Obligatorisk innlevering 1 - SQL

Gruppenummer: 33

9. februar 2026

Innhold

1 Introduksjon

Denne oppgaven omhandler utvikling av en databaseløsning for registrering av bompasseringer. Formålet er å kunne lagre informasjon om biler, deres eiere, samt detaljer om hver enkelt passering ved en bomstasjon. Vi skal løse spesifikke krav knyttet til datadefinisjon (DDL), datamanipulering (DML) og uthenting av statistikk.

2 Metode

For å løse oppgaven har vi valgt en relasjonsdatabasedesign. Vi har vurdert normalisering for å unngå dataduplisering, spesielt med tanke på forholdet mellom eiere og kjøretøy.

Valg	**Fordel**	**Ulempe**
Flere tabeller	Bedre dataintegritet og mindre redundans.	Mer komplekse spørringer (JOINs).
Bruk av NULL	Håndterer manglende skiltlesing.	Krever bevisst bruk av Outer Joins i spørringer.

Tabell 1: Metodevalg og vurderinger.

3 Oppgaven

3.1 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.

3.1.1 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)
);
```

3.1.2 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
);
```

3.1.3 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)
);
```

3.1.4 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),
);
```

3.1.5 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
);
```

3.2 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);
```

3.3 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;
```

3.4 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'.

3.5 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;
```

3.6 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;
```

3.7 g) Nå med relasjonsalgebra

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

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

3.8 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;
```

3.9 i) Vis siste passering med registreringsnummer AA100000

```
SELECT MAX(passeringstid)
FROM bompassering p
JOIN kjoretoy k ON p.kjoretoyid = k.id
WHERE k.registreringsnummer = `AA10000`;
```

3.10 j) Vis antall passeringer uten registreringsnummer

```
SELECT COUNT(*)
FROM bompassering
WHERE kjoretoyid IS NULL;
```

3.11 k) Utenlandske registreringsnummer

Løsningen kan håndtere disse så lenge `VARCHAR`-feltet for registreringsnummer er fleksibelt nok. En utfordring er manglende tilgang til utenlandske eierregistre, som gjør fakturering vanskelig.

3.12 l) Andre nyttige tabeller

Selv om oppgaven nevner to tabeller, har vi inkludert **bomstasjon** for å unngå å lagre tekstnavn repetitivt. En tabell for **faktura** eller **prisgrupper** (takster basert på tid/kjøretøytype) ville også vært nyttig i et reelt system).

4 Diskusjon

Gjennom denne oppgaven har vi etablert en normalisert databasestruktur som ivaretar sporbarhet for bompasseringer. Valget om å bruke en mellomtabell (`kjoretoy_eier`) gjør at systemet støtter at en bil kan ha flere eiere over tid, eller at en eier har flere biler. Ved bruk av `LEFT JOIN` har vi sikret at også tapte"inntekter (uleselige skilt) kan loggføres og analyseres.