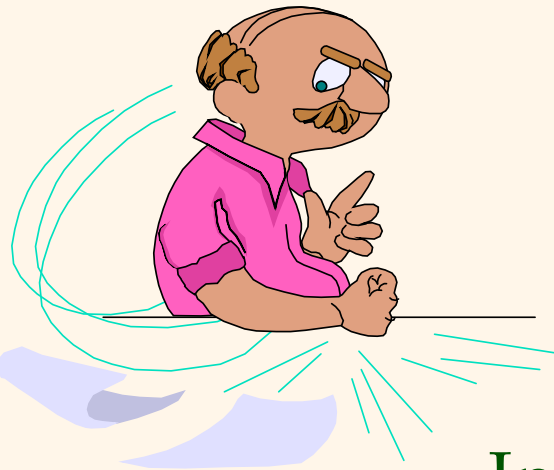


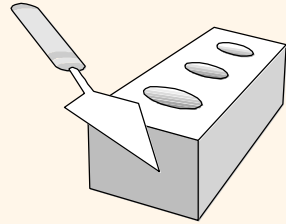
# *Database Management Systems (DBMS)*

## *Chapter 1*

Instructor: Mohamed Mokbel  
mokbel@cs.umn.edu



# Welcome to CSCI 4707



## ❖ Your Host:

Mohamed F. Mokbel

Office: KHKH 4-207

Web: [www.cs.umn.edu/~mokbel](http://www.cs.umn.edu/~mokbel)

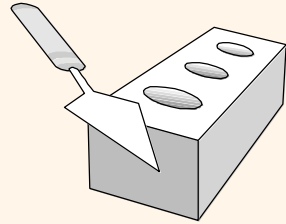
Email: [mokbel@cs.umn.edu](mailto:mokbel@cs.umn.edu) (**Include [CSCI 4707] on the Subject line**)

Office Hours: TuThu: 12:00 – 13:00 PM (or by appointment)


## ❖ TA:

- Christopher Jonathan  
Office KHKH: 2-209      Office hours: Mon: 09:00 - 10:00 AM  
Email: [cjonathan@umn.edu](mailto:cjonathan@umn.edu)
  
- Harshada Chavan  
Office KHKH: 2-209      Office hours: Wed: 9:00 - 11:00 AM  
Email: [chava057@cs.umn.edu](mailto:chava057@cs.umn.edu)
  
- Saif Alharthi  
Office KHKH: 2-209      Office hours: Fri: 9:30 - 11:30 AM  
Email: [alhar035@cs.umn.edu](mailto:alhar035@cs.umn.edu)

# Grading Policy



- ❖ See it on  
<http://www-users.cselabs.umn.edu/classes/Fall-2016/csci4707/index.php?page=grading>

**UNIVERSITY OF MINNESOTA**  
**Driven to Discover<sup>SM</sup>**

myU > One Stop >

Search U of M Web Sites

COLLEGE OF Science & Engineering

[CSE Home](#) | [CSE Directory](#) | [Give to CSE](#) | [Student Dashboard](#)

[Home](#)  
[Grading](#)  
[Schedule](#)  
[Moodle](#)

## CSci 4707: Practice of Database Systems

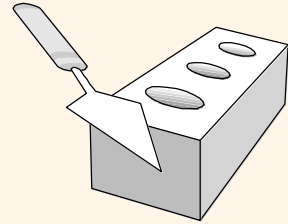
### Grading Evaluations

- \* **Final Exam:** 25% (Comprehensive).
- \* **Three Midterm Exams:** 25%.  
Best two of three midterms are considered: 12.5% each.
- \* **Four Homeworks:** 12% (3% each).
- \* **Four Labs:** 38%.
  - Lab # 1: 8%. (Individual)
  - Lab # 2: 10%. (Group of 2)
  - Lab # 3: 10%. (Group of 2)
  - Lab # 4: 10%. (Group of 2)

- 24 hours late submission is allowed for each assignment and it costs 10% from the total grade of the assignment.

