



APPLIED DATABASE MANAGEMENT SYSTEM BCSC0014



Dr. Neeraj Gupta
Assistant Professor, Dept. of CEA
neeraj.gupta@gla.ac.in



OUTLINE

- ☐ DBMS
- ☐ DBMS Applications
- ☐ File system vs Database system
- ☐ Data Abstraction
- ☐ Data Models and Their Categories
- ☐ Schemas, Instances, and States
- ☐ Three-Schema Architecture



DBMS

DBMS contains information about a particular enterprise

- Collection of interrelated data
- Set of programs to access the data
- An environment that is both *convenient* and *efficient* to use



shutterstock.com • 700346665

DATABASE APPLICATIONS

- Banking: transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Online retailers: order tracking, customized recommendations
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

Databases can be very large.

Databases touch all aspects of our lives

UNIVERSITY DATABASE EXAMPLE

Application program examples

- Add new students, instructors, and courses
- Register students for courses, and generate class rosters
- Assign grades to students, compute grade point averages (GPA) and generate transcripts

In the early days,

Database applications were built directly on top of file systems

DRAWBACKS OF USING FILE SYSTEMS TO STORE DATA

Data redundancy and inconsistency

- Multiple file formats, duplication of information in different files

Difficulty in accessing data

- Need to write a new program to carry out each new task

Data isolation

- Multiple files and formats

Integrity problems

- Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
- Hard to add new constraints or change existing ones

DRAWBACKS OF USING FILE SYSTEMS TO STORE DATA

Atomicity of updates

- Failures may leave database in an inconsistent state with partial updates carried out
- Example: Transfer of funds from one account to another should either complete or not happen at all

Concurrent access by multiple users

- Concurrent access needed for performance
- Uncontrolled concurrent accesses can lead to inconsistencies
 - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time

Security problems

- Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

DATA ABSTRACTION



Real Life Example of Abstraction



DATA MODELS

Data Model:

- A set of concepts to describe the **structure** of a database, the **operations** for manipulating these structures, and certain **constraints** that the database should obey.

Data Model Structure and Constraints:

- Constructs are used to define the database structure
- Constructs typically include **elements** (and their **data types**) as well as groups of elements (e.g. **entity, record, table**), and **relationships** among such groups
- Constraints specify some restrictions on valid data; these constraints must be enforced at all times



DATA MODELS

Data Model Operations:

- These operations are used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.
- Operations on the data model may include
 - ***basic model operations*** (e.g. generic insert, delete, update) and
 - ***user-defined operations*** (e.g. compute_student_gpa, update_inventory)



CATEGORIES OF DATA MODELS

Conceptual (high-level, semantic) data models:

- Provide concepts that are close to the way many users perceive data.
 - (Also called *entity-based* or *object-based* data models.)

Physical (low-level, internal) data models:

- Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals

Implementation (representational) data models:

- Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).



SCHEMAS VERSUS INSTANCES

Database Schema:

- The ***description*** of a database.
- Includes descriptions of the database structure, data types, and the constraints on the database.

Schema Diagram:

- An ***illustrative*** display of (most aspects of) a database schema.

Schema Construct:

- A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.



SCHEMAS VERSUS INSTANCES

Database State:

- The actual data stored in a database at a ***particular moment in time***. This includes the collection of all the data in the database.
- Also called database instance (or occurrence or snapshot).
 - The term *instance* is also applied to individual database components, e.g. *record instance*, *table instance*, *entity instance*



DATABASE SCHEMA VS. DATABASE STATE

Database State:

- Refers to the **content** of a database at a moment in time.

Initial Database State:

- Refers to the database state when it is initially loaded into the system.

Valid State:

- A state that satisfies the structure and constraints of the database.



DATABASE SCHEMA VS. DATABASE STATE

Distinction

- The **database schema** changes very infrequently.
- The **database state** changes every time the database is updated.

Schema is also called **intension**.

State is also called **extension**.



EXAMPLE OF A DATABASE SCHEMA

STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

Figure 2.1

Schema diagram for the database in Figure 1.2.



EXAMPLE OF A DATABASE STATE

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.



THREE-SCHEMA ARCHITECTURE

Proposed to support DBMS characteristics of:

- **Program-data independence.**
- Support of **multiple views** of the data.

Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization



THREE-SCHEMA ARCHITECTURE

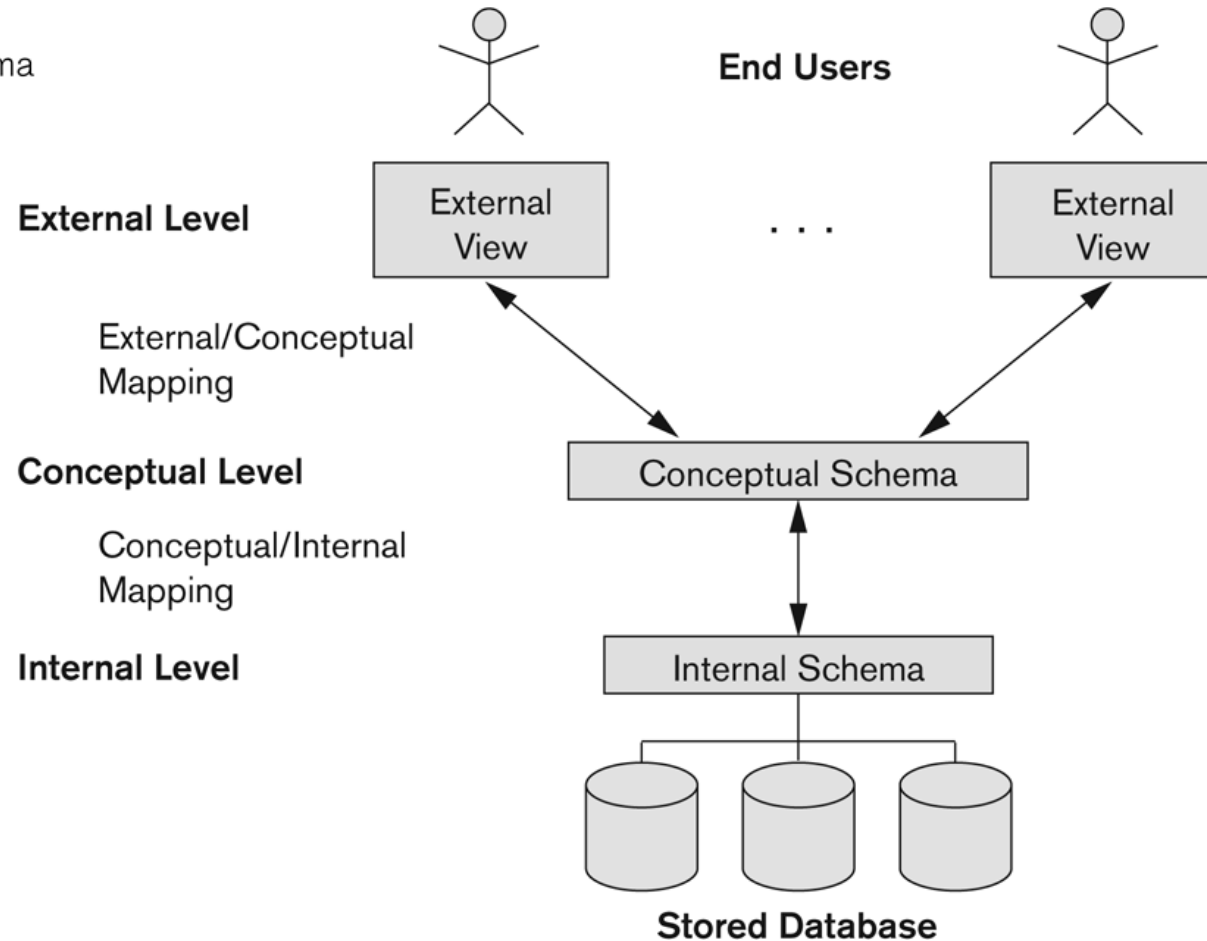
Defines DBMS schemas at **three** levels:

- **Internal schema** at the internal level to describe physical storage structures and access paths (e.g indexes).
 - Typically uses a **physical** data model.
- **Conceptual schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users.
 - Uses a **conceptual** or an **implementation** data model.
- **External schemas** at the external level to describe the various user views.
 - Usually uses the same data model as the conceptual schema.



THE THREE-SCHEMA ARCHITECTURE

Figure 2.2
The three-schema architecture.



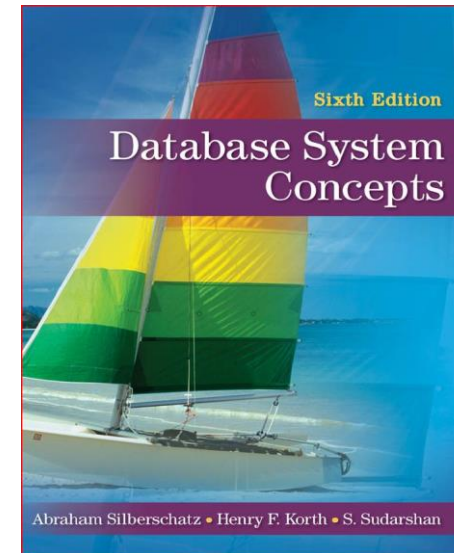
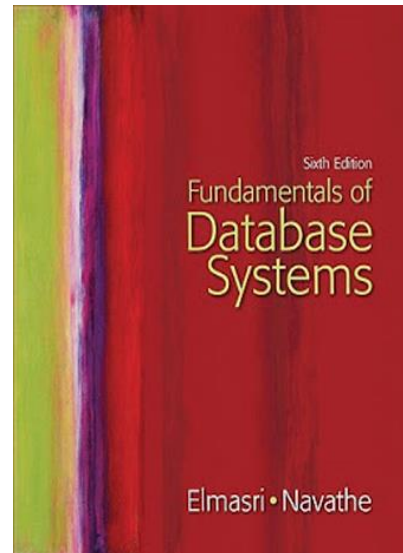
THREE-SCHEMA ARCHITECTURE

Mappings among schema levels are needed to transform requests and data.

- Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
- Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)



REFERENCE BOOKS





Keep Learning
Keep Growing

