BillettID, RadNummer, RadBokstav, Klasse, Pris, Innsjekk,
    BillettKjøpID)
VALUES (9, 3, "C", "budget", 599, NULL, 1);

INSERT INTO Billett(BillettID, RadNummer, RadBokstav, Klasse, Pris, Innsjekk,
    BillettKjøpID)
VALUES (10, 3, "D", "budget", 599, NULL, 1);
```

Del 1: ER modellen

Entiteter, deres antagelser og restriksjoner

Flyselskap

Flyselskapskode er nøkkel, ellers lagrer vi bare navn foreløpig.

Flytype

Navn er nøkkel. Vi lagrer produksjonsstart og produksjonsslutt. Flyprodusenten er koblet via ProdusertaV Relasjonen. Hvis flyet produseres fortsatt setter vi NULL-verdi for produksjonsslutt. Til slutt lagrer vi informasjon om setekonfiguarsjonen til en flytype. Her tar vi med antall rader (RadAntall) og antall seter (SeteAntall), siden dette ikke er unikt bestemt av seter per rad og antall rader. Velger å ikke ta med antall seter per rad siden dette ikke er unik innenfor hvert fly. Vi oppretter en en egen sete entitet som lagrerer den resterende informasjonen som er mer relevant for billetkjøp senere.

Flyprodusent

Nøkkel er navnet. Ellers lagrer vi nasjonalitet og stiftelsesår. Nasjonalitet er et flerverdiattributt for å kunne håndtere tilfellet med internasjonale flyprodusenter.

Fly

Tolker det slik at et registreringsnummeret er unikt på tvers av alle fly, og blir dermed nøkkel. Videre lagrer vi serienummer og første driftsår (FørsteDriftsår). Vi lagrer også navn, som blir NULL vis flyet ikke har navn.

Sete

Vi modellerer et sete som en svak entitetsklasse som har en unik kode innenfor hver fly. Denne koden er definert ved et tall som representerer raden og en bokstav som sier hvor i raden setet er. Vi lagrer også en boolsk variable kalt nødutgang.

Flyplass

Flyplasskode er nøkkel, med flyplass navn som alternativ nøkkel.

Delreise

Dette er eventuelle delreiser som kan inngå i en flyrute. En direkteflyvning vil bare bestå av en delreise, mens man ved flere delreiser har flere. Vi modellerer dette som en svak entitetsklasse hvor flyrute blir den definerende entiteten. Vi lager en svak nøkkel kalt SekvensNummer, som angir rekkefølgen for delreisene i en flyrute. For eksempel 1 for første delreise og 2 for andre. Vi tillater også å lagre hele reiser som vi gir SekvensNummer 999, disse er da strengt tatt ikke ekte delreiser. Vi lagrer også (planlagt) avgangstid og ankomstid for denne entiteten. Startsflyplass og endeflyplass blir lagret som relasjoner til flyplass. Vi må også lagre prisene til delreisen, som vi legger i budsjett, økonomi og premium attributtene. Prisene på en flyrute med mange delreiser må utledes.

Flyrute

Dette er en samling av en eller flere faste delreiser. Flyrutenummer er nøkkel. Start og slutt for den totale flyruten samt tider utledes fra delreisene (ønsker å unngå redundans). Vi lagrer ukedagskode, denne er oppgitt i et format som gjør at vi ikke trenger å lagre flere verdier selv om en flyrute flys flere ganger samme uke. Dette formater er en delmengde av strengen "1234567", så en rute som går mandag og lørdag er for eksempel beskrevet ved "16". Selskapet som betjener ruten og dets flytype er modellert via BetjenesAv relasjonen. Til slutt lagrer vi oppstartdato og eventuelt sluttdato for flyruten, vis det ikke er planlagt noen sluttdato setter vi NULL-verdi.

Flyvning

Nøkkel er løpenummer. Vi registrer hvilket fly som skal brukes i flys av relasjonen. Vi lagrer statusen til flyvningen (planned, active, completed, cancelled). Vi lager vi en relasjon PlanlagtSom som beskriver hvilken flyrute flyvningen representerer. Den faktiske avgangs og ankomstiden utledes av de faste flytidene i flyruten først, før de faktiske tidene blir satt nærmere avgang.