- Lab 2 - 4 must be done in a group of 2. Group of 1 needs permission from the T.A.

# Welcome to CSCI 4707



## ❖ Text Book:

Database Management Systems,  
3e (ISBN: 0-07-246563-8)

Raghu Ramakrishnan and Johannes  
Gehrke

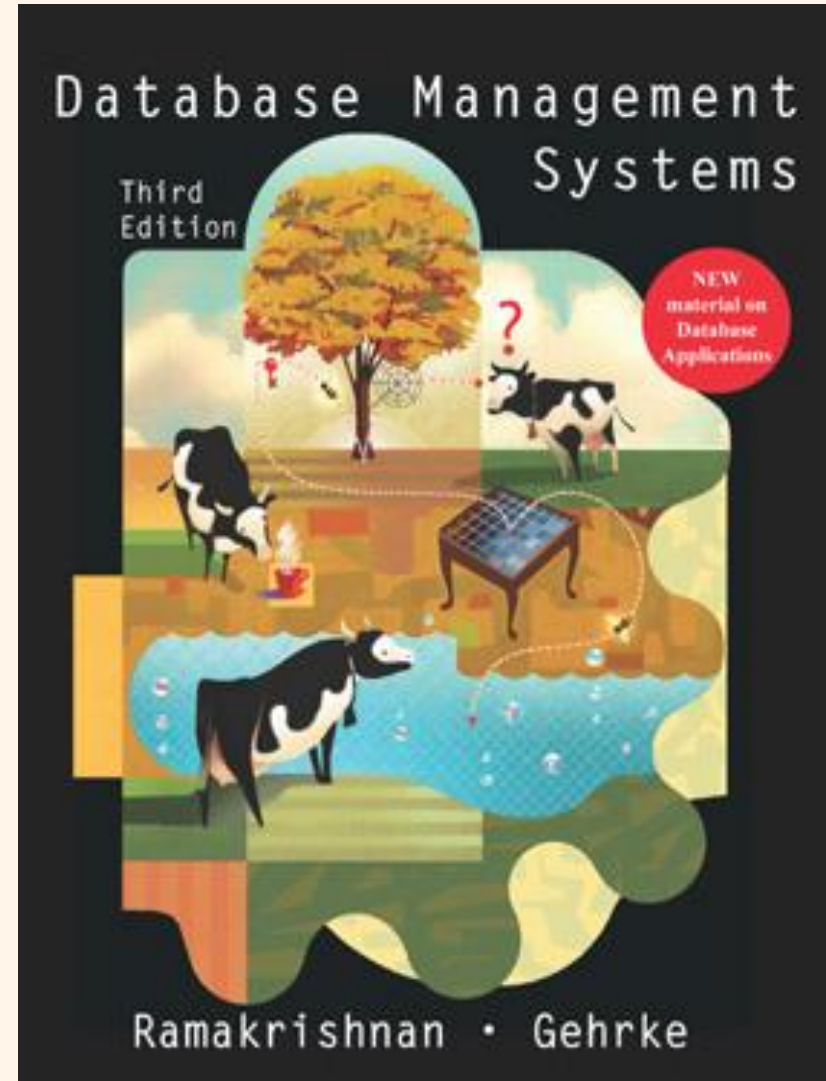
Book website:

<http://www.cs.wisc.edu/~dbbook/>

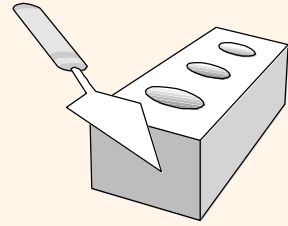
- Chapters: 1-5, 8, 9, 12-20

## ❖ Class web page:

- <http://www-users.cselabs.umn.edu/classes/Fall-2016/csci4707/index.php>
- Tentative scheduling is in the course web site
- Stay tuned for announcements, labs, homeworks, and grades

