NTNU

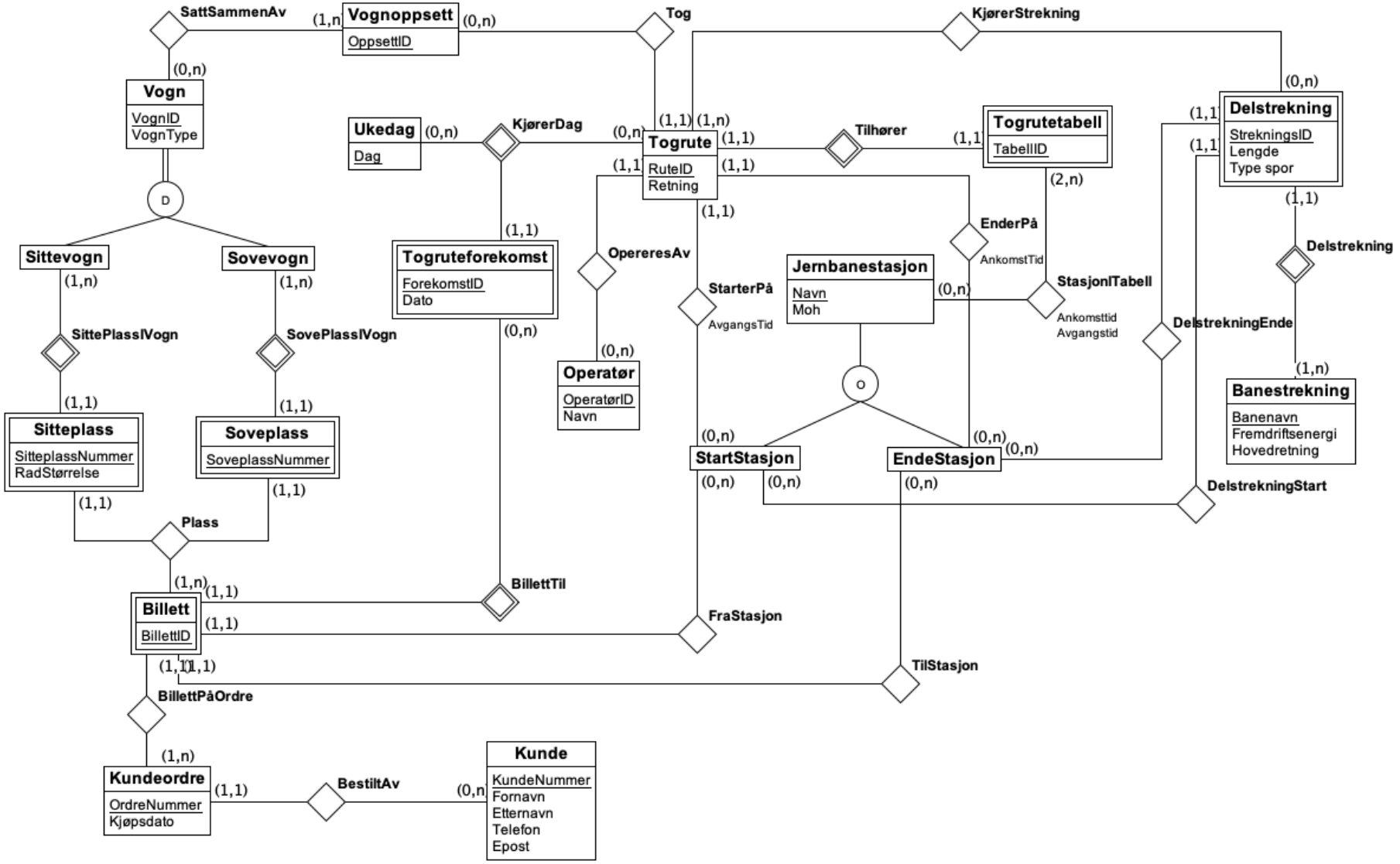
TDT4145 - Datamodellering og Databasesystemer

Prosjekt DB1 : Gruppe 121

John Gøthesen, Timon Selnes, Viktor Tingstad

Innholdsfortegnelse

A) EER-diagram



B) Relasjonsskjema

Jernbanestasjon(Navn, Moh) *(4NF)*
StartStasjon(Navn, Moh) *(4NF)*
EndeStasjon(Navn, Moh) *(4NF)*
Banestrekning(Banenavn, Fremdriftsenergi, Hovedretning) *(4NF)*
Delstrekning(StrekningsID, Lengde, TypeSpor, Navn, Navn) *(4NF)*
- Navn er en fremmednøkkel mot StartStasjon
- Navn er en fremmednøkkel mot EndeStasjon

DelbanerPåBane(StrekningsID, Banenavn) *(4NF)*
- Hvor Banenavn er en fremmednøkkel mot Banestrekning

Operatør(OperatørID, Navn) *(4NF)*
Togrute(RuteID, Retning, OperatørID) *(4NF)*
- OperatørID er en fremmednøkkel mot Operatør

