



Økt 4 (av 12)

DB1102 Databaser

Per Lauvås / per.lauvas@kristiania.no

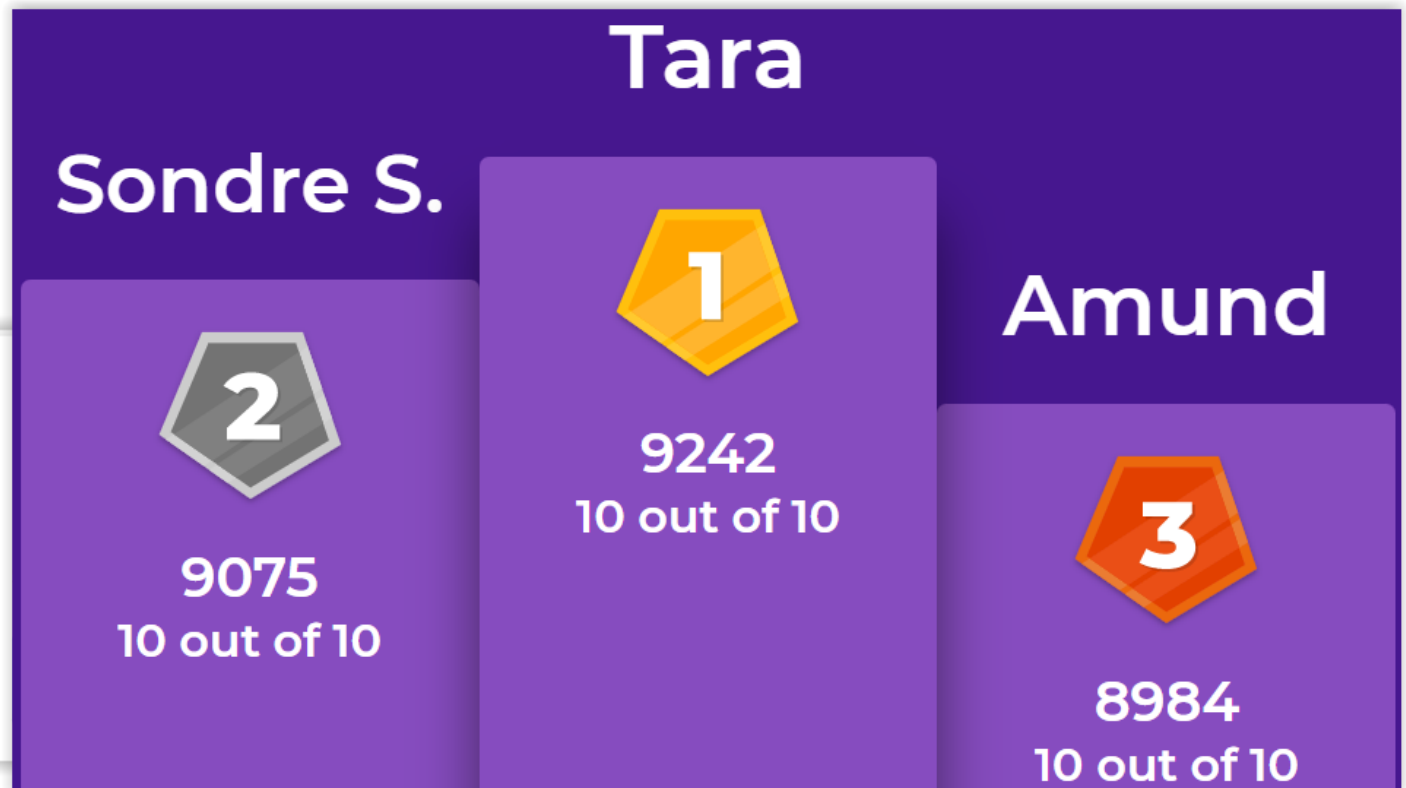
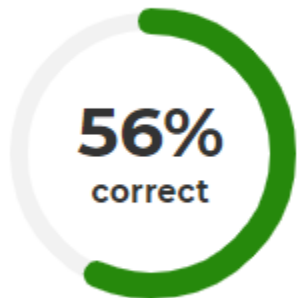
Yuan Lin / yuan.lin@kristiania.no

Dagens temaer

Dagens tema: [Spørringer mot flere tabeller \(JOIN\)](#).