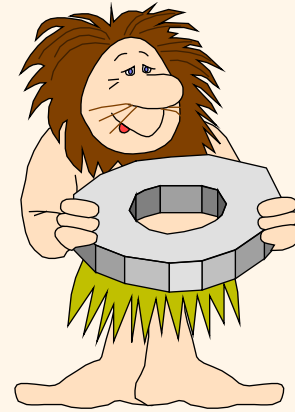


# What we will study in the Course



- ❖ **PART 1:** *Outside* the Database Engine as a *User*
  - How to do conceptual database design (Chapter 2)
  - How to do a logical database design (Chapter 3)
  - How to query the database (Chapters 4, 5)
- ❖ **PART 2:** *Inside* the Database Engine
  - Storage and indexing modules (Chapters 8-10)
  - Query processing & optimization (Chapters 12-15)
  - Transaction Processing (Chapters 16-18)
- ❖ **PART 3:** *Outside* the Database Engine as a *DBA*
  - How to refine your schema design (Chapter 19)
  - How to tune the database design (Chapter 20)

# *What Is a DBMS?*

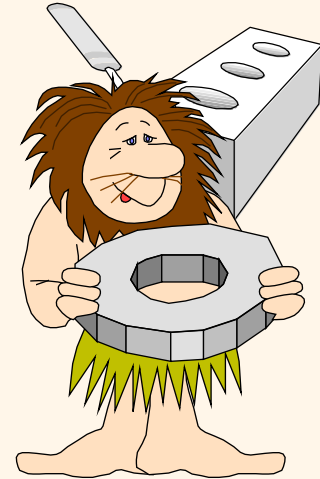


- ❖ How do you store your friend's phone numbers:

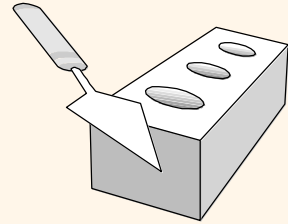
| Name | Home | Cell | Address | Email |
|------|------|------|---------|-------|
|      |      |      |         |       |

- ❖ Designing, storing, and managing such “simple” tables is the core of DBMS

# What Is a DBMS?



- ❖ A **VERY LARGE**, integrated collection of data.
- ❖ Models real-world:
  - Entities (e.g., universities, departments, students, courses)
  - Relationships (e.g., Students are taking courses)
- ❖ A Database Management System (DBMS) is a software package designed to store and manage data.



# *History of DBMS*

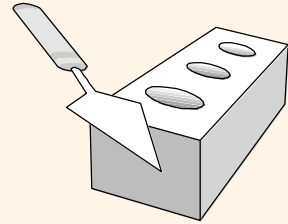
## ❖ Early 60's

- First general-purpose DBMS by Charles Bachman at General Electric (First recipient of ACM Turing Award in 1973)

## ❖ Late 60's

- Hierarchical data model developed at IBM
- The SABRE system for airline reservation is jointly developed by American Airlines and IBM where several people can access the same data through a network





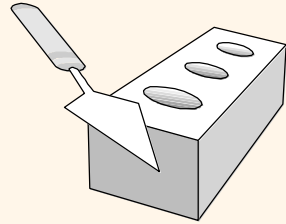
# *History of DBMS*

## ❖ 70's

- The relational data model is proposed by Edgar Codd at IBM (ACM Turing Award at 1981)
- Two main prototypes for relational database management systems are developed. Ingres at UCB and System R at IBM.
- Peter Chen (MIT) proposed the entity-relationship model

## ❖ 80's

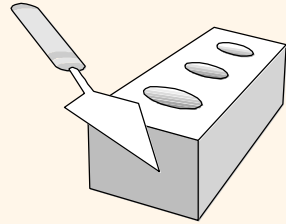
- SQL query language is developed (part of System R)
- The concept of read/write transactions is developed to allow concurrent execution of database operations (Jim Gray receives the ACM Turing Award at 1999)
- Commercial databases are in the market (DB2, Oracle, Informix)



