Introduction to Databases, Fall 2005
IT University of Copenhagen

Lecture 1: Introduction

August 29, 2005

Lecturer: Rasmus Pagh

About your teachers

Rasmus Pagh (lecturer):

- PhD in computer science, Aarhus University, Denmark, 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*.

Teaching assistants:

- Priya Seetharaman (INT student at ITU since 2003).
- Simer Sawhney (INT student at ITU since 2004).

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 goal is that you should obtain a firm background in database implementation. After the course, you should be able to design and implement moderate size relational databases. In particular:

- make a database design in the E/R model and convert it to the relational model
- program complex queries in SQL
- use theoretical tools (normalization and relational algebra) to improve database design and implementation
- create database constraints such as referential integrity
- 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.
- Too many students at past exams have lacked, not in knowledge, but in understanding and skill, and gotten a poorer grade than they ought to.

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, hand-ins, and the Gradiance system is your chance to get feedback, and the best way of checking that you really understood each subject.

Course format

Lectures and problem sessions: (Mondays 13.30-16.00, Aud. 4)

Mix of lectures and short problem sessions without preparation.

Exercise sessions: (Mondays 16.00-18.30, room 4A 56 and 4A 58.)

Exercises will be based on the material of the lecture on the same day. You will probably need to keep working on the exercises even after the session to get through them all.

Hand-ins: **Approval of mandatory hand-ins required to enter exam.**
There will be a group project consisting of 3 mandatory hand-ins.
Additionally there will be 3 individual hand-ins, at least one of which must be approved.

Exam: Written, 4 hours, on January 16, 2006.

Group project

The group project will start in two weeks, after you have handed in your first individual hand-in.

Groups will consist of 3-4 students, and will be formed at random (i.e., you do not choose your group).

If you are *not a full-time student*, and have problems participating in group work (e.g. due to your job), please inform me as soon as possible, and you will be given an alternative possibility.

Course material

- Course book: *Database systems – the complete book* (GUW).
- Supplementary material will be made available.

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

The course homepage

`www.itu.dk/people/pagh/IDB05/`

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.

Problem session: (5 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.

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.

“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 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.

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**.
- 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

During the course we will also consider:

- OLAP (data analysis queries).
- Database efficiency.
- Some commercial database management systems.

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/

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

- `ssh ssh.itu.dk`
- `sqlplus oracle_user_name@studora`

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 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 interface to Oracle called TOra.

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 exercise sheet for next week).