Database Management Systems(CO301)

By

Dr. M.Venkatesan

Assistant Professor

Department of Computer Science and Engineering

National Institute of Technology Karnataka

Mangalore

Database Management Systems(CO301)

Credits (L-T-P): 04 (3-1-0)

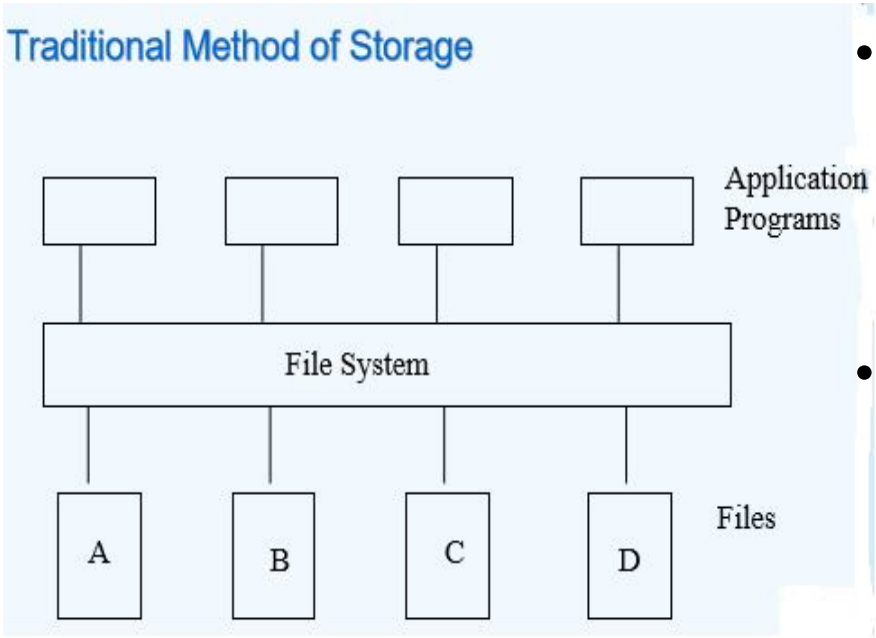
Content:

Introduction
E-R Models
Relational Models , Relational Algebra & Calculus
SQL Queries, programming and triggers
Data Storage, File Handling, Security,
Parallel & distributed data, Internet database,
Data Mining, Object Database systems, Real Time Database systems.

References:

1. R. Ramaksrishnan and Johannes G, "Database Management System", McGraw Hill Publishers.
2. Elmasri, Rames, Shamkant B Navathe, "Fundamentals of database systems", 2003.
3. J.O. Ullman, "Principles of Database systems", Galgotia Publishers.
4. Stamper and Price, "Database Design and Management-An Applied Approach", McGraw Hill Publications.

File System



- Store information in flat files which are maintained by the file system under the operating system's control.
- Application programs go through the file system to access these flat files

File System(cont..)