# *History of DBMS*

## ❖ 90's - present

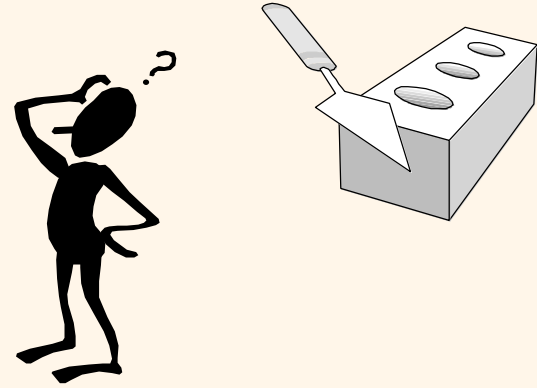
- DBMSs are well-established in industry and academia
- New database applications:
  - Data warehousing
  - Multimedia databases
  - Spatial/Spatio-temporal databases
  - Data mining
  - Scientific database
  - Bioinformatics
  - Digital libraries
  - Web-databases



# *Files vs. DBMS*

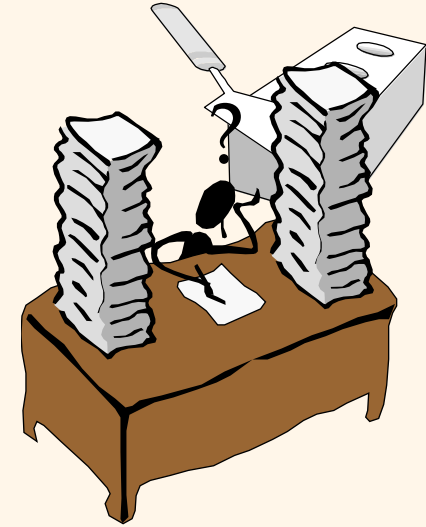
- ❖ Special code for different queries
- ❖ Must protect data from inconsistency due to multiple concurrent users
- ❖ Crash recovery
- ❖ Security and access control

# *Why Use a DBMS?*

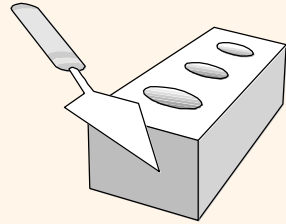


- ❖ **Data independence**
  - Applications are independent from data representation
- ❖ **Efficient data access**
  - Through indexing and query optimization techniques
- ❖ **Data integrity and security**
  - DBMS enforce integrity constraints and access control
- ❖ **Concurrent access**
  - Multiples users are allowed to use the same tables
- ❖ **Crash recovery**
  - DBMS protects the user from the system failure
- ❖ **Reduced application development time**
  - DBMS supports functions that are common to many applications

# Why Study Databases??

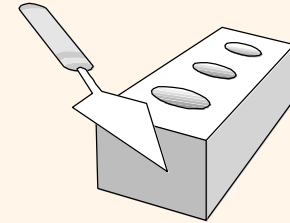


- ❖ Shift from computation to information
  - at the “low end”: scramble to webspace (a mess!)
  - at the “high end”: scientific applications
- ❖ Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, Earth Observation project
  - ... need for DBMS exploding
- ❖ Good job market as a DBA, system analyst,
- ❖ 3-credit course(s)



