



TDT4145 - Datamodellering og Databasesystemer

Prosjekt DB2 : Gruppe 121

John Gøthesen, Timon Selnes, Viktor Tingstad

Innholdsfortegnelse

Informasjon om innlevering:	1
Endringer fra DB1	2
Brukerhistorier	3

Informasjon om innlevering:

I mappen denne .pdf filen ligger vil du også finne mappen 'Deliverables' her finner du kildekoden til prosjektet i en mappe 'Python' med Queries.py og TogDB.py. Utenfor denne kan du se TogDB.sql som vil opprette hele databasen med nødvendig data for brukerhistoriene. I tillegg til denne ligger det allerede en tom databasefil TogDB.db i filstrukturen, det er denne som blir referert til fra Python programmet vårt, tabellene er allerede opprettet ut fra TogDB.sql.

Programmet kan kjøres fra TogDB.py, da vil du i terminalvinduet finne en liste over tilgjengelige funksjoner for å utføre tester til brukerhistoriene, under denne teksten vil du finne eksempler på hvordan resultatene ser ut.

Database er oppdatert med data på en slik måte at de gjelder for én uke, vi har dermed antatt at togruter er like hver uke av året. De er lagt inn tre togruter som hver går på to ulike dager, 2023-04-03 og 2023-04-04 som spesifisert i brukerhistoriene.

Det kan være greit å bemerke seg at listen over ledige plasser som returneres har et plassnummer. Dette plassnummeret er ikke representativt over hvor mange plasser et togoppsett har, men var en enklere abstraksjon å produsere i prosjektet. Dermed vil togoppsett to ha sitteplasser 25 til 36, fremfor 1 til 12.

Eksempel over brukergrensesnitt: Input er én enkel bokstav f.eks: 'F'.

Actions:

```
F - Get all routes stopping at a chosen station on said day.  
B - Get all routes running between two chosen stations on a chosen day.  
R - Register a new customer.  
A - Buy tickets for a route between two stations.  
O - Get all future orders for a customer.
```

Press 'Enter' to quit...

Select action:

Endringer fra DB1

Etter første innlevering oppdaget vi en rekke ting som måtte endres, og ting vi rett og slett hadde glemt å legge til innleveringer, blant disse var det en rekke tabeller i relasjonsskjemaet vårt som ble oversett.

Tabeller som nå har blitt lagt inn er, StasjonITabell, som holder styr over relasjonen mellom Togrutetabell og Jernbanestasjon. Denne skulle allerede ha vært med i forrige innlevering men ble oversett.

Vi valgte så å fjerne superklassen Vogn med Subklassene Sovevogn og Sittevogn. Disse erstattet vi med en vanlig entitetsklasse Vogn, som vi heller brukte attributter til å merke hvilken type vogn den representerer. Vi satt det som et krav at én av disse verdiene måtte være NULL, som kunne løses i programvare. Denne ser nå slik ut:

Vogn(VognID, VognType, RadStørrelse, Kupeer)

Vi bestemmer her at VognType forteller hvilken type vogn dette er, i tillegg har vi attributtene RadStørrelse som blir aktuell for sittevogner, og Kupeer som er aktuell for sovevogner. Én av de to siste attributtene må være NULL.

Neste tabell som måtte opprettes var TogRuteKjørerDag-tabellen, en tabell vi igjen hadde oversett fra første innlevering, men som var nødvendig for å løse funksjonaliteten i programmet vårt.

TogRuteKjørerDag(RuteID, Dag)

Kunde-tabellen har vi endret fra å ha et unikt kundenummer til å bruke e-post som primærnøkkel, dette gjorde søk til kunder enklere i tillegg til å fjerne eventuelle problemer som kunne oppstått ved at to brukere har samme epost, til for eksempel innlogging. SittplassPåBillett og SoveplassPåBillett ble så opprettet som igjen var glemt i det opprinnelige relasjonsskjemaet; med fremmednøkler mot Billett og Sitteplass/Soveplass.