Kunde

Nøkkel er KundeNr. Vi lagrer navn, telefon, epost og nasjonalitet. Vi lagrer også om kunden et med i et fordelspogram, og setter NULL hvis ikke.

Billett

Vi definerer først en enkelt billett for en delreise. Senere kan vi sette sammen disse billettene for å lage alle mulige slags reiser. En billett vil ha et radnummer og radbokstav som referer til det aktuelle setet, eventuelt NULL-verdi om det ikke er bestemt enda. Vi lagrer også klasse og pris på billetten. Det er lettest å generere en egen nøkkel for Billett, vi kaller denne BillettID. For å enkelt kunne sikre gyldige kjøp av billett er billett koblet til en flyvning, og en delreise. Dermed sikrer vi at billetten hører til en faktisk flyvning, og vi har friheten til å velge ut delreiser fra flyvningen. Vi lagrer også tid for innsjekk. Bagasjeinnsjekk tar vi på bagasje-entiteten.

Bagasje

Enhver billett kan ha tilhørende bagasje. Nøkkel er registreringsnummer. Vi lagrer bagasjens vekt og innleveringstidspunkt (BagasjeInnsjekk).

Billettkjøp

Nøkkel er referansenummer. Alle delreisene i billetkjøpet får vi via kjøp relasjonen. Vi lagrer totalprisen på billettkjøpet, som blir en avledet atttribut fra alle prisene på billettene som hører til reisen.

Kjøp

Relasjon som knytter en kunde opp mot et billettkjøp. En kunde alt mellom ingen og mange billetter, og et billettkjøp finnes ikke uten en kunde.

Relasjoner, deres antagelser og restriksjoner

Flåte

For hvert flyselskap bruker minst en flytype, og har en flåte av fly i denne flytypen. Dette blir en tertiær relasjon mellom disse entitetene. Om vi vil ha alle flyene i et flyselskap senere kan dette utledes fra denne informasjonen. Det blir også enkelt å hente ut alle flyene av en bestemt flytype i en flåte. Et flyselskap må minst tilhøre en flåte forekomst. Vi tenker oss at et fly kan maksimalt tilhøre et flyselskap, men det kan eksistere uten å være eid av et flyselskap. Hver flytype kan finnes uten å inngå i en flåte, og kan også inngå i mange flåter.

ProdusertAv

En flytype må være produsert av en flyprodusent. Vi antar også at en flyprodusent har minst produsert en flytype.

SeteIFly

Et sete må tilhøre et fly, og et fly har mange seter.

ErAvTypen

Hvert fly har en flytype, og det kan finnes mange fly av en gitt flytype. Det virker rimelig å anta at det må finnes et fly av hver type.

RuteTil

En delreise må ha en endeflyplass. En endeflyplass kan mota mange delreiser (men må ikke).

RuteFra

En delreise må ha en startsflyplass. En startsflyplass kan være start for mange delreiser (men må ikke).

BetjenesAv

Relasjon som bestemmer hvilken flytype og hvilket flyselskap som betjener en flyrute. At flytypen faktisk er en del av flåten er en restriksjon vi må sjekke ved oppdateringer. Et vi tenker oss at et flyselskap betjener minst en flyrute. En flytype kan inngå i et vilkårlig antall flyruter. En flyrute må inngå i BetjenesAv, men ikke mer enn en gang.

FlysAv

Bestemmer hvilket fly som skal brukes på en konkret flyvning. En flyvning flys av kun et fly, men det kan settes opp med nullverdi til vi vet det konkretet flyet som skal brukes. Et fly kan registreres til et vilkårlig antall flyvninger. Her tenker vi oss at selskapet har kontroll over om flyet faktisk er tilgjengelig i et tidsrom før det settes opp for en flyvning. Eventuelt er dette noe vi må sjekke selv.

PlanlagtSom

Det kan planlegges ingen eller mange av en gitt flyrute. En bestemt flyvning hører til kun en flyrute.

