

Database-støttet Web-publicering, efterår 2001

Forelæsning 7, tirsdag den 9. oktober 2001.

Databaseprogrammering med SQL

- Tcl-kommandoen `regsub`
- Eksempler på brug af `regsub`-kommandoen
- Fire kategorier af sites
- Konstruktion af sites som er databaser
- Filsystemet som database
- Rigtige databaser - RDBMS's (the ACID test)
- SQL (Structured Query Language)
- Data Definition Language – `create`, `alter` og `drop`
- Data Manipulation Language – `insert`, `select`, `joins`, `delete` og `update`
- Eksempel på en database: studenter, kurser, eksamener
- Flere eksempler på SQL
- Databasesystemet Oracle8i

IT-højskolen

Databasestøttet Web-publicering, efterår 2001

Side 7-1

Tcl-kommandoen `regsub`, fortsat

`regsub` *exp string subSpec varName*

Enkelt tegn i *subSpec* har speciel betydning for et given match:

- `&` og `\` står for den del af *string* som matcher *exp*

Eksempel: givet lokalnummer N NN konstruer et fuldt nummer: 38 16 8N NN:

```
% set t1fs "8 34, 8 55, 8 67"
8 34, 8 55, 8 67
% regsub -all {[0-9] [0-9][0-9]} $t1fs "38 16 8&" res
3
% set res
38 16 88 34, 38 16 88 55, 38 16 88 67
% regsub -all {[0-9] [0-9][0-9]} $t1fs {38 16 8\0} res
3
% set res
38 16 88 34, 38 16 88 55, 38 16 88 67
```

- N står for den del af *string* som matcher det N'de regulære udtryk omgivet af parenteser i *exp* - læst fra venstre mod højre.

Eksempel: givet lokalnummer M NN konstruer et fuldt nummer 38 16 89 NN, hvor M er udskiftet med 9:

```
% set t1fs "8 34, 8 55, 8 67"
8 34, 8 55, 8 67
% regsub -all {[0-9]} {[0-9][0-9]} $t1fs {38 16 89 \2} res
3
% set res
38 16 89 34, 38 16 89 55, 38 16 89 67
```

IT-højskolen

Databasestøttet Web-publicering, efterår 2001

Side 7-3

Tcl-kommandoen `regsub`

Bruges til at udskifte tegnsekvenser som matcher et mønster med andre tegnsekvenser.

`regsub` *exp string subSpec varName*

Kommandoen matcher det regulære udtryk *exp* mod *string* og kopierer *string* til variabelen *varName*.

Under kopieringen erstattes de dele af *string* der matcher *exp* med *subSpec*.

Eksempel, udskiftning af `<` med `<t;`:

```
% set html "<html><body>Hej</body></html>"
<html><body>Hej</body></html>
% regsub -all < $html {\&lt;t;} res
4
% set res
&lt;t;html>&lt;t;body>Hej&lt;t;/body>&lt;t;/html>
%
```

Eksempel, udskiftning af lokalnummer 853 med 824

```
% set t1f "38 16 88 53"
38 16 88 53
% regsub "8 53" $t1f "8 24" res
1
% set res
38 16 88 24
%
```

IT-højskolen

Databasestøttet Web-publicering, efterår 2001

Side 7-2

Flere eksempler på brug af `regsub`-kommandoen

Templates:

```
% set template "Hi name,\nHappy birthday!\nCheers, Niels"
% regsub name $template "George Bush" message
1
% set message
Hi George Bush,
Happy birthday!
Cheers, Niels
```

Udskiftning af email-adresser:

```
% set text "Min email adresse er nh@itu.dk, Frank's er bjergoe@itu.dk
og lise's er lise@itu.dk."
Min email adresse er nh@itu.dk, Frank's er bjergoe@itu.dk
og lise's er lise@itu.dk.
% regsub -all "@itu.dk" $text "@it-c.dk" text
3
% set text
Min email adresse er nh@it-c.dk, Frank's er bjergoe@it-c.dk
og lise's er lise@it-c.dk.
% regsub -all "@it-c.dk" $text "@it.edu" text
3
% set text
Min email adresse er nh@it.edu, Frank's er bjergoe@it.edu
og lise's er lise@it.edu.
```

Husk, at tilføjelsen `-all` gør at alle match fanges!

