Innhold

1 Oppgaven **1**

1.1 **a)** Opprette tabeller 1
1.1.1 Eier 1
1.1.2 Kjøretøy 1
1.1.3 Kjøretøy eier 2
1.1.4 Bompassering 2
1.1.5 Bomstasjon 2
1.2 **b)** Legge inn test data 2
1.3 **c)** Dropp telefonnummer 3
1.4 **d)** Håndtere passering med manglende skilt 3
1.5 **e)** List all informasjon 3
1.6 **f)** List all informasjon der det er registrert registreringsnummer . 4
1.7 **g)** Nå med relasjonsalgebra 4
1.8 **h)** Vis antall passeringer på ulike registreringsnummer 4
1.9 **i)** Vis siste passering med registreringsnummer AA100000 4
1.10 **j)** Vis antall passeringer uten registreringsnummer 4
1.11 **k)** Kan vi håndtere utenlandske registreringsnummer? 4
1.12 **l)** Oppgaven ønsker egentlig 2 tabeller, hvilke andre ønsker vi? . 4

1 Oppgaven

1.1 a) Opprette tabeller

For å oppfylle kravene oppgitt i oppgaven om å lagre informasjon om bilene og eierne, ble det vurdert å lage 5 tabeller. Nedenfor er SQL-koden for å opprette tabellene med forklaring.

1.1.1 Eier

Vi oppretter tabellen **eier** for å lagre informasjon om eierne, inkludert navn, adresse, telefonnummer og e-postadresse. Hver eier får en unik ID som primærnøkkel.

```
CREATE TABLE IF NOT EXISTS eier (
    id SERIAL PRIMARY KEY,
    navn VARCHAR(255) NOT NULL,
    adresse VARCHAR(255) NOT NULL,
    telefon VARCHAR(20),
    epost VARCHAR(255)
);
```

1.1.2 Kjøretøy

Deretter oppretter vi tabellen **kjoretoy** for å lagre informasjon om kjøretøyene, som for eksempel registreringsnummer. Hver kjøretøy får en unik ID som pri-

mærnøkkel. Tanken med denne tabellen er å kunne lagre flere kjøretøy for hver eier, og dermed håndtere situasjoner der en eier har flere biler.

```
CREATE TABLE IF NOT EXISTS kjoretoy (
    id SERIAL PRIMARY KEY,
    registreringsnummer VARCHAR(20) UNIQUE NOT NULL
);
```

1.1.3 Kjøretøy eier

kjoretoy_eier tabellen bruker vi som en mellomtabell for å hondtere forholdet mellom kjøretøy og eiere, vi tenkte at en eier kan ha flere kjøretøy.

```
CREATE TABLE IF NOT EXISTS kjoretoy_eier (
    kjoretoyid INTEGER,
    eierid INTEGER,
    FOREIGN KEY (kjoretoyid) REFERENCES kjoretoy(id),
    FOREIGN KEY (eierid) REFERENCES eier(id)
);
```

1.1.4 Bompassering

I **bompassering** tabellen lagrer vi informasjon om hver passering, inkludert tidspunktet for passeringen, kjøretøyets ID og bomstasjonens ID. Vi bruker fremmednøkler for å referere til kjøretøy og bomstasjoner.

```
CREATE TABLE IF NOT EXISTS bompassering (
    id SERIAL PRIMARY KEY,
    passeringstid TIMESTAMP NOT NULL

    kjoretoyid INTEGER,
    bomstasjonid INTEGER,
    FOREIGN KEY (kjoretoyid) REFERENCES kjoretoy(id),
    FOREIGN KEY (bomstasjonid) REFERENCES bomstasjon(id),
);
```

1.1.5 Bomstasjon

bomstasjon tabellen lagrer informasjon om bomstasjonene, inkludert navn og plassering.

```
CREATE TABLE IF NOT EXISTS bomstasjon (
    id SERIAL PRIMARY KEY,
    navn VARCHAR(255) NOT NULL,
    plassering VARCHAR(255) NOT NULL
);
```

1.2 b) Legge inn test data

Vi legger inn data for å teste ulike scenarier, inkludert biler med og uten eier, og passeringer uten skilt.

```
INSERT INTO eier
    (navn, adresse, telefon, epost)
VALUES
    (`Ola Nordmann`, `Osloveien 1`, `90000000`, `ola@test.no`),
    (`Kari Posten`, `Bergenstorget 2`, `40000000`, `kari@test.
no`);

INSERT INTO kjoretoy
    (registreringsnummer)
VALUES
    (`AA10000`),
    (`BT20000`),
    (NULL);

INSERT INTO bomstasjon
    (navn, plassering)
VALUES
    (`Hovedveien 1`, `Oslo`),
    (`Fjordkryssing`, `Bergen`);

INSERT INTO bompassering
    (passeringstid, kjoretoyid, bomstasjonid)
VALUES
    (CURRENT_TIMESTAMP, 1, 1),
    (CURRENT_TIMESTAMP, NULL, 2);
```

1.3 c) Dropp telefonnummer

Hvis det ikke er behov å lagre telefonnummeret i eier tabellen, så er det mulig å fjerne det på denne måten.

```
ALTER TABLE eier DROP COLUMN telefon;
```

1.4 d) Håndtere passering med manglende skilt

I vår løsning håndteres dette ved at feltet `kjoretoyid` i tabellen `bompassering` tillater `NULL`-verdier. Når et skilt ikke kan leses, registreres passeringen med tid og sted, men uten referanse til et kjøretøy. En alternativ måte kunne vært å registrere et Dummy-kjøretøy"med registreringsnummer 'UKJENT'.

1.5 e) List all informasjon

Her bruker vi `LEFT JOIN` for å sikre at passeringer uten bilreferanse inkluderes.

```
SELECT p.*, k.registreringsnummer, e.navn, e.epost
FROM bompassering p
LEFT JOIN kjoretoy k ON p.kjoretoyid = k.id
LEFT JOIN kjoretoy_eier ke ON k.id = ke.kjoretoyid
LEFT JOIN eier e ON ke.eierid = e.id;
```

1.6 f) List all informasjon der det er registrert registreringsnummer

Her bruker vi `INNER JOIN`, som ekskluderer rader hvor det ikke er match (dvs. uleselige skilt).

```
SELECT p.*, k.registreringsnummer, e.navn, e.epost
FROM bompassering p
JOIN kjoretoy k ON p.kjoretoyid = k.id
JOIN kjoretoy_eier ke ON k.id = ke.kjoretoyid
JOIN eier e ON ke.eierid = e.id;
```

1.7 g) Nå med relasjonsalgebra

Dette uttrykkes ved projeksjon (π) og naturlig join ($\bowtie$):

$$\pi_{p.*,k.regnr,e.navn,e.epost}(bompassering \bowtie kjoretoy \bowtie kjoretoy\_eier \bowtie eier)$$

1.8 h) Vis antall passeringer på ulike registreringsnummer

```
SELECT k.registreringsnummer, COUNT(p.id) AS antall_passeringer
FROM kjoretoy k
JOIN bompassering p ON k.id = p.kjoretoyid
GROUP BY k.registreringsnummer;
```

1.9 i) Vis siste passering med registreringsnummer AA100000

1.10 j) Vis antall passeringer uten registreringsnummer

1.11 k) Kan vi håndtere utenlandske registreringsnummer?

1.12 l) Oppgaven ønsker egentlig 2 tabeller, hvilke andre ønsker vi?