DelBillett

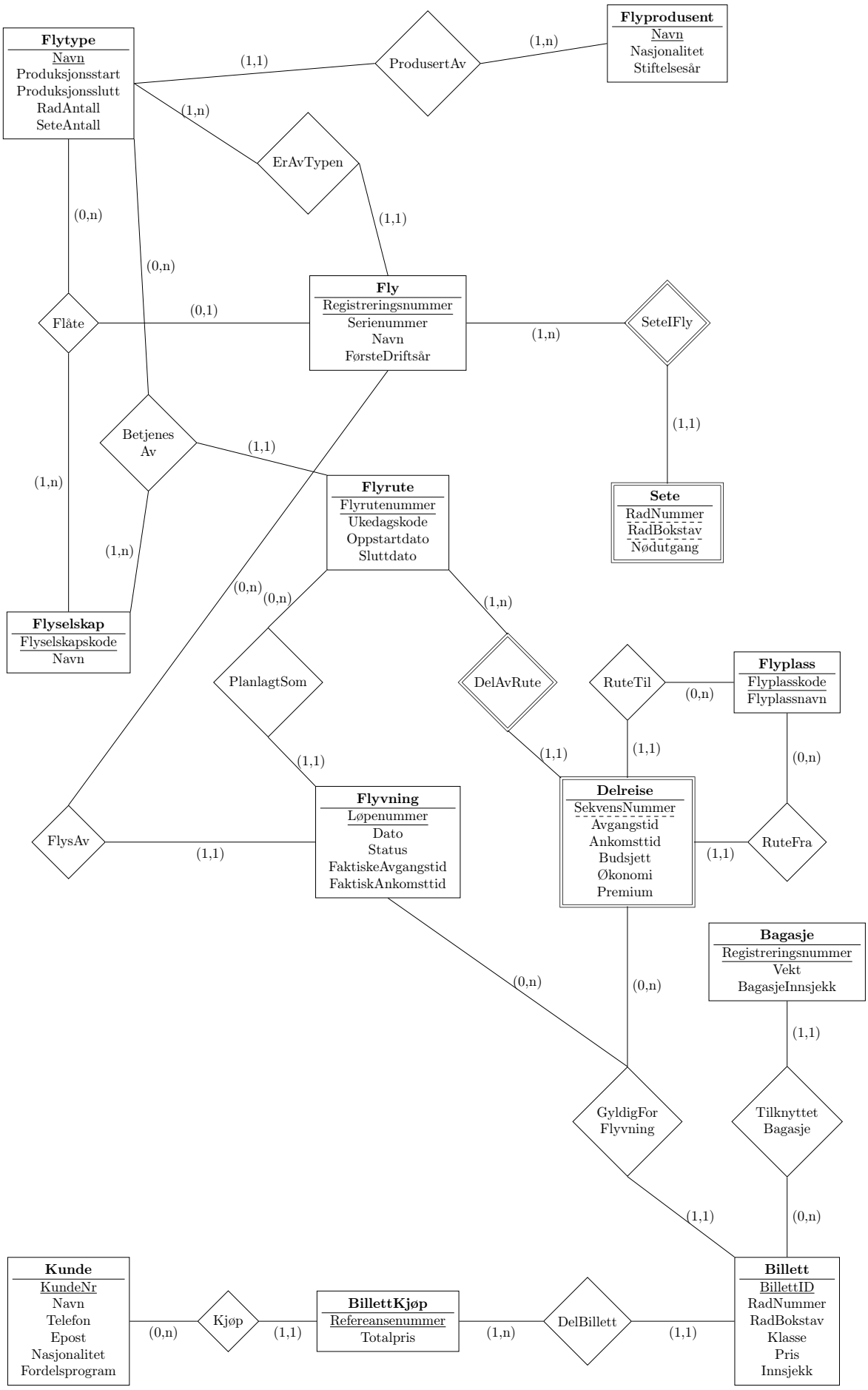
Et billettkjøp inneholder kjøp av minst en billett, opp til vilkårlig mange. En billett må være en del et kun et billettkjøp, så vi bruker den som definerende relasjon for billett entiteten.

GyldigForFlyvning

En billett må være en del kun ett GyldigForFlyvning, mens en flyvning og en delreise kan være del av vilkårlig mange. Relasjonen brukes for å sikre at vi kan kjøpe delreiser, og at delreisene faktisk finnes.

