Oppgaver til Databaser: DDL & CRUD

Oppgave - begreper & terminologi

Gå sammen to og to, forklar **kort/enkelt** følgende terminologi og databasebegreper for hverandre/sammen:

- Primærnøkkel ("Primary Key" – PK)
- Fremmednøkkel ("Foreign Key" – FK)
- Løpenummer, altså SERIAL

Oppgave – DDL og CRUD

For disse oppgavene, opprett en tom eksempeldatabase. Kall den **`TestDDL`** e.l. Jobb mot denne databasen.

Skriv SQL mot den nye databasen og utfør følgende:

1. Opprett tabellen `person`. Tabellen skal inneholde:
   `ID`, `SERIAL`.
   `Personnummer`, `CHAR(11)`, skal ikke kunne være `null` og skal være unik.
   `Navn`, `TEXT`, skal ha defaultverdi `'Ukjent'`.
   `EPost`, `TEXT`, skal være unik.
   Som primærnøkkel skal kolonnen `ID` angis.

   Sjekk at tabellen er opprettet, evt. ved å refresh'e DB-tabell oversikten først.

2. Legg inn en rad i tabellen person. Spesifiser følgende data: (opprett gjerne en kvinne i stedet)
   Personnummer: `'12345678901'`
   Navn: `'Ola Nordmann'`
   EPost: `'ola@nordmann.no'`

   Skriv en passende spørring for å se at dataene ligger i tabellen.

3. Du finner ut at det er kjekt å registrere fødselsdato også. Modifiser den eksisterende `person` tabellen, så den også inneholder kolonnen:
   `Foedselsdato`, `DATE`.

   Sjekk at tabellen har fått med seg endringene. (Tilsvarende fremgangsmåte som i oppg 1.)

4. Legg inn en ny rad i tabellen person. Spesifiser følgende data:
   personnummer: `'98765432109'`

   Skriv en passende spørring for å se at data ligger i tabellen som forventet.

5. Oppdater raden med personnummeret '98765432109', slik at Foedselsdato til denne blir 1. januar 1990. **Merk:** Datofelt formateres internt i databasen, en mulighet er å oppgi det slik: 'yyyy-mm-dd'. (I dette tilfellet er y = år, m = måned d = dag.) Evt. google hvordan du kan sette inn dette på ønsket format (hvordan du bruker en funksjon for å angi dato på et selvvalgt format).

6. Legg inn deg selv i person tabellen! Fyll inn alle felter (trenger ikke bruke ekte info da).

7. Vi innser at databaseløsningen vår er litt snevert designet: Hver person kan bare ha 1 epost adresse. Opprett en egen tabell epost med følgende kriterier:
   `Adresse, TEXT`, skal ikke kunne være `null` og skal være unik.
   `Type`, `TEXT`.
   `Person_ID`, `INT`, skal ikke kunne være `null`.
   Som primærnøkkel skal kolonnen `Adresse` angis.
   Som fremmednøkkel `Person_ID` som refererer tabellen `person` sin kolonne `ID`.

   Skriv en passende spørring for å se at tabellen er opprettet som forventet.

8. Lag en tabell tilsvarende den i punkt 1 som heter `personCopy`. (Du trenger ikke kopiere data inn i den.) Sjekk at denne er opprettet, før du sletter den igjen. (Poenget er å få testet kommandoen for å slette tabell, uten å ødelegge de to fine tabellene du har jobbet med en stund nå.)

   Sjekk at tabellen er slettet.

9. **VANSKELIG!** Skriv en SQL som kopierer epostadresser og IDer fra person tabellen over til Adresse og Person_ID i epost tabellen. (Google detaljene, her må vi bruke en SELECT statement som del av vår INSERT INTO statement.)

   Skriv en passende spørring for å se at data ligger i tabellen som forventet.

10. (Hvis du har klart oppgaven over)
    Nå som vi har flyttet epost ut i en egen tabell, så kan vi slette epostkolonnen fra `person`-tabellen. Vi ønsker å fjerne hele kolonnen, ikke bare dataene som ligger der.

11. **VANSKELIG:** Hvis vi henter ut navn og fødselsdato fra person tabellen, kommer dato på databasen sitt format. Omformater output så dato har formatet:
    31.01.2001 (<-- Eksempel på dato med ønsket format.)
    Tips: Google PostgreSQL sine funksjoner for å omformatere dato output. (NB: Navn på funksjon er avhengig av database, her er det forskjell på PostgreSQL, SQL Server og MySQL.)

    Legg også inn dato for Ola Nordmann. Han skal ha fødselsdatoen 20.02.1912 (gammel mann!). Skriv den inn på *dette formatet*, bruk en funksjon som gjør at databasen skjønner hvilket format datoen kommer på.