Introduction to Database Systems

Module 1, Lecture 1

M. Valenta - KSI CTU FIT in Prague
Michal.Valenta@fit.cvut.cz

Based on slides:

R. Ramakrishnan (raghu@cs.wisc.edu)

Materials

Web site:

https://edux.fit.cvut.cz/courses/BIE-DBS/

there: lectures, seminar, materials

Materials: slides + recommended books

What Is a DB?

- A *database* is a very large, integrated collection of data.
- Models real-world *enterprise.*
  - Entities (e.g., students, courses)
  - Relationships (e.g., Madonna is taking CS564)
- A *Database Management System (DBMS)* is a software package designed to store and manage databases.

Why Use a DBMS?

- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.

Why Study Databases??

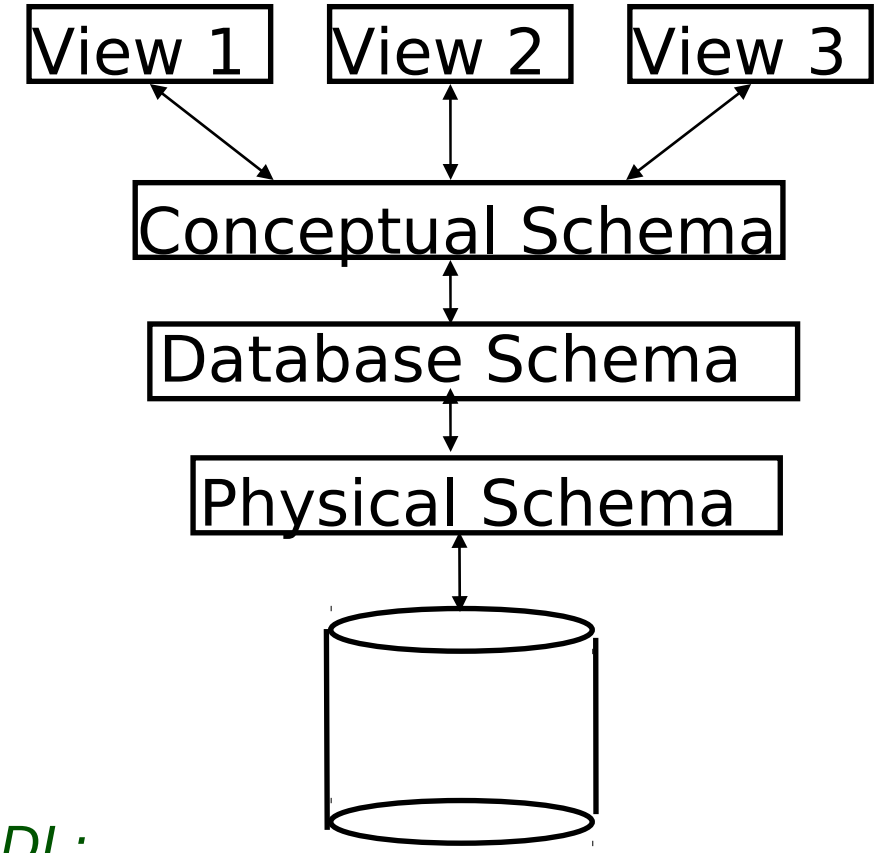
- Shift from *computation* to *information*
  - at the “low end”: scramble to webspace (a mess!)
  - at the “high end”: scientific applications
- Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, EOS project
  - ... need for DBMS exploding
- DBMS encompasses most of CS
  - OS, languages, theory, “AI ”, multimedia, logic

Data Models

- A *data model* is a collection of concepts for describing data.
- A *schema* is a description of a particular collection of data, using the a given data model.
- The *relational model of data* is the most widely used model today.
  - Main concept: *relation*, basically a table with rows and columns.
  - Every relation has a *schema*, which describes the columns, or attributes.

Levels of Abstraction

- Many *views*, single *conceptual schema, database (logical)* and *physical schema*.
  - Views describe how users see the real world and/or data,
  - Conceptual schema defines abstract objects of real world
  - Database schema defines logical structure
  - Physical schema describes the files and indexes used.



- *Schemas are defined using DDL;*
- *data is modified/queried using DML*.

Example: University Database

- Database schema:
  - Students(sid: string, name: string, login: string, age: integer, gpa:real)
  - Courses(cid: string, cname:string, credits:integer)
  - Enrolled(sid:string, cid:string, grade:string)
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View):
  - Course_info(cid:string,enrolment:integer)

Data Independence

- Applications insulated from how data is structured and stored.
- *Logical data independence*: Protection from changes in *logical* structure of data.
- *Physical data independence*: Protection from changes in *physical* structure of data.