IT-højskolen

Databasestøttet Web-publicering, efterår 2001

Side 7-4

Opgaver til regulære udtryk og regex-kommandoer

1. Omform strengen `abe`, `abe` til `abekat`, `abekat`:
% set text "abe abe"
abe abe
% regexsub -all {_____} \$text {_____} res
2
% set res
abekat abekat
2. Omform strengen "nh, lise, bjergoe" til `nh@it.edu`, `lise@it.edu`, `bjergoe@it.edu`:
bjergoe@it.edu:
% set text "nh, lise, bjergoe"
nh, lise, bjergoe
% regexsub -all {(_____) } \$text {_____} res
3
% set res
nh@it.edu, lise@it.edu, bjergoe@it.edu
3. Omform strengen "hej <i>Jonas</i>" til "hej <u>Jonas</u>":
% set text "hej <i>Jonas</i>"
% regexsub -all {_____} \$text {_____} res
2
% set res
hej <u>Jonas</u>

Vink: Benyt /? til at matche både <i> og </i>.

Konstruktion af sites som er databaser

- Konstruktion af datamodel
- hvilken information skal gemmes og hvordan skal den repræsenteres
 - dette er den svære del
- Udvikling af legale data-transaktioner
- hvordan indsættes data i databasen
 - hvordan udtrækkes data fra databasen
- Konstruktion af web-forms til at implementere transaktionerne
- brugergrænsefladen er html-kode (forms)
 - SQL (Structured Query Language) bruges til de egentlige database-transaktioner
 - dette er den nemme del

Fire kategorier af sites

1. Traditionelle sites
 - online aviser, vejruddsigter, aktiekurser
 - høje omkostninger - indtjening ved reklamer, sponsorer, abonnenter
2. Sites som publicerer kollaborativt indsamlet information
 - information opsamles elektronisk via forms
 - eksempel: brugerundersøgelser
3. Sites som tilbyder en service via et webserver-program
 - bryllups-service, WimpyPoint, CourseGrader
 - indtjening ved abonnenter, fokuseret reklame
4. Sites som definerer en standard som tillader brugere at tilgå flere databaser
 - bilbasar, auktionssites
 - store indtjeningsmuligheder - kun udgifter til teknologien

Sites i de sidste tre kategorier er database-sites!

Filsystemet som database

- Fordele:
- løsning nem at forstå - data gemmes som tekst i filer
 - ingen ekstra software kræves
- | | |
|----------------|------------------|
| nh@it.edu | Niels Hallenberg |
| kenneth@it.edu | Kenneth Riis |
| lise@it.edu | Lise Bennedsen |

Nogle ulemper:

- umiddelbare performance-problemer (lineær søgning)
- manglende abstraktion fra datarepræsentation
- problemer med håndtering af samtidige brugere
- ingen standard for integration med andre systemer
- problemer med fejlhåndtering

Man ender med at opbygge hvad firmaer har været gode til siden 60'erne: (R)DBMS's.

Rigtige databaser - RDBMS's (the ACID test)

En god database skal bestå "the ACID test":

Atomicity – en transaktion er enten fuldt udført eller slet ikke udført

- overførsel mellem bankkonti: enten er begge konti opdateret ellers er ingen af dem opdateret

Consistency – transaktioner sender databasen fra en legal tilstand til en anden legal tilstand

- sikring af rapportering af pengeoverførsler til skattemyndighederne

Isolation – transaktion er usynlig for andre transaktioner indtil transaktionen er komplet

- overførsel mellem bankkonti samtidig med generering af regnskabsrapport

Durability – komplette transaktioner overlever fremtidige systemcrash

Databasebegreber

En *relationsdatabase* består af en samling navngivne relationer (= tabeller).

En *relation* (= tabel) består af to ting:

- Et skema (= tabeloverskrift):

Skemaet er relationens form; det er uforanderligt.

Skemaet angiver navn og type på relationens attributter (felter).

- En samling tupler (= tabellinjer):

Samlingen af *tupler* er relationens indhold; den kan variere over tid.

Hvert tupel indeholder et sæt værdier for relationens attributter (felter).

