Introduction to Databases, Fall 2003
IT University of Copenhagen

Lecture 1: Introduction

August 26, 2003

Lecturer: Rasmus Pagh

Today's lecture (2-3 hours)

- Why study databases?
- What you will get from the course.
- Practical information.
- Introduction to the course material.
- System demonstration.

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.

Why study databases?

Eternity student's reason:

Need last 7,5 ECTS. . .

Wrong course!

You will be expected to work hard and independently to pass this course.

But when you go for a job interview it may prove worth it!

What you will get from the course

A firm background in database implementation that makes you able to:

- Design and implement moderate size relational databases:
  - Do data modeling and query programming (SQL).
  - Use theoretical tools to improve design and implementation.
- Use the basic ways of improving database efficiency.
- Reason about basic database system concepts (transactions, concurrency, error recovery, information integration).

What you will get from the course

Practical experience with implementing databases and the connection to the **theoretical foundation** is emphasized.

The course is primarily aimed at those who see themselves actively using database systems in the future.

Database course overview

Database Systems (in Danish, fall 2003)

Course book: *Data Management: Databases and Organizations*.

Assumes: Introductory programming.

Introduction to Databases (in English, fall 2003)

Course book: *Database systems – the complete book* (GUW).

Assumes: Curiosity.

Advanced Database Technology (in English, spring 2004)

Course book: *Database systems – the complete book*.

Assumes: Introduction to algorithms and a database course.

Course format

Lectures and problem sessions: (Tuesdays 12.30 to around 15.30-16.00)

Mix of lectures and short problem sessions without preparation.

Exercises: (10.00-12.00 for day students, 16.15-18.15 for open education)

You are expected to prepare before the exercises, and continue to work on problems in class with teaching assistant present.

Hand-ins: **Satisfactory hand-ins required to enter exam.**

Exam: Written, 4 hours, on January 20, 2004.

About the mandatory hand-ins

There will be hand-ins due at noon on most Fridays, starting two weeks from now.

Hand-ins must be completed **individually**. You are allowed to discuss hand-ins with fellow students, but you must understand and prepare your own solution.

If you feel tempted to search the Internet for a solution or “inspiration”, note that you may very well be cheating yourself: **Working independently on hand-ins and other problems is a very important way of learning (more important than attending lectures).**
Also, this is what will prepare you for a successful exam.

Manning of course

Lecturer:

Rasmus Pagh, `pagh@itu.dk`, office 1.23.

Teaching assistants:

Lars Bengtsson (`larsb@itu.dk`)
Tøger G. Nørgaard (`tgn@ostervold.dk`)

Available during exercises, and Thursdays before hand-ins 10-11 AM.

The course homepage

`www.itu.dk/people/pagh/IDB03/`

Contains:

- News (will also be posted on the news group).
- Reading directions for each lecture.
- Lecture slides.
- Problems for hand-ins and exercises.
- Useful links to on-line resources.
- The IDB dictionary.

Bookmark it now!

The IDB dictionary

Background:

- All major database textbooks with technical emphasis are aimed at computer science or electrical engineering students.
- In CS and EE programmes, database courses are typically placed after the first year.
- Therefore GUW may use terms that you do not know, because (most of) you don't have CS or EE background.

Please submit terms you wish to have included to `pagh@itu.dk`.
(Remember to state where you saw the term used.)

The course news group

`itu.courses.idbi`

You may use the newsgroup for any business related to the course, e.g.:

- questions (and answers)
- tips for other students
- organizing a study group
- etc.

The course teachers will try to read the newsgroup regularly.

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.

Problem session: (10 minutes, discuss in groups of 2-4 students)

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 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 about some important database applications using special purpose software, take the course **Advanced Database Technology** in the spring semester.

- Last time it covered e.g. text indices, geographical information systems, and data streams. . .
- . . . in addition to lots of material on relational databases.
- Remember to first study **Introduction to Algorithms**!

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 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 4.)

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;
```

During the course you will learn how to program much more complex queries in SQL.

— 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**).

You will learn next week what happens 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.

Theoretical basis of SQL

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

Lecture 7 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.

Rewriting may result in a gain in simplicity or efficiency.

[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**.
- Manage users, security, rights, etc.
- Control **transactions** in a multi-user environment.
- ...

These aspects are expected to be covered in lectures 2, 6, and 8.

Other subjects treated in the course

Towards the end of the course we will consider:

- Data warehousing and decision support (pending guest lecturer).
- Database efficiency.
- Some commercial database management systems.

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 ...`".
- Be able to start working with MySQL.