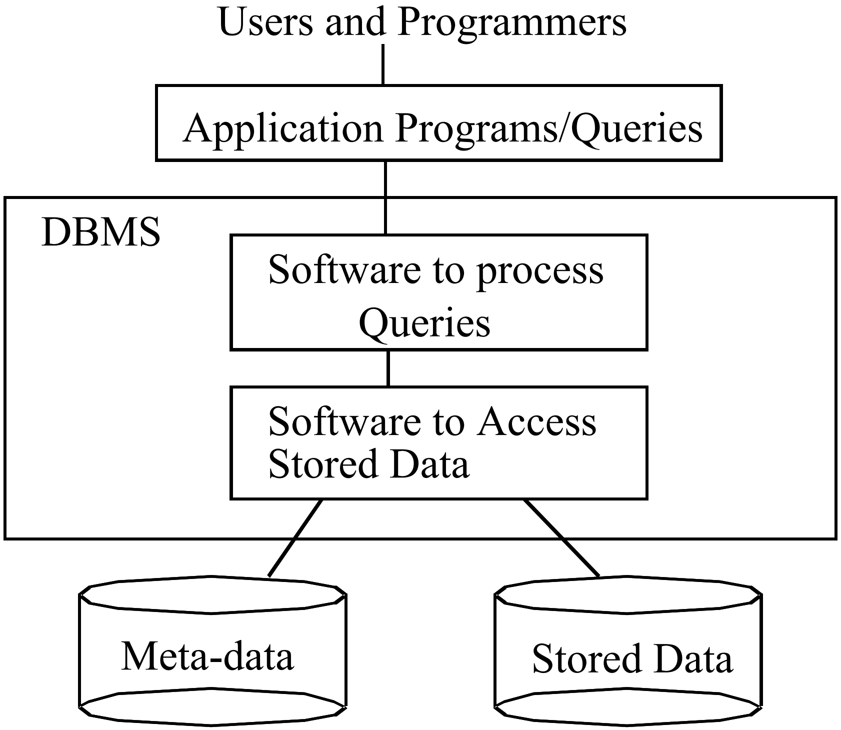
Principles of modern database systems

Doctoral student course 2007

Tore Risch
Dept. of information technology
Uppsala University
Sweden

Tore Risch
Uppsala University, Sweden

- What is a *database*?
  A database is a collection of related data stored in a computer managed by a DataBase Management System (DBMS)
- What is a *DBMS*?
  A DBMS is a collection of programs for creating, searching, updating and maintaining large databases