Rækkefølgen af tuplerne i en relation er ligegyldig.

Der er mange lighedspunkter mellem en tabel og et regneark.

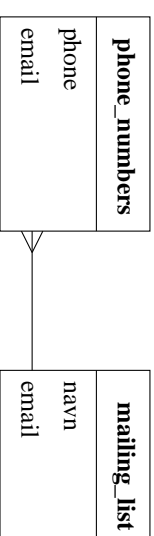
Eksempel på en mailingliste: mailing_list, phone_numbers

Et udsnit af relationen mailing_list:

email	navn
kenneth@it.edu	Kenneth Riis
nh@it.edu	Niels Hallenberg

Et udsnit af relationen phone_numbers:

email	phone
kenneth@it.edu	44 84 34 94
nh@it.edu	38 16 88 88
nh@it.edu	38 16 88 24



Til hver person, unikt identificeret ved email, kan der være mere end et telefonnummer. "Gæffen" angiver en "en-til-mange-relation".

SQL (Structured Query Language)

Når man arbejder med relationsdatabaser bruger man sproget SQL

SQL er opfundet af IBM ca. 1970.

I SQL skelner man ikke mellem store og små bogstaver (i modsætning til T_{cl}).

- **Data Definition Language**
 - design af datastrukturer
- **Data Manipulation Language**
 - manipulation af data

Data Definition Language (ma11.db.sql)

Eksempel på create table kommandoer:

```
create table mailing_list (  
  email varchar(100) primary key,  
  name varchar(100) not null  
);  
  
create table phone_numbers (  
  email references mailing_list,  
  phone varchar(20) not null  
);
```

primary key betyder *primærnøgle*: feltet email bliver unikt og ikke tom.

not null betyder at feltet er ikke tomt.

varchar(100) betyder at feltet maksimalt kan være på 100 tegn.

email references mailing_list betyder at email *refererer* til et tilsvarende felt i tabellen mailing_list. Der skal findes et række i mailing_list, hvor feltet email har samme værdi som email.

Andre kommandoer:

- alter table – tilføj eller modificer kolonne
- drop table – slet tabel fra database

Data Manipulation Language - select

Kommandoen select bruges til at udtrække data fra en database.

Eksempel:

```
column email format a20  
column name format a25  
  
select * from mailing_list;
```

Endnu et eksempel:

```
select name from mailing_list;
```

En where-clause bruges til at begrænse rækkerne i resultatet:

```
select name  
from mailing_list  
where email = 'kenneth@it.edu';
```

Data Manipulation Language - insert

Eksempler på insert kommandoer:

```
insert into mailing_list (name, email)  
values ('Kenneth Riss', 'kenneth@it.edu');  
  
insert into mailing_list (name, email)  
values ('Niels Hallenberg', 'nh@it.edu');  
  
insert into phone_numbers (email, phone)  
values ('kenneth@it.edu', '44 84 34 94');  
  
insert into phone_numbers (email, phone)  
values ('nh@it.edu', '35 28 23 04');  
  
insert into phone_numbers (email, phone)  
values ('nh@it.edu', '38 16 88 43');
```

Data Manipulation Language - Joins

Join:

```
select * from mailing_list, phone_numbers;
```

En bedre join:

```
select * from mailing_list, phone_numbers  
where mailing_list.email = phone_numbers.email;
```

Eller endnu bedre:

```
select name, mailing_list.email, phone  
from mailing_list, phone_numbers  
where mailing_list.email = phone_numbers.email;
```

Data Manipulation Language - delete og update

- Kommandoen delete bruges til at slette rækker fra en tabel:

```
-- Virker ikke pga. 'integrity constraint'  
delete from mailing_list where email = 'nh@it.edu';
```
 - Vis skal først slette fra phone_numbers:

```
delete from phone_numbers where email = 'nh@it.edu';  
delete from mailing_list where email = 'nh@it.edu';
```
 - Kommandoen update bruges til at opdatere kolonner i en tabel:

```
delete from phone_numbers where email = 'kenneth@it.edu';
```
- ```
update mailing_list
set email = 'kenneth@it-c.dk'
where email = 'kenneth@it.edu';
```
- Pas på where-betingelserne!!