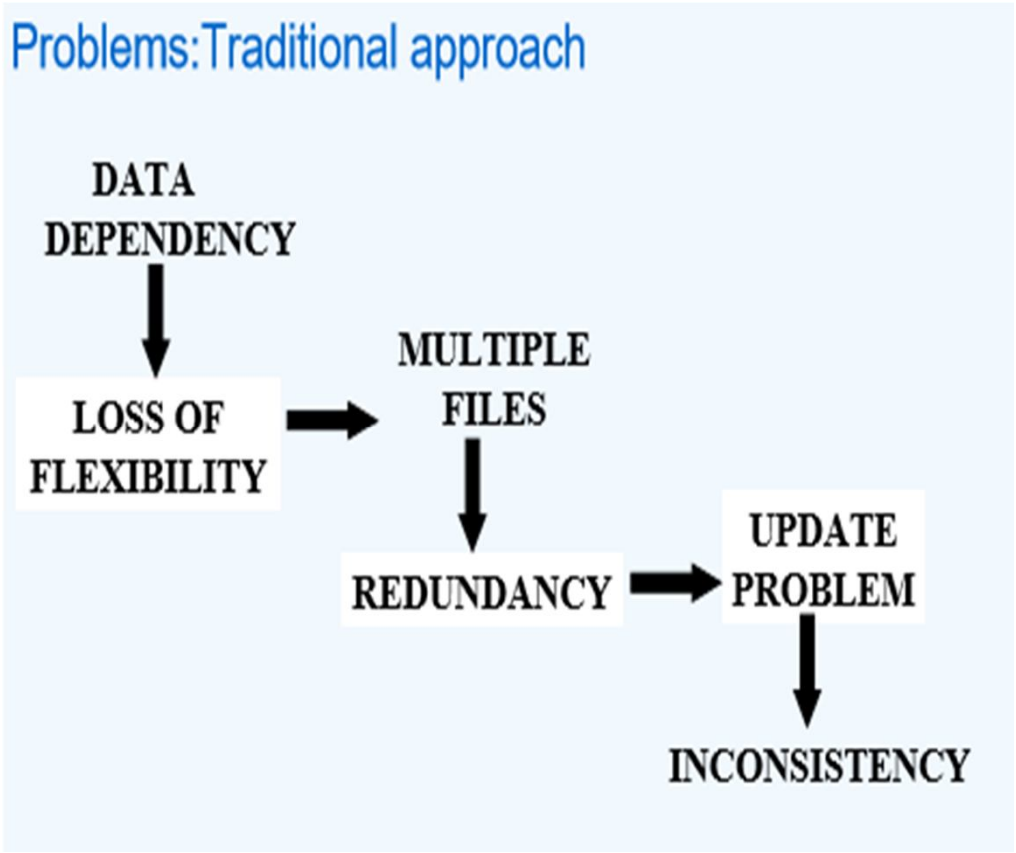
Ways of storing data in files

4176	Aniruddha Sarkar	SBU1
4181	Manoj Saha	SBU1
4183	Moushumi Dharchoudhury	SBU1
4203	Suryanarayana D.V.S.S.	SBU1
4204	Vivek Rai	SBU1

```
4176 AniruddhaSarkar SBU1
4181 ManojSaha SBU1
4183 MoushumiDharchoudhury SBU1
4203 SuryanarayanaD.V.S. SBU1
4204 Vivek Rai SBU1
```

- Data used to be stored in the form of records in the files.
- Records consist of various fields which are delimited by a space , comma , tab etc

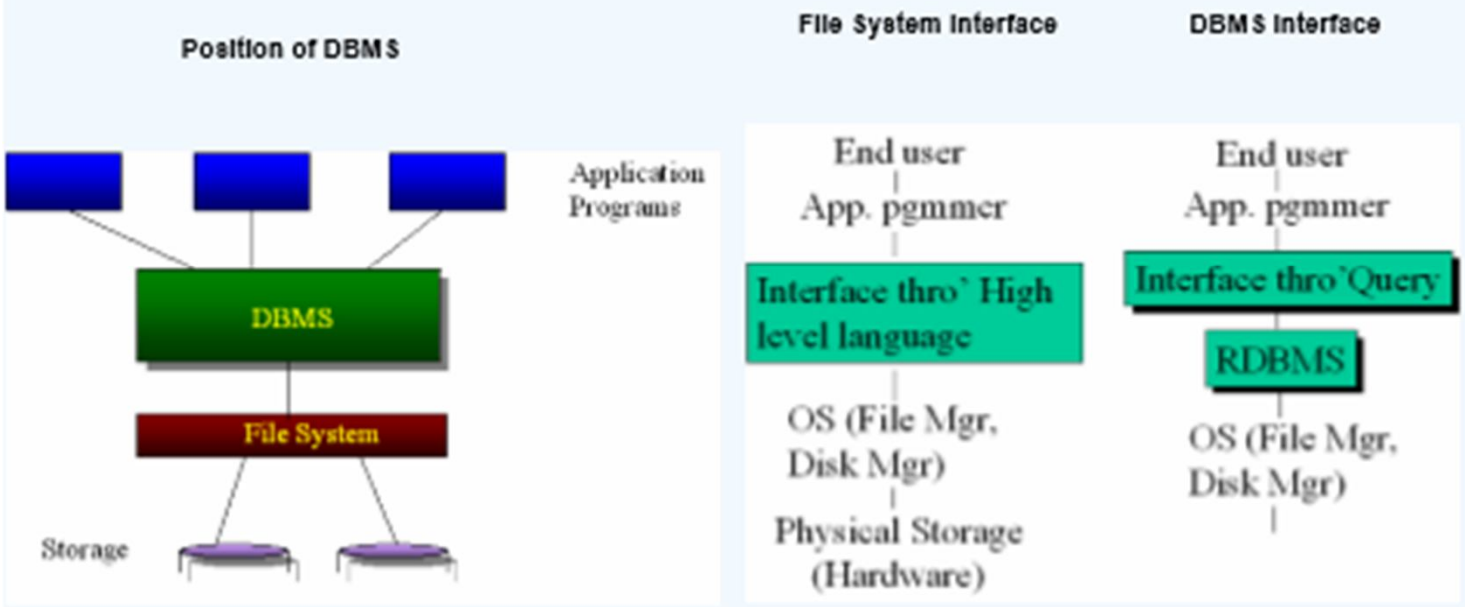
File System(cont..)