- *One of the most important benefits of using a DBMS!*

Concurrency Control

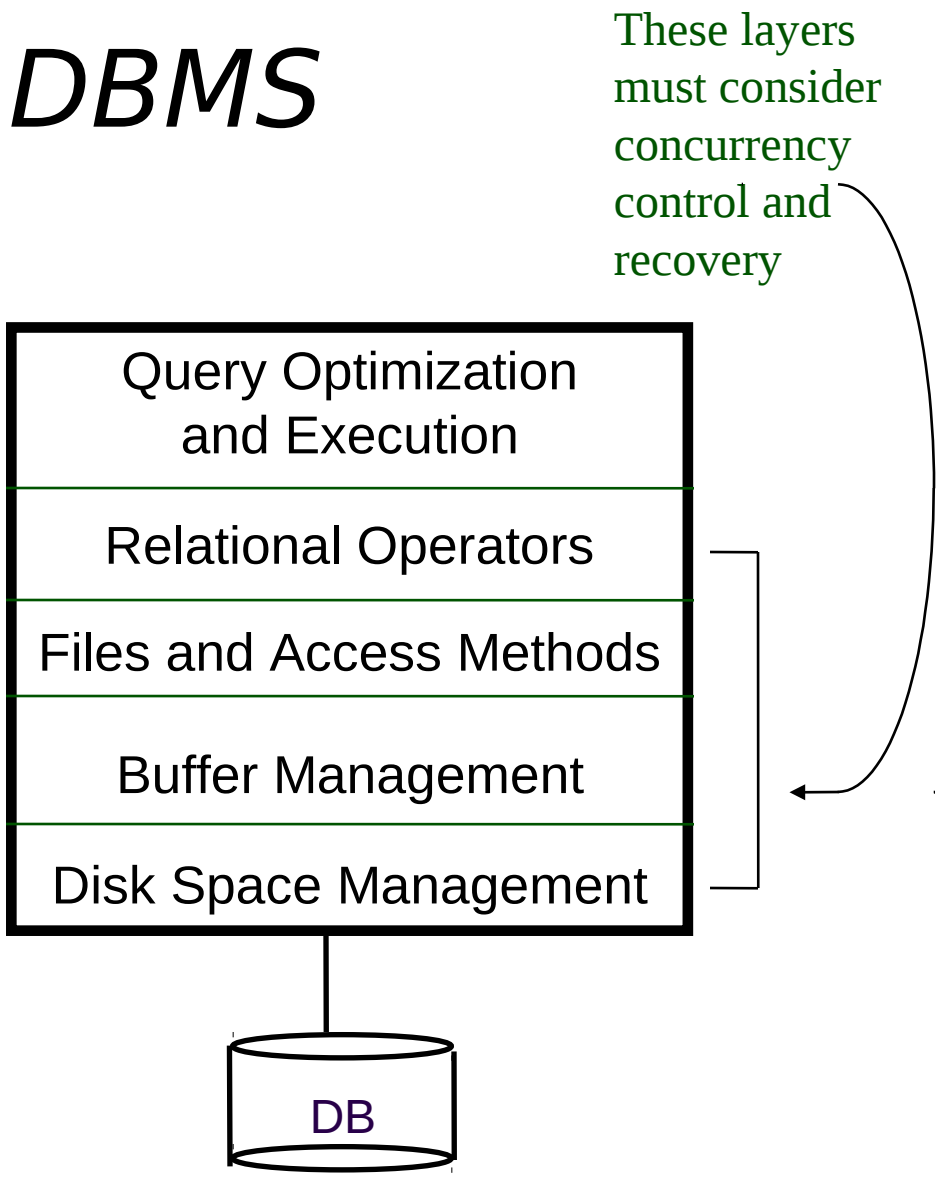
- Concurrent execution of user programs is essential for good DBMS performance.
  - Because disk accesses are frequent, and relatively slow, it is important to keep the cpu humming by working on several user programs concurrently.
- Interleaving actions of different user programs can lead to inconsistency: e.g., check is cleared while account balance is being computed.
- DBMS ensures such problems don’t arise: users can pretend they are using a single-user system.

Transaction: An Execution of a DB Program

- Key concept is *transaction*, which is an *atomic* sequence of database actions (reads/writes).
- Each transaction, executed completely, must leave the DB in a *consistent state* if DB is consistent when the transaction begins.
  - Users can specify some simple *integrity constraints* on the data, and the DBMS will enforce these constraints.
  - Beyond this, the DBMS does not really understand the semantics of the data. (e.g., it does not understand how the interest on a bank account is computed).
  - Thus, ensuring that a transaction (run alone) preserves consistency is ultimately the user's responsibility!

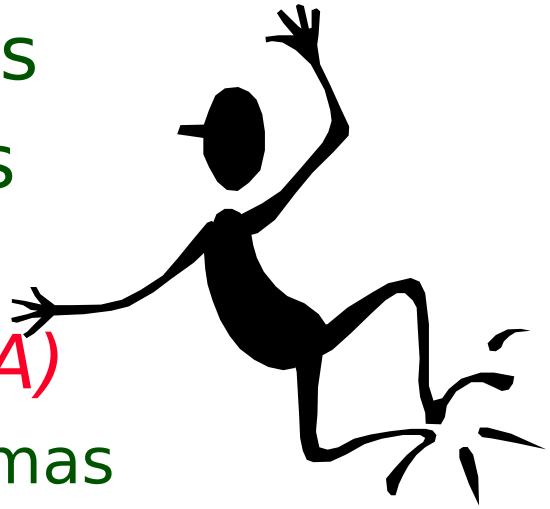
Structure of a DBMS

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.
- This is one of several possible architectures; each system has its own variations.



Databases in practice ...

- End users and DBMS vendors
- DB application programmers
  - E.g. smart webmasters
- *Database administrator (DBA)*
  - Designs logical /physical schemas
  - Handles security and authorization
  - Data availability, crash recovery
  - Database tuning as needs evolve



Must understand how a DBMS works!

Summary

- DBMS used to maintain, query large datasets.
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
- Levels of abstraction give data independence.
- A DBMS typically has a layered architecture.
- DBAs hold responsible jobs and are well-paid!
- DBMS R&D is one of the broadest, most exciting areas in CS.