METU Department of Computer Eng

Ceng 302 Introduction to DBMS

Basic Concepts

by

Pinar Senkul

resources: mostly froom Elmasri, Navathe

and other books

Data Models

- **Data Model**: A set of concepts to describe the *structure* of a database, and certain *constraints* that the database should obey.

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.)
  e.g. ER model
- **Implementation** (**representational**) data models: Provide concepts that fall between the above two, balancing user views with some computer storage details.

  e.g. Relational model, OO model, network model
- **Physical** (**low-level**, **internal**) data models: Provide concepts that describe details of how data is stored in the computer.

History of Data Models

- Network Model: the first one to be implemented by Honeywell in 1964-65 (IDS System). Adopted heavily due to the support by CODASYL (CODASYL - DBTG report of 1971). Later implemented in a large variety of systems - IDMS (Cullinet - now CA), DMS 1100 (Unisys), IMAGE (H.P.), VAX -DBMS (Digital Equipment Corp.).
- Hierarchical Data Model: implemented in a joint effort by IBM and North American Rockwell around 1965. Resulted in the IMS family of systems. The most popular model. Other system based on this model: System 2k (SAS inc.)
- Relational Model: proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82. Now in several commercial products (DB2, ORACLE, SQL Server, SYBASE, INFORMIX).

History of Data Models

- Object-oriented Data Model(s): several models have been proposed for implementing in a database system. One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE). Additionally, systems like $O_{2,}$ ORION (at MCC - then ITASCA), IRIS (at H.P.- used in Open OODB).
- Object-Relational Models: Most Recent Trend. Started with Informix Universal Server. Exemplified in the latest versions of Oracle-10i, DB2, and SQL Server etc. systems.

Network Model

- ADVANTAGES:
  - Network Model is able to model complex relationships and represents semantics of add/delete on the relationships.
  - Can handle most situations for modeling using record types and relationship types.
  - Language is navigational; uses constructs like FIND, FIND member, FIND owner, FIND NEXT within set, GET etc. Programmers can do optimal navigation through the database.
- DISADVANTAGES:
  - Navigational and procedural nature of processing
  - Database contains a complex array of pointers that thread through a set of records.
    Little scope for automated "query optimization”

Hierarchical Model

- ADVANTAGES:
  - Hierarchical Model is simple to construct and operate on
  - Corresponds to a number of natural hierarchically organized domains - e.g., assemblies in manufacturing, personnel organization in companies
  - Language is simple; uses constructs like GET, GET UNIQUE, GET NEXT, GET NEXT WITHIN PARENT etc.
- DISADVANTAGES:
  - Navigational and procedural nature of processing
  - Database is visualized as a linear arrangement of records
  - Little scope for "query optimization"

Relational Model

- Data is described as a set of **relations** (can be thought of as a set of **records**, or a **table** of values)
- Records are not considered to be linear (as opposed to previous models), therefore access to the data is more efficient
- Sophisticated algorithms for query optimization

Schemas vs. Instances

- **Database Schema**: The *description* of a database. Includes descriptions of the database structure and the constraints that should hold on the database.
- **Schema Diagram**: A diagrammatic display of (some aspects of) a database schema.
- **Database Instance**: The actual data stored in a database at a *particular moment in time*. Also called **database state** (or **occurrence**).

Database Schema Vs. Database State

- **Database State:** Refers to the content of a database at a moment in time (snapshot).
- **Initial Database State:** Refers to the database when it is loaded
- **Valid State:** A state that satisfies the structure and constraints of the database.
- **Distinction**
  - The **database schema** changes *very infrequently*. The **database state** changes *every time the database is updated.*
  - **Schema** is also called **intension**, whereas **state** is called **extension**.

Three-Schema Architecture

- Defines DBMS schemas at *three levels*:
  - **Internal (physical) schema** at the internal level to describe physical storage structures and access paths. 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 level.

