Økt 5 (av 12)

DB1102 Databaser

Per Lauvås / per.lauvas@kristiania.no

Yuan Lin / yuan.lin@kristiania.no

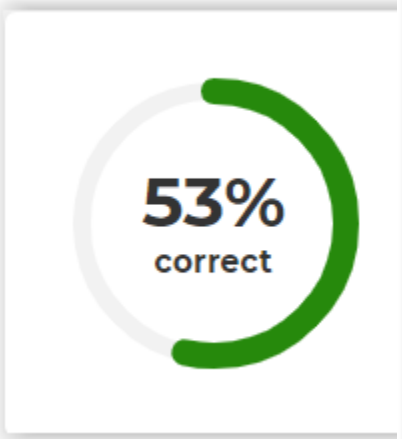
Dagens temaer

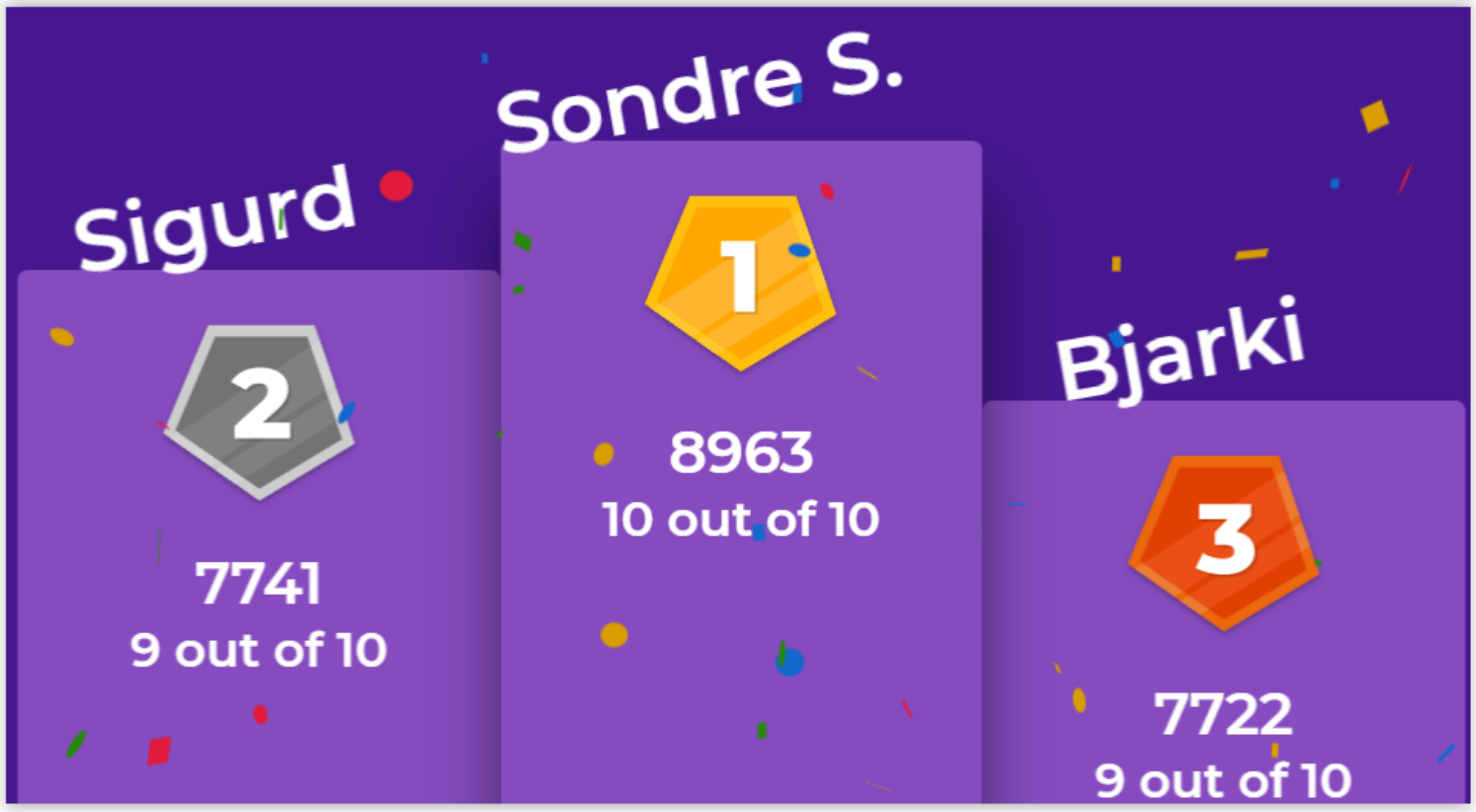
Dagens tema: Avanserte spørringer.