Togruteforekomst(ForekomstID, RuteID, Dato) *(4NF)*
- RuteID er en fremmednøkkel mot TogRute

Togrutetabell(TabellID, RuteID) *(4NF)*
- RuteID er en fremmednøkkel mot TogRute

Ukedag(Dag) *(4NF)*
Vognoppsett(OppsettID)
Sovevogn(VognID, VognType)
Sittevogn(VognID, VognType)
Sitteplass(VognID, SitteplassNummer, VognType, RadStørrelse) *(4NF)*
- VognID er en fremmednøkkel mot VognType

(Antar at SitteplassNummer resetter per vogn, og dermed trenger VognID som en del av nøkkelen)
Soveplass(VognID, SoveplassNummer) *(4NF)*
- VognID er en fremmednøkkel mot VognType

(Antar at SoveplassNummer resetter per vogn, og dermed trenger VognID som en del av nøkkelen)
Billett(BillettID, ForekomstID, Navn, Navn) *(4NF)*
- ForekomstID er en fremmednøkkel mot Togruteforekomst
- Navn er en fremmednøkkel mot StartStasjon
- Navn er en fremmednøkkel mot EndeStasjon

Kundeordre(Ordrenummer, Kjøpsdato, Kundenummer) *(4NF)*
- Kundenummer er en fremmednøkkel mot Kunde

Kunde(Kundenummer, Fornavn, Etternavn, Telefon, Epost) *(4NF)*

Relasjonene og tabellene i skjemaet oppfyller 4NF, det er ingen redunante flerverdiattributter som strider med 4NF; og heller ingen funksjonelle avhengigheter som ville skapt redundans.

DelbanerPåBane *er laget for å kunne oppnå en høyere normalform, ettersom StrekningsID gjør Banenavn redundant dersom den ikke legges i en egen tabell. Dermed får vi 4NF.*

For ***Sitteplass*** *og* ***Soveplass*** *bruker vi kompositttnøkler med både Nummer som ID til subklassen, men også med VognID, derfor trenger vi ikke en egen tabell for disse.*

C) SQL - Antagelser og spesifikasjoner

SQL-scriptet leveres som en separat fil (se sql.sql)

Vi har valgt å ta ut lengde fra **Banestrekning** for å fjerne uønsket redundans, denne lengden kan finnes ut fra total av lengder av de relaterte **delstrekningene** dermed slipper vi å måtte oppdatere flere steder dersom det skulle vært aktuelt.

Kunderegister var en entitet vi opprinnelig startet med, men vi ser at tabellen over kunder effektivt fungerer som et kunderegister, vi har dermed besluttet å fjerne entiteten siden **Kunde**-tabellen utfører oppgaven vi antar trenges.

En del av oppgavene til databasen vi har modellert er nødvendig å løse i programvare. Blant problemet som må løses her har vi hvilke rader og vogner en **Sitteplass** eller **Soveplass** tilhører. Her har vi sett for oss at vi kan bruke SitteplassNummer i vognen for å finne dets tilhørende rad basert på *RadStørrelse*. Ettersom hver **Vogn** har sin egen VognID blir det mulig å finne riktig vogn i programvare også. Det samme konseptet vil gjelde for **Soveplass**. Her vil vi kunne bruke faktumet at hver kupé inneholder to soveplasser til å finne hvilken kupé billetten tilhører, i samspill med VognID for å finne riktig vogn.

Her kommer vi videre til et problem vi blir nødt til å løse i programvare, når én sengeplass bestilles til en **Togruteforekomst** må vi sørge for at denne gjelder for hele togruten, og at én seng tilsvarer en hel kupé; slik at man ikke deler med andre som spesifisert. Logikken rundt når et sete blir ledig og ikke, blir også fikset i programvare, men informasjonen om start- og endestasjon for plassen står naturligvis i databasen.

Vi antar at systemet fungerer på lignende måte som i virkeligheten, og at **Vognoppsett** tilhører en togrute og dermed banestrekningen, fremfor at de har noen reell relasjon til operatøren. Vi antar videre at en togrute har ett fast **Vognoppsett** derfor har vi modellert kardinaliteten til å være en én-til-én relasjon.

Vi har valgt å ha to subklasser ut fra **Jernbanestasjon** for å representere start- og endestasjoner. For at entiteten fortsatt skal kunne være en vanlig stasjon, altså et stopp, har vi gjort at entitetsklassene til subklassene er delvis overlappende slik at en **Jernbanestasjon** kan stå for seg selv uten å nødvendigvis måtte være en start- eller endestasjon.
Hvilke stasjoner en **Togrute** stopper på finnes i **Togrutetabell**-tabellen. Ettersom **Togrutetabell** er en svak entitet av Togrute synes vi det passer godt at hvilke stasjoner det stoppes på finnes her, fremfor en til relasjon mot **Jernbanestasjon**.

Vi har ikke tatt med **Vogn** i relasjonsskjema eller i SQL ettersom dette er en superklasse med totale subklasser, dermed vil aldri **Vogn** være reell å produsere. I motsetning til med **Jernbanstasjon** superklassen, som er delvis, og vil kunne brukes.