Three-Schema Architecture

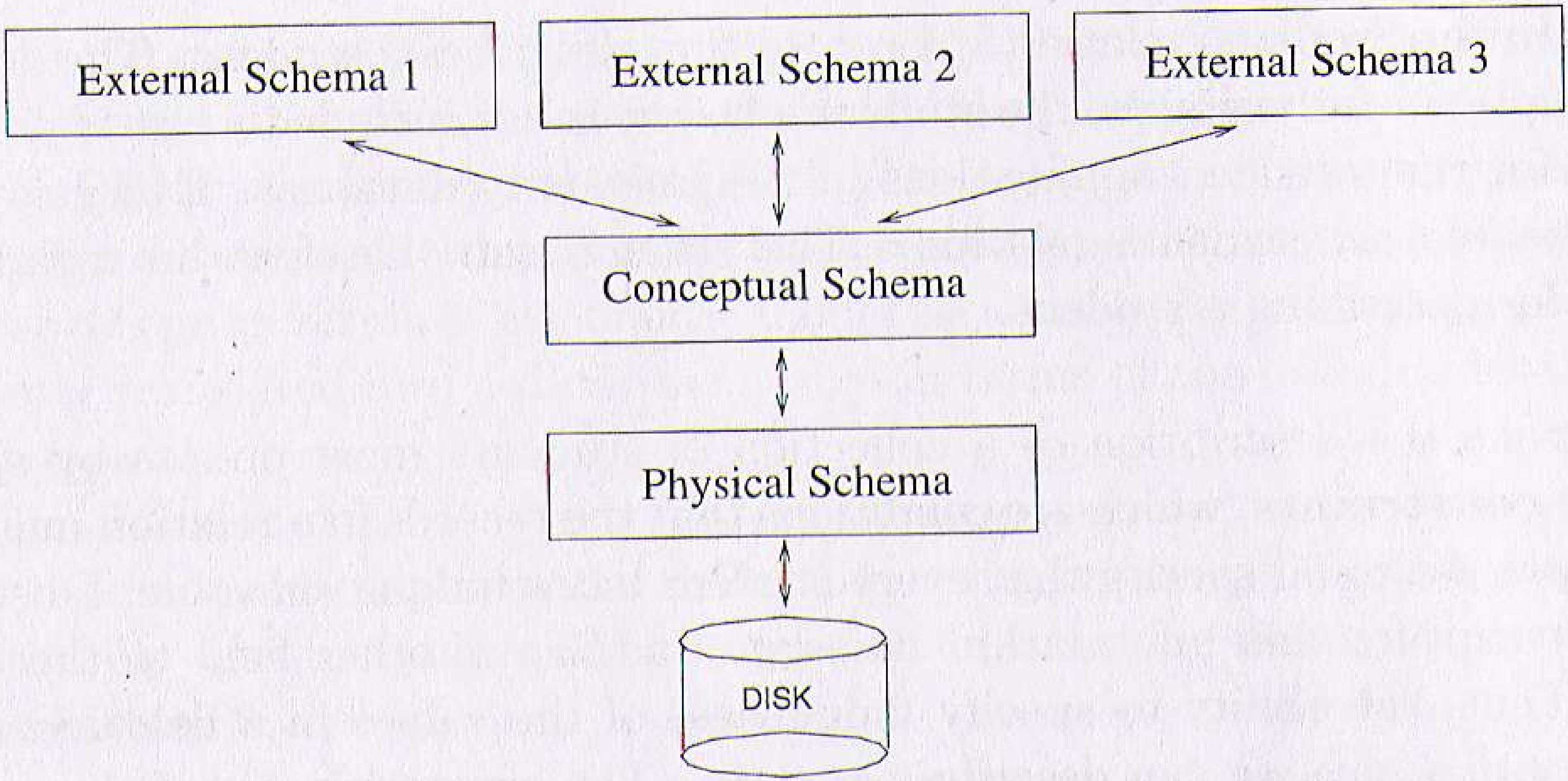

Figure 1.2 Levels of Abstraction in a DBMS

Data Independence

- **Logical Data Independence**: The capacity to change the conceptual schema without having to change the external schemas and their application programs.
- **Physical Data Independence**: The capacity to change the internal schema without having to change the conceptual schema.

DBMS Languages

- **Data Definition Language (DDL)**: 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.

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 or an Assembly Language.
  - Alternatively, *stand-alone* DML commands can be applied directly (**query language**).

DBMS Languages

- SQL is the relational database language
- It contains DDL, VDL, DML
- SDL was a component in the early versions but it has been removed in the later versions so that SQL becomes a language for external and conceptual levels only.

Transaction Management

- Transaction: atomic execution of a user program in DBMS (sequence of read and write operations)
- DBMS should schedule the concurrent transactions so that each user can safely ignore the fact that others are accessing the data concurrently.
- To provide a correct interleaving of transactions, **locking protocol** is used

Recovery

- In case of failure, DBMS must ensure the correctness of the date
- The results of complete transactions should still hold.
- The effect of incomplete transactions should be undone.
- To provide this DBMS keeps the log of write operations.

