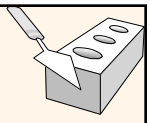
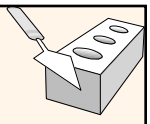


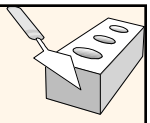
Database Management Systems

Chapter 1



What Is a Database/DBMS?

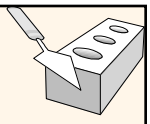
- A very large, integrated collection of data.
- Models real-world *scenarios*
  - Entities (e.g., students, courses)
  - Relationships (e.g., John is taking CS 5120)
- A *Database Management System (DBMS)* is a software package designed to store and manage databases.



Files vs. DBMS

Scenario: 100 TB of data of employees, products, salaries, ….

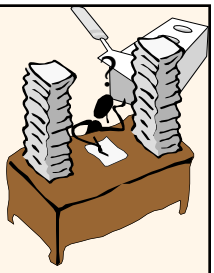
- Application must stage large datasets between main memory and secondary storage (e.g., buffering, page-oriented access, etc.)
- Special code for different queries
- Must protect data from inconsistency due to multiple concurrent users
- Crash recovery
- Security and access control



Why Use a DBMS?

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

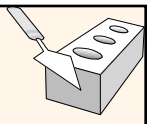
Why Study Databases??



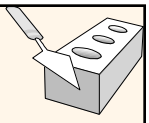
- Shift from *computation* to *information*
  - Web applications, scientific applications, etc.
- Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, Earth Observation System project
  - ... need for DBMS exploding
- DBMS encompasses most of CS
  - OS, programming languages, theory, AI, multimedia, etc.

https://www.ibm.com/analytics

Data Models

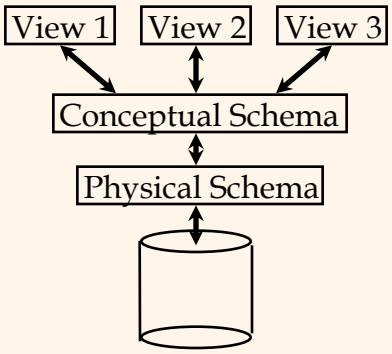


- A *data model* is a collection of concepts for describing data.
- A *schema* is a description of a particular collection of data, using the given data model.
- The *relational model of data* is the most widely used model today.
  - Main concept: *relation*, basically a table with rows and columns.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

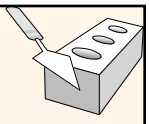


Levels of Abstraction

- Many *views*, single *conceptual (logical) schema* and *physical schema*.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indices used.

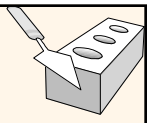


☛ *Schemas are defined using DDL; data is modified/queried using DML.*



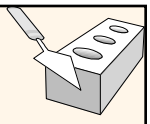
Example: University Database

- Conceptual schema (Chapters 2 and 19):
  - *Students(sid: string, name: string, login: string, age: integer, gpa: real)*
  - *Courses(cid: string, cname: string, credits: integer, fname: string)*
  - *Enrolled(sid: string, cid:string, grade: string)*
  - *Faculty(fid: string, fname: string, sal: real)*
- Physical schema (Chapter 20):
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View) (Chapters 3 and 25):
  - *Course_info(cid:string,enrollment:integer,fname:string)*



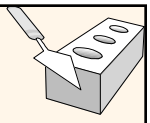
Data Independence

- Applications are insulated from how data is structured and stored.
- *Logical data independence*: Protection from changes in *logical* structure of data.
  - *Faculty(fid: string, fname: string, sal: real)*
  - *Faculty_Public(fid: string, fname: string, office: integer)*
  - *Faculty_Private(fid: string, sal: real)*
- *Physical data independence*: Protection from changes in *physical* structure of data.
  - Indices, file structure, disks …



Relational Algebra

- Basic operations:
  - *Selection* (σ) Selects a subset of rows from a relation.
  - *Projection* (π) Deletes unwanted columns from relation.
  - *Cross-product* ($\times$) Allows us to combine two relations.
  - *Set-difference* ($-$) Tuples in reln. 1, but not in reln. 2.
  - *Union* ($\cup$) Tuples in reln. 1 or in reln. 2 (or both).
- Additional operations:
  - Intersection, *join*, division, renaming: Not essential, but (very!) useful.
- Since each operation returns a relation, operations can be *composed*!



The SQL Query Language

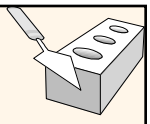
- To find all 18 years old students, we can write:

```
SELECT *
FROM Students S
WHERE S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

- To find just names and logins, replace the first line:

```
SELECT S.name, S.login
```

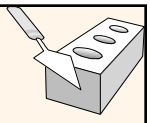
Creating Relations in SQL

- Creates the Students relation. Observe that the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.
- As another example, the Enrolled table holds information about courses that students take.

```
CREATE TABLE Students
        (sid: CHAR(20),
         name: CHAR(20),
         login: CHAR(10),
         age: INTEGER,
         gpa: REAL)
```

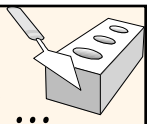
```
CREATE TABLE Enrolled
        (sid: CHAR(20),
         cid: CHAR(20),
         grade: CHAR(2))
```

https://www.w3resource.com/sql/sql-table.php



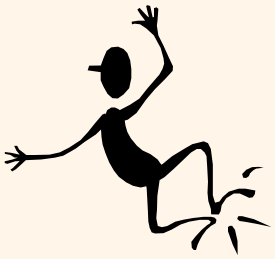
Concurrency Control

- 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*.



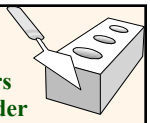
Databases make these folks happy ...

- 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



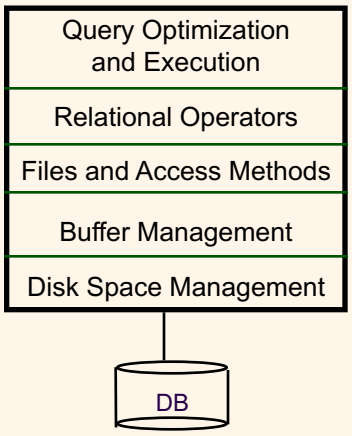
Must understand how a DBMS works!

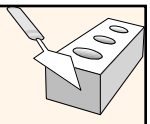
Structure of a DBMS



These layers must consider concurrency control and recovery

- 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.





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.