SittplassPåBillett(BillettID, SittplassID)

SoveplassPåBillett(BillettID, SoveplassID)

Brukerhistorier

c) For en stasjon som oppgis, skal bruker få ut alle togruter som er innom stasjonen en gitt ukedag:

```
Select action: F
Station: Mosjøen
Day: Onsdag

Rute: 1, 2, 3, kjører gjennom Mosjøen på Onsdag
```

```
Select action: F
Station: Fauske
Day: Lørdag

Rute: 2, kjører gjennom Fauske på Lørdag
```

d) Bruker skal kunne søke etter togruter som går mellom en startstasjon og en sluttstasjon, med utgangspunkt i en dato og et klokkeslett. Alle ruter den samme dagen og den neste skal returneres, sortert på tid:

```
Select action: B
Start Station: Steinkjer
End Station: Fauske
Enter a date in YYYY-MM-DD format: 2023-04-04

Rutetabell fra 2023-04-04 mellom Steinkjer og Fauske:
| 2 | Steinkjer | 00:57 | Fauske | 08:19 | Tirsdag |
| 1 | Steinkjer | 09:51 | Fauske | 16:49 | Tirsdag |
| 2 | Steinkjer | 00:57 | Fauske | 08:19 | Onsdag |
| 1 | Steinkjer | 09:51 | Fauske | 16:49 | Onsdag |
```

```
Select action: B
Start Station: Trondheim
End Station: Mo i Rana
Enter a date in YYYY-MM-DD format: 2023-04-03
```

Rutetabell fra 2023-04-03 mellom Trondheim og Mo i Rana:

1	Trondheim	07:49	Mo i Rana	14:31	Mandag	
2	Trondheim	23:05	Mo i Rana	05:55	Mandag	
1	Trondheim	07:49	Mo i Rana	14:31	Tirsdag	
2	Trondheim	23:05	Mo i Rana	05:55	Tirsdag	

e) En bruker skal kunne registrere seg i kunderegisteret:

```
Select action: R
Email: test@testemail.com
First name: Ola
Last name: Nordmann
Phone: 12345678
Kunde registrert! ('test@testemail.com', 'Ola', 'Nordmann', '12345678')
```

Dersom en epost allerede er registrert:

```
Select action: R
Email: test@testemail.com
First name: Test
Last name: Duplicate
Phone: 12345678
Kunde ikke registrert! Feilmelding: UNIQUE constraint failed: Kunde.Epost
```

g) Registrerte kunder skal kunne finne ledige billetter for en oppgitt strekning på en ønsket togrute og kjøpe de billettene hen ønsker:

```
Select action: A
If you wish to see all routes between two stations, please use the "B" action.

Route: 1
Start Station: Trondheim
End Station: Mosjøen

Sitteplasser:
['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23', '24']
Soveplasser:
[]

Email: test@testemail.com
Sitte- eller soveplass? (sitte/sove): sitte
SitteplassID: 4
Billett kjøpt!
```

Vi kan så se hvis kjører en spørring hvor sete 4 nå er tatt at den ikke vil være mulig å kjøpe dersom den er en del av strekningen du så spør etter:

```
Route: 1
Start Station: Steinkjer
End Station: Mo i Rana

Sitteplasser:
['1', '2', '3', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23', '24']
Soveplasser:
[]
```

Dette oppdateres naturligvis til så mange plasser som ikke lenger er tilgjengelige:

```
Route: 1
Start Station: Steinkjer
End Station: Bodø

Sitteplasser:
['1', '3', '5', '6', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23', '24']
Soveplasser:
[]
```

h) For en bruker skal man kunne finne all informasjon om de kjøpene hen har gjort for fremtidige reiser:

Slik ser vi alle være kjøpte billetter til en kunde:

```
Select action: 0
Email: test@testemail.com

Future tickets for test@testemail.com:

Dato: 2023-04-03      Fra: Trondheim  Til: Mosjøen  Sitte/Soveplassnummer: 4
Dato: 2023-04-03      Fra: Steinkjer  Til: Fauske   Sitte/Soveplassnummer: 8
Dato: 2023-04-03      Fra: Steinkjer  Til: Fauske   Sitte/Soveplassnummer: 7
Dato: 2023-04-03      Fra: Steinkjer  Til: Fauske   Sitte/Soveplassnummer: 2
```

I programmet er det lagt til noe validering av inputs, det vil for eksempel ikke være mulig å oppgi stasjoner som ikke finnes, da dette slås opp i databasen før en spørring. Vi har også lagt til en enkel Regex validering av e-post adresser i tillegg til av vis sjekker dagen du etterspør finnes, disse ligger dog hardkodet i Queries.py.