Databasesystemer, forår 2006

IT Universitetet i København

Forelæsning 1: Introduktion

2. februar 2006

Forelæser: Rasmus Pagh

Lidt om jeres undervisere

Rasmus Pagh (forelæser):

- Ph.d. i datalogi fra AU, 2002. Forskning i algoritmik, bl.a. i spørgsmål omkring effektiv implementation af DBMSer, softwaren bag databaser.
- Har undervist i databaser på ITU siden foråret 2003, på såvel introducerende som videregående databasekurser.

Esben Rune Hansen (forelæser):

- Cand. it. og ph.d. studerende på ITU.
- Forskning indenfor brug af logik i design af brugergrænseflader.

Hjælpelærere:

- Knut Tveitane (SWU stud. på ITU).
- Martin Scheil Corneliussen (SWU stud. på ITU).

Kursussprog: Danglish

- Kursusbog på engelsk.
- Forelæsninger på dansk.
- Tekniske termer uden entydig oversættelse dog altid på engelsk.
- Slides, opgaver, etc. på blandet dansk og engelsk.

Forelæsningen i dag

- Hvorfor studere databaser?
- Målsætning og indhold.
- Praktisk information.
- Introduktion til kursets indhold.
- Øvelser: Brug af ITUs Oracle database.

Hvorfor studere databaser?

Akademikerens grund:

Databaser har berøring med mange fascinerende emner indenfor datalogi.

Programmørens grund:

Har brug for databaser til at udvikle programmer.

Informationspilotens grund:

Vil arbejde med, og trække information ud fra, store, dynamiske datamængder.

Kapitalistens grund:

Alle har brug for databaser – her er der penge at tjene!

Målsætning for kurset

Du får kendskab til de grundlæggende regler for datamodellering og datamanipulation i (relationelle) databaser, og således baggrund for at arbejde med systemer, hvor databaser indgår.

Specifikke kompetencer:

- Lave et database-design i E-R modellen, og oversætte det til en relationel datamodel.
- Programmere forespørgsler i SQL, der involverer f.eks. gruppering, aggregering, og underforespørgsler.
- Foretage normalisering for at forbedre en relationel datamodel.
- Skabe database constraints, såsom referenceintegritet.
- Anvende indeksering til at forbedre en databases effektivitet.
- Analysere opførslen af transaktioner, der udføres parallelt.

Kursusindhold

Kursusindhold i hovedpunkter:

- Introduktion til database management.
- Database design ved brug af entity-relationship datamodellen, EER modellen, eller UML.
- Den relationelle model og normalisering.
- Programmering af databaseforespørgsler i SQL.
- Effektiv databaseimplementation.
- Transaktionshåndtering.
- OLAP.
- XML i databaser.

Kursusformat

Forelæsninger: Torsdag 17.00-ca. 19.00, Aud. 3 (fra 9/2).

Blanding af forelæsninger og korte problemsessioner uden forberedelse.

Øvelser: Torsdag 19.00-21.00

For hver forelæsning stilles en blanding af lette øvelser, der kan løses med baggrund i forelæsningen, og lidt sværere opgaver, der typisk kræver hjemmearbejde (læsning, tænkning, etc.) Det er op til **jer** hvornår I vil arbejde på hvilke opgaver. Løsninger gennemgås **ikke** i plenum.

Obligatoriske afleveringer: 3 individuelle og 3 gruppeopgaver.

Eksamen: Skriftlig, 4 timer, d. 8. juni 2006. Tidligere eksamenssæt findes på kursushjemmesiden.

Hvorfor obligatoriske (gruppe)afleveringer?

Det er en udbredt opfattelse, at obligatoriske afleveringer bare er en måde at svinge pisken, og få jer til at arbejde mere!

Men der er andre grunde:

- Basis for at give konstruktiv og omfattende **feedback** i løbet af semesteret.
- Projektarbejde om et udviklingsforløb giver erfaringer, som det er svært at teste til eksamen.
- Projektarbejde i en gruppe giver erfaringer, som det er svært at opnå på egen hånd.