- Dagens pensum: [Læreboka, kapittel 4](#).
- Nytt innhold: Spørringer mot flere tabeller
 - Ny SQL i denne sammenheng: JOIN, LEFT JOIN, RIGHT JOIN

Kahoot



Lærdom fra Kahoot

Sp.2: Hva slags datatype er en ENUM?



En nummererings-datatype



En datatype for ekstra lange nummer.



En datatype med et predefinert verdsett.



En løpende teller



No answer

country

- Columns
 - Code
 - Name
 - Continent**
 - Region
 - SurfaceArea
 - IndepYear
 - Population
 - LifeExpectancy
 - GNP
 - GNPOld
 - LocalName
 - GovernmentForm
 - HeadOfState

Administration Schemas

Information

Column: Continent

Collation: utf8mb4_0900_ai_ci

Definition:

enum('Asia','Europe','North America','Africa','Oceania','Antarctica','South America')

Lærdom fra Kahoot

Sp.9: Hva gjør: DELETE FROM world.Country; ?

	Sletter alle rader i world.Country.	✓ 	70
	Sletter alle kolonner i world.Country.	✗ 	54
	Sletter hele tabellen world.Country.	✗ 	53
	Sletter hele world-databasen.	✗ 	9
	No answer	✗ 	20

Husk at det er viktig å huske hva de ulike reservede ordene betyr:

- *DELETE FROM* vs. *DROP TABLE*
- *UPDATE* vs. *ALTER TABLE*
- *INSERT INTO* vs. *CREATE TABLE*

Kartesisk produkt

- Kartesisk produkt operasjonen gir som output mengden som kombinerer hver eneste rad ("tuple") i tabell R med hver eneste rad i tabell S.

– Formel: $R \times S$

R

a
b

S

1
2

$R \times S$

a	1
a	2
b	1
b	2

- Hva blir det kartesiske produktet ($R \times S$)?

- Kartesisk produkt: Vi plusser sammen kolonner, ganger sammen rader.

Oppgave!

- Hvordan ser det kartesiske produktet ut når vi har disse to tabellene?

Eier_id	Navn
1	Per Persen
2	Ola Olsen
3	Kari Normann

Bileier

Regnr	Modell	Eier_id
KH22222	Skoda	1
KH33333	Ferrari	NULL
DE22222	Volvo	2

Bil

SQL: Søk over flere tabeller

- Hva om vi vil hente ut data fra flere tabeller?

Eier_id	Navn
1	Per Persen
2	Ola Olsen
3	Kari Normann

Bileier

Regnr	Modell	Eier_id
KH22222	Skoda	1
KH33333	Ferrari	NULL
DE22222	Volvo	2

Bil

- Eks: "Jeg vil hente ut bileiers navn + bileiers registrerte biler" (registreringsnummer og modell).

Kartesisk produkt og SQL

- **IKKE** ok oppsett: Får for mange radkombinasjoner returnert!

```
SELECT * FROM bileier, bil;
```



- (Det **kartesiske produktet**, som dere kjenner til☺)

Result Grid					
Filter Rows: <input type="text"/>					
Export: 					
	eier_id	navn	regnr	modell	eier_id
▶	1	Per Persen	DE22222	Volvo	2
	2	Ola Olsen	DE22222	Volvo	2
	3	Kari Normann	DE22222	Volvo	2
	1	Per Persen	KH22222	Skoda	1
	2	Ola Olsen	KH22222	Skoda	1
	3	Kari Normann	KH22222	Skoda	1
	1	Per Persen	KH33333	Ferrari	NULL
	2	Ola Olsen	KH33333	Ferrari	NULL
	3	Kari Normann	KH33333	Ferrari	NULL

Natural join

- Eksempel:

```
SELECT * FROM bileier NATURAL JOIN bil;
```

Result Grid   Filter Rows: <input type="text"/>				
	eier_id	navn	regnr	modell
▶	2	Ola Olsen	DE22222	Volvo
	1	Per Persen	KH22222	Skoda

- Naturlig å slå sammen tabellene basert på felles kolonnenavn (eier_id). Kolonnene må også ha samme datatype!
 - Vi henter kun de radene som har felles eier_id.
 - Vi trenger ikke å ha med begge eier_id-kolonnene: De har jo samme innhold.
 - SQL syntaks for dette: **NATURAL JOIN**

Natural join – forts.

