Advanced Database Technology

Rasmus Pagh and Srinivasa Rao

IT University of Copenhagen

Spring 2006

Overview of lectures

January 30, 2006

Warm-up problem session

Pretend you are a DBMS. . .

Lecture 2: Data storage devices. I/O model.

When dealing with large data sets one needs storage outside of the computer's main memory, e.g.:

- Disks.
- Tapes.
- The memory of other computers.

The access time for a given byte on such devices is *thousands or millions* of times larger than retrieving a byte in main memory.

Lecture 2 concerns the characteristics of external memory devices. Such an understanding is necessary to devise efficient external memory algorithms.

Lecture 2, cont.

Block transfers:

Because of the large access time, every external memory access is used to transfer a whole *block* of adjacent data.

- E.g., disk blocks are typically 4-16 kilobytes.

Question: Why do block transfers help?

A particularly simple model of external memory is the *I/O model*, which will be used for most of the material in the course.

As a primer, we consider efficient *sorting* in the I/O model.

Lecture 3: Representing data elements.

In lecture 3 we ask: How does one store relations in a blocked memory?

title	*year*	*length*	*filmType*
Star Wars	1977	124	color
Mighty Ducks	1991	104	color
Wayne's World	1992	95	color

Schema:Movie; Star Wars; 1977; 124; color; Mighty Ducks; 1991; 104; color

Schema:Movie; Wayne's World; 1992; 95; color; . . .

Problems with updates:

What if we want to add "Episode 4" to "Star Wars" but there is not sufficient space in the block?

Lecture 3 and 4: Index structures.

Relational operations can sometimes be computed much faster if we have precomputed a suitable data structure.

Most notably, two kinds of so-called *index structures* are essential to database performance:

- B-trees.
- External hash tables.

For example, hash tables may speed up relational operations that involve finding all occurrences in a relation of a particular value.

Lectures 3 and 4 concern *how* such index structures work, *why* they are useful, and what the *price* paid is.

Lectures 5 & 6: Impl. of relational operations.

We consider the question: Given a query in e.g. SQL, how can the result be efficiently computed in the database?

One problem in answering this is that the best way of doing things may depend crucially on the data in the relations.

Example:

ID	*Name*
1	P. Persson
2	J. Jensen
3	P. Schlüter
...	...
1447	S. Isted

$\bowtie$

ID	*HSAS*	*Mark*
1	ADBT	n/a
1	DSK	7
2	WEB	10
...	...	...
1447	OOP	n/a

$\bowtie$

ID	*Project*	*Done*
211	XML	n/a
347	XML	n/a
790	XML	n/a

Lecture 7: Concurrency control.

Transactions with the ideal ACID properties resolve the semantic problems that arise when many concurrent users access and change the same database.

In practice, transactions may be at different *isolation levels*, that approximate the ACID properties to different degrees.

This lecture is about how transactions are implemented using locking mechanisms, or "optimistic" approaches.

This knowledge is useful in database programming, e.g., it makes it possible in some cases to avoid (or reduce) rollbacks of transactions, and generally make transactions wait less for each other.

Lecture 8: Geometric index structures.

Many large data sets are geometric in nature, e.g., satellite images and map data (road networks, altitude maps, etc.)

Examples of geometric queries:

- Show a specific part of a map.
- Show the 10 gas stations nearest to your location.
- Show the shortest path to some destination.

This lecture will introduce some of the problems in geometric databases, and some of the most important tools for dealing with such problems.

Lecture 9: Text indexing.

Searching in large amounts of text has gained importance due to the success of the World Wide Web.

For example, the search engine Google facilitates rapid keyword searches in over 8 billion web pages.

- Possible only due to a huge precomputed data structure that is regenerated a few times a year.

This lecture introduces basic tools in (large scale) text indexing, in particular *suffix trees* and *suffix arrays* (on secondary memory).

Lecture 10: Data mining.

Data mining means that one tries to find interesting **patterns** in large amounts of data, e.g., from a database. The idea is to find these patterns **without knowing where to look** for them.

Example: When the ITU board want to find out why some students drop out, and why others stay and get their degree, they may ask the database about patterns in age, gender, etc.

The data mining way of doing it is to ask "What influences drop out?" without specifying what attributes to look at. Maybe the zip code together with first semester courses is the important factor for drop out, but no one would ever suggest to look at this.

We will look at some interesting algorithmic problems that arise in connection with data mining: Finding association rules, and building classifiers.

— Lecture 11: System failures and media failures.

Things go bad. . .

. . . and we want to be able to recover.

The recovery system of a DBMS may also affect performance. Thus, it is important to understand it also from this point of view.

How do we cope with system failures, e.g.:

- disk crash in the middle of a transaction?
- power failure during recovery?

Lecture 12: ITU research in databases

An overview of some recent database related research at ITU, including possible topics for thesis work.

Finally, we will go through some previous exam problems.