

# CH1

---

- CH1
  - What is a DBMS, in particular, a relational DBMS?
  - Why should we consider a DBMS to manage data?
  - How is application data represented in a DBMS?
  - Why study Databases?
  - How is data in a DBMS retrieved and manipulated?
  - How does a DBMS support concurrent access and protect data during system failures?
  - What are the main components of a DBMS?
  - Who is involved with databases in real life?

What is a DBMS, in particular, a relational DBMS?

## *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 3ed, R. Ramakrishnan and J. Gehrke

2

### Database

a collection of data, typically describing the activities of one or more related organizations.

For example, a university database might contain information about the following:

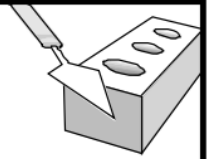
- **Entities** such as students, faculty, courses, and classrooms.
- **Relationships** between entities, such as students' enrollment in courses, faculty teaching courses, and the use of rooms for courses.

Database management system, DBMS

a software designed to assist in maintaining and utilizing large collections of data

Why should we consider a DBMS to manage data?

## *Files vs. DBMS*



- ❖ Application must stage large datasets between main memory and secondary storage (e.g., buffering, page-oriented access, 32-bit addressing, etc.)
- ❖ Special code for different queries
- ❖ Must protect data from inconsistency due to multiple concurrent users
- ❖ Crash recovery
- ❖ Security and access control

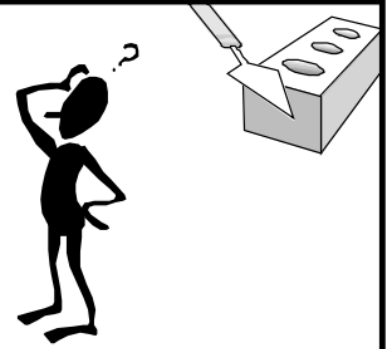
Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

3

!

How is application data represented in a DBMS?

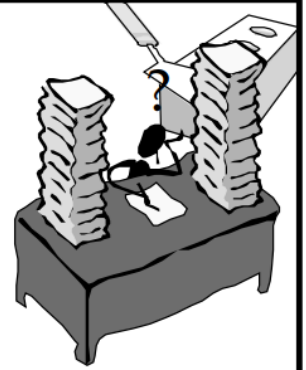
# *Why Use a DBMS?*



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

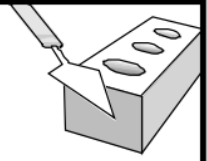
Why study Databases?

# 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

How is data in a DBMS retrieved and manipulated?



# 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 3ed, R. Ramakrishnan and J. Gehrke

6

data model

a collection of high-level data description constructs that hide many low-level storage details

schema

A description of data in terms of a data model

In the relational model, the schema for a relation specifies its name, the name of each **field** (or **attribute** or **column**), and the type of each field.

As an example, student information in a university database may be stored in a relation with the following schema:

`Students(sid: string, name: string, login: string,  
age: integer, gpa: real)`

The preceding schema says that each record in the Students relation has five fields, with field names and types as indicated. An example instance of the Students relation appears in Figure 1.1.

<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

**Figure 1.1** An Instance of the Students Relation

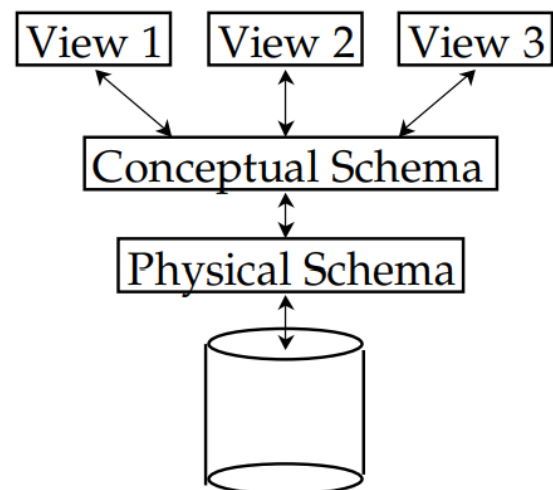
Every row follows the schema of the Students relation. The schema can therefore be regarded as a template for describing a student.

integrity constraints

conditions that the records in a relation must satisfy.

## Levels of Abstraction

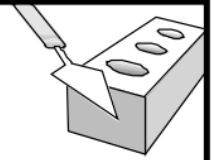
- ❖ Many views, single conceptual (logical) schema and physical schema.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used.



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

is used to define the external(view) and conceptual schemas

## 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*)

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

8

physical schema

specifies additional storage details.