Users and Programmers

Application Programs/Queries

DBMS

Software to process Queries

Software to Access Stored Data

Meta-data

Stored Data

Database Management Systems

- DBMS:
  Software to manage *large volumes* of data

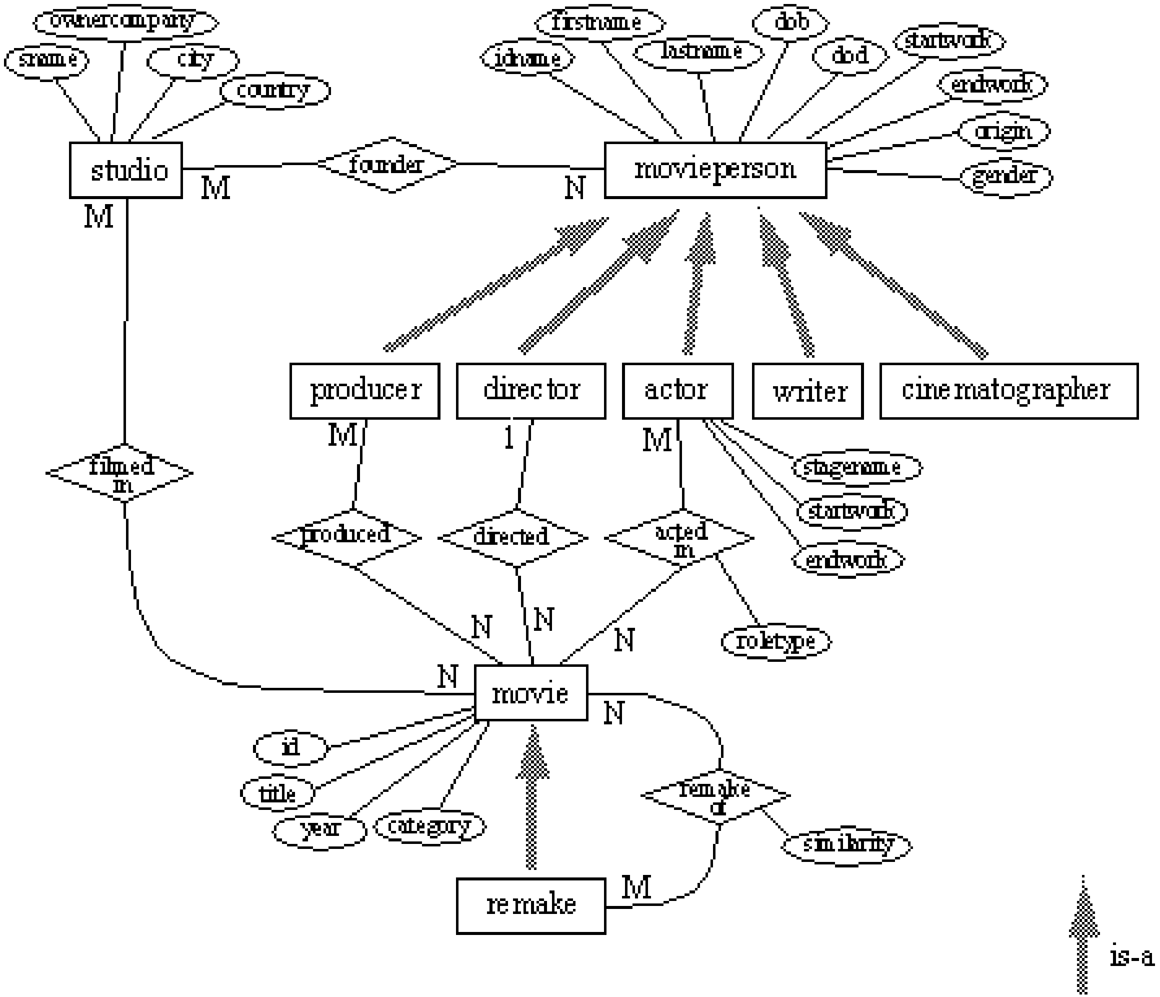
- DBMS very useful in many applications, *including scientific work of all kinds!*

- Enabling technologies:
- Efficient *search* and *update* of large datasets
- Security, authorization, integrity
- Many different data representations
  (tables, statistics, arrays, XML, text, time series, images)

Database Design

- Designing meta-data
  understanding data
  selecting relevant data
  designing database schema
  adaptation to database schemas
- Documentation
- Availability strategy
- Privacy and security strategy
- Archiving strategy

Extended ER schema



Database Design

Logical Database Design:

How to translate a schema in the conceptual data model (e.g. extended ER-schemas) to a schema in the DBMS data model (e.g relational tables)

PROBLEM:

emantics may disappear or be blurred when data is translated rom extended ER-model to less expressive relational data model

Database Design

- *Physical Database Design*:

  E.g by indexes:

  - permit fast matching of records in table satisfying certain search conditions.

PROBLEM:

New applications may require data and index structures that are not supported by the DBMS.

E.g. calendars, numerical arrays, geographical data, images, text, voice, etc.

Database Manipulation

- Typical query language *operations* are:
  - *Searching* for records fulfilling certain selection conditions
  - Iterating over entire tables applying *update operations*

PROBLEM: Would like to be able to customize and extend query language for different application areas, maps, time series, images, etc.

Database Manipulation

•*Query language*:

Originally a QL could only specify database searches.

Now the standard query language SQL is a *general* language fo
interactions with the database.

•Typical query language *operations* are:

