



Økt 6 (av 12)

DB1102 Databaser

Per Lauvås / per.lauvas@kristiania.no

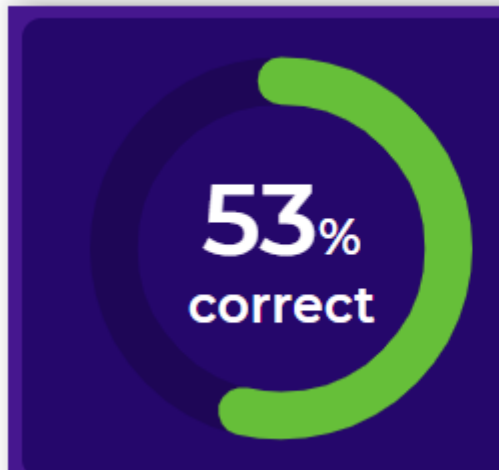
Yuan Lin / yuan.lin@kristiania.no

Ukens temaer

Dagens tema: [Relasjonsmodellen](#).

- Dagens pensum: [Læreboka](#), kapittel 6.
- Relasjonsalgebra
- Arbeidskrav
- I øvingstimene: 2 timer multiple choice, *for egentesting*.

Kahoot



Spørsmål 4

4 -Quiz Hvilken forutsetning er korrekt for oppdatering av underliggende data via views.

1 of 3



Viewet kan referere til flere tabeller



39



DISTINCT kan være en del av viewet



14



GROUP BY eller HAVING kan benyttes i viewet



22



Alle elementer i view'ets select-del må være kolonner



36



No answer



5

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.).

Spørsmål 7

7 -Quiz Hva er et materialized view?

2 of 3 < >



Et view som omhandler materialer

✗ ●

10



Et view som kun er en lagret spørring

✗ —

32



En tabell laget på bakgrunn av en spørring

✓ —

36



Den teknologien som et DBMS benytter for å lage et view

✗ —

26



No answer

✗ ●

12

Materialiserte view

Pensumboka, s. 122:

- Et materialisert view (materialized view) er et view der selve spørreresultatet blir fysisk lagret i databasen. DBHS sørger automatisk for å holde innholdet i slike views oppdaterte. (Støttes ikke i MySQL)

Spørsmål 8

8 -Quiz Resultat? `SELECT * FROM city WHERE ID IN(SELECT capital FROM country);`

3 of 3



Informasjon om alle hovedsteder.



38



Informasjon om alle byer som har en ID.



18



Informasjon om alle land som har en hovedstad.



41



Informasjon om alle kapitalistiske stater.



1

2



No answer



17

Litt repetisjon av join

- Takk for deling!

<https://joins.spathon.com/>

Relasjonsalgebra

Relasjonsalgebra

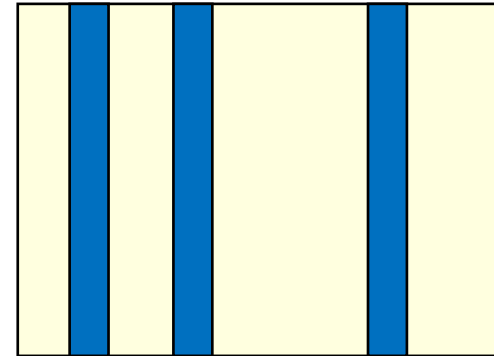
- Relasjonsalgebra er et teoretisk språk.
 - Definert av Codd.
 - SQL er basert på dette.
- Fra emnebeskrivelsen, læringsutbytte:
 - Etter å ha fullført emnet skal studenten kunne:
 - ... beskrive hva relasjonsalgebra er, og forklare begreper som **kartesisk produkt**.

Relasjonsalgebra – forts.

- Relasjonsalgebra (og dermed også SQL) har noen viktige prinsipper:
- Resultatet skal dannes uten å endre kildene (SELECT).
- Resultatet skal følge samme format som kildene.
 - Resultatet fra én operasjon kan dermed være kilden til en annen! (Ref. subqueries og views.)
- Resultatet av en operasjon omtales gjerne som en mengde.