## Relationer og skemaer i eksemplet

Relationer: Student, Kursus, Eksamen

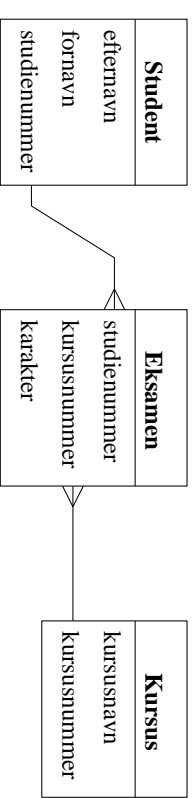
Relationernes skemaer:

Student: (efternavn VARCHAR(200), fornavn VARCHAR(200), studienummer VARCHAR(200))

Kursus: (kursusnavn VARCHAR(200), kursusnummer INT)

Eksamen: (studienummer VARCHAR(200), kursusnummer INT, karakter INT)

Ud for hvert felt angives feltets type, f.eks. VARCHAR(n) og INT.



## Eksempel på en database: studenter, kurser, eksamener (studiedb.sql)

Et udsnit af relationen Student:

| efternavn | fornavn | studienummer |
|-----------|---------|--------------|
| Olesen    | Peter   | L2143        |
| Hansen    | Etika   | Jø0007       |
| Funder    | Ulrik   | Hg0014       |

Et udsnit af relationen Kursus:

| kursusnummer | kursusnavn             |
|--------------|------------------------|
| 10181        | Databehandling         |
| 45621        | Landbrugszoologi       |
| 15351        | Miljømodeller          |
| 15311        | Matematisk Grundkursus |

Et udsnit af relationen Eksamen:

| studienummer | kursusnummer | karakter |
|--------------|--------------|----------|
| L2143        | 010181       | 8        |
| L2143        | 045621       | 9        |
| Jø0007       | 010181       | 11       |
| Jø0007       | 015311       | 8        |
| L2143        | 015311       | 10       |
| Hg0014       | 015311       | 7        |

## Opretelse af relationer (SQL CREATE) (studiedb.sql)

```
CREATE TABLE student (efternavn VARCHAR(200), fornavn VARCHAR(200),
studienummer VARCHAR(200));
```

```
CREATE TABLE kursus (kursusnummer INT, kursusnavn VARCHAR(200));
```

```
CREATE TABLE eksamen (studienummer VARCHAR(200), kursusnummer INT,
karakter INT);
```

## Oracle8i

- SQL\*Plus ved brug af ssh (Secure Shell) til hug.it.edu
- Brug kommandoen  
sql  
når du er logget på hug.it.edu

### Indsættelse af værdier i relationerne (SQL INSERT) (studiedb.sql)

```
INSERT INTO student (efternavn, fornavn, studienummer)
VALUES ('Olesen', 'Peter', 'I2143');
INSERT INTO student (efternavn, fornavn, studienummer)
VALUES ('Hansen', 'Erika', 'J00007');
INSERT INTO student (efternavn, fornavn, studienummer)
VALUES ('Funder', 'Ulrik', 'Hg0014');
INSERT INTO kursus (kursusnummer, kursusnavn)
VALUES (10181, 'Databehandling');
INSERT INTO kursus (kursusnummer, kursusnavn)
VALUES (45621, 'Landbrugszoologi');
INSERT INTO kursus (kursusnummer, kursusnavn)
VALUES (15351, 'Miljømodeller');
INSERT INTO kursus (kursusnummer, kursusnavn)
VALUES (15311, 'Matematisk Grundkursus');
INSERT INTO eksamen (studienummer, kursusnummer, karakter)
VALUES ('L2143', 010181, 8);
INSERT INTO eksamen (studienummer, kursusnummer, karakter)
VALUES ('L2143', 045621, 9);
INSERT INTO eksamen (studienummer, kursusnummer, karakter)
VALUES ('J00007', 010181, 11);
INSERT INTO eksamen (studienummer, kursusnummer, karakter)
VALUES ('J00007', 015311, 8);
INSERT INTO eksamen (studienummer, kursusnummer, karakter)
VALUES ('L2143', 015311, 10);
INSERT INTO eksamen (studienummer, kursusnummer, karakter)
VALUES ('Hg0014', 015311, 7);
```