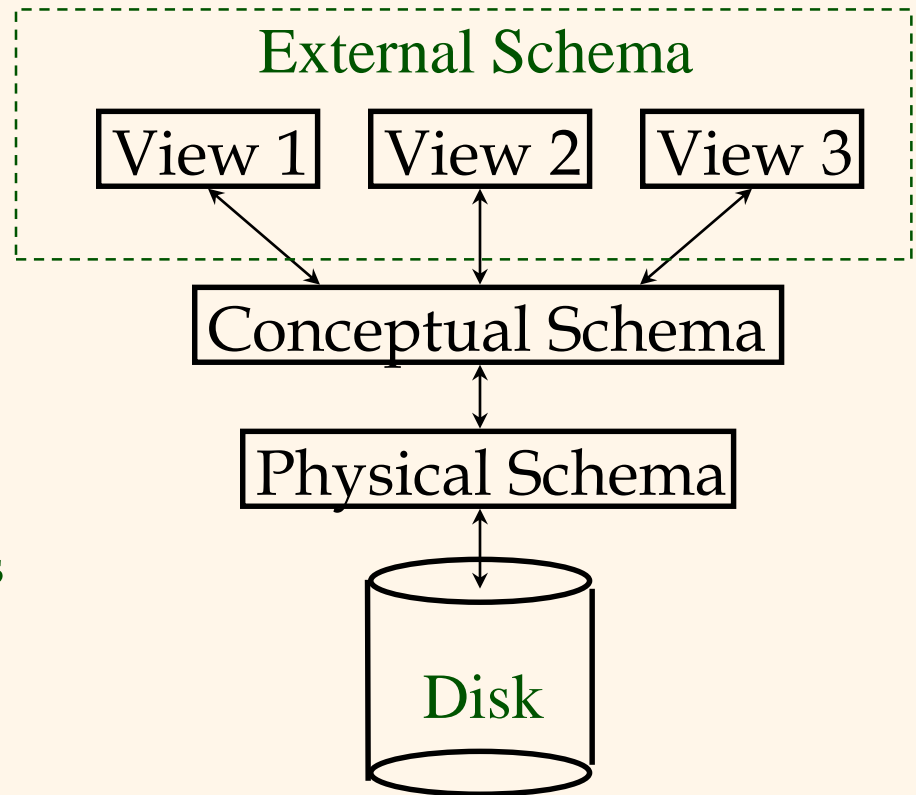
# Concepts of Data Modeling

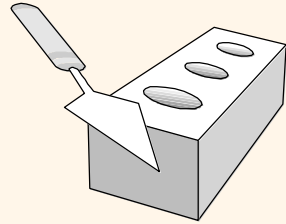
- ❖ A data model is a collection of concepts for describing data.
- ❖ A schema is a description of a particular collection of data, using the a given data model.
- ❖ The relational model of data is the most widely used model today.
  - Main concept: relation, basically a table with rows and columns.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



# Levels of Abstraction

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





# *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)*

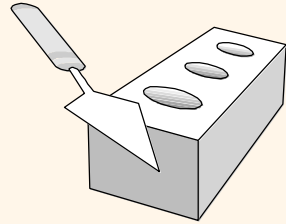
## ❖ Physical schema:

- Relations stored as unordered files.
- Index on first column of Students.

## ❖ External Schema (View):

- *Course\_info(cid: string, enrollment: integer)*

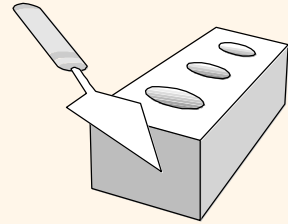




# *Data Independence \**

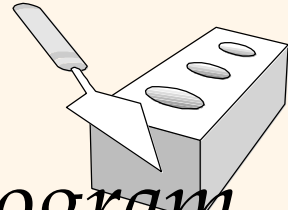
- ❖ Applications insulated from how data is structured and stored.
- ❖ Logical data independence: Protection from changes in *logical* structure of data.
- ❖ Physical data independence: Protection from changes in *physical* structure of data.

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



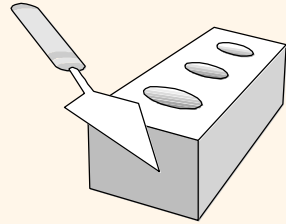
# *Concurrency Control*

- ❖ 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 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!