Seleksjon og projeksjon

- En projeksjon av tabellen (relasjonen) R er en mengde bestående av kolonner fra R .



- En seleksjon av tabellen R er en mengde bestående av rader fra R .



Kartesisk produkt (repetisjon)

- Kartesisk produkt har vi snakket om tidligere.
 - Se også læreboka, side 97 og 155.
- **Kartesisk produkt** operasjonen gir som output mengden som kombinerer hver eneste rad ("tuple") i tabell R med hver eneste rad i tabell S.
 - Formel: $R \times S$
 - Huskeregel: "*Plusse kolonner, gange rader.*"

R	
a	
b	

S	
1	
2	

R x S	
a	1
a	2
b	1
b	2

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

Jeg vil hente ut bileiers navn + bileiers registrerte biler (registreringsnummer og modell).

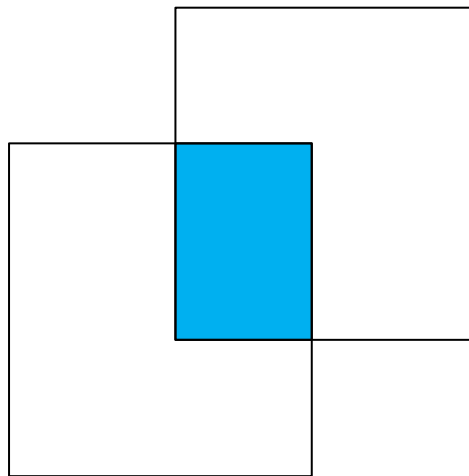
Kartesisk produkt

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

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

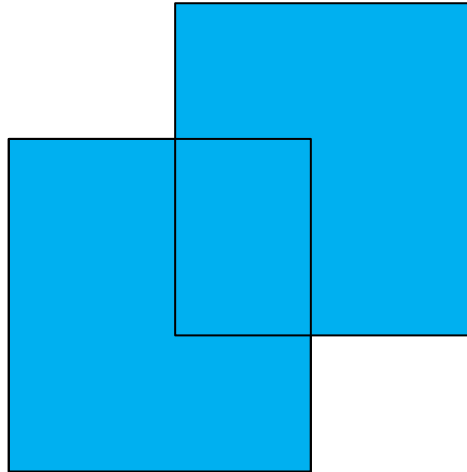
Snitt

- **Snittet** av R og S er definert som mengden som inneholder radene som finnes i **både R og S**.
 - Formel: $R \cap S$



Union

- **Unionen** av R og S er definert som mengden som inneholder radene som finnes i **R eller S**. (Duplikater tas ikke med.)
 - Formel: $R \cup S$



Union i SQL

[w3schools:](https://www.w3schools.com/sql/union.asp)

The SQL UNION Operator

The **UNION** operator is used to combine the result-set of two or more **SELECT** statements.

- Every **SELECT** statement within **UNION** must have the same number of columns
- The columns must also have similar data types
- The columns in every **SELECT** statement must also be in the same order

UNION Syntax

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

Union eksempel

- Navn på land som begynner på bokstavene Ar: (3 rader)

```
SELECT Name  
FROM country  
WHERE Name LIKE 'Ar%';
```

name
Aruba
Argentina
Armenia

- Navn på byer som begynner på bokstavene Ar:

```
SELECT Name  
FROM city  
WHERE Name LIKE 'Ar%';
```

30 rader, deriblant:

name
Amhem
Aracaju
Arapiraca
Araraquara
Araçatuba
Araguaína
Araras
Araguari
Arica
Arayat
Arrah (Ara)
Arak
Ardebil
Arezzo
Armenia
Arequipa
Arecibo
Argenteuil
Arad
Araçatuba
Arusha

Union eksempel

- Navn på land og byer som begynner på bokstavene Ar:

```
SELECT Name
FROM country
WHERE Name LIKE 'Ar%'
UNION
SELECT Name
FROM city
WHERE Name LIKE 'Ar%';
```