Overlevelse 1: Obligatoriske afleveringer

- Afleveres som hovedregel til hjælpelæreren i forbindelse med øvelserne, eller via e-mail.
- **Alle opgaver bortset fra højst 1 individuel opgave skal godkendes for at kunne gå til eksamen.**
- Der er mulighed for 1 genaflevering af hver opgave, men kun hvis der er gjort et seriøst forsøg på at besvare den til afleveringsfristen.
- Opgaver ligger på kursushjemmesiden mindst 2 uger før fristen.
- Gruppeopgaver **skal** laves i grupper med 3 eller 4 studerende.
  (For studerende med fuldtidsarbejde kan der dispenseres for dette.)

Overlevelse 2: Mad

- Nærmeste åbne kantine er på KUA (2 min. gang fra ITU) – åbent til 19.30.
- Diverse slikautomater på ITU.
- Mad og drikke må **ikke** indtages i auditoriet
- . . . men pauserne burde række til en sandwich.

Kursusmateriale

- Kursushjemmeside: Nyheder, læseanvisninger, slides, øvelser, afleveringsopgaver.
  URL: `www.itu.dk/people/pagh/DBS06/`
- Lærebog: Modern Database Managament, 7th ed, af Hoffer, Prescott og McFadden.
- Supplerende materiale on-line på
  `www.itu.dk/people/pagh/DBS06/Intranet/`
  (kræver brugernavn og password, som I vil få tilsendt per e-mail).
- Supplerende litteratur om SQL (ikke pensum):
  SQL Clearly Explained af Jan L. Harrington.

Lærebogen sælges i Samfundslitteratur (ved siden af ITUs information).
Den supplerende bog er bestilt hjem, men der er pt. ikke eksemplarer til alle.

Om Modern Database Management

- Amerikansk bog: Mange sider, mange eksempler, en del redundans.
- Styrker: Giver et godt billede af den sammenhæng databasesystemer indgår i, med fokus på udvikling af databaser med forretningsdata.
- Svaghed: På visse steder manglende præcision og dybde omkring tekniske aspekter.

Forelæsningerne vil være fokuseret på det **databasespecifikke** materiale, og vil forsøge at supplere bogen hvor der mangler noget.

Det er altså **ikke** alt materialet fra bogen, der dækkes af forelæsningerne.

Efter pausen: Kursusoverblik

- What is a database?
- What is a database management system (DBMS)?
- What is a relational database?
- How are databases designed?
- How are databases programmed?
- . . . and other subjects of the course.

What is a database?

According to Webster's dictionary:

> **da·ta·base**
> a usually large collection of data organized especially for rapid search and retrieval (as by a computer)

Remark:
The need for (and the ability to give) rapid answers to a multitude of **queries** about data is increasing. Databases have thus grown to perform much more advanced processing than search and retrieval.

What is a database management system?

Database management system (DBMS):

Software system used when implementing databases

more precisely

System for providing **efficient**, **convenient**, and **safe** storage of and **multi-user** access to (possibly **massive**) amounts of **persistent** data.

What is a relational database?

All major general purpose DBMSs are based on the so-called **relational data model**. This means that all data is stored in a number of tables (with named columns), such as:

accountNo	*balance*	*type*
12345	1000.00	`savings`
67890	2846.92	`checking`
32178	-3210.00	`loan`
...	...	...

For historical, mathematical reasons such tables are referred to as **relations**. This course is mainly on relational databases, and on relational database management systems (RDBMSs).

How are relational databases designed?

It is often far from obvious to decide how to store data from an application as relations. A considerable part of the course will deal with a methodology for good relational database design.

Problem session: (5 minutes, discuss in groups of 2-4 students)

Suggest how to represent the following types of data as one or more relations:

- An address book (names, addresses, phone numbers).
- A phone operator's record of phone calls.

Can you avoid (or reduce) duplication of information?

