IN2090

Grunnleggende SQL

Fjerninnlogging på IFI-maskin

1. Åpne terminalen på pcen din
2. Skriv inn:

```
>> ssh -YC brukernavn@login.ifi.uio.no
```

Bytt ut "brukernavn" med ditt eget brukernavn på uio.

- Ved feilmelding:
  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  @ Warning: REMOTE HOST IDENTIFICATION HAS CHANGE CHANGED!@
  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

  Skrive dette i terminalen en gang:

  ```
  >> ssh-keygen –R login.ifi.uio.no
  ```

3. Passordet for å logge inn er det dere bruker når dere logger inn med feide.

Logge seg på databasene

- Når dere har fått logget dere på ifi-maskinene, så kan dere logge dere på de forskjellige databasene derfra. Det gjør man på følgende måte:
- (fyll inn deres eget brukernavn der hvor det står brukernavn)
- For å logge på filmdatabasen skriver man:

```
>> psql -h dbpg-ifi-kurs03 -U brukernavn -d fdb
```

- Northwind-databasen:

```
>> psql -h dbpg-ifi-kurs03 -U brukernavn -d northwind
```

- Deres private database:

```
>> psql -h dbpg-ifi-kurs03 -U brukernavn -d brukernavn
```

Kjøre .sql-fil I databasen

- For å kjøre en .sql fil i databasen, så må filen ligge på en ifi-maskin. Det kan dere enkelt gjøre ved å kopiere filen over fra deres egen pc med kommandoen:
  `>> scp filnavn.sql brukernavn@login.ifi.uio.no:</path/til/mappe>`

- MEN, da må man logge inn på databasen fra samme mappe man kopierte filen til.

- Mens denne kommandoen vil legge filen på home directory på ifi maskinen:

`>> scp filnavn.sql brukernavn@login.ifi.uio.no:~`

- Deretter kan man bare fjernlogge seg inn som vanlig, logge på databasen og kjøre filen med kommandoen:

`>> \i filnavn.sql`

Noen kommandoer i PostgreSQL

- logge ut fra PostgreSQL

```
>> \q
```

- hjelp om SQL

```
>> \h
```

- hjelp om postgreskommandoer

```
>> \?
```

- liste opp tabeller (relasjoner)

```
>> \d
```

- liste opp kolonner og skranker for tabellen

```
>> \d <tabellnavn>
```

- lese filen `<filnavn>` som input

```
>> \i <filnavn>
```

- åpner en teksteditor (f.eks.Vim eller Emacs) som man så kan skrive SQL i og som blir kjørt når man så la grer og går ut av editoren

```
>> \e
```

Spørringer

- På formen:

```
1   SELECT <attributter/kolonner> -- alle attributter som skal vises i svaret
2   FROM <tabeller>
3   WHERE <betingelse>; --velger(selekterer) det som oppfyller betingelsen
```

- Store bokstaver på SELECT, FROM og WHERE er ikke nødvendig, men det er fin "kodeskikk"
- Spørringer avsluttes med ;

EKSEMPLER

```sql
SELECT filmid
FROM film; -- gir oss bare filmid

SELECT filmid, title
FROM film; --gir oss filmid og title

SELECT *
FROM film; --gir oss alle attributtene/kolonnene

SELECT
DISTINCT; --fjerner duplikater
```

Mer SELECT

```
18  --Fra filmdatabasen
19  --Person(personid, firstname, lastname, gender)
20  SELECT concat(firstname, ' ', lastname) AS Name
21  FROM Person;
22
23  --Eller
24
25  SELECT firstname || ' ' || lastname AS Name
26  FROM Person;
27  --Slår sammen 2 kolonner til 1
28
29  --Kan også bruke matematiske operasjoner som +, -, * og / rett i select-delen.
30  --F.eks:
31  SELECT 2*2+4-1;
32  --Vil gi tallet 7.
```

SELECT med aggregering

- sum – summen
- avg – gjennomsnitt
- max – maksimum
- min – minimum
- count – teller alle rader (ikke null verdier)
- count(*) teller alle rader inkludert null verdier

EKSEMPLER

```
34    SELECT count(filmid) -- gir antall filmid i film-tabellen
35    FROM film;
36
37    SELECT max(prodyear) --gir deg den/de nyeste filmene i databasen (høyest tall)
38    FROM film;
```

WHERE – klausulen

- Fra tabellen film(filmid, title, prodyear), velg ut navn på alle filmene som er produsert i perioden 2006-2008
- Bruker where-betingelsen til å filtrere ut filmene i den perioden.
- WHERE-betingelser kan kombineres med "AND", "OR" og "NOT".

```
42  SELECT title
43  FROM film
44  WHERE prodyear > 2005 AND prodyear < 2009;
45
46  --eller
47
48  SELECT title
49  FROM film
50  WHERE prodyear >= 2006 AND prodyear <= 2008;
51
52  --eller
53
54  SELECT title
55  FROM film
56  WHERE prodyear BETWEEN 2006 AND 2008;
```

