IN2090

Datamanipulering, skranker og views

Typer SQL-spørringer

- SELECT henter informasjon (svarer på et spørsmål)
- CREATE lager noe (f.eks. en ny tabell)
- INSERT setter inn rader i en tabell
- UPDATE oppdaterer data i en tabell
- DELETE sletter rader fra en tabell
- DROP sletter en hel ting (f.eks. en hel tabell)

CREATE

- Brukes for å lage tabeller, brukere, skjemaer, osv.
- For å lage skjema

```
>> CREATE SCHEMA <skjemanavn>;
```

- For å lage tabeller

```
>> CREATE TABLE <tabellnavn> (<kolonner>)
```

- En kolonne-deklarering inneholder:
  - et kolonnenavn, og
  - en type,
  - og en liste med skranker (contraints)

Skranker

- NOT NULL – tillater ikke NULL-verdier I en kolonne
- UNIQUE – hvis vi ønsker at en kolonne aldri skal gjenta en verdi/ikke inneholde duplikater.
- PRIMARY KEY - gjør verdien til primærnøkkel I tabelle, kan bare ha en per tabell. PRIMARY KEY er en kombinasjon av NOT NULL og UNIQUE, kan derfor lage kandidatnøkler på denne måten.

EKSEMPEL

Man skriver skranker til slutt når man ønsker skranker over flere kolonner. F.eks her er kombinasjonen av StdName og StdBirthdate alltid unik

```
1   CREATE TABLE Students (
2       SID int PRIMARY KEY,
3       StdName text NOT NULL,
4       StdBirthdate date
5   );
6
7   CREATE TABLE Students (
8       SID int,
9       StdName text NOT NULL,
10      StdBirthdate date
11      CONSTRAINT sid_pk PRIMARY KEY (SID),
12      CONSTRAINT name_bd_un UNIQUE (StdName, StdBirthdate)
13  );
```

REFRENCES

- Fremmednøkler

```
15    CREATE TABLE TakesCourse (
16        SID int REFERENCES Students (SID),
17        CID int REFERENCES Course (CID),
18        Semester text
19    );
```

INSERT

- Brukes for å sette inn data I en tabell

```
>> INSERT INTO <tabell>
>> VALUES (<rad>),
          (<rad>),
           ...,
          (<rad>);
```

```
21    INSERT INTO Students
22    VALUES (0, 'Anna Consuma ', '1978 -10 -09 '),
23           (1, 'Peter Young ', '2009 -03 -01 ');
```

Alternative metoder

```sql
CREATE TABLE Students2018 (
    SID int PRIMARY KEY,
    StdName text NOT NULL
);

INSERT INTO Students2018
SELECT S.SID , S.StdName
    FROM Students AS S INNER JOIN TakesCourse AS T
        ON (S.SID = T.SID)
    WHERE T.Semester LIKE '%18 ';

CREATE TABLE Students2018 AS
SELECT S.SID , S.StdName
    FROM Students AS S INNER JOIN TakesCourse AS T
        ON (S.SID = T.SID)
    WHERE T.Semester LIKE '%18 ';
```

DEFAULT og SERIAL

- DEFAULT – gi en kolonne en standard verdi, brukes dersom vi ikke oppgir en verdi for kolonnen
- SERIAL – kan brukes for primærnøkler som bare er heltall, databsen genererer unike heltall for hver rad

```
43  CREATE TABLE personer (
44      pid int PRIMARY KEY,
45      navn text NOT NULL,
46      nationalitet text DEFAULT 'norge'
47  );
```

```
49  CREATE TABLE Students (
50      SID SERIAL PRIMARY KEY, -- merk ingen type
51      StdName text NOT NULL,
52      StdBirthdate date
53  );
```

DROP

- Brukes for å slette ting (tabeller, skjemaer, brukere, osv.)

```
>> DROP TABLE <tabellnavn>;
>> DROP SCHEMA <skjemanavn>;
```

- For å slette alt som avhenger av det vi sletter

```
>> DROP TABLE <tabellnavn> CASCADE;
```

- Denne kommandoen sletter tabellen, men også tabeller som avhenger av denne.

DELETE

- Brukes for å slette rader I tabell

```
>> DELETE
>> FROM <tabellnavn>
>> WHERE <betingelse>
```

```
55    DELETE
56    FROM Students
57    WHERE StdBirthdate > '1990 -01 -01 ';
```

ALTER

- For å oppdatere skjemaelementer (endre navn)

```
>> ALTER TABLE <tabellnavn>
>> RENAME TO <nytt tabellnavn>
```

- Legge til kolonner

```
>> ALTER TABLE <tabellnavn>
>> ADD COLUMN <kolonne>;
```

- Legge til skranker

```
>> ALTER TABLE <tabellnavn>
>> ADD CONSTRAINT <betingelse>;
```

UPDATE

- Oppdaterer verdiene I en tabell

```
>> UPDATE <tabellnavn>
>> SET <oppdateringer>
>> WHERE <betingelse>
```

- Oppdateringene blir eksekvertt for hver rad som oppfyller betingelsen

CHECK

- Lar oss bruke et genrelt uttrykk for å avgjøre om verdier kan settes inn i kolonnen eller ikke

```
>> CREATE TABLE <tabellnavn> (
>>  <kolonner>
>> CHECK (betingelse)
>>);
```