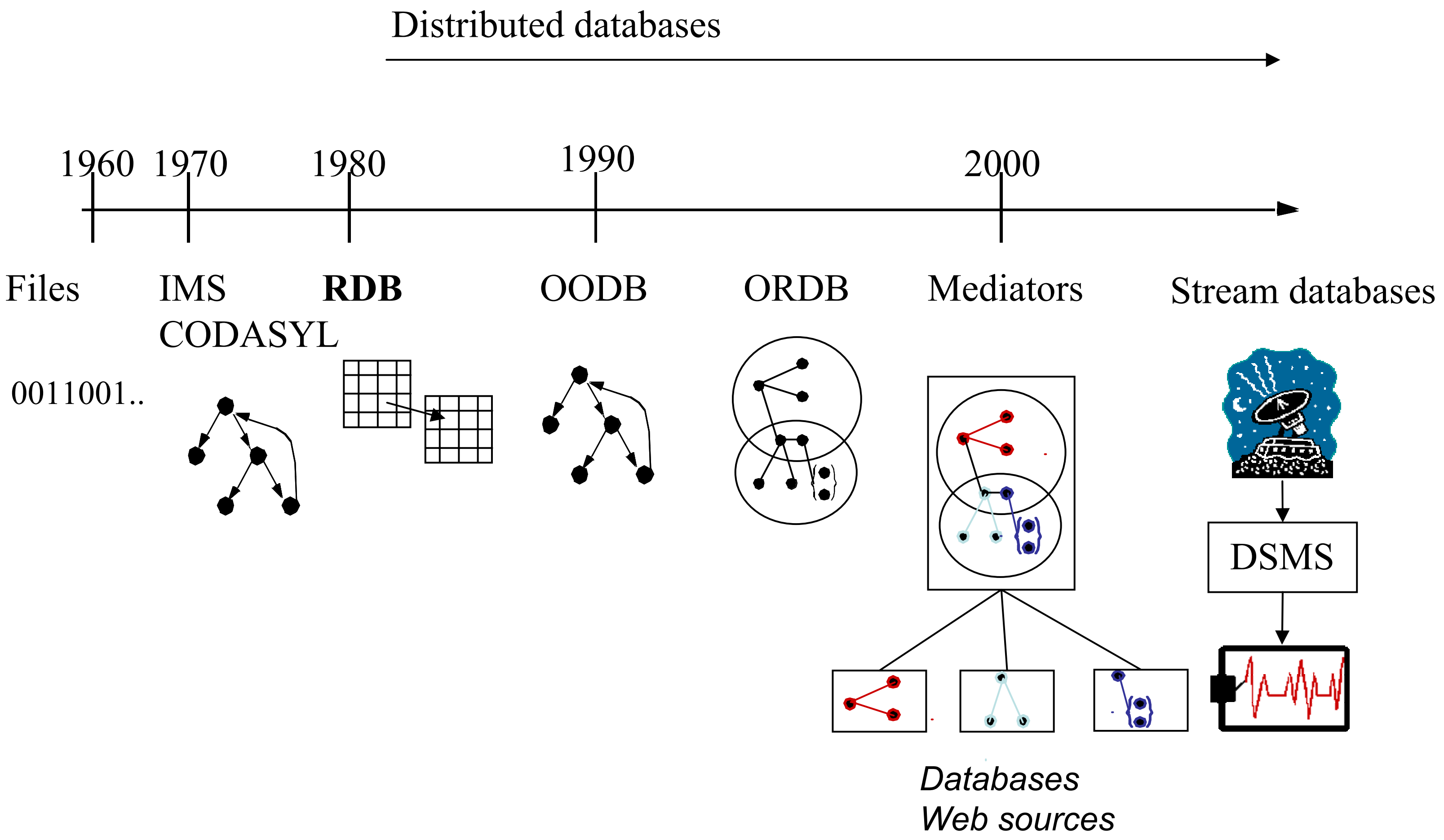
- *Searching* for records fulfilling certain selection conditions
- Iterating over entire tables applying *update operations*
- *Schema definition* and *evolution* operators
- Object-Oriented Databases have create and delete *objects*

PROBLEM: Would like to be able to customize and extend query language for different application areas

E.g. temporal, numerical, image queries

Tore Risch
Uppsala University, Sweden

Evolution of DBMS technology



Distributed databases

1960 1970 1980 1990 2000

Files

0011001..

IMS
CODASYL

RDB

OODB

ORDB

Mediators

Databases
Web sources

Stream databases

DSMS

Topics in basic DBMS course

Database design, logical and physical

Relational query languages, *SQL*, calculus, and algebra

Transaction processing

DBMS APIs

Basic query processing

Object-relational databases and query language (Amos II)

Data warehouses: Large relational databases for decision support, e.g. advanced queries, *statistic* *spreadsheets*, *trends*, *OLAP*

Modern DBMS research areas

Query processing (fast search) is a *central* database research area:

How to find correct result *fast* from *large* database

New kinds of data to search among:

Not only *tables*

temporal data, representation of *time* in databases

unstructured data, free *text*, *documents, HTML, bitmaps*

semistructured data, *XML*, *RDF*

sequence data, e.g *XML, arrays, time sequences, streams*

spatial data, e.g. *points*, *lines*, *surfaces*, *maps*, etc.

multi-media data, search of *voice*, *video*, *music*

Modern DBMS research areas

Representation and search of *unstructured* textual data

- Free text *indexing* in database server (e.g. Oracle)
- Search text *similar* to other text (c.f. Google)

```
select x.name,s
from Documents x, myDocument y
where s = similarity(x,y) and s >0.9
     and x.name like ‘%database%’
order by s
stop after 10
```

- Mixing *structured* and *free* text search
- Find *similar* or *close* sentences or words

Modern DBMS research areas

Representation and search of semistructured data

Usually XML structures

Tree structures, some structure known

Path expressions (XPath) combined with queries (XQuery)

Searching *multi-media* data

Representation of *very large objects*

Streamed (real-time) retrieval, *QoS*

Searching for *sections, scenes, patterns, similarities, etc.*

Modern DBMS research areas

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*, *text*, A *follows* B, A *contains* B

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.

