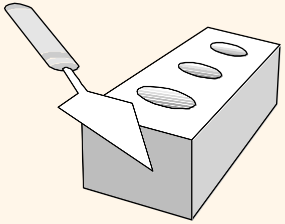
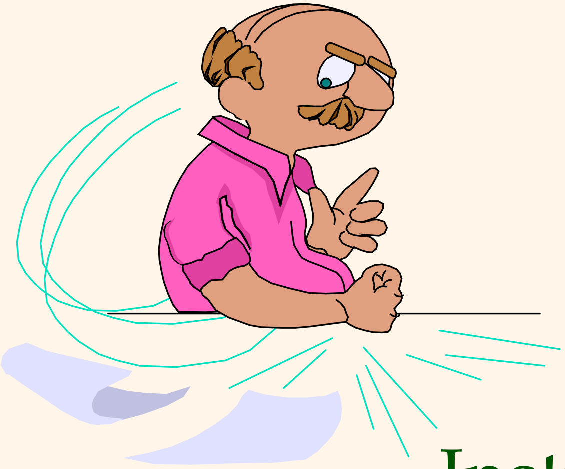
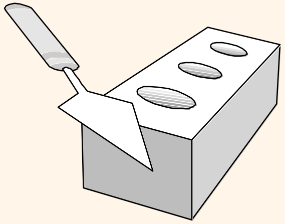


Database Management Systems

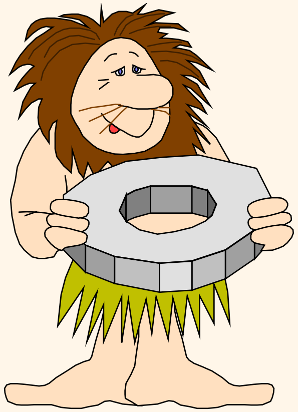
Chapter 1



Instructor: Raghu Ramakrishnan
raghu@cs.wisc.edu
Hamed Vahdat-Nejad



What Is a DBMS?



- 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.

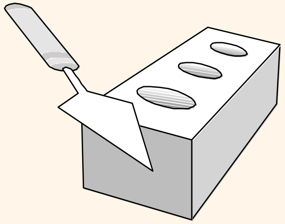
Files vs. DBMS

- Application must store and transfer large datasets between main memory and secondary storage
- 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 from applications layer and efficient access.
- Reduced application development time.
- Data integrity and security.
- Concurrent access

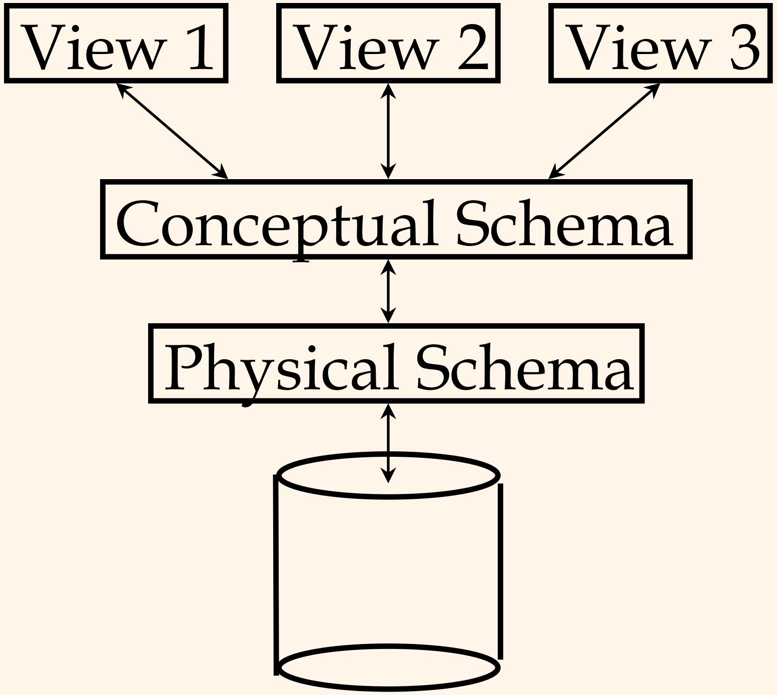
Data Models



- 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 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 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 isolated 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.

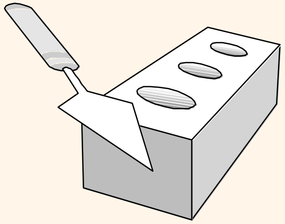
* *One of the most important benefits of using a DBMS!*

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 active by working on several user programs concurrently.
- Interleaving actions of different user programs can lead to inconsistency.
- 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.



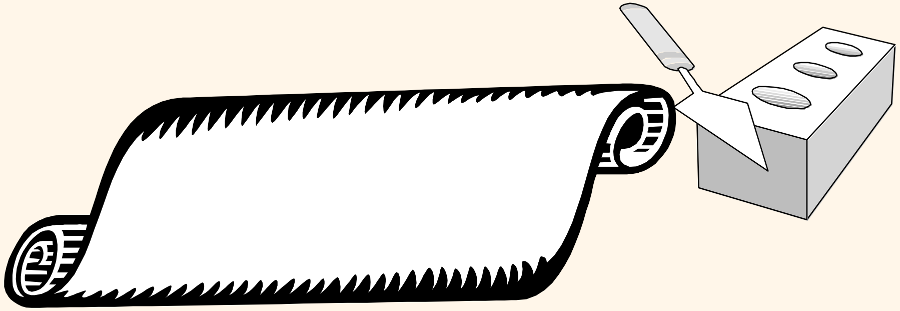
Scheduling Concurrent Transactions

- DBMS ensures that execution of {T1, ... , Tn} is equivalent to some *serial* execution T1' ... Tn'.
  - 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. Idea: If an action of Ti (say, writing X) affects Tj (which perhaps reads X), one of them, say Ti, will obtain the lock on X first and Tj is forced to wait until Ti completes; this effectively orders the transactions.
  - What if Tj already has a lock on Y and Ti later requests a lock on Y? (Deadlock!) Ti or Tj is aborted and restarted!

Ensuring Atomicity

- DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of an act.
- Idea: Keep a *log* (history) of all actions carried out by the DBMS while executing a set of acts:
  - 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.

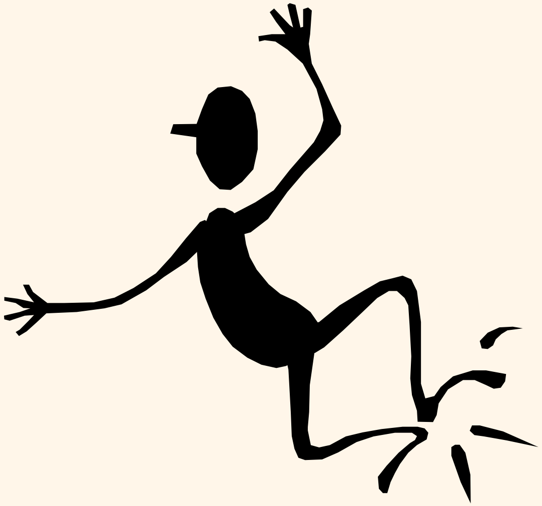
The Log



- The following actions are recorded in the log:
  - *writes an object*: The old value and the new value.
    - Log record must go to disk *before* the changed page!
  - *commits/aborts*: A log record indicating this action.
- Log is often *duplexed (replicated)* and *archived* on "stable" storage.

Databases make these people 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



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.