TilknyttetBagasje

En billett kan ha rett til innsjekking av ingen eller flere kolli. Et kolli må være knyttet til en billett.


Flytype
Navn
Produksjonsstart
Produksjonsslutt
RadAntall
SeteAntall
ProdusertAv
Flyprodusent
Navn
Nasjonalitet
Stiftelsesår
ErAvTypen
Fly
Registreringsnummer
Serienummer
Navn
FørsteDriftsår
Flåte
SeteIFly
Betjenes Av
Flyrute
Flyrutenummer
Ukedagskode
Oppstartdato
Sluttdato
Sete
RadNummer
RadBokstav
Nødutgang
Flyselskap
Flyselskapskode
Navn
PlanlagtSom
DelAvRute
RuteTil
Flyplass
Flyplasskode
Flyplassnavn
Flyvning
Løpenummer
Dato
Status
FaktiskeAvgangstid
FaktiskAnkomsttid
FlysAv
Delreise
SekvensNummer
Avgangstid
Ankomsttid
Budsjett
Økonomi
Premium
RuteFra
Bagasje
Registreringsnummer
Vekt
BagasjeInnsjekk
GyldigFor Flyvning
Tilknyttet Bagasje
Kunde
KundeNr
Navn
Telefon
Epost
Nasjonalitet
Fordelsprogram
Kjøp
BillettKjøp
Refereansenummer
Totalpris
DelBillett
Billett
BillettID
RadNummer
RadBokstav
Klasse
Pris
Innsjekk
(1,1)
(1,n)
(0,n)
(0,1)

Relasjonsdatabaseskjema

Kunde

Vi oversetter til Kunde(KundeNr, Navn, Telefon, Epost, Nasjonalitet, Fordelsprogram). Her er det ingen avhengigheter utenom fra nøkkelen, så tabellen er på 4NF.

BillettKjøp og Kjøp

Vi oversetter til BillettKjøp(Referansenummer, Totalpris, Kjøper). Kjøper er fremmednøkkel mot Kunde, kan ikke ha NULL-verdi. Dette dekker da også Kjøp relasjonen. Her er det ingen avhengigheter utenom fra nøkkelen, så tabellen er på 4NF.

Bagasje

Vi oversetter til Bagasje(Registreringsnummer, Vekt, BagasjeInnsjekk, BillettID). BillettID er fremmednøkkel mot Billett, kan ikke ha NULL-verdi. Dette dekker TilknyttetBagasje relasjonen. Her er det ingen avhengigheter utenom fra nøkkelen, så tabellen er på 4NF.

Flyprodusent

Vi oversetter til Flyprodusent(Navn, Stiftelsesår). Her er det ingen avhengigheter utenom fra nøkkelen, så tabellen er på 4NF. Vi trenger tabellen Nasjonalitet(FlyprodusentNavn, Land) siden nasjonalitet er en flerverdiattributt. Her er FlyprodusentNavn fremmednøkkel mot Flyprodusent, kan ikke ta NULL-verdi.

Flytype og ProdusertAv

Vi oversetter til Flytype(Navn, Produksjonsstart, Produksjonsslutt, RadAntall, SeteAntall, FlyprodusentID). FlyprodusentID er fremmednøkkel mot Flyprodusent, kan ikke ta nullverdier. Vi har kun avhengigheter ut av nøkkelen, tabellen er på 4NF.

Fly og ErAvTypen

Vi oversetter til Fly(Registreringsnummer, Serienummer, Navn, FørsteDriftsår, FlytypeID). FlytypeID er fremmednøkkel mot Flytype, kan ikke ha NULL-verdi. Dette tar hånd om ErAvTypen relasjonen. Her er serienummer og produsent en supernøkkel, men siden vi har en nøkkel av lengde en er tabellen fortsatt 4NF.

Sete og SeteIFly

Vi oversetter til Sete(FlyRegNummer, RadNummer, RadBokstav, Nødutgang). FlyRegNummer er fremmednøkkel mot Fly, kan ikke ta NULL-verdi. Dette tar hånd om den definerende SeteIFly relasjonen. Vi har ingen avhengigheter innad i nøkkelen, tabellen er på 4NF.

Flyselskap

Vi oversetter til Flyselskap(Flyselskapskode, Navn). Tabellen er trivielt på 4NF.