Tore Risch
Uppsala University, Sweden

Course topics

Database technology evolution (this lecture)

Extensible query optimization (this lecture)

Mediator/wrapper approach (heterogeneous databases)

Querying heterogeneous databases.

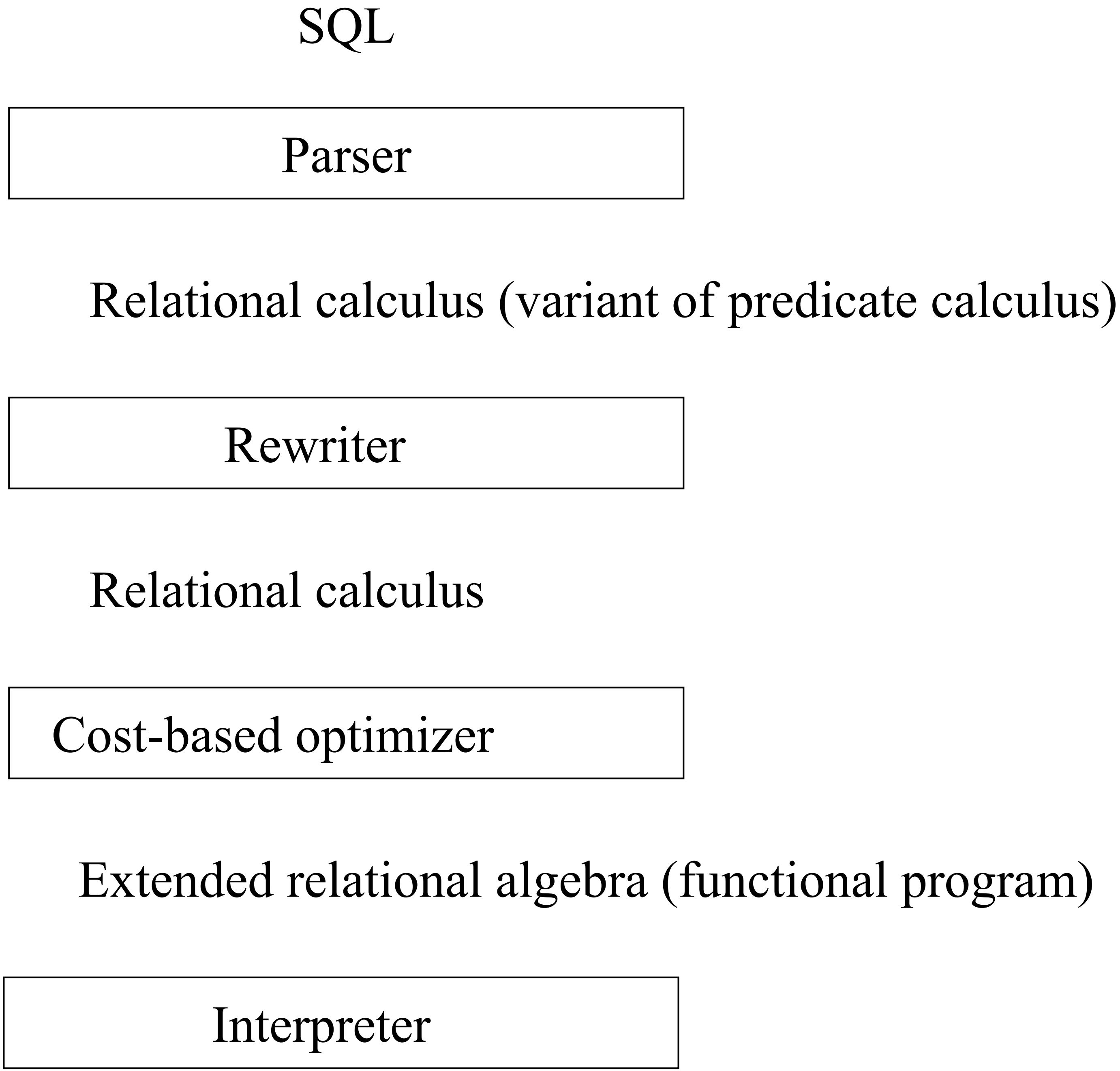
Data Stream Management Systems

Semi-structured databases (XML, RDF)

Modern parallel and distributed databases

Support for numerical data

Multi-media 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 executio 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