- Dagens pensum: Læreboka, kapittel 5.1-5.3.

- Nytt innhold: Avanserte spørringer
  - Ny SQL i denne sammenheng: `VIEW` og subqueries, LIMIT, COALESCE, IN mm.

Kahoot


53%
correct


Sondre S.
1
8963
10 out of 10
Sigurd
2
7741
9 out of 10
Bjarki
3
7722
9 out of 10

Hva med kapittel 6?

- En liten oppdatering i fremdriftsplanen
- Kapittel 6 i økt 6 (fremdriftsplanen er oppdatert).

Fra gårsdagens Zoom

- Jeg forsnakker meg av og til…
  - 5:54 og 5:59 Her mente jeg fremmednøkler (og ikke primærnøkler).
- I oppgave 4 ble det krøll, og nå vet jeg hvorfor…

View

Likner veldig på en tabell

Hva er et View?

- Et View er en forhåndslaget spørring fra en eller flere tabeller. Eksempel:

SNR	NAVN	GATE	STED	STILLING	MLOENN	ANSDATO	FNR
3	JON	BRUVEIEN 7	STAVANGER	LEDER	35000	08-SEP-95	1
2	MARIE	STRILEGATEN 8	BERGEN	LEDER	30000	01-JAN-95	2
1	SUSANNE	SKRAPLODDVEIEN 62	OSLO	LEDER	45000	01-JUL-94	3
20	OLAV	GALMANNSVEIEN 4	STAVANGER	SENIORMEGLER	26000	07-JUL-97	1
5	DAVID	GULERLEVEIEN 43	STAVANGER	SEKRETÆR	18000	14-JUN-95	1
4	ANNE	STRANDGATEN 5	[illegible]VANGER	MEGLER	12000	12-DEC-96	1
8	GUSTAV	NORDLYSVEIEN 78				01-JAN-96	1
9	OLAVA	LOMVIVEIEN 57				01-JAN-98	1
11	LEONORA	RØDVEIEN 6	OSL[illegible]			01-JUL-94	3
10	TEODOR	TULIPAN 12	OSL[illegible]			30-MAY-97	3
6	JONNAS	KIRKEVEIEN 7	BÆRU[illegible]			19-APR-96	3
12	TULLA	BLÅVEIEN 7a	OSL[illegible]			09-SEP-95	3
18	FREDRIK	LASSOVEIEN 37	OSL[illegible]			01-JUL-94	3
7	KARL	OLAVSGATE 7	OSL[illegible]			10-SEP-95	3
16	SMUKKA	GRAUTSTIEN 43	OSL[illegible]			01-JUL-94	3
17	KARL	BLÅVEISVEIEN 7	OSLO	MEGLER	20500	12-MAY-96	3

SNR	NAVN	GATE
3	JON	BRUVEIEN 7
20	OLAV	GALMANNSVEIEN 4
5	DAVID	GULERLEVEIEN 43
4	ANNE	STRANDGATEN 5
8	GUSTAV	NORDLYSVEIEN 78
9	OLAVA	LOMVIVEIEN 57

Hvorfor bruke View?

- Sikkerhet: Begrense datatilgangen i databasen.

- Redusere kompleksitet når man senere skal skrive SQL spørringer:
  - Gjøre komplekse spørringer enklere.
  - Redusere datamengden for bruker.