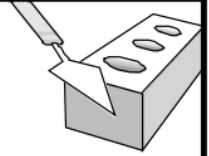
Essentially, the physical schema summarizes how the relations described in the conceptual schema are actually stored on secondary storage devices such as disks and tapes.

External schemas

which usually are also in terms of the data model of the DBMS, allow data access to be customized (and authorized) at the level of individual users or groups of users.

How does a DBMS support concurrent access and protect data during system failures?

# 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!*

## Data independence

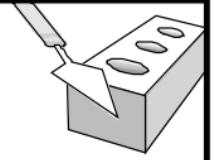
application programs are insulated from changes in the way the data is structured and stored

## Transaction

any one execution of a user program in 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 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.

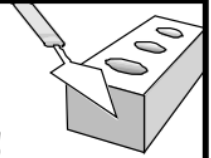
## Transaction: An Execution of a DB Program



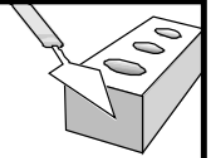
- ❖ Key concept is transaction, which is an *atomic* sequence of database actions (reads/writes).
- ❖ Each transaction, executed completely, must leave the DB in a consistent state if DB is consistent when the transaction begins.
  - Users can specify some simple integrity constraints on the data, and 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!



## 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

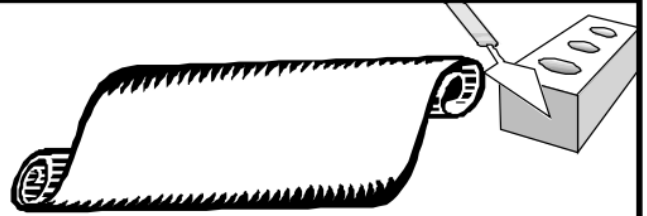
- ❖ DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of a Xact.
- ❖ Idea: Keep a log (history) of all actions carried out by the DBMS while executing a set of Xacts:
  - Before a change is made to the database, the corresponding log entry is forced to a safe location. (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 3ed, R. Ramakrishnan and J. Gehrke

13

A crucial property of the log is that each write action must be recorded in the log (on disk) before the corresponding change is reflected in the database itself—otherwise, if the system crashes just after making the change in the database but before the change is recorded in the log, the DBMS would be unable to detect and undo this change. This property is called **Write-Ahead Log**, or **WAL**.

# The Log

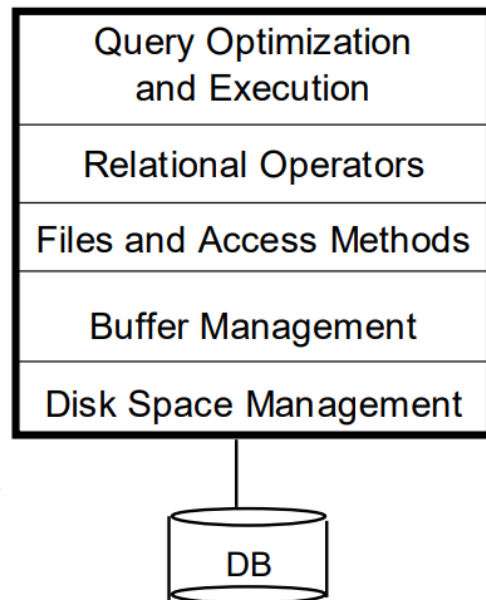


- ❖ The following actions are recorded in the log:
  - *Ti writes an object*: the old value and the new value.
    - Log record must go to disk before the changed page!
  - *Ti commits/aborts*: a log record indicating this action.
- ❖ Log records chained together by Xact id, so it's easy to undo a specific Xact (e.g., to resolve a deadlock).
- ❖ Log is often *duplexed* and *archived* on “stable” storage.
- ❖ All log related activities (and in fact, all CC related activities such as lock/unlock, dealing with deadlocks etc.) are handled transparently by the DBMS.

What are the main components of a DBMS?

# 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.

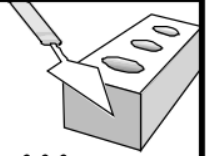


These layers must consider concurrency control and recovery

Who is involved with databases in real life?



## Databases make these folks happy ...

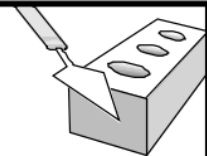


- ❖ End users and DBMS vendors
- ❖ DB application programmers
  - E.g. smart webmasters
- ❖ Database administrator (DBA)
  - Designs logical / physical schemas
  - Handles security and authorization
  - Data availability, crash recovery
  - Database tuning as needs evolve



*Must understand how a DBMS works!*

## 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.