- Data spread across multiple files where dependent on each other . This led to loss of flexibility
- The same information was repeated in multiple files. This led to redundancy
- The inconsistent data

Role of DBMS

Where does the RDBMS fit in?



File Processing VS DBMS

File Processing

- data definition is part of application programs
- programs & data are interdependent

DBMS

- self-describing
- program-data independence
- support of multiple views of data
- provides concurrency control & transaction processing capabilities
- provides mechanisms for backup & recovery
- support for query languages
- provides access control

Data? Database? DBMS?

- **Database:**
  - A collection of related data.
- **Data:**
  - Known facts that can be recorded and have an implicit meaning.

- **Database Management System (DBMS):**

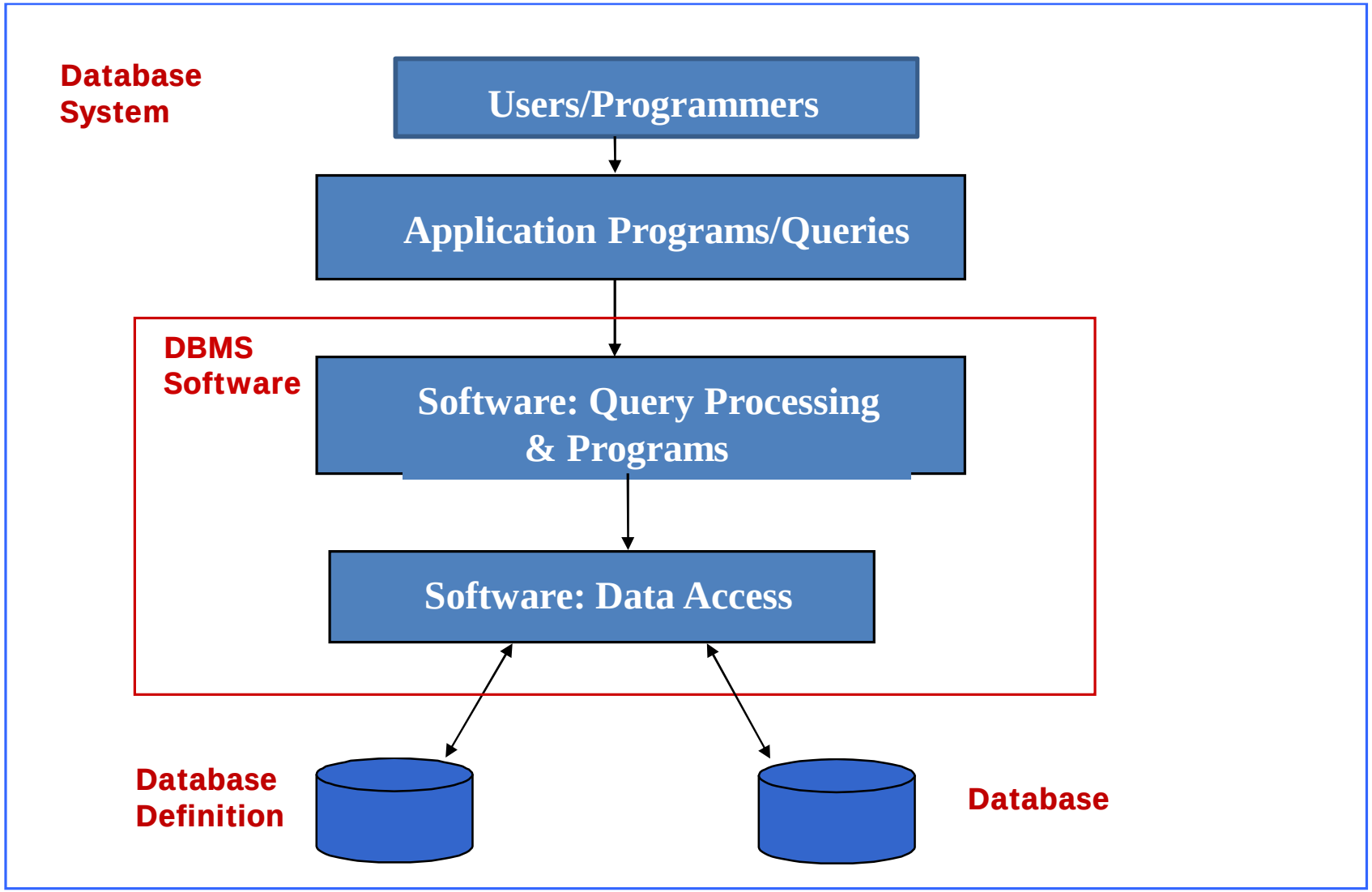
  A software package/ system to facilitate
  - **Define** – specify data types, structures & constraints for the data to be stored in the database
  - **Construct** – store the data
  - **Manipulate** – post queries to retrieve specific data, update data or generate reports based on the data

- **Database System:**
  - The DBMS software together with the data itself. Sometimes, the applications are also included.