31 rader, deriblant:

Name
Aruba
Argentina
Armenia
Arnhem
Aracaju
Arapiraca
Araraqu...
Araçá...
Aragua...
Araras
Araguari
Arica
Arayat
Arrah (...)
Arak

Union ALL eksempel

- Navn på land og byer som begynner på bokstavene Ar:

```
SELECT Name
FROM country
WHERE Name LIKE 'Ar%'
UNION ALL
SELECT Name
FROM city
WHERE Name LIKE 'Ar%';
```

33 rader, deriblant:

Name
Aruba
Argentina
Armenia
Arnhem
Aracaju
Arapiraca
Araraqu...
AraËga...
Aragua...
Araras
Araguari
Arica
Arayat
Arrah (...)
Arak

Hvilke verdier er duplikater?

- Kanskje vi kan finne duplikatene ved å ta snittet? (Snitt = INTERSECT)

```
SELECT Name
FROM country
WHERE Name LIKE 'Ar%'
INTERSECT
SELECT Name
FROM city
WHERE Name LIKE 'Ar%';
```

Ser ikke lovende ut...

```
SELECT Name
FROM country
WHERE Name LIKE 'Ar%'
INTERSECT
SELECT Name
FROM city
WHERE Name LIKE 'Ar%';
```

- MySQL, reserverte ord (utdrag):

INSERT_METHOD	INSTANCE	INSTANCE
INT (R)	INT1 (R)	INT2 (R)
INT3 (R)	INT4 (R)	INT8 (R)
INTEGER (R)	INTERVAL (R)	INTO (R)
INVOKER	IO	IO_AFTER_GTIDS (R)
IO BEFORE GTIDS (R)	IO THREAD	IPC

Eksempel på variasjon i DBMS'er

- Oracle DB, [reserveerte ord](#) (utdrag):

```
INDEX  
INITIAL  
INSERT *  
INTEGER *  
INTERSECT *  
INTO *  
IS *  
LEVEL  
LIKE *  
LOCK  
LONG  
MAXEXTENTS  
MINUS  
MLSLABEL  
MODE
```

Kan vi få det til uten INTERSECT?

- Vi ønsker altså å finne navnene som forekommer i både land og by, men MySQL har ikke INTERSECT.
- Kan vi gjøre det i MySQL med en annen teknikk dere allerede har lært?

```
SELECT Name
FROM country
WHERE Name IN (
  SELECT Name
  FROM city
  WHERE Name LIKE 'Ar%');
```

- Finner 1 felles navn: "Armenia".
- Men hvor er det andre duplikatet? Hvis bare Armenia er delt mellom Country (3 rader) og City (30 rader), hvorfor får vi 31 rader (ikke 32) med UNION?
 - Det er to byer som har samme navn ("Arlington"), når vi benytter UNION er det som å bruke DISTINCT både på hver spørring og på resultatet!

Altså...

- Vi har direkte støtte for **UNION** i MySQL.
- Vi kan benytte **UNION ALL** hvis vi vil beholde duplikater.
- Vi har **ikke noe reservert ord for snitt i MySQL** (INTERSECT), men vi kan finne det i andre DBMSer, som f.eks Oracle.
- Vi kan utføre snitt ved å bruke **subquery**.
 - For subquery bruk til snitt, ta gjerne en titt på **EXISTS** også.
 - Kan også få til snitt ved å bruke JOIN (men subquery er gjerne et bedre valg).

Arbeidskrav

Hva er arbeidskrav?

- Noe du må **bestå** for å få ta eksamen.
- Er det vanlig i norsk IT-utdanning?

Institution	n1	MCW%
NTNU	75	73
UiT	38	74
HVL	31	100
UiB	28	96
UiA	38	82
UiS	31	65
UiO	81	91
HK	76	32
Hiof	52	73
OsloMet	69	67
Nord	25	48
USN	124	90

Hvordan opplever studenter arbeidskrav?

