

Databasestøttet Webpublicering, forår 2002

Forelæsning 7

Databaseprogrammering med SQL

- 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 Oracle!

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 ind sættes data i databasen
- hvordan udrækkes 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, vejrudsigter, aktiekurser
 - høje omkostninger - indtjening ved reklamer, sponsorer, abonnenter
 2. Sites som publicerer kollaborativt indsamlet information
 - information opsamlles 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
 - bilbaser, auktionssites
 - store indtjeningsmuligheder - kun udgifter til teknologien
- Sites i de sidste tre kategorier er database-sites!

Filsystemet som database

Fordele:

- ingen ekstra software kræves
- løsning nem at forstå - data gemmes som tekst i filer:
 - nh@it.edu Niels Hallenberg
 - kenneth@it.edu Kenneth Rils
 - lise@it.edu Lise Bennedsen
 - gates@microsoft.com Bill Gates

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 (= en tabels form):
Skemaet er relationens form: det er uforanderligt.
Skemaet angiver navn og type på relationens attributter (felter).
- En samling tupler (= tabelliner):
Samlingen af *tupler* er relationens indhold; den kan variere over tid.
Hvert tuple 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.

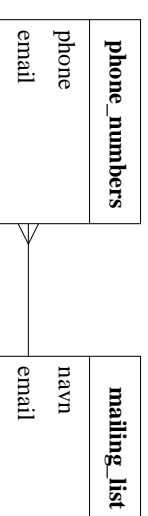
Eksempel på en mailingliste: mailing_list, phone_numbers

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. "Gaffen" 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 Tcl).

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

Data Definition Language (mai1ldb .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 tomt.
 - 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 udrække data fra en database.

Eksempel:
column email format a15
column name format a20

```
select * from mailing_list;
```

Endnu et eksempel:

```
select name from mailing_list;
```

En where-clause bruges til at begrænse rækkekerne 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 Riis', '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';
- Vi 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!!!

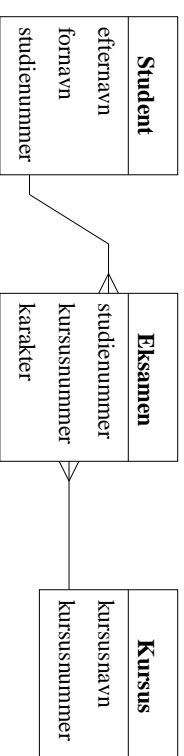
Relationer og skemaer i eksemplet

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	Erika	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

Oprettelse 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) (studi.ecb .sql)

```
insert into student (efternavn, fornavn, studienummer)
values ('Olesen', 'Peter', 'L2143');
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, 'Miljømedier');
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 der involverer flere relationer (sammenkøring: join)

Vis kursusnavn på alle kurser der har været holdt eksamen i.

Hvordan? Kursusnavnet står jo ikke i relationen Eksamen!

Lav en sammenkøring (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, kursus
where student.studienummer = eksamen.studienummer
and kursus.kursusnummer = eksamen.kursusnummer
and kursus.kursusnavn = 'Databehandling';
```

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
```

Sletning (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 in
(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;
```

Modelering og manipulation af data

Ved fastlæggelse af datamodel og initialisering af webste:

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

Når slettet kører:

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

Aggregerede udtræk: count, sum, AVG, MIN, max

count: Vis antal eksamener hver studerende har gået til:

```
select studienummer, count(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;
```