Tore Risch
Uppsala University, Sweden

UDBL

Topics in basic DBMS course

Database design

Transaction processing

Relational query languages (SQL), calculus, and algebra

DBMS APIs

Database tuning (physical database design)

Basic query processing (ch 9)

Object-relational databases and query language (Amos II)

Multi-media database overview

http://user.it.uu.se/~udbl/dbt-ht2004/

Tore Risch
Uppsala University, Sweden

UDBL

Modern DBMS research areas

Query processing is a *central* database research area:

How to find correct result fast from *large* database

Queries over *new scalable storage types*:

Text, XML, images, voice, music, …

Indexing new storage types.

New DBMS *architectures*

Wrapper/mediators, P2P databases, main memory, real time,..

New *query optimization methods*

Adaptive, new heuristics, rule based, …

Tore Risch
Uppsala University, Sweden

UDBL

Modern DBMS research areas

Representation and search of *spatial data*

- Representation of points, lines, surfaces, bitmaps, maps, etc.
- Find *similar* or *close* objects, regions, etc.
- *Spatial* indexes, indexing *mobile* data.

Representation and search of *voice*, *video*, *music*, and other *multi-media* data

- Representation of very large objects
- Streamed (real-time) retrieval, QoS
- Searching for sections, scenes, patterns, similarities, etc.

Tore Risch
Uppsala University, Sweden

UDBL

Modern DBMS research areas

Stream databases

- Queries over indefinite *stream* of data, not disk tables
- *Continous* rather than passive queries
- *Data reduction* queries yield new smaller streams
- *Combine* with passive data.

Representation and search of *temporal data*

- Time stamping of all data
- Queries over time, trends, etc.
- Temporal indexing

Representation and search of *ordered data*,

- e.g. *sequences* and *arrays*.

Tore Risch
Uppsala University, Sweden

UDBL

Modern DBMS research areas

Representation and search of *unstructured textual data*

- Free text search/indexing in database server (e.g. Oracle)
- Search text similar to other text (c.f. Google)
- Mixing structured and free text search

Representation and search of *semistructured data*

- Usually XML structures
- Tree structures, some structure known
- *Path expressions* combined with queries

Tore Risch
Uppsala University, Sweden

UDBL

Modern DBMS research areas

Data warehouses, large relational databases for *enterprise data storage and analysis*

Very large relational databases

OLAP, Advanced queries, data cubes, statistics, trends

Database support for statistical analysis of data from large databases, *data mining*

Data mining is large scale statistical inference!

E.g. discovery of clusters, trends, correlations, etc.

Data mining utilizes databases and data warehouses

DBMS requirements: Performance of stat. extractions, statistically sound query results

Tore Risch
Uppsala University, Sweden

UDBL

Modern DBMS research areas

High *performance* and *availability* DBMSs based on *distributed* and *parallel computations*

Very high performance, reliability, scalability

Use of modern hardware,

e.g. large main memories, fast communication, P2P

New kinds of storage structures:

SDDSs, *Scalable Distributed Data Structures*

LH – LH*

Tore Risch
Uppsala University, Sweden

UDBL

Course topics

Object-Relational query optimization (this lecture)

Mediator/wrapper approach (heterogeneous databases)

Lecture.

Temporal Databases

Real-time Databases

Stream Databases

Scalable distributed data structures (LH*)

Semi-structured databases (XML)

Uncertainty in databases

Parallel and distributed databases

Meta-optimization

Spatial databases

SQL

Parser

Relational calculus (variant of predicate calculus)

Rewriter

Relational calculus

Cost-based optimizer

Extended relational algebra (functional program)

Interpreter

The Query Processing problem

Transform:

High-Level Declarative Query --> Low-Level *Execution Plan*

Normally*:*

Relational Calculus --> Annotated Physical Relational Algebra

The execution plan is a (functional) program which is interpreted by the *evaluation engine* to produce the query result

Problem: For every query there may be very many possible execution plans:
$O(2^{|Q|})$ where |Q| is number of operations in query

The Query Processing problem

The optimal plan can be millions of times faster than an unoptimized plan!

Why? The complexity of optimal plan improved automatically,
e.g. index used instead of linear search of database.

```
select name from person where ssn=123456
select ssn from person where name like 'To%'
```

E.g from $O(N^2)$ to $O(1)$, where N is size of database!
Query optimization may have huge payoff!

However: Query optimization time may be significant!

Cost-based query optimization

1. Generate *all likely* execution plans (heuristics to avoid some unlikely ones)

2. Estimate the *cost* of executing each of the generated plans

3. Choose the *cheapest* one

The cost depends on *amount of data* processed (disk blocks accessed).

-> DBMS maintains *statistical model* of data distribution in tables.

E.g. select ssn from person where name > ‘M’

Optimization criteria:

a. # of *disk blocks* read (dominates)

b. *CPU* usage

c. *Communication* time

Normally *weighted average* of different criteria.

Cost depends on *query execution strategy*, *storage methods*, and *indexing* used

Optimizing the optimizer (meta-optimization):

Naïve approach (trying all execution orders and indexes): O(|Q|!)
Dynamic programming O($|Q|^2$) – O($3^{|Q|}$) generates optimal plan.
Normally used.
Hillclimbing O($|Q|^2$) may generate suboptimal plans.
Randomized methods O($|Q|^2$) converge to optimal plan.