- Tilpasning: Oppnå forskjellige syn (views) av datagrunnlaget. (Brukergrupper/applikasjoner)

Ulemper med View

- Mer kompleksitet knyttet til oppdateringer og endringer:
  - Det er begrensninger på hvordan man kan endre underliggende data gjennom et view.
  - Et views struktur blir bestemt når det opprettes, og vil ikke automatisk endre seg senere (VIEW basert på SELECT * FROM...)

- Ytelse:
  - Et view kan joine mange tabeller, og dermed være en relativt tung spørring.
  - Dette kommer ikke alltid tydelig frem for brukeren.

Definere et View

```
CREATE [OR REPLACE] VIEW navn
AS
subquery;
```

OR REPLACE	Overskriver view om det eksisterer
subquery	En fullstendig SELECT setning

Eksempel 1: world databasen

```
CREATE OR REPLACE VIEW EuropeCountry_view
AS
SELECT Code, Name, Population
FROM country
WHERE Continent = 'Europe';
```

```
SELECT *
FROM EuropeCountry_view;
```

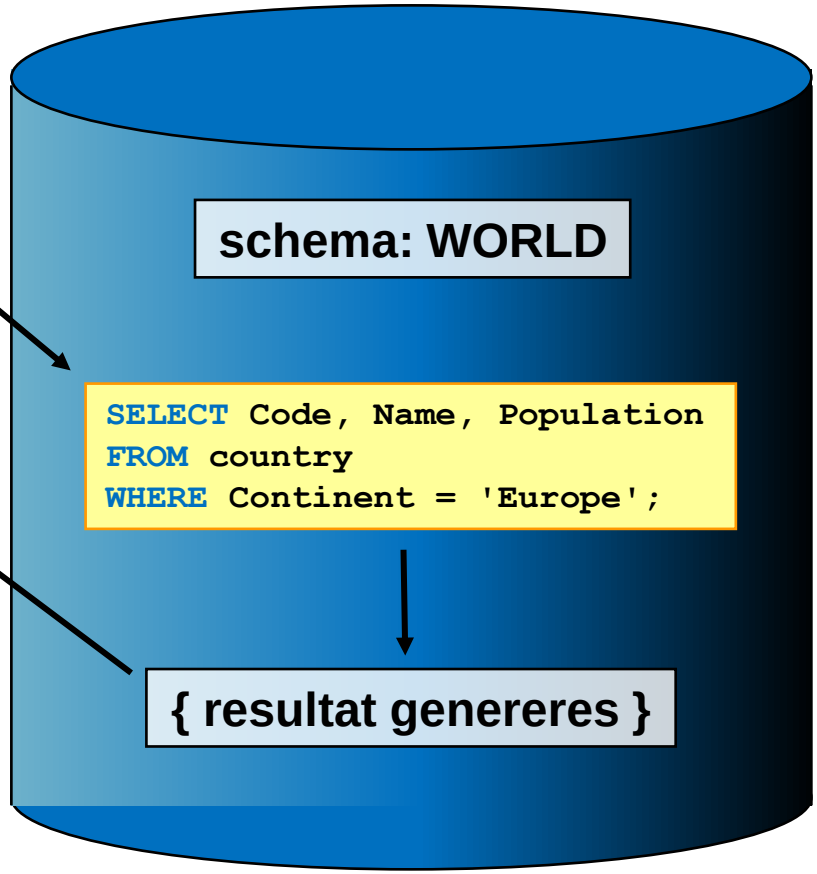
Code	Name	Population
ALB	Albania	3401200
AND	Andorra	78000
AUT	Austria	8091800
BEL	Belgium	10239000
BGR	Bulgaria	8190900
BIH	Bosnia and Herzegovina	3972000
BLR	Belarus	10236000
CHE	Switzerland	7160400
CZE	Czech Republic	10278100
DEU	Germany	82164700
DNK	Denmark	5330000
ESP	Spain	39441700
...		

- Et view behandles på samme måte som en tabell.

Slik fungerer spørring mot view

```
SELECT *
FROM EuropeCountry_view;
```

```
Code  Name            Population
----  ------------    ----------
ALB   Albania         3401200
AND   Andorra         78000
AUT   Austria         8091800
BEL   Belgium         10239000
BGR   Bulgaria        8190900
...
```



- View definisjonen (ikke resultatet!) er lagret i databasen.