- **NATURAL JOIN** gir oss alle kolonner, men kun for de relevante, **sammenhengende radene fra begge tabeller**.
 - Vi kan f.eks. hente ut "bileiers navn, reg.nummer og modell":
(dataene er fordelt på bileier- og bil-tabellene)


```
SELECT navn, regnr, modell  
FROM bileier NATURAL JOIN bil;
```

Result Grid			
Filter Rows:			
	navn	regnr	modell
▶	Ola Olsen	DE22222	Volvo
	Per Persen	KH22222	Skoda

Natural left join

- Men hva om vi også vil ha med personer som ikke eier en bil?
 - Da benytter vi **NATURAL LEFT JOIN**: (med bileier som venstre tabell)

```
SELECT * FROM bileier NATURAL LEFT JOIN bil;
```

Result Grid  Filter Rows: <input type="text"/> Export				
	eier_id	navn	regnr	modell
▶	2	Ola Olsen	DE22222	Volvo
	1	Per Persen	KH22222	Skoda
	3	Kari Normann	NULL	NULL

- Vi tar **utgangspunkt i venstre tabell** (*bileier*) og tar med relevant data fra **høyre tabell** (*bil*) **der det finnes en kopling** mellom tabellene.
 - Med **LEFT JOIN** får vi med Kari Normann, selv om hun ikke eier en bil.

Natural right join

- Hva om vi heller vil ha med alle biler, også de uten eier?
 - Da benytter vi **NATURAL RIGHT JOIN**: (med bileier som venstre tabell)

```
SELECT * FROM bileier NATURAL RIGHT JOIN bil;
```

Result Grid				
Filter Rows: <input type="text"/>				
	eier_id	regnr	modell	navn
▶	1	KH22222	Skoda	Per Persen
	2	DE22222	Volvo	Ola Olsen
	NULL	KH33333	Ferrari	NULL

- Vi tar **utgangspunkt i høyre tabell** (bil) og inkluderer data fra venstre tabell (bileier) der det finnes en kopling for disse.
 - Med **RIGHT JOIN** får vi med KH33333 Ferrari, selv om den ikke har noen eier.

"Unaturlig" join

- På foregående slides hadde kolonnene vi matchet samme navn (eier_id).
- Det vanligste er at vi slår sammen tabeller basert på kolonner som **IKKE har samme navn**. Da kan vi ikke benytte *NATURAL JOIN*.
 - (Vanligvis er disse primærnøkkel i den ene tabellen og fremmednøkkel i den andre tabellen, og vanligvis tilsier dette litt forskjellige kolonnenavn.)
- Når vi slår sammen tabeller basert på kolonner som *ikke* har samme navn, må vi beskrive hvilke kolonner vi vil benytte til å joine tabellene.
 - Dette vil også gjøre det lettere å lese SQL-en.
 - (Eksempler på kommende slides.)

