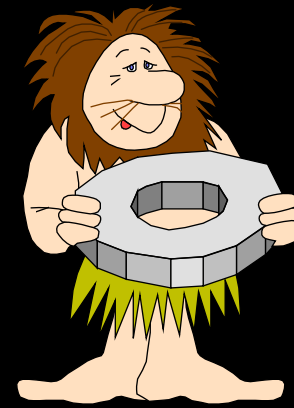


INSTITUTE OF INDUSTRIAL AND INFORMATION TECHNOLOGY



Database Management Systems

What is a database?



- A very large, integrated collection of data.
- Models real-world application :
 - Entities (e.g., students, courses)
 - Relationships (e.g., Madonna is taking CS564)
- Usually data is too large to fit into main memory, and often used by many users

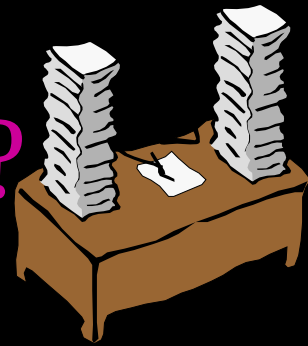
Database applications ?

- E-commerce : Amazon.com, etc.
- Airlines and travel services
- Scientific data such as biology, oceanography, etc.
- Spatial data such as maps, travel networks,
- World Wide Web
- Digital libraries of artifacts of any kind

What is a DBMS ?

- DBMS stands for Database Management System
- software package designed to store, manage and provide access to databases.

Why Study Databases??



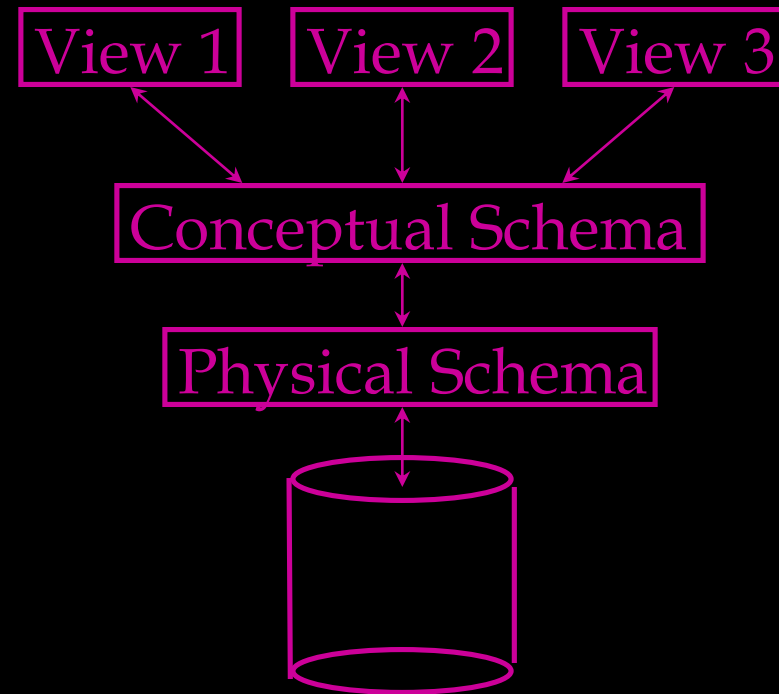
- Shift from computation to information
- Datasets increasing in diversity and volume.
 - Digital libraries, interactive video, Human Genome project, EOS project
 - ... need for DBMS exploding
- DBMS encompasses most of CS :
 - OS, languages, theory, AI, multimedia, logic

Terminology : 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
 - 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:
 - Views describe how users see the data.
- Single conceptual (logical) schema
 - Conceptual schema defines logical structure
- Single physical schema:
 - 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, 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*, *enrollment:integer*)
- *CS542Students*(*sid: string, grade:string*)

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!*

Files vs. DBMS



If we were to use files, we would have to :

- Stage large datasets between main memory and secondary storage (buffering, page-oriented access)
- Must write special code for different queries
- Must protect data from inconsistency due to multiple concurrent users
- Must manage crash recovery in some special-purpose manner
- Must provide good methods for access control

Why Use a DBMS?

- Reduced application development time.
- Data independence
- Efficient data access.
- Data integrity under updates.
- Concurrent access
- Recovery from crashes.
- Security
- Uniform data administration.

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 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 data inconsistency problems don't arise:
 - E.g., users can pretend they are using a single-user system

Key Concepts of CC

- Key concept is transaction, which is an atomic sequence of multiple database actions (reads/writes)
- Each transaction, executed completely, must leave the DB in a consistent state
- Utilize **locking** of resources and other **protocols** for guaranteeing consistency.