Flytype

Vi oversetter til Flytype(Navn, Pro)

Flåte

Vi lager en egen tabell for relasjonen Flåte. Flåte(FlyID, FlytypeID, FlyselskapID). FlyID er fremmednøkkel mot Fly, kan ikke ta NULL-verdi. FlytypeID er fremmednøkkel mot Flytype, kan ikke ta NULL-verdi. FlyselskapID er fremmednøkkel mot Flyselskap, kan ikke ta NULL-verdi. Her blir referansen til fly nøkkel siden et fly maksimalt inngår i en flåte. Siden fly er med i relasjonen, vil ikke flyselskap flerverdi bestemme flytype, så tabellen er på 4NF.

Flyrute

Vi oversetter til Flyrute(Flyrutenummer, Ukedagskode, Oppstartsdato, Sluttdato). Vi har ingen avhengigheter utenom fra nøkkelen, så tabellen er på 4NF.

BetjenesAv

Vi lager en tabell for relasjonen BetjenesAv. BetjenesAv(FlyruteID, FlytypeID, FlyselskapID). FlyruteID er fremmednøkkel mot Flyrute, kan ikke ta NULL-verdi. FlytypeID er fremmednøkkel mot Flytype, kan ikke ta NULL-verdi. FlyselskapID er fremmednøkkel mot Flyselskap, kan ikke ta NULL-verdi. Her blir referansen til Flyrute nøkkel siden det er en total avhengighet med maksimal kardinalitet en. Siden flyrute er en del av tabellen har vi ingen MVD problemer mellom flyselskap og flytype. Tabellen er på 4NF.

Flyvning, FlysAv og PlanlagtSom

Vi oversetter til Flyvning(Løpenummer, Dato, Status, FaktiskAvgangstid, FaktiskAnkomstid, FlyID, FlyruteID). FlyID er fremmednøkkel mot Fly, kan ta NULL-verdi. FlytureID er fremmednøkkel mot Flyrute, kan ikke ta NULL-verdi. Fremmednøkkelene tar hånd om FlysAv og PlanlagtSom relasjonene. Fortsatt ingen transitive avhengigheter og denslags, tabellen er på 4NF.

Flyplass

Oversetter til Flyplass(Flyplasskode, Flyplassnavn). Her har vi kun nøkkelattributter, så tabellen er på 4NF.

Delreise, DelAvRute, RuteTil og RuteFra

Vi oversetter til Delreise(Flyrutenummer, SekvensNummer, Avgangstid, Ankomsttid, Budsjett, Økonomi, Premium, Startflyplass, Endeflyplass, FlyruteID). Startsflyplass er fremmednøkkel mot Flyplass, kan ikke ha NULL-verdi. Endeflyplass er fremmednøkkel mot Flyplass, kan ikke ha NULL-verdi. FlyruteID er fremmednøkkel mot Flyrute, kan ikke ha NULL-verdi. Dette tar hånd om RuteFra, RuteTil og DelAvRute relasjonene. Vi har ingen avhengigheter innad i nøkkelen, og heller ingen avhengigheter utenom fra nøkkelen. Tabellen er på 4NF.

Billett og DelBillett

Vi oversetter til Billett(BillettID, RadNummer, RadBokstav, Klasse, Pris, Innsjekk, BillettKjøpID). BillettKjøpID er fremmednøkkel mot BillettKjøp. Dette tar hånd om DelBillett relasjonen. Her har vi at pris er funksjonelt avhengig av klasse, altså et brudd på 3NF. Vi kunne ha splittet opp tabellene, men det virker ikke verdt den økte kompleksiteten i vårt tilfelle.

GyldigForFlyvning

Vi lager en tabell med BillettID som nøkkel. GyldigForFlyvning(BillettID, Løpenummer, Flyrutenummer, SekvensNummer). BillettID er fremmednøkkel mot Billett, kan ikke ta NULL-verdi. Løpenummer er fremmednøkkel mot Flyvning, kan ikke ta NULL-verdi. Flyrutenummer og SekvensNummer er fremmednøkkel mot Delreise, kan ikke ta NULL-verdi. Tabellen er på 4NF.