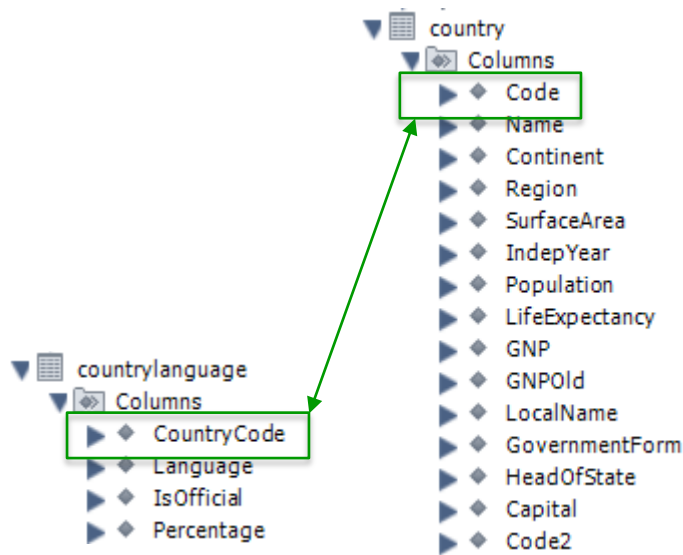
SQL: Søk over flere tabeller

- World-databasen: Vi ønsker å finne hvilke land som har spansk som offisielt språk.
- Fra før av har vi kunnskap nok til å gjøre dette mot countrylanguage tabellen.
 - Kan da imidlertid ikke få ut navnet på landet, bare landkodene:

```
SELECT CountryCode, Percentage
FROM countrylanguage
WHERE Language = 'spanish' AND IsOfficial = 'T'
ORDER BY CountryCode;
```

Join

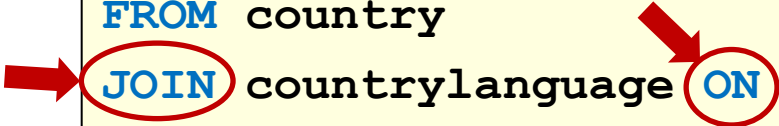
- Hva om vi ønsker å vise både navnet på landet (Country-tabellen) og prosentandelen i landet som snakker spansk (CountryLanguage-tabellen) samtidig?
- Vi kan bruke join, men vi har ikke matchende kolonnenavn:



Join – forts.

- Vi benytter **ON** for å angi kolonnenavnene. Eksempel:

```
SELECT Name, Percentage
FROM country
JOIN countrylanguage ON Code = CountryCode
WHERE Language = 'spanish' AND IsOfficial = 'T'
ORDER BY Name;
```



- Denne type **join** kalles også **inner join** ("likekobling" i læreboka).

Left Join og Right Join

- (INNER) JOIN tar bare med resultater som har data i alle involverte tabeller.
- Det finnes andre varianter av join også:
 - LEFT JOIN og RIGHT JOIN (felles kalt *outer join*).
 - Boka omtaler disse som "venstre/høyre ytre kobling".
- Som for natural join variantene av left og right, er dette koplinger som tar med rader som ikke matches i alle involverte tabeller.

Left Join og Right Join – forts.


- Eks.: Vi ønsker å hente ut landene som begynner på 'An', og byene i dem.

```
select country.Name, city.Name
from Country
join City on Code = CountryCode
where country.Name like 'An%'
order by country.Name;
```

- Men: *Henter denne alle land på 'An%'?*
 - Tja, time will show!
 - Eller neste slide da ... ;-)

Left Join og Right Join – forts.

- Antarctica mangler, fordi det ikke inneholder noen byer!
 - For å få med Antarctica må vi gjøre en LEFT JOIN:



```
SELECT country.Name, city.Name
FROM Country
LEFT JOIN City ON Code = CountryCode
WHERE country.Name like 'An%'
ORDER BY country.Name;
```

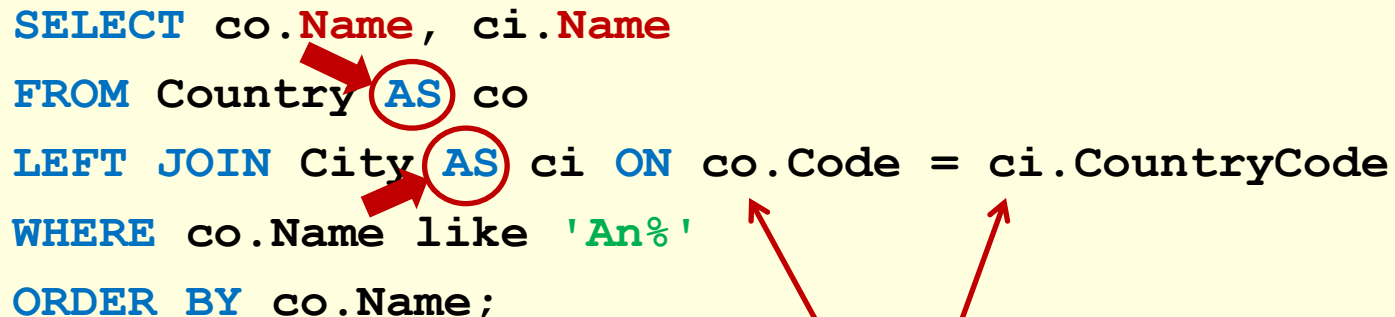
Left Join og Right Join – forts.