Typer

- Numeriske typer (int, decimal, serial, osv)
- Strengtyper (char, text)
- Dato- og tidstyper (timestap, date, time)
- Boolean
- Array

VIEW

- Ofte er vi ikke interessert I dataene slik de er lagret
- Av og til vil en bestemt spørring bli eksekvert veldig ofte
- Det er da upraktisk å måtte skrive den ut hver gang
- Vi kan derfor bruke VIEW
- Et VIEW er egentlig bare en navngitt spørring
- Et VIEW kan brukes som en vanlig tabell, men blir beregnet på nytt hver gang den brukes

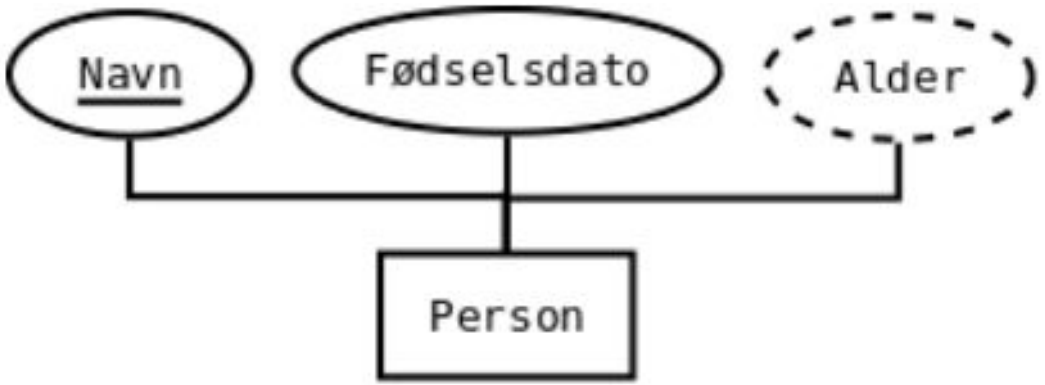
EKSEMPEL

>> CREATE VIEW <navn på view> (<kolonner>) AS
<spørring>

```
59    CREATE VIEW StudentTakesCourse ( StdName text , CourseName text )
60    AS
61    SELECT S.StdName , C. CourseName
62    FROM Students AS S,
63    Courses AS C,
64    TakesCourse AS T
65    WHERE S.SID = T.SID AND C.CID = T.CID
```

EXTRACT

- Views for utledbare verdier (fra ER)




```
67    CREATE VIEW person_alder AS
68    SELECT navn ,
69           fødselsdato ,
70           EXTRACT(year FROM age(current_date ,fødselsdato )) AS alder
71    FROM person
```

Materialiserte Views

- Dersom et view brukes veldig ofte kan det lønne seg å materialisere det
- Et materialisert view lagres som en vanlig tabell på disk
- De er derfor like effektive å kjøre spørringer mot som en vanlig tabell

```
>> CREATE MATERIALIZED VIEW <navn> AS (spørring)
```

- Men, den kan enkelt oppdateres når de tabellene den avhenger av oppdateres
- Dette skjer derimot ikke automatisk, man må kjøre følgende for å oppdatere det:

```
>> REFRESH MATERIALIZED VIEW person_alder;
```

SQL-scripts

- Inneholder SQL-kommandoer som lager skjemaene, tabellene og views, osv.
- Dersom man forsøker å opprette en tabell som allerede finnes eller slette en tabell som ikke finnes så feiler kommandoen
- Dette kan hindres ved å bruke IF EXISTS og IF NOT EXISTS i kommandone

IF EXISTS og IF NOT EXISTS

```
>> CREATE TABLE IF NOT EXISTS persons(name text, born date); -- Lager ny tabell
>> CREATE TABLE IF NOT EXISTS persons(name text, born date); -- Gir ingen error/lykkes
>> CREATE TABLE persons(name text, born date); -- Gir ERROR og feiler
>> DROP TABLE IF EXISTS persons; -- Sletter tabellen
>> DROP TABLE IF EXISTS persons; -- Gir ingen error/lykkes
>> DROP TABLE persons; -- Gir error , og feiler
```

Transaksjoner

- Når man oppdaterer databasen og noe går galt underveis ønsker man ofte at ingen av oppdateringene skal ha skjedd
- Vi pakker derfor inn oppdateringer som skal utføres som en ”enhet” i transaksjoner
- Transaksjoner omsluttes av BEGIN og COMMIT slik:

```
>> BEGIN;
>> UPDATE balances SET balance = balance - 100 WHERE id = 1;
>> UPDATE balances SET balance = balance + 100 WHERE id = 2;
>> COMMIT;
```

Jobb med ukesoppgaver/Innlevering 2

- Innelvering 2 (Enkel SQL)
  innlevering2.pdf (uio.no)
- Frist for Innlevering 2 (Enkel SQL): 12.Oktober kl 23.59!
- Ukesoppgaver (uke 6: SQL: Datamanipulering)
  IN2090-ukesoppgaver: Uke 6 – Universitetet i Oslo (uio.no)