Database Examples

- Company Databases
  - employees, departments, projects ...
- Airline Reservation Systems
  - flights, fares, customers, reservations ..
- Library Databases
  - authors, titles, publishers, videos ...
- Bank Databases
  - accounts, customers ...

Database System Environment



Desirable Capabilities

- control redundancy
- restrict access
- provide persistent storage for program objects & data structures
- permit inferencing & actions by using rules
- provide multiple user interfaces
- represent complex relationships among data
- enforce integrity constraints
- provide back-up & recovery

Main Characteristics of the Database Approach

- **Self-describing nature of a database system:**
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
  - The description is called **meta-data**.
  - This allows the DBMS software to work with different database applications.
- **Insulation between programs and data:**
  - Called **program-data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs

Main Characteristics of the Database Approach

- **Data Abstraction:**
  - A data model is used to hide storage details and present the users with a conceptual view of the database.
  - Programs refer to the data model constructs rather than data storage details
- **Support of multiple views of the data:**
  - Each user may see a different view of the database, which describes only the data of interest to that user.

Main Characteristics of the Database Approach

- **Sharing of data and multi-user transaction processing:**
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - *Concurrency control* within the DBMS guarantees that each **transaction** is correctly executed or aborted
  - *Recovery* subsystem ensures each completed transaction has its effect permanently recorded in the database
  - **OLTP** (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.

Historically....

1960's

- Hierarchical model (IBM's IMS)
- Network Model (CODASYL)

1970's

- Network DBMS's introduced
- Codd introduces Relational Model (1971)
- Chen develops ER model (1976)
- Relational DB's appear in late '70's

Historically...(con’t)

1980’s

- Relational DBMS’s dominate
- Preliminary SQL standard published
- Object-oriented concepts
- Distributed DBMS’s become an important area of research

1990’s

- Client-server takes over
- Legacy DBs become a major problem
- New areas such as data warehousing, multimedia emerge

Historically (con’t)

2000

- Databases survive Y2K!

2001

- IBM buys Informix
- DB2 and Oracle in major PR war for customers
- Industry depending heavily on DBMSs for everything
- Move back to mainframes and DBMSs for everything from data storage to web servers

Database Players

- DBA ($$$$$$)
  - access authorization, coordination & monitoring database usage, problem determination, performance tuning etc
- Designers
  - identify the requirements & chose the appropriate structures to represent & store the data
- Users
- System analysts & application programmers
- DBMS system designers & implementers
- Tool developers
- Operators & maintenance personnel

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 (continued)

- **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 (continued)