- En LEFT JOIN tar med rader fra den første/venstre tabellen (Country) som ikke finnes i den siste/høyre tabellen (City).
- Tilsvarende har vi RIGHT JOIN (eller "RIGHT OUTER JOIN") som tar med rader fra den siste/høyre tabellen som ikke finnes i den første/venstre.
- Dette er tilsvarende det vi så med biler og personer i natural join-eksemplene.

Noen tips for Join

- Vi må bruke tabell-prefix for å skille identiske kolonnenavn i flere tabeller fra hverandre.
 - *MERK!* Det er lov å bruke alias også her:

```
SELECT co.Name, ci.Name
FROM Country AS co
LEFT JOIN City AS ci ON co.Code = ci.CountryCode
WHERE co.Name like 'An%'
ORDER BY co.Name;
```



- Det er ikke nødvendig med tabell-prefix (eks: `co.Code`) dersom kolonnen kun er definert i én av tabellene. Men du vil oppnå bedre ytelse med det.

Join – oversikt

- Disse skal dere kunne:

Vi skriver	Kan også skrives
JOIN	INNER JOIN
LEFT JOIN	LEFT OUTER JOIN
RIGHT JOIN	RIGHT OUTER JOIN

- Hvor ble det av NATURAL JOIN?
 - Du kan alltid bruke (LEFT/RIGHT) JOIN ... ON isteden for NATURAL (LEFT/RIGHT) JOIN.
 - NATURAL JOIN er ikke mye i bruk, og vi vil ikke bruke den videre i DB1102.
 - NATURAL JOIN er kun med her som en enkel introduksjon til JOIN.

Andre typer Join

- *NB:* Det finnes andre typer outer join og, som FULL OUTER JOIN.
 - Men left/right er de vanligste, og de som er viktige (pensum) i DB1102.
- FULL OUTER JOIN støttes heller ikke av MySQL.

Join / natural join

```
SELECT * FROM bileier NATURAL JOIN bil;
```

Result Grid				
	eier_id	navn	regnr	modell
▶	2	Ola Olsen	DE22222	Volvo
	1	Per Persen	KH22222	Skoda

```
SELECT * FROM bileier JOIN bil ON bileier.eier_id = bil.eier_id;
```

Result Grid					
	eier_id	navn	regnr	modell	eier_id
▶	2	Ola Olsen	DE22222	Volvo	2
	1	Per Persen	KH22222	Skoda	1

Join / kartesisk produkt med betingelse

- Det er veldig vanlig å hente ut data fra flere tabeller.
 - Derfor har vi også en egen syntaks for å gjøre dette: JOIN
- Vi kan skrive det uten bruk av ordet JOIN om vi foretrekker det.
 - Her er to eksempler som gir samme resultat, men kun én av dem bruker JOIN:

```
SELECT ci.Name, ci.Population, co.name
FROM Country AS co, City AS ci
WHERE co.Code = ci.CountryCode
      AND co.continent >= 'Europe';
```

```
SELECT ci.Name, ci.Population, co.name
FROM Country AS co JOIN City AS ci
ON co.Code = ci.CountryCode
WHERE co.continent >= 'Europe';
```

Hvorfor JOIN?

- I stedet for JOIN kunne vi lagret data dobbelt opp, i flere tabeller.
- Dette kalles **redundans**.
- Mange grunner til å unngå dette – detaljene kommer med modellering og normalisering, senere.

Neste gang

- Kapittel 5: Avanserte spørringer.