FEE232
Database Management Systems (DBMS)

Database Management Systems - Ramakrishnan-Gherke-3rd. Ed. McGraw-Hill, 2003.

Main areas to cover:

1. Introduction to databases
2. Database conceptual design (Entity-Relationship model)
3. Database Logical design (Relational model)
4. Relational Database theory (Schema refinement)
5. Relational Query Languages

What Is a DBMS?

- A very large, integrated collection of data describing activities of organizations.
- Models real-world.
  - Entities (e.g., students, courses)
  - Relationships (e.g., Raul is taking FEE232)
- A *Database Management System (DBMS)* is a software package designed to store and manage databases.

A Little Bit of History

- First DBMS: Bachman at General Electric, early 60's (***Network Data Model***). Standardized by CODASYL.
- Late 60's : IBM's IMS (Inf. Mgmt. Sys.) (***Hierarchical Data Model***).
- 1970: Edgar Codd (at IBM) proposed the ***Relational Data Model***. Strong theoretical basis.
- 1980's -90's: Relational model consolidated. Research on query languages and data models => logic-based languages, OO DBMSs => Object-relational data model (extend DBMSs with new data types)

Why Use a DBMS?

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

Files vs. DBMS

- Application must transfer large datasets between main memory and secondary storage (e.g., buffering, page-oriented access, 32-bit addressing, etc.)
- Special code for different queries
- Must protect data from inconsistency due to multiple concurrent users
- Crash recovery
- Security and access control

Describing Data: Data Models

- A *data model* is a collection of concepts and constructs for describing data.
- A *schema* is a description of a particular collection of data, using 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 fields.

Describing Data: Data Models (cont.)

- The data model of the DBMS hides details - *Semantic Models* assist in the DB design process.
- Semantic Models allow an initial description of data in the "real world".
- A DBMS do not support directly all the features in a semantic model.
- Most widely used: **Entity-Relationship model (E/R).**

The Relational Model (Introduction)

- Central construct: the RELATION : a set of records.
- Data is described through a SCHEMA specifying the name of the relation, and name and type of each field:
  - *Students(sid: string, name: string, login: string,*
    age*: integer, average:real)*

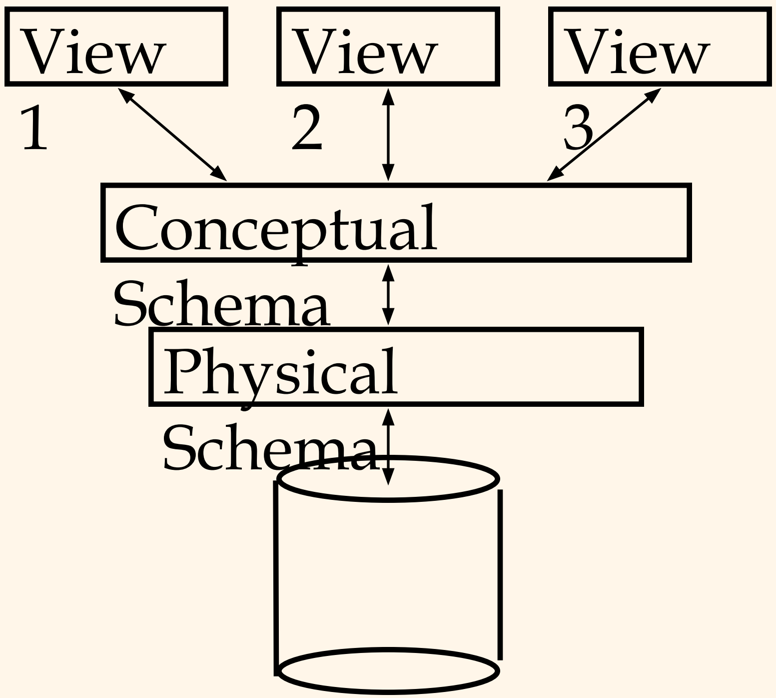
The Relational Model (Introduction)

- Actual data: instance of the relations : a set of *tuples, e.g.:*
  {<F17/1001/2016,Paul,paul@uonbi,20,80>,
  <F17/1002/2016,Alex,alex@uonbi,21,85>,
  <F17/1003/2016,Jane,jane@uonbi,20,70>,
  ...}

student_id	name	login	age	marks
F17/1001/2016	Paul	paul@uonbi	20	80
F17/1002/2016	Alex	alex@uonbi	21	85
F17/1003/2016	Jane	jane@uonbi	20	70
F17/1004/2016	Irene	irene@uonbi	19	60

Levels of Abstraction

- Data is described at three Levels of Abstraction
- Many *views*, single *conceptual (logical) schema* and *physical schema*.
  - Views describe how users see the data (data tailored to different user groups) .
  - Conceptual 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

- Conceptual schema:
  - *Students(sid: string, name: string, login: string,*
    ***age**: integer, average:real)*
  - *Courses(cid: string, cname:string, hours:integer)*
  - *Enrolled(sid:string, cid:string, marks:int)*
    - *describes data in terms of the data model of the DBMS*
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View):
  - *Course_info(cid:string, enrollment:integer)*

Data Independence

- Advantage of using a DBMS: applications are isolated from changes in the way data is structured and stored.
- *Logical data independence*: Protection from changes in *logical* structure of data (if the CS is changed, views can be redefined in terms of the new relations).
- *Physical data independence*: Protection from changes in *physical* structure of data.
- *One of the most important benefits of using a DBMS!*

Querying a DBMS

- A DBMS provides a Query Language.
- Query languages allow querying and updating a DMBS in a simple way.
- Most popular DML (Data Manipulation Language) : SQL (Structured Query Language).
- Queries:
  - List the name of student with sid=F17/1003/2016
  - Name and age of students enrolled in FEE232

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 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.