### Forespørgsler på databasen (SQL SELECT)

```
SELECT: Vis alle kurser:
SELECT * FROM kursus ;

ORDER BY: Vis alle kurser sorteret efter kursusnavn:
SELECT * FROM kursus ORDER BY kursusnavn;

WHERE: Vis alle kurser med kursusnummer 010181:
SELECT * FROM kursus WHERE kursusnummer = 010181;

Vis kursusnumre på alle kurser der har været holdt eksamen i:
SELECT kursusnummer FROM eksamen;

DISTINCT: Samme, uden dubletter:
SELECT DISTINCT kursusnummer FROM eksamen;
```

### Formattering af kolonner

```
COLUMN efternavn FORMAT a10
COLUMN fornavn FORMAT a10
COLUMN kursusnavn FORMAT a25
COLUMN studienummer FORMAT a6
```

### Forespørgsler der involverer flere relationer (sankøring: join)

Vis kursusnavn på alle kurser der har været holdt eksamen i:  
Hvordan — kursusnavnet står jo ikke i relationen Eksamen?  
Lav en sankøring (join) mellem relationerne Eksamen og kursus:

```
SELECT DISTINCT kursusnavn
FROM kursus, eksamen
WHERE kursus.kursusnummer = eksamen.kursusnummer;

Vis alle studerende der har været til eksamen i kursus 10181:
```

```
SELECT DISTINCT fornavn, efternavn FROM student, eksamen
WHERE student.studienummer = eksamen.studienummer
AND kursusnummer = 10181;
```

Vis alle studerende der har været til eksamen i kurset 'Databehandling':

```
SELECT DISTINCT fornavn, efternavn FROM student, eksamen, kursus
WHERE student.studienummer = eksamen.studienummer
AND kursus.kursusnummer = eksamen.kursusnummer
AND kursus.kursusnavn = 'Databehandling';
```

### Støtning (SQL DELETE)

Vis alle karakterer under 9:

```
SELECT * from eksamen WHERE karakter < 9;
```

Fjern alle eksamener med karakterer under 9:

```
DELETE FROM eksamen WHERE karakter < 9;
```

Vis alle studerende der har fået mindst 11 i en eksamen:

```
SELECT * FROM student, eksamen
WHERE student.studienummer = eksamen.studienummer
AND eksamen.karakter >= 11;
```

Fjern alle studerende der har fået mindst 11 i en eksamen.

```
DELETE FROM student
WHERE student.studienummer =
(SELECT studienummer FROM eksamen WHERE karakter >= 11);
```

### Opdatering (SQL UPDATE)

Vis alle karakterer mellem 7 og 11:

```
SELECT * from eksamen
WHERE 7 <= karakter AND karakter <= 11;
```

Nedsæt alle karakterer mellem 7 og 11 med 1:

```
UPDATE eksamen
SET karakter = karakter-1
WHERE 7 <= karakter AND karakter <= 11;
```

### Introduktion til Øvelse 6

- Oracle8i på hug.it.edu
- SQL\*Plus ved brug af ssh (Secure Shell) til hug.it.edu
- Brug kommandoen  
sql

når du er logget på hug.it.edu

### Aggregerede udtræk: COUNT, SUM, AVG, MIN, MAX

**COUNT:** Vis antal eksamener hver studerende har gået til:

```
SELECT studienummer, COUNT(karakter) AS karakter FROM eksamen
GROUP BY studienummer;
```

**Opgave:** Vis summen af alle karakterer pr. studerende

**Opgave:** Vis den højeste karakter for hver studerende

**Opgave:** Vis snittet for hver studerende

### Slet en hel relation (SQL DROP TABLE)

```
DROP TABLE student;
DROP TABLE kursus;
DROP TABLE eksamen;
```

### Modelering og manipulation af data

#### Ved fastlæggelse af datamodel og initialisering af website:

- SQL's modelerings-kommandoer, såsom create table, udføres fra f.eks. SQL\*Plus ved brug af @kommandoen
- initieft data indsættes i databasen

#### Når sitet kører:

- data indsættes, slettes og modificeres når brugere tilgår databasen via web-forms
- kun sjældent oprettes og slettes der nye tabeller