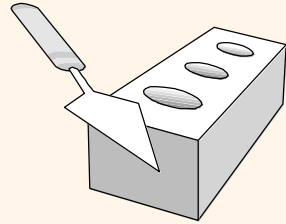


# Example

- ❖ Consider two transactions (*Xacts*):

|     |       |              |            |     |
|-----|-------|--------------|------------|-----|
| T1: | BEGIN | $A=A+100$ ,  | $B=B-100$  | END |
| T2: | BEGIN | $A=1.06*A$ , | $B=1.06*B$ | END |

- ❖ Intuitively, the first transaction is transferring \$100 from B's account to A's account. The second is crediting both accounts with a 6% interest payment.
- ❖ There is no guarantee that T1 will execute before T2 or vice-versa, if both are submitted together. However, the net effect *must* be equivalent to these two transactions running serially in some order.



## Example (Contd.)

- ❖ Consider a possible interleaving (schedule):

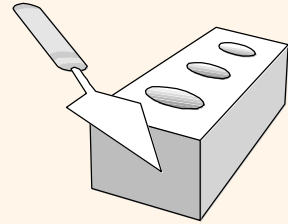
|     |                 |                |
|-----|-----------------|----------------|
| T1: | $A = A + 100,$  | $B = B - 100$  |
| T2: | $A = 1.06 * A,$ | $B = 1.06 * B$ |

- ❖ This is OK. But what about:

|     |                              |               |
|-----|------------------------------|---------------|
| T1: | $A = A + 100,$               | $B = B - 100$ |
| T2: | $A = 1.06 * A, B = 1.06 * B$ |               |

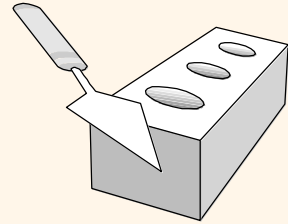
- ❖ The DBMS's view of the second schedule:

|     |                          |              |
|-----|--------------------------|--------------|
| T1: | $R(A), W(A),$            | $R(B), W(B)$ |
| T2: | $R(A), W(A), R(B), W(B)$ |              |



# Scheduling Concurrent Transactions

- ❖ DBMS ensures that execution of  $\{T_1, \dots, T_n\}$  is equivalent to some serial execution  $T_1' \dots T_n'$ .
  - Before reading/writing an object, a transaction requests a lock on the object, and waits till the DBMS gives it the lock. All locks are released at the end of the transaction. (Strict 2PL locking protocol.)
  - **Idea:** If an action of  $T_i$  (say, writing  $X$ ) affects  $T_j$  (which perhaps reads  $X$ ), one of them, say  $T_i$ , will obtain the lock on  $X$  first and  $T_j$  is forced to wait until  $T_i$  completes; this effectively orders the transactions.
  - What if  $T_j$  already has a lock on  $Y$  and  $T_i$  later requests a lock on  $Y$ ? (Deadlock!)  $T_i$  or  $T_j$  is aborted and restarted!

