

SQL fortsat

- ▶ DEMO: databaseoprettelse og tilgang til ITU's MySQL-server
- ▶ Aggregering og gruppering med GROUP BY
- ▶ Kombination af GROUP BY og HAVING
- ▶ Venstre- eller højrestyret samkøring: LEFT JOIN og RIGHT JOIN
- ▶ Indekser og effektivitet
- ▶ Primærnøgler
- ▶ Navngivning af tabeller i forespørgsler, self-joins
- ▶ Diverse hints: Hvilke tabeller findes i databasen? Hvordan loades data fra en fil?

Databaseoprettelse

Se kursushjemmeside:

<http://www.itu.dk/courses/DSDS/E2007>

Tilgang til MySQL-server med `mysql`

Se siden

<http://www.itu.dk/courses/DSDS/E2007/mysql-da.html>

Eksempel: Kursusdata

Student:

snavn	snr	adresse
Ole Olesen	L1234	Hjemmevej 7
Rikke Richardsen	L2345	Udvej 9
Søren Sørensen	J0007	I dybet 13
Ulrikka Funder	B7117	Skovvej 11

Kursus:

knr	knavn	lærer
W2	Databasestøttet Webpublicering	Martin Elsman
GP	Grundlæggende Programmering	Morten Rhiger

Tilmelding:

snr	knr
L1234	W2
L2345	W2
J0007	W2
J0007	GP
L2345	GP

Kommandoen `create table` benyttes til oprettelse af tabeller:

```
CREATE TABLE Student (snavn VARCHAR(80),  
                        snr VARCHAR(10) NOT NULL,  
                        adresse VARCHAR(80));
```

```
CREATE TABLE Kursus (knr INT NOT NULL,  
                      knavn VARCHAR(80),  
                      laerer VARCHAR(80));
```

```
CREATE TABLE Tilmelding (snr VARCHAR(10) NOT NULL,  
                          knr INT NOT NULL);
```

```
INSERT INTO ...;
```

```
...
```

Aggregerede udtræk i SELECT, omdøbning af felter med AS, og GROUP BY

Vis samlede antal kursustilmeldinger:

```
SELECT COUNT(*) FROM Tilmelding;
```

Felter kan navngives eller omdøbes med AS:

```
SELECT COUNT(*) AS antal FROM Tilmelding;
```

Vis antal studerende der er tilmeldt hvert kursus:

```
SELECT knr, COUNT(snr) FROM Tilmelding  
GROUP BY knr;
```

Navn	Betydning
COUNT(udtryk)	Antal værdier
MIN(udtryk)	Mindste værdi (eller NULL)
MAX(udtryk)	Største værdi (eller NULL)
SUM(udtryk)	Summen af værdierne (eller NULL)
AVG(udtryk)	Middelværdi af værdierne

Bemærk: COUNT tæller antallet af ikke-NULL værdier.

Et andet eksempel: udgifter i en virksomhed

Model og Data

```
CREATE TABLE Udgift (art VARCHAR(100), afd VARCHAR(100),  
                    aar INT,                beloeb INT);
```

```
INSERT INTO Udgift (art, afd, aar, beloeb)  
VALUES ('Loen', 'Indkoeb', 2001, 490000);
```

...

Tabellen Udgift efter indsættelse af data

art	afd	aar	beloeb
Loen	Indkoeb	2001	490.000,-
Loen	Salg	2002	1.500.000,-
Loen	Indkoeb	2002	500.000,-
Kaffe	Indkoeb	2003	800,-
Kaffe	Salg	2003	300,-
Loen	Salg	2003	1.600.000,-
Loen	Indkoeb	2003	510.000,-

- ▶ Aggregering og GROUP BY kan bruges til at lave subtotaler.
- ▶ Eksempel: Samlede udgifter opgjort på art og afdeling for årene 2002–2003:

```
SELECT art, afd, SUM(beloeb)
FROM Udgift
WHERE 2002 <= aar AND aar <= 2003
GROUP BY art, afd;
```

- ▶ Find de totale udgifter opgjort på art for årene 2002–2003?
- ▶ Find, de totale udgifter opgjort på år?

