INF105 gruppetime 11

Del 1 - Relasjonell modell fra CSV: eksempel til manuell konvertering

1. Last ned vår kopi av CSV filen til akvakulturregisteret fra
   https://www.inf105.org/Akva.csv
   (kilde: https://www.fiskeridir.no/registre/akvakulturregisteret , 2026-11-04)
2. Lag en ny tom sqlite database: `sqlite3 akva.db`
3. I sqlite-kommandolinjen, importer alle kolonnene fra CSV-filen i en stor sqlite3-tabell:
   ```
   .mode csv
   .separator ";"
   .import Akva.csv akva_orig
   ```
4. Sjekk den nye tabellen, f.eks
   ```
   SELECT count(*) FROM akva_orig; -- burde vise 27402 rader
   .schema
   SELECT DISTINCT POSTNR FROM akva_orig ORDER BY POSTNR DESC;
   ```
5. Skru på foreign key checking for senere bruk: `PRAGMA foreign_keys=1;`
6. Vi lager en ny tabell `akva` som kopi, for å beholde referanse-data uendret:
   `CREATE TABLE akva AS SELECT * FROM akva_orig;`
7. Nå begynner vi normalisering: vi lager en ny tom tabell til organisasjoner:
   CREATE TABLE `org (nr INTEGER PRIMARY KEY, navn NOT NULL, gate, postnr, poststed);`
8. Når vi prøver å fylle *org* fra den gamle tabellen med
   ```
   INSERT OR ROLLBACK INTO org
   SELECT distinct "ORG.NR_PERS.NR" AS nr, NAVN AS navn, ADRESSE AS gate,
   POSTNR AS postnr, POSTSTED AS poststed FROM akva;
   ```

   får vi “feil datatype” errrors. (Hva gjør “or rollback” her?)
   Grunnen ser vi når vi sjekker output til bare SELECT statementet: noen rader i “ORG.NR_PERS.NR” er tomme (personnummer til individer må skjules fra fiskeridirektoratet sin side). Disse verdiene kan ikke konverteres til INTEGER PRIMARY KEY. Skal vi beholde disse radene, må vi komme med en løsning, f.eks legge inn fiktive personnumrene fra https://norske-testdata.no/fiktivt-fnr/ . Men la oss bare droppe dem fra datasettet den gangen:
   `DELETE from akva where "ORG.NR_PERS.NR" = '';`
   (dette er 2 single quotes!) Prøv INSERT på nytt, det burde gå nå, med 494 orgs.
9. Vi kan nå slette de kolonnene vi har knyttet til org.nr:
   ```
   ALTER TABLE akva DROP COLUMN NAVN;   -- osv...
   ```

10. Nå kan vi prøve å knytte tillatelser til org med en foreign key:
    ```
    CREATE TABLE till (nr PRIMARY KEY, dato, orgnr INTEGER,
    FOREIGN KEY (orgnr) REFERENCES org(nr));

    INSERT OR ROLLBACK INTO till SELECT DISTINCT TILL_NR as nr,
    TILDELINGSTIDSPUNKT as dato, "ORG.NR_PERS.NR" from akva;
    ```

    Så kan vi igjen fjerne de redundante kolonnene fra akva (ikke fjern de kolonnene vi bruker som primary key!)

11. Det finnes mange flere redundante kolonner, f.eks *art - art_nr*, eller *kommune_nr - navn - fylkenr - fylkenavn* i to ulike steder.
12. Fortsett med å lage et normalisert sett med tabeller som bruker primary keys og foreign keys for å øve på kommandoene. Vi legger ut et ferdig forslag neste uke.

Del 2 - SQL join - movies.db

Last ned https://www.inf105.org/movies.db

1. Lag noen spørsmål som krever å knytte sammen flere av tabellene, f.eks
   Hvile roller har vi i databasen til skuespilleren Sam Neill? Tabellen *people* har navnet til Sam Neill mens rollene er i tabellen *actors*. Her må vi bruke JOIN ... ON
   ```
   SELECT a.character, a.movie_id
   FROM actors as a
     JOIN people as p ON a.actor_id=p.id
   WHERE p.name = 'Sam Neill';
   ```

2. For å få navn til filmer må vi JOINe en tabell til:
   ```
   SELECT a.character, m.title
   FROM actors as a
      JOIN people as p ON a.actor_id=p.id
      JOIN movies as m ON a.movie_id=m.id
   WHERE p.name = 'Sam Neill';
   ```

3. Nå kan vi lage komplisertere spørsmål. Hvilken av disse filmene var mest profitabelt?
   ```
   SELECT a.character, m.title, (m.revenue-m.budget) as profit
   FROM actors as a JOIN people as p ON a.actor_id=p.id
   JOIN movies as m ON a.movie_id=m.id WHERE p.name = 'Sam Neill'
   ORDER BY profit DESC LIMIT 3;
   ```

4. Lag egne spørsmål til de andre rundt bordet. Også ta med spørsmål der en tabell brukes flere ganger: hvilke 2 skuespillere er oftest sammen, alle filmer med King og Queen i roller, ...

En komplisertere database til å utforske er https://www.sqlitetutorial.net/sqlite-sample-database/