- 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

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

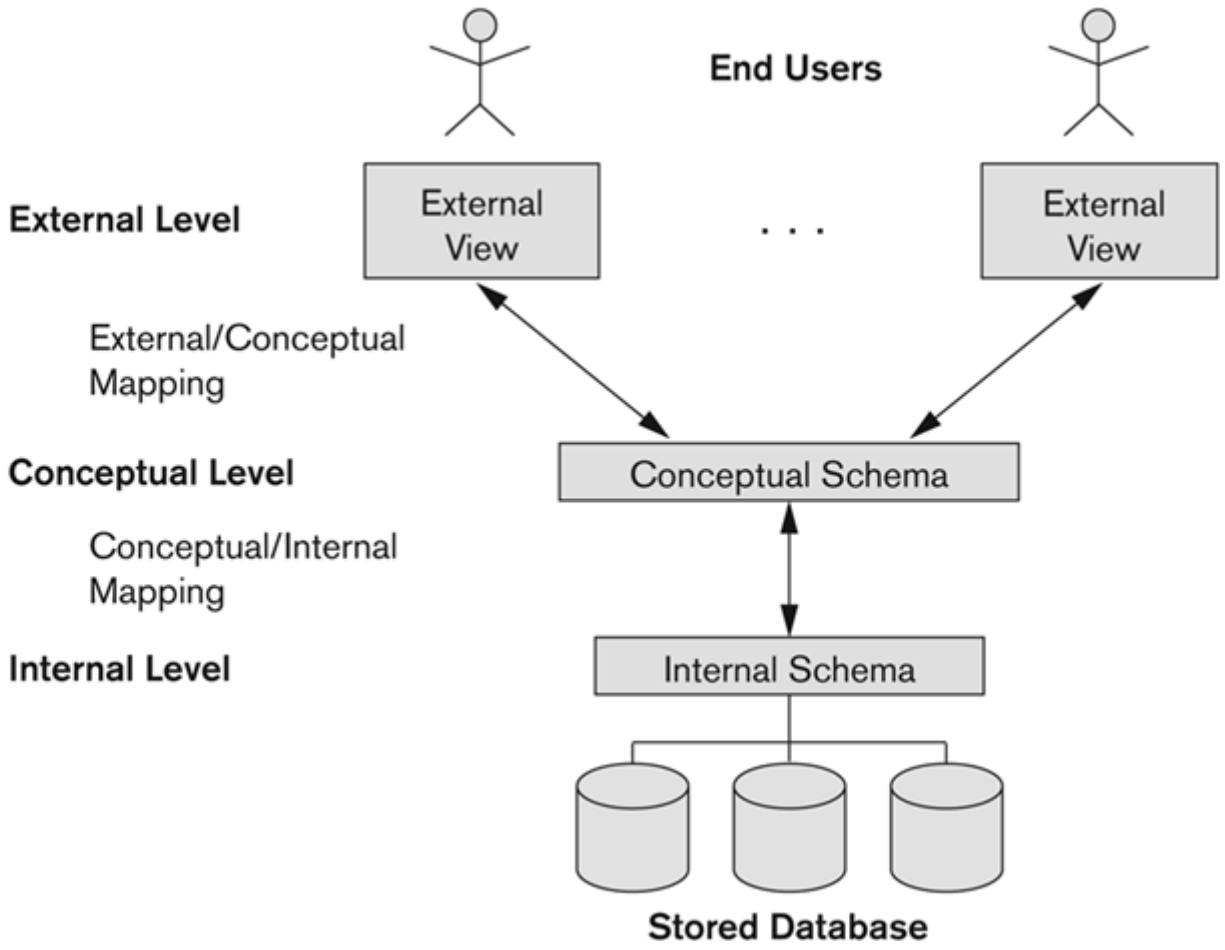
Three-Schema Architecture

- 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



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)

Data Independence

- **Logical Data Independence:**
  - The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.
- **Physical Data Independence:**
  - The capacity to change the internal schema without having to change the conceptual schema.
  - For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance

Data Independence (continued)

- When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence.
- The higher-level schemas themselves are **unchanged**.
  - Hence, the application programs need not be changed since they refer to the external schemas.

DBMS Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
  - High-Level or Non-procedural Languages: These include the relational language SQL
    - May be used in a standalone way or may be embedded in a programming language
  - Low Level or Procedural Languages:
    - These must be embedded in a programming language

DBMS Languages

- **Data Definition Language (DDL):**
  - Used by the DBA and database designers to specify the conceptual schema of a database.
  - In many DBMSs, the DDL is also used to define internal and external schemas (views).
  - In some DBMSs, separate **storage definition language (SDL)** and **view definition language (VDL)** are used to define internal and external schemas.
    - SDL is typically realized via DBMS commands provided to the DBA and database designers

DBMS Languages

- **Data Manipulation Language (DML):**
  - Used to specify database retrievals and updates
  - DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.
    - A library of functions can also be provided to access the DBMS from a programming language
  - Alternatively, stand-alone DML commands can be applied directly (called a *query language*).

Types of DML

- **High Level or Non-procedural Language:**
  - For example, the SQL relational language
  - Are “set”-oriented and specify what data to retrieve rather than how to retrieve it.
  - Also called **declarative** languages.
- **Low Level or Procedural Language:**
  - Retrieve data one record-at-a-time;
  - Constructs such as looping are needed to retrieve multiple records, along with positioning pointers.

DBMS Interfaces

- Stand-alone query language interfaces
  - Example: Entering SQL queries at the DBMS interactive SQL interface (e.g. SQL*Plus in ORACLE)
- Programmer interfaces for embedding DML in programming languages
- User-friendly interfaces
  - Menu-based, forms-based, graphics-based, etc.

User-Friendly DBMS Interfaces

- Menu-based, popular for browsing on the web
- Forms-based, designed for naïve users
- Graphics-based
  - (Point and Click, Drag and Drop, etc.)
- Natural language: requests in written English
- Combinations of the above:
  - For example, both menus and forms used extensively in Web database interfaces