Betingelser i grupperinger

- ▶ Men hvad hvis man ikke vil have uddata-rækker med hvor udgift-summen er mindre end 1.000.000?
- ▶ Man kan ikke bruge `WHERE SUM(beloeb) >= 1000000`.
- ▶ Man kan derimod tilføje `HAVING SUM(beloeb) >= 1000000` efter `GROUP BY`:

```
SELECT aar, SUM(beloeb)
FROM Udgift
GROUP BY aar
HAVING SUM(beloeb) >= 1000000;
```

- ▶ For kursuseksemplet, hvordan kan man bruge `GROUP BY ... HAVING` til at finde navnet på de kurser der har mindst 2 tilmeldte?

Problemet med Indre Joins

Problem: Samkøring (almindelige joins) medtager kun en linie hvis der er et match...

Eksempel: For hvert kursus, vis studienumre på de tilmeldte studerende:

```
SELECT knavn, snr FROM Kursus, Tilmelding
WHERE Kursus.knr = Tilmelding.knr;
```

Eller: for hvert kursus, vis *antal* tilmeldte studerende:

```
SELECT knavn, COUNT(snr) FROM Kursus, Tilmelding
WHERE Kursus.knr = Tilmelding.knr
GROUP BY knavn;
```

Mangel: kun kurser der faktisk har studerende tilknyttet bliver vist.

Ydre joins: LEFT JOIN og RIGHT JOIN

- ▶ Med LEFT JOIN tages **alle linier fra venstre tabel**, selvom der ikke er en matchende linie i højre tabel.

```
SELECT Kursus.knavn, Tilmelding.snr
FROM Kursus LEFT JOIN Tilmelding
      ON Kursus.knr = Tilmelding.knr;
```

- ▶ Vis antal tilmeldte for hvert kursus:

```
SELECT knavn, COUNT(snr)
FROM Kursus LEFT JOIN Tilmelding
      ON Kursus.knr = Tilmelding.knr
GROUP BY knavn;
```

- ▶ Normal samkøring (join) ville glemme de kurser der har 0 tilmeldte, så LEFT JOIN er vigtig!
- ▶ Med RIGHT JOIN tages **alle linier fra højre tabel**, selvom der ikke er en matchende linie i venstre tabel.

Hvor lang tid tager en forespørgsel?

- ▶ En samkøring (join) af to tabeller kan tage meget lang tid.
- ▶ Grundlæggende skal hver linie i den ene tabel matches med hver linie i den anden.
- ▶ Med 10.000 linier i den ene tabel og 10.000 i den anden kræves der altså $10.000 * 10.000 = 100.000.000$ sammenligninger.
- ▶ Når Large1 og Large2 er på 10.000 linier tager dette 20 sekunder:

```
SELECT id1
FROM Large1, Large2
WHERE Large1.id1 = Large2.id2.
```

- ▶ Med *indekser* på id1 og id2 tager det 0.05 sekunder; det er **400 gange hurtigere**.
- ▶ Jo flere linier der er, jo vigtigere er det at bruge et indeks.
- ▶ Med 1.000.000 linier i hver tabel vil indekser gøre forespørgslen 15.000 gange hurtigere!

Hvad er et indeks?

Et indeks svarer til en “**telefonbog**” for et felt i en tabel.

I en almindelig telefon(navne)bog gælder det:

- ▶ Det er svært finde et navn tilhørende et nummer.
- ▶ Det er nemt at finde nummeret tilhørende et navn.
- ▶ Man skal kun se på 20–30 navne for at finde ét bestemt navn.

Med et indeks kan man finde en bestemt værdi med kun $\log_2(N)$ sammenligninger, når der er N mulige værdier.

Eksempel: Folkeregister med 5.000.000 personer (linier) i en tabel.