Eksempel 1: noen varianter

- Vi kan gi view'et egne kolonnenavn:

```
CREATE OR REPLACE VIEW EuropeCountry_view AS
SELECT Code AS ID, Name AS Country, Population
FROM country
WHERE Continent = 'Europe';
```

```
CREATE OR REPLACE VIEW EuropeCountry_view
(ID, Country, Population)
AS
SELECT Code, Name, Population
FROM country
WHERE Continent = 'Europe';
```

```
SELECT *
FROM EuropeCountry_view;
```

ID	Country	Population
ALB	Albania	3401200
AND	Andorra	78000
AUT	Austria	8091800
BEL	Belgium	10239000
BGR	Bulgaria	8190900
BIH	Bosnia and Herzegovina	3972000
BLR	Belarus	10236000
CHE	Switzerland	7160400
CZE	Czech Republic	10278100
DEU	Germany	82164700
DNK	Denmark	5330000
ESP	Spain	39441700
...		

Summering av kolonner i view

```
CREATE OR REPLACE VIEW ContinentPopulation_view AS
SELECT Continent, SUM(Population) AS Population
FROM country
GROUP BY Continent
ORDER BY Population ASC;
```

```
SELECT *
FROM ContinentPopulation_view;
```

Continent	Population
Antarctica	0
Oceania	30401150
South America	345780000
North America	482993000
Europe	730074600
Africa	784475000
Asia	3705025700

- Gruppefunksjoner kan inngå i et view.
  - (Det er en fordel å bruke AS/alias for å tilpasse kolonnenavnene.)

Oppgave

- Lag et view som viser de byene i verden som har en befolkning større enn 5 millioner. Viewet skal vise bynavn, befolkning og hvilket land byen ligger i (navnet på landet), og være sortert synkende på innbyggertall.

```
CREATE OR REPLACE VIEW largeCities
(city, population, country)
AS
SELECT ci.name, ci.population, co.name
FROM city ci LEFT JOIN country co ON ci.countryCode = co.Code
WHERE ci.Population > 5000000
ORDER BY population DESC;
```

city	population	country
Mumbai (Bombay)	10500000	India
Seoul	9981619	South Korea
SÃ£o Paulo	9968485	Brazil
Shanghai	9696300	China
Jakarta	9604900	Indonesia
Karachi	9269265	Pakistan
Istanbul	8787958	Turkey

Hente ut de 7 største:

```
SELECT * FROM largeCities LIMIT 7;
```

Hvorfor LEFT JOIN? Hmmm…

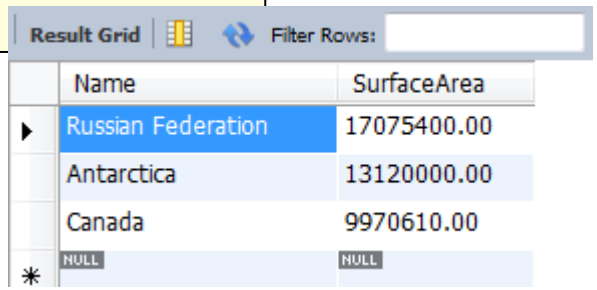
Nytt SQL ord: LIMIT

- LIMIT begrenser antall rader i resultatet ditt.

- Eksempel:

  Jeg vil finne de tre største landene i verden.

```
SELECT Name, SurfaceArea
FROM Country
ORDER BY SurfaceArea DESC
LIMIT 3;
```



Tips ved bruk av CREATE VIEW

Samme tips ved opprettelse av VIEW som for endringer på data basert på spørringer: (UPDATE og DELETE FROM)

- Kjør alltid SELECT først, så vet du om viewet fungerer *før* du oppretter det!

```
CREATE OR REPLACE VIEW largeCities
(city, population, country)
AS SELECT ci.name, ci.population, co.name
FROM city ci LEFT JOIN country co ON ci.countryCode = co.Code
WHERE ci.Population>5000000
ORDER BY population DESC;
```

Oppdatering via view

Som nevnt kan view benyttes til å oppdatere data i underliggende tabell(er).

