Database Systems, Fall 2006
IT University of Copenhagen

Lecture 1: Introduction

August 29, 2006

Lecturer: Rasmus Pagh

About your teachers

Rasmus Pagh (lecturer):

- PhD in computer science, Aarhus University, 2002.
- Research in algorithms, among other things into questions on efficient implementation of DBMSs, the software behind databases.
- Has taught database courses at ITU since spring 2003, on the courses *Introduction to databases*, *Advanced database technology*, and *Databasesystemer*.

Esben Rune Hansen (lecturer):

- Cand. it. and PhD student at ITU.
- Teaches Database Systems for the second time.

About your teachers, cont.

Teaching assistants:

- Martin Scheil Corneliussen (PhD student at ITU).
- Knut Tveitane (SWU student at ITU).

Today's lecture (2-3 hours)

- Why study databases?
- What you will get from the course.
- Practical information.
- Introduction to the course material.
- Using ITUs Oracle DBMS.

Why study databases?

Academic's reason:

Databases touch upon many interesting topics in computer science.

Programmer's reason:

Need to use databases when programming applications.

Information pilot's reason:

Want to work with and extract information from large, changing data sets.

Capitalist's reason:

Everybody needs databases, so there is a lot of money to be made.

Goals of course

The overall goal of the course is to provide you with a background for designing and implementing databases based on the relational model (the dominant database paradigm). In particular, after the course you should be able to:

- make a database design in the entity-relationship model or UML and convert it to the relational model
- use theoretical tools such as normalization and relational algebra to improve database design and implementation
- create database constraints such as referential integrity
- program complex queries in SQL
- apply the basic ways of improving database efficiency
- analyze the behavior of concurrent transactions

Knowledge-understanding-skill

To study in the most fruitful way, you should be aware that you should obtain three things during the course (and any other university level course):

- **Knowledge** is what you obtain by reading the book or attending lectures. Concrete facts, terms, methods and theories.
- **Understanding** is mainly obtained by actively using your knowledge - e.g. in practical or theoretical exercises. This is where your TA can help.
- **Skills** are obtained by actively using knowledge and understanding in many contexts. Becoming skilled is simply hard work!

- Skill (and understanding) is the focus of the exam.

Knowledge-understanding-skill 2

In a nutshell:

- **Knowledge** is what will get you a job.
- **Understanding** is what will allow you to keep it.
- **Skill** is what will give you a promotion :-)

How-to?

- To know that you have progressed from knowledge to understanding and skill, you need **feedback** from your teachers and fellow students.
- The exercises and hand-ins are chances to get feedback, and the best way of checking that you really understood each subject.

Course format

Lectures and problem sessions: (Tuesdays 17.00-19.00, Aud. 4)

Mix of lectures and short problem sessions without preparation.

I do not assume that you have read the material in advance.

Exercise sessions: (Tuesdays 19.00-21.00, rooms 4A 56 and 4A 58.)

Exercises will be based on the material of the lecture on the same day.

You will probably need to work on the exercises even after the session.

Hand-ins: **Approval of mandatory hand-ins required to enter exam.**
There will be a group project consisting of 3 mandatory hand-ins.
Additionally, two of three individual hand-ins must be approved.

Exam: Written, 4 hours, January 2007. You can find previous exams for (very) similar courses on the course home page.

Why mandatory (group) hand-ins?

Many students think that mandatory hand-ins is just a way of forcing them to work harder.

However, there are other reasons:

- Basis for constructive and comprehensive **feedback** during the semester.
- Project work on a development project gives experience that is difficult to test at an exam.
- Working in a group gives experience that is hard to obtain when working on your own.

Group project

The group project will start in two weeks, after you have handed in your first individual hand-in. It consists of a small database development project.

Groups will consist of 3-4 students. Formation of groups will be coordinated by Martin Scheil Corneliussen (mscheil@itu.dk). Contact Martin if:

- You have wishes about who should be in your group.
- You have special constraints, e.g. due to full time job or family.
- You decide not to take the course anyway.
- You already qualified for the exam in “Databasesystemer”.

We will then try to form the groups according to your wishes.

Course material

- Main course textbook: Database Management Systems, 3rd edition, ISBN 0071230572.
- Two other good textbooks (not curriculum for the course) are mentioned in the course description. They are relevant for students especially interested in SQL programming and business use of database systems, respectively.
- Supplementary material will be made available.

The books can be bought at Samfundslitteratur (next to the ITU information desk).

The course homepage

`www.itu.dk/people/pagh/DBSE06/`

Contains:

- News.
- Reading directions for each lecture.
- Lecture slides.
- Problems for hand-ins and exercises.
- Previous exams.
- **Useful links to on-line resources.**

After the break: Course overview

- 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.

Brainstorm session:

Think of examples of databases where each of the words in **bold** are important.

What is a relational database?

All major general purpose DBMSs are based on the so-called **relational data model**. This means that **conceptually** 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 **solely** on relational databases, and on relational database management systems (RDBMSs).

If you want to learn more

If you want to know more about database technology, take the course **Database Tuning** in the spring semester.

- Previous editions of this advanced database course covered e.g. text indexes, geographical information systems, and data mining...
- ...in addition to lots of material on relational databases.
- Remember to first study **Performance and Test** (at least the “performance” part), or similar.

How are relational databases designed?

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

Problem session: (10 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.
- 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 2.)
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 8.)

— 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;
```

Even more SQL

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>;
```

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

Later in the course we will describe in detail what is computed 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.

“Joining” information in two relations

SQL’s `NATURAL JOIN` operator combines information from two relations, by merging tuples that agree on the common attributes.

Example:

cpr	*street*	*city*
300266-3278	Louis Ln 1	Louisiana
300266-3278	Blixen Park 4	Ørestad
310671-2343	Glentevej 67	København
311177-2342	Grønager 222	Vejle

cpr	*course*	*grade*
300266-3278	DBS	9
300266-3278	ADBT	8
310671-2343	OOP	10
310671-2343	IPBR	9
310671-2343	IDBI	6
311177-2342	IPBR	7

The natural join of these relations results in a new relation with 8 tuples, where each tuple contains cpr number, an address, and a grade of a student.

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.

Theoretical basis of SQL

SQL is based on a mathematical formalism called **relational algebra**.

Lecture 4 is devoted to relational algebra and its relation[a] to SQL.

Knowledge of relational algebra allows formal reasoning about database queries – in particular how to correctly **rewrite** queries.

Perhaps more importantly, it gives an alternative, supplementary understanding of SQL.

[a] In the usual sense of the word.

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**.
- Control **transactions** in a multi-user environment.
- Setting up indexes to improve performance.
- Manage users, security, rights, etc.
- . . .

These aspects are expected to be covered in lectures 3, 5, 8, 9, and 12.

The RDBMS used in the course

You will be working with an RDBMS from the vendor Oracle.

To create your Oracle work space, use the setup at `itu.dk/sysadm/db/`

To log into Oracle:

- Log onto `ssh.itu.dk` using an SSH client.
- Type: `sqlplus your_oracle_user_name@studora`

During the first step you will be prompted for mail user name and mail password. In the second step you will be prompted for Oracle password.

When working on the command prompt, it is a good idea to use a text editing program on the side for writing your queries.

At ITU you may also use a graphical database interface called Rational Data Architect. (Recommendation: Start with SQL*Plus.)

Most important points in this lecture

As a minimum, you should after this lecture:

- Know the significance of knowledge, understanding, and skill.
- 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 ...`".
- Be able to start working with Oracle (see also today's exercise sheet).