System Crash : Ensuring Atomic

- If system crashes in the middle of a Xact, then DBMS ensures *atomicity*
- **Idea:** Keep a log (history) of all actions carried out by the DBMS while executing a set of Xacts:
 - **Before** a change is made to database, corresponding log entry is **forced** to a safe location (commit of transaction)
 - After a crash, the effects of partially executed transactions are undone using the log (rollback of transaction)

Databases make these folks happy ...

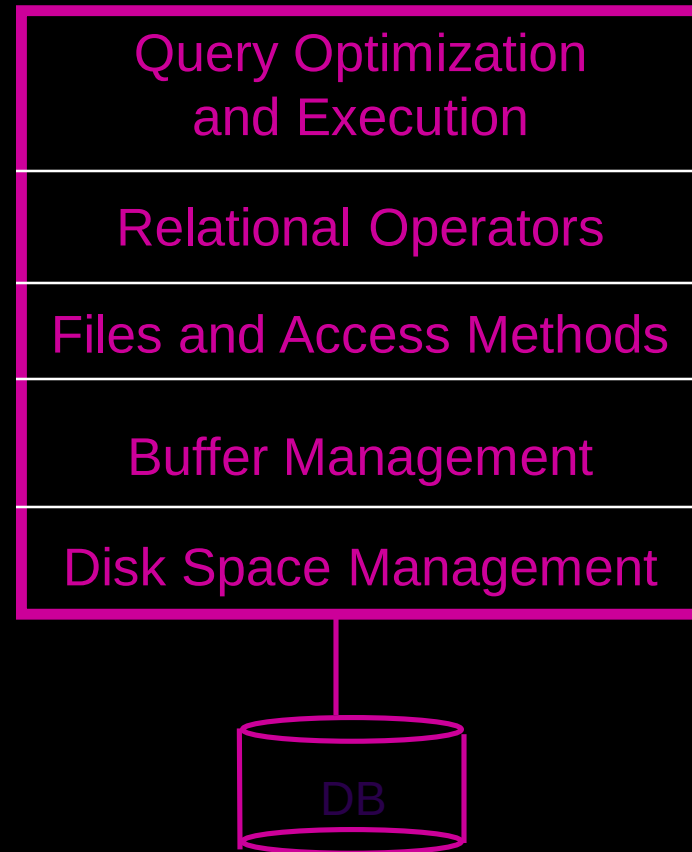
- End users and DBMS vendors
- DB application programmers
 - E.g., smart webmasters
- *Database administrator (DBA)*
 - Designs logical /physical schemas
 - Handles security and authorization
 - Data availability, crash recovery
 - Database tuning as needs evolve

Must understand how a DBMS works!

Structure of a DBMS

- A typical DBMS has a layered architecture.
- Concurrency control and recovery components not shown.

These layers must consider concurrency control and recovery



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 rewarding jobs. 😊
 - **DBMS R&D is one of the broadest, most exciting areas in CS.**

Introductory Material

Sets, Relations and Functions

Sets

- Unordered collection of objects
- Characteristics
 - Unordered
 - No duplicates (no object appears more than once in a set)
- Eg: Set of passengers, set of flights
- Recall the main set operations
 - Union, intersection, complement
 - Check subset

Relations

- Given multiple sets $A_1, A_2, \dots, A_n$, a relation is a set of n -tuples of the form $(a_{11}, a_{12}, \dots, a_{1n})$, where a_{11} is an element of A_1 , a_{12} is an element of A_2 , and so on.
- Eg: suppose the set of course = $\{DB1, DB2\}$, the set of TAs = $\{Hong, Song\}$, then a relation between these two sets could be $\{(DB1, Hong), (DB1, Song), (DB2, Hong)\}$

Functions

- Given two sets A , B , a function f from A to B is denoted as $f: A \rightarrow B$. This maps any value of A to one value of B .
 - Eg: consider function from faculty members to depts
 - $\{(\text{Mike Gennert} \rightarrow \text{CS}), (\text{Peter Hansen} \rightarrow \text{Humanities})\}$
- Characteristics
 - A is called domain
 - B is called range
 - No value of A can map to multiple B 's.

Functions

- Injection (one to one):
 - No 2 values in A map to the same B
 - Eg: set of Husbands $\rightarrow$ set of wives
- Surjections (onto)
 - Every value in B has at least 1 value in A that maps to it
- Bijections
 - One to one and onto