Structure of a DBMS

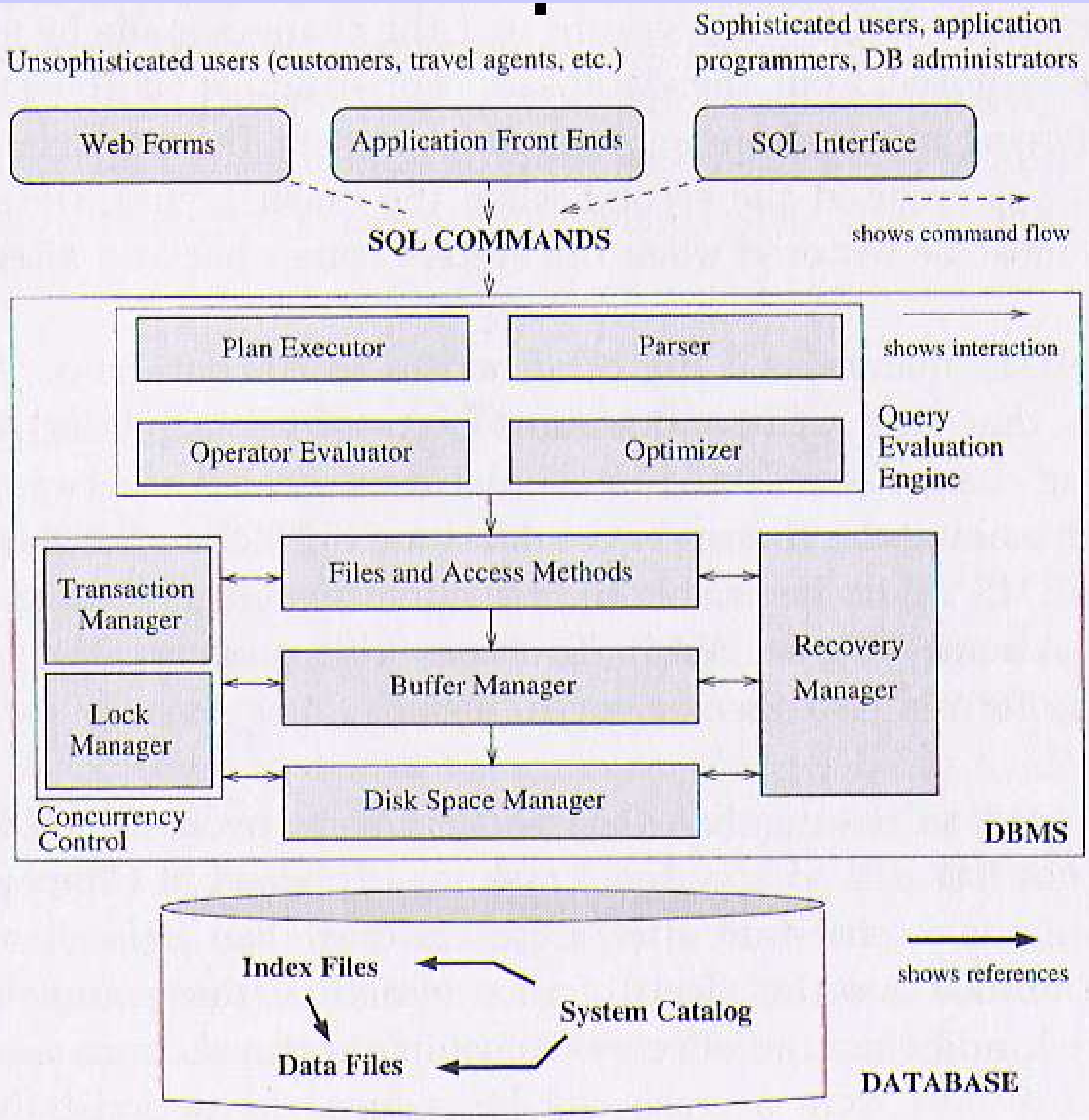

Figure 1.3 Architecture of a DBMS

Centralized and Client-Server Architectures

- **Centralized DBMS:** combines everything into single system including- DBMS software, hardware, application programs and user interface processing software.

Basic Client-Server Architectures

- Specialized Servers with Specialized functions
- Clients
- DBMS Server

Specialized Servers with Specialized functions:

- File Servers
- Printer Servers
- Web Servers
- E-mail Servers

Clients:

- Provide appropriate interfaces and a client-version of the system to access and utilize the server resources.
- Clients maybe diskless machines or PCs or Workstations with disks with only the client software installed.
- Connected to the servers via some form of a network.
  (LAN: local area network, wireless network, etc.)

DBMS Server

- Provides database query and transaction services to the clients
- Sometimes called query and transaction servers

Two Tier Client-Server Architecture

- **User Interface Programs and Application Programs** run on the client side
- Interface called **ODBC (Open Database Connectivity)** provides an Application program interface (API) allow client side programs to call the DBMS. Most DBMS vendors provide ODBC drivers.

Two Tier Client-Server Architecture

- A client program may connect to several DBMSs.
- Other variations of clients are possible: e.g., in some DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc. In such situations the server may be called the **Data Server**.

Three Tier Client-Server Architecture

- Common for **Web applications**
- Intermediate Layer called **Application Server** or **Web Server:**
  - stores the web connectivity software and **the rules and business logic (constraints)** part of the application used to access the right amount of data from the database server
  - acts like a conduit for sending partially processed data between the database server and the client.
- **Additional Features- Security:**
  - encrypt the data at the server before transmission
  - decrypt data at the client

Classification of DBMSs

- **Based on the data model used**:
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented, Object-relational.
- **Other classifications:**
  - Single-user (typically used with micro- computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed /client-server(uses multiple computers, multiple databases)

Variations of Distributed Environments:

- Homogeneous DDBMS
- Heterogeneous DDBMS
- Federated or Multidatabase Systems