- NB: ISO restriksjoner på hvordan et view er laget m.t.p. oppdateringer. Bl.a.:
  - View'et kan bare referere til én tabell.
  - `DISTINCT` kan ikke være del av view'et.
  - Alle elementer i view'ets select del må være kolonner (ikke konstanter, summeringer, etc. ...)
  - Ingen `GROUP BY` eller `HAVING`.
  - Rad som blir lagt til må følge integritetsreglene for underliggende tabell (not null, etc.).

Oppdatering via view – forts.

- Merk at et view oppdaterer dataene i tabellen!
  (Ikke bare data for view'et selv.)

```
UPDATE EuropeCountry_view
SET ID = 'A_Z'
WHERE Country = 'Austria';
```

```
SELECT ID, Country
FROM EuropeCountry_view;
```

ID	Country
ALB	Albania
AND	Andorra
A_Z	Austria
BEL	Belgium
BGR	Bulgaria
...	

```
SELECT Code, Name, Population
FROM country
WHERE Name = 'Austria';
```

CODE	Name	Population
A_Z	Austria	8091800

Definere et View – del 2.

```
CREATE [OR REPLACE] VIEW navn
[(alias [,alias ] . . .)]
AS
subquery
[WITH CHECK OPTION]
```

- WITH CHECK OPTION
  - Spesifiserer at rader som et view kan aksessere ikke kan endres gjennom dette view'et om det resulterer i at de forsvinner fra view'et.

View med check *option*

- Check option for view gjør at vi *ikke* kan oppdatere en rad slik at den forsvinner ut av viewet:

```
CREATE OR REPLACE VIEW EuropeCountry_view AS
SELECT Code, Name, Continent
FROM country
WHERE Continent = 'Europe'
WITH CHECK OPTION;
```

```
SELECT *
FROM EuropeCountry_view;
```

Code	Name	Continent
ALB	Albania	Europe
AND	Andorra	Europe
A_Z	Austria	Europe
BEL	Belgium	Europe
BGR	Bulgaria	Europe
...		

```
UPDATE EuropeCountry_view
SET Continent = 'Asia'
WHERE Name = 'Austria';
```

```
Error Code: 1369. CHECK OPTION failed
'world.europecountry_view'
```

Slette et view

- Syntaks for å slette et view er nesten som for tabell:
  (bytt ut table med view)

```
DROP VIEW EuropeCountry_view;
```

Delspørringer (subqueries)

Spørringer inne i spørringer

(eller spørringer inne i VIEW, UPDATE, DELETE FROM…)

Subqueries

- Som nevnt, er resultatet av en SELECT formatert som en ny tabell:
  - Det danner kolonner og rader på samme måte som databasens eksisterende tabeller.

- Derfor er det ikke noe problem å bruke resultatet av en SELECT som et element i en annen!

- Å putte en SELECT inne i en annen kalles en subquery.
  - Eller i et VIEW, en UPDATE…

Subqueries II

- Noen ganger trenger vi svaret fra den ene spørringen før vi kan begynne på den andre.

- Eks.: Ønsker å finne hvor mange byer som har et innbyggertall over eller likt gjennomsnittet.
  - Før vi kan fullføre denne spørringen, må vi vite hva gjennomsnittet er!

Subqueries III

- Oppgave: «hvor mange byer har et innbyggertall over eller likt gjennomsnittet.»
  - Løser det med en subquery, på denne måten:

```
select count(*)
from City
where Population >=
(select avg(Population) from City);
```

- Eller hva med: "Ønsker å se prosentandelen som innbyggerne i et land utgjør av jordas totale befolkning"? (<-Vanskelig)
  - Denne er blant ukens øvingsoppgaver! :-)

- Husk å lese pensumbokas beskrivelse av delspørringer (subqueries, kap. 5.3). Der vil du finne mange flere eksempler på bruk.

Noen andre temaer fra pensumboka

- CASE
- IF
- COALESCE
- IFNULL
- IN (SOME, ANY, ALL, EXISTS)
- DISTINCT

- La meg demonstrere…