Database design methodology

We will cover the dominant design methodology for relational databases, which consists of three steps:

1. Identify all relevant **E**ntities and **R**elationships, and describe them using so-called **E-R model notation**. (Lecture 3.)
2. Convert the E-R model to a number of relations. (Lecture 3.)
3. Eliminate (or reduce) redundancy by splitting relations. This process is called **normalization**. (Lecture 6.)

In real, complex systems the process is often iterated several times before a final design is reached.

How are relational databases programmed?

The success of relational databases is largely due to the existence of powerful **programming languages** for writing database queries.

The most important such language is **SQL** ("Structured Query Language", sometimes pronounced "sequel").

Important properties:

- **convenient**: queries can be written with little effort
- **efficient**: even for large data sets, a good DBMS can answer queries written in SQL quickly (compared to the fastest possible)

SQL example

Consider the following relation, which we give the name `Accounts`:

accountNo	*balance*	*type*
12345	1000.00	`savings`
67890	2846.92	`checking`
32178	-3210.00	`loan`

SQL to get the balance of account 67890:

```
SELECT balance
FROM Accounts
WHERE accountNo = 67890;
```

More SQL examples

```
SELECT accountNo, balance
FROM Accounts
WHERE type = 'loan' AND balance < -10000;
```

```
SELECT *
FROM Accounts
WHERE accountNo > balance;
```

The * is a shorthand for the list of all column names (or **attributes**).

Querying several relations, an example

Suppose we have a relation `Holders` related (!) to `Accounts`:

accountNo	*name*	*address*
12345	Scrooge	Money Tank
67890	Donald Duck	Apple Rd 13
67890	Daisy Duck	Apple Rd 13
32178	Gyro Gearloose	Inventor's lane 1

SQL to get names of all holders of checking accounts:

```
SELECT name
FROM Accounts, Holders
WHERE Accounts.accountNo = Holders.accountNo;
```

— General form of "simple" SELECT-FROM-WHERE

```
SELECT name1, name2, ...
FROM relation1, relation2, ...
WHERE <some condition>;
```

You will learn next week what happens in general when there is more than one relation involved in the SELECT-FROM-WHERE.

Problem session

Consider again the relation `Accounts`:

accountNo	*balance*	*type*
12345	1000.00	`savings`
67890	2846.92	`checking`
32178	-3210.00	`loan`

Problem session: (5 minutes, discuss in groups of 2-4 students)

Write an SQL query that finds all accounts (i.e., account number and type) that have positive balance but not more than a balance of 2000.

Syntax and semantics of SQL

As you have seen, SQL queries resemble questions in English. Often, the effect of an SQL query can easily be intuitively understood.

During the course you will learn how to program much more complex queries in SQL. To be able to do that you need a precise understanding of SQL's:

- **Syntax.** The *way* SQL may be written.
- **Semantics.** The *meaning* of an SQL statement.

Other aspects of SQL

In addition to queries, SQL can be used to express many types of database operations:

- Define new relations.
- Perform changes to data.
- Set up **constraints** and **triggers**.
- Manage users, security, rights, etc.
- Control **transactions** in a multi-user environment.
- ...

These aspects are covered later in the course.

Most important points in this lecture

As a minimum, you should after this lecture:

- Know how to qualify for the exam.
- Know a little about some key concepts: Relation, (R)DBMS, SQL queries, normalization, relational algebra, and know how they fit into this course.
- Understand how a subset of a relation `R` can be obtained using "`SELECT ... FROM R WHERE ...`".

Læsevejledning

Skemaet angiver MDM kapitel 1 og 2 som litteratur til denne forelæsning.

Disse kapitler giver et bredt billede af databasesystemers rolle i virksomheder, og af de processer, der bruges når systemer skal udvikles, altså også emner jeg ikke har berørt ved forelæsningen.

Vi er startet på materiale fra MDM kapitel 7, og resten af kapitlet gennemgås næste gang.

Næste uge

Næste uge taler Esben videre om relationer og SQL:

- Hvad er den relationelle datamodel?
- Hvad er en relation helt præcist?
- Hvad er de grundliggende måder at skrive SQL udtryk på?
- Hvordan opdaterer man data i en relation?
- Hvordan skaber man nye relationer i SQL?