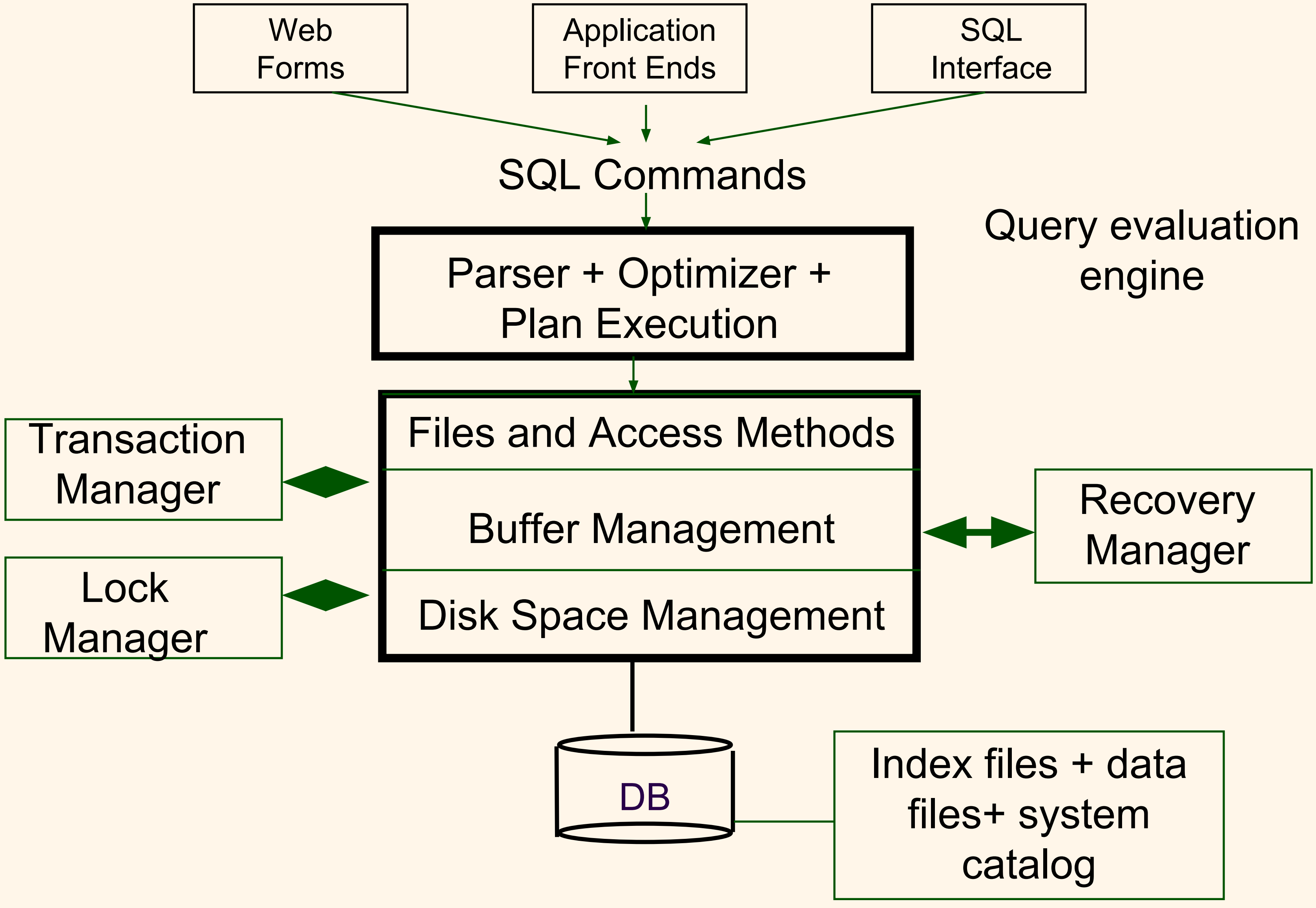
Transaction: An Execution of a DB Program

- 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.
  - Thus, ensuring that a transaction (run alone) preserves consistency is ultimately the user's responsibility!

Ensuring Atomicity

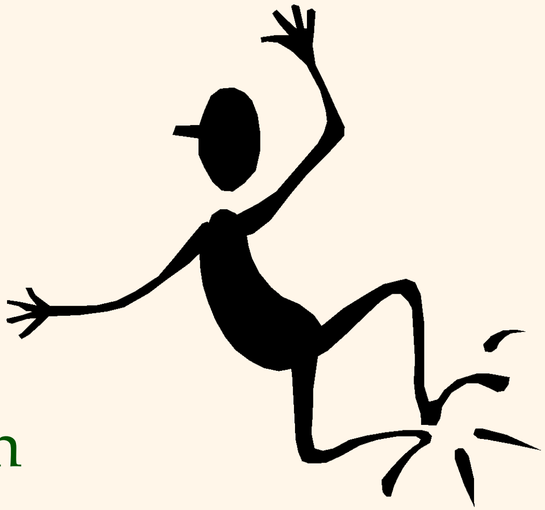
- DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of a transaction.
- Idea: Keep a *log* (history) of all actions carried out by the DBMS while executing a set of transactions:
  - Before a change is made to the database, the corresponding log entry is forced to a safe location.
  - After a crash, the effects of partially executed transactions are *undone* using the log. (the change was not applied to database but to the log itself!)

Structure of a DBMS (cont.)



Typical users of a DBMS…

- End users and DBMS vendors
- DB application programmers
- *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!
- DBMS R&D is one of the broadest, most exciting areas in CS.