- They fit best within technical topics, such as computer programming.
- **They should be engaging and at an appropriate level of difficulty.** Since it may be difficult, or even impossible, to find *one* assignment that fits all students, consider having multiple assignments that students can choose from.
- They must be coordinated with other activities in parallel courses.
- **They must provide value. To many students, this means they must be relevant to the upcoming exam.**
- Be aware of the learning environment students solve assignments in. Fellow students and TAs play an important role in the learning process, and they should have a good understanding of how to provide help.
- **Provide more feedback (or rather: feed forward) to the students than simply the pass/fail verdict.**

Arbeidskravet i DB1102

- Modellere en database
- Valgfritt:
 - Opprette databasen med db-script (og fylle med data)
 - Kjøre spørringer mot databasen
 - Si noe om normaliseringsgraden
- Du kan velge å jobbe alene eller i gruppe.

Arbeidskrav

- Krav til å lykkes?
 - Krav til å arbeide?
-
- Ikke se på dette arbeidskravet som et hinder på veien mot eksamen. Se på det som en læringsaktivitet!

Tidsplan (tar imot innspill)

- Økt 6 (den vi er inne i nå): Finne ditt eget case. Hvilken database ønsker du å modellere? Ta utgangspunkt i noe du synes er interessant!
- 27. oktober: Levere modell (første utkast – kan inneholde feil/mangler) i Canvas. Modellen må ha en medfølgende forklaring.
- 3. november: Frist for å gi tilbakemelding til medstudent. Tilbakemeldinger gjennomføres individuelt.
- 17. november: Frist for å levere inn arbeidskravet.

Leveransen 27. oktober

- Hva om du ikke får det til?
- En ekstra seanse i Zoom (13. okt?), der jeg presenterer et case og hjelper dere i gang.

Leveransen 17. nov

- Modell med tilhørende forklaring.
- Valgfritt:
 - Db-script for å opprette databasen (med data)
 - Utvalgte spørringer mot databasen
 - Vurdering av normalisering
- Valgfritt format (video, tekst).
- Har du ikke fått det til? Lever en på forklaring på hvorfor du ikke kom i mål.

Blir du med?

- La fremtidige studenter lære av deg.
- Ha noe mer å vise til mer enn karakterkortet.
- Bruk målgruppen «en som vil lære om databaser».
- Eksempler fra tidligere studenter publiseres 6. oktober.
 - OBS! Du trenger ikke å velge video-format!

Lære av å lære bort

«[...] because you have to understand why it's like that. Because you have to explain it in your own way with your own database and your own modeling, your own db script, your own normalization. So you have to make everything yourself. And that's what I think is great. Because now, I've learned all about it.»

Ukens øvingsoppgaver

1. Tidligere deleksamen

- Tidligere hadde databaseemnet en deleksamen som utgjorde 25% av karakteren i emnet. Slik er det ikke lenger. Nå har vi en avsluttende 24t hjemmeeksamen som teller 100%.
- Denne uken skal dere få teste dere selv med en tidligere del-eksamen. Dere velger selv hvordan dere gjennomfører den, men her er min anbefaling:
 - Bruk tiden fram mot øving til å forberede deg (repetere)
 - Sett av to timer (f.eks øvingstiden).
 - Ta eksamen slik den var (uten hjelpemidler).
 - Sammenlikne dine svar med fasit.
 - Forsøk å forstå riktig svar hvis du svarte feil.
- Fasiten på deleksamen publiseres onsdag kl 12, som vanlig.

2. Database til arbeidskrav

- Forsøk å tenke ut et case som du synes er interessant, og som du tenker det vil være mulig å lage en database for.
- En passende størrelse på en database (for dette arbeidskravet) kan være mellom 5 og 10 tabeller.
- Tenk igjennom om du ønsker å jobbe med arbeidskravet ditt alene eller sammen med andre.
- Husk at vi vil bruke de to neste ukene til å lære oss å modellere 😊

Neste gang

- Neste gang, nytt tema: [Modellering av databaser](#)
 - Pensum for de neste to øktene: [Kapittel 7 – 8.1.](#)