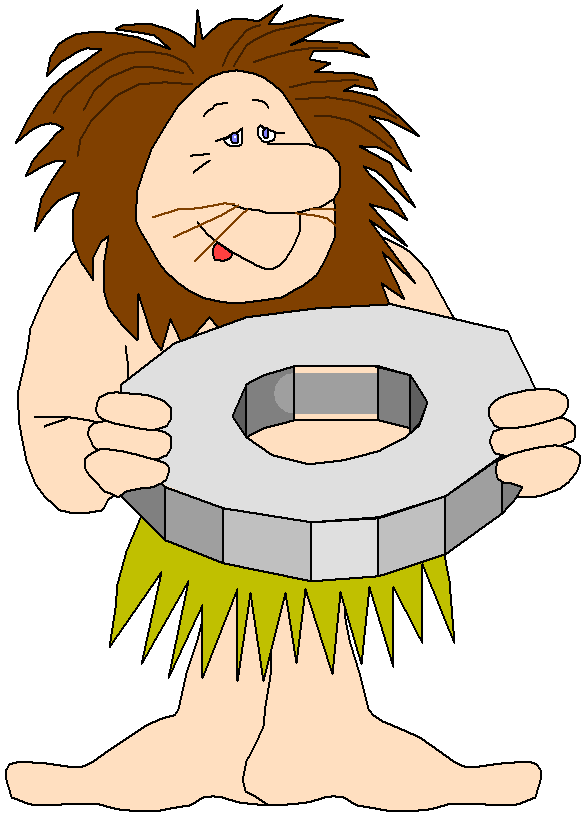


*Introduction to Database Systems*

*Module 1, Lecture 1*

Database Management Systems 1



*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.

Database Management Systems 2

*Why Use a DBMS?*

❖ Data independence and efficient access.

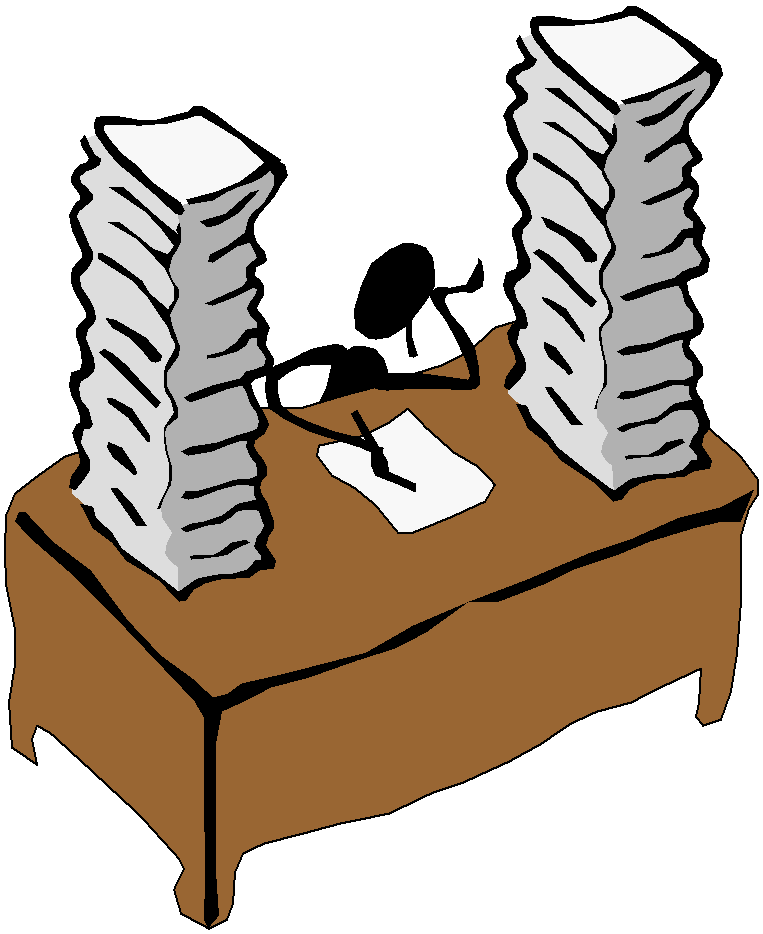
❖ Reduced application development time.

❖ Data integrity and security.

❖ Uniform data administration.

❖ Concurrent access, recovery from crashes.

Database Management Systems 3



?

*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, EOS project

– ... need for DBMS exploding

❖ DBMS encompasses most of CS

– OS, languages, theory, “A”I, multimedia, logic

Database Management Systems 4

*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.

Database Management Systems 5

*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.

View 1 View 2 View 3

Conceptual Schema

Physical Schema

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

Database Management Systems 6

*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!*

Database Management Systems 8

*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.

Database Management Systems 9

*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!

Database Management Systems 10

*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. (Strict 2PL locking protocol.)

– 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!

Database Management Systems 11

*Ensuring Atomicity*

❖ DBMS ensures *atomicity* (all-or-nothing property)

even if system crashes in the middle of a Xact.

❖ 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 the database, the corresponding log entry is forced to a safe location. (*WAL protocol*; OS support for this is often inadequate.)

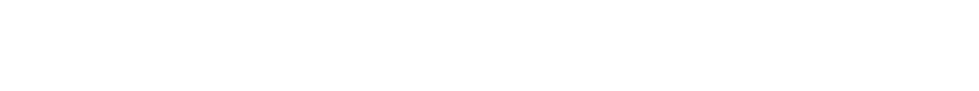
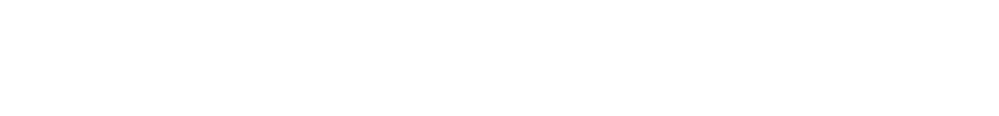
– After a crash, the effects of partially executed

transactions are  *undone* using the log. (Thanks to WAL, if

log entry wasn’t saved before the crash, corresponding

change was not applied to database!)

Database Management Systems 12



*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.

|  |  |
| --- | --- |
| Query Optimization and Execution | |
| Relational Operators | |
| Files and Access Methods | |
| Buffer Management | |
| Disk Space Management | |
|  |  |

❖ The figure does not show the concurrency control and recovery components.

❖ This is one of several possible architectures; each system has its own variations.

DB

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 responsible jobs and are well-paid!

❖ DBMS R&D is one of the broadest, most exciting areas in CS.