Søke i tekst

- For å søke i tekst bruker man LIKE og %.
- Hvis vi ser på tabellen film(filmid, title, prodyear) igjen, så vil:

```
60   SELECT title
61   FROM film
62   WHERE title LIKE 'Lord of the Rings'; --vil gi oss akkurat filmen(e) med den tittelen.
63
64   SELECT title
65   FROM film
66   WHERE title LIKE '%Lord of the Rings'; --tittelen slutter på "Lord of the Rings"
67
68   SELECT title
69   FROM film
70   WHERE title LIKE 'Lord of the Rings%'; --tittelen starter på "Lord of the Rings"
71
72   SELECT title
73   FROM film
74   WHERE title LIKE '%Lord of the Rings%'; --kommer tekst før og etter 'Lord of the Rings'
```

EKSEMPEL

- Vil ikke gi det samme svaret.
- På sist spørring kommer det et mellomrom etter Rings før det kommer noe mer, så filmen med navnet "Lord of the Rings: The Two Towers" ville ikke blitt med siden det kommer et ":" etter Rings

```
78    SELECT title
79    FROM film
80    WHERE title LIKE 'Lord of the Rings%';
81
82    --og
83
84    SELECT title
85    FROM film
86    WHERE title LIKE'Lord of the Rings %';
```

NULL-verdier

- NULL-verdier er ukjente verdier. Kan ikke bruke "=" eller "LIKE" for å finne disse. Man bruker: "IS NULL" og "IS NOT NULL".

```
94    SELECT *
95    FROM film
96    WHERE prodyear IS NULL
97    LIMIT 3;
```

- "LIMIT" begrenser output til her tre rader.

JOINS i SQL

- Bruker JOINS for å slå sammen tabeller. Nå bruker vi:

```
INNER JOIN
NATURAL JOIN
SELF JOIN
```

- Senere i kurset:

```
FULL OUTER JOIN
RIGHT OUTER JOIN
LEFT OUTER JOIN
```

Finne fremmednøkler

- Finner fremmed nøkler ved å skrive

```
>> \d <tabellnavn>
```

```
fdb=> \d film
                Table "public.film"
  Column  |  Type   | Collation | Nullable | Default
----------+---------+-----------+----------+---------
 filmid   | integer |           |          |
 title    | text    |           | not null |
 prodyear | integer |           |          |
Indexes:
    "filmpkey" UNIQUE CONSTRAINT, btree (filmid)
    "filmtitleindex" btree (title)
    "filmyearindex" btree (prodyear)
Foreign-key constraints:
    "filmfkey" FOREIGN KEY (filmid) REFERENCES filmitem(filmid)
```

INNER JOIN

- INNER JOIN joiner på en betingelse

```
101    SELECT * from film f
102    INNER JOIN filmitem fi ON f.filmid = fi.filmid
103    WHERE prodyear = 2011;
```

NATURAL JOIN

- Joiner på alle kolonner med likt navn

```
107    SELECT *
108    FROM film NATURAL JOIN filmitem
109    WHERE prodyear = 2011 AND filmtype = 'C';
```

- Funker kun fordi film har fremmednøkkel:

```
Foreign-key constraints:
    "filmfkey" FOREIGN KEY (filmid) REFERENCES filmitem(filmid)
```

NATURAL JOIN

- NATURAL JOIN mellom series-tabellen og filmitem-tabellen, hadde ikke fungert fordi series(seriesid) -> filmitem(filmid)

```
fdb=> \d series
                 Table "public.series"
    Column     |  Type   | Collation | Nullable | Default
---------------+---------+-----------+----------+---------
 seriesid      | integer |           |          |
 maintitle     | text    |           | not null |
 firstprodyear | integer |           |          |
Indexes:
    "seriesmaintitleindex" btree (maintitle)
    "seriespkey" UNIQUE CONSTRAINT, btree (seriesid)
Foreign-key constraints:
    "seriesfkey" FOREIGN KEY (seriesid) REFERENCES filmitem(filmid)
Referenced by:
    TABLE "episode" CONSTRAINT "episodefkeyseriesidseries" FOREIGN KEY (seriesid) REFERENCES series(seriesid)
```

Nestede spørringer

- Husk at tingene i en FROM-klausul er tabeller
- Husk også at resultatet av en SELECT-spørring er en tabell
- Så, vi kan putte en SELECT-spørring i FROM-klausulen

```
111    SELECT <kolonner>
112    FROM(SELECT <kolonner>
113         FROM <tabell>
114         WHERE <betingelse>) AS tab
115    WHERE <betingelse>;
```

EKSEMPEL

- For å finne antall unike kombinasjoner av land og by for alle kunder:

```
117    SELECT count(*)
118    FROM (SELECT DISTINCT country , city
119          FROM customers ) AS d;
```

- Finne navnet på alle produkter med en “supplier” fra Tyskland:

```
122    SELECT product_name
123    FROM products
124    WHERE supplier_id IN (SELECT supplier_id
125                          FROM suppliers
126                          WHERE  country = 'Germany ')
```

Jobb med ukesoppgaver/ Innlevering 2

- Innlevering 2 (Enkel SQL): innlevering2.pdf (uio.no)
- Frist for Innlevering 2 (Enkel SQL): 12.Oktober kl 23.59!
- Ukesoppgaver (uke 5: SQL: Grunnleggende SQL)
  IN2090-ukesoppgaver: Uke 5 – Universitetet i Oslo (uio.no)