Introduction to Databases, Fall 2004

IT University of Copenhagen

Lecture 1: Introduction

August 27, 2004

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.
- Using the ITU 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.

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.

Course format

Lectures and problem sessions: (Fridays 9.00-12.00, room 2A 20)

Mix of lectures and short problem sessions without preparation.

Exercises: (Fridays 13.00-15.00, room 2A 52)

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

Hand-ins: **Satisfactory hand-ins required to enter exam.**
First one due 3 weeks from now. Hand-ins will appear on the home page. (Note: Previously, there has been a clear correlation between students working well on hand-ins and students passing the exam.)

Exam: Written, 4 hours, on January 7, 2004. (Last year's exam available

Course material

- Course book: *Database systems – the complete book* (GUW).
- Compendium for IDBI 2004.

Both (soon) for sale at Samfundslitteratur (next to the information desk).

The course homepage

www.itu.dk/people/pagh/IDB04/

Contains:

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

Bookmark it now!

The course news group

`it-c.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.

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 indexes, geographical information systems, and data mining...
- ...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;
```

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**).

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

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.

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.
- Database efficiency.
- Some commercial database management systems.
- Databases in real life (guest lecturer from Oracle).

The RDBMS used in the course

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

To create your Oracle work space, use the setup at itu.dk/sysadm/db/ (should work shortly).

To log into Oracle type (at a command prompt):

- `ssh ssh.itu.dk`
- sqlplus

After the first line you will be prompted for mail user name and mail password. After the second line you will be prompted for Oracle user name and password.

The ITU system administrators are working hard to make a graphical user interface to Oracle available on student machines asap.

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 Oracle.

Next lecture

The next lecture is **two weeks** from now, due to the official opening of ITU next Friday.

This means you have lots of time to get started reading GUW!

You can see the material I plan to cover in each lecture at the course home page. There may be minor changes, and there will be additional material for some lectures.