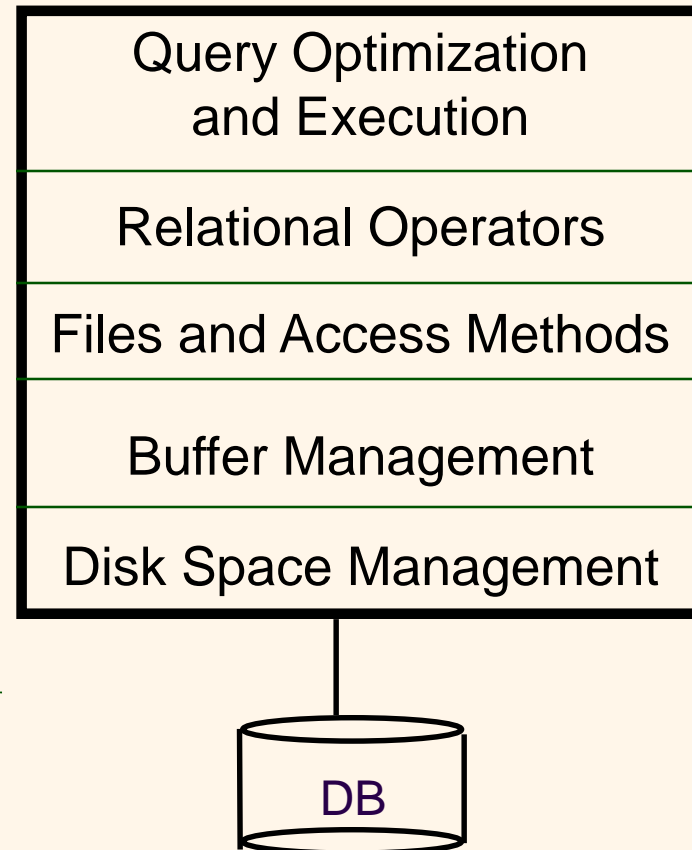


# Ensuring Atomicity

- ❖ 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.

# Structure of a DBMS

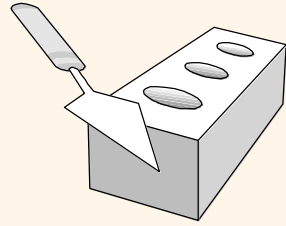
- ❖ A typical DBMS has a layered architecture.
- ❖ The figure does not show the concurrency control and recovery components.
- ❖ This is one of several possible architectures; each system has its own variations.



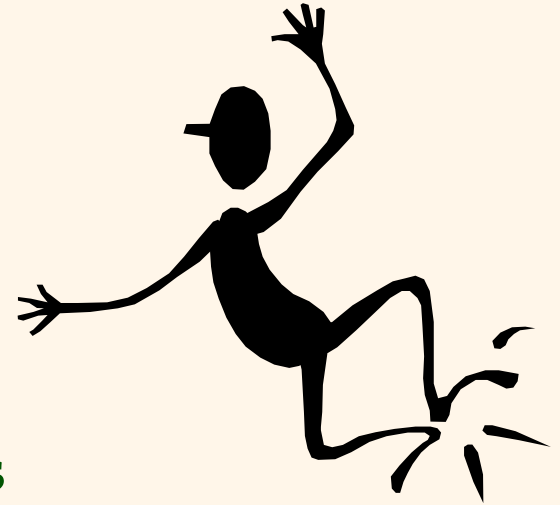
These layers must consider concurrency control and recovery

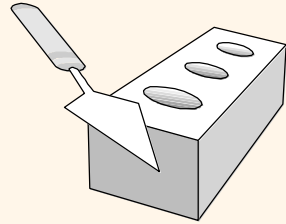


# *Databases make these folks happy ...*



- ❖ End users
- ❖ 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





# *Steps for Designing a Database*

## 1. Requirement analysis

- ❖ An informal discussion with the customers
- ❖ Understand what are the requirements, how different entities relate to each other, what are the frequent operations to be performed

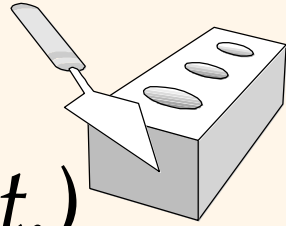
## 2. Conceptual Database Design

- ❖ Develop a high-level description of the data
- ❖ Develop an ER (entity-relationship) model that capture the semantics of the data

## 3. Logical Database Design

- ❖ Convert the ER model into a (relational) database schema

# *Steps for Designing a Database (Cont.)*



## 4. Scheme Refinement

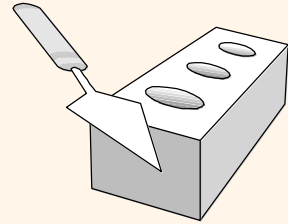
- ❖ Identify potential problems in your schema design
- ❖ This process can be guided by some elegant and powerful theory

## 5. Physical Database Design

- ❖ Study the expected workload of the system
- ❖ Tune the performance by building indexes and clustering tables

## 6. Application and Security Design

- ❖ Identify which parts of the database are accessible to whom



# 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.