- ▶ Find et CPR-nummer uden indeks: 2.500.000 sammenligninger i gennemsnit.
- ▶ Find et CPR-nummer med indeks: $\log_2(5.000.000) = 22,3$ sammenligninger i gennemsnit

Oprettelse af indekser på felter i tabeller

- ▶ En tabel kan have indekser på felter, og på feltkombinationer (felter der indgår skal erklæres NOT NULL).
- ▶ Oprettelse af indeks på feltet id1 i tabel Large1:

```
CREATE INDEX Large1_id1 ON Large1 (id1);
```

- ▶ Man kan bestemme at en bestemt felt-værdi må forekomme højst 1 gang i en tabel:

```
CREATE UNIQUE INDEX Student_snr ON Student (snr);
```

Dette indeks betyder at det er umuligt at give to studerende samme studienummer snr. Prøv!

- ▶ Man kan lave indekser på flere felter samtidig sådan:

```
CREATE UNIQUE INDEX Tilmelding_snr_knr  
ON Tilmelding (snr, knr);
```

Det gør opslag på kombinationer af felter meget hurtigere end hvis der var indekser på felterne hver for sig.

- ▶ Hvad betyder UNIQUE i indekset Tilmelding_snr_knr?

Primærnøgle i en tabel

- ▶ En *primærnøgle* er en kombination af felter som entydigt identificerer en linie i tabellen.
- ▶ Typisk udgør primærnøglen den vigtigste måde at slå op i tabellen.
- ▶ Eksempler:
 - ▶ snr er primærnøgle i Student
 - ▶ knr er primærnøgle i Kursus
 - ▶ (snr, knr) er primærnøgle i Tilmelding
- ▶ Man kan erklære primærnøglen når man opretter tabellen:

```
CREATE TABLE Student (snavn VARCHAR(80),  
                        snr VARCHAR(10) PRIMARY KEY,  
                        adresse VARCHAR(80));
```

- ▶ Når primærnøglen består af flere felter skal PRIMARY KEY skrives for sig selv:

```
CREATE TABLE Tilmelding (snr VARCHAR(10) NOT NULL,  
                          knr VARCHAR(10) NOT NULL,  
                          PRIMARY KEY (snr, knr));
```

- ▶ Om en erklæret primærnøgle gælder der:
 - ▶ Alle felter i primærnøglen er automatisk NOT NULL.
 - ▶ Der er automatisk UNIQUE INDEX på kombinationen af primærnøglens felter.

Navngivning af tabeller i forespørgsler, self-joins

- ▶ Ønske: Find alle par af kurser der har samme lærer.
- ▶ Dvs. find alle par (k_1, k_2) af kurser hvor k_1 's lærer er den samme som k_2 's lærer.
- ▶ Det kan laves ved en samkøring af `Kursus`-tabellen med sig selv (self-join).
- ▶ Man kan navngive tabellerne i en samkøring ved at skrive et nyt navn (f.eks. `k1`) efter tabelnavnet:

```
SELECT k1.knr, k2.knr
FROM Kursus k1, Kursus k2
WHERE k1.laerer = k2.laerer;
```

- ▶ Forespørgslen giver unødigt mange resultater: Både $(15341, 15311)$ og $(15311, 15341)$, og desuden f.eks. $(10188, 10188)$.
- ▶ Tilføj `AND k1.knr < k2.knr` for at udelukke uønskede rækker.

Hvilke tabeller findes i databasen? Hvilke felter har de? Osv

Vis mine tabeller:

```
SHOW TABLES;
```

Vis navn og type på felter i en tabel:

```
SHOW FIELDS FROM Tabel;
```

Vis indholdet af en tabel:

```
SELECT * FROM Tabel;
```

Importer data fra tekstfil til eksisterende tabel:

```
LOAD DATA LOCAL INFILE "wwwstats.txt" INTO TABLE Weblog;
```

Øvelser

- ▶ Aggregerede forespørgsler med SQL
- ▶ Analyse a Weblog med SQL (Ekstra opgave)

Næste gang

- ▶ Forbindelse til MySQL database fra PHP på webserveren
- ▶ Websites der er databaser