Database Management System (15ECSC208)

UNIT I: Chapter 1: Introduction to DBMS and ER-Model

Database System Concepts and Architecture

Part 2

Data Models

- Data abstraction refers to suppression of details of data organization, storage, and essential features for an improved understanding of data.
- **Data model provides means to achieve data abstraction.**
- Data model is collection of concepts that describe the structure of a database(data type, relationships, and constraints).

- **Basic operations**
  - Specify retrievals and updates on the database
- **Dynamic aspect or behavior** of a database application

Categories of Data Models

- **High-level or conceptual data models**
  - Close to the way many users perceive data
- **Low-level or physical data models**
  - Describe the details of how data is stored on computer storage media
- **Representational (implementation) data models**
  - Easily understood by end users
  - Also similar to how data organized in computer storage
  - Hide many details of data storage on disk but can be implemented on a computer system

Conceptual Data Model

- **Entity**
  - Represents a real-world object or concept
- **Attribute**
  - Represents some property of interest
  - Further describes an entity
- **Relationship among two or more entities**
  - Represents an association among the entities
  - **Entity-Relationship model**

Other Categories of Data Models

- **Relational data model**
  - Represent data by using record structures
- **Object data model**
  - New family of higher-level implementation data models Closer to conceptual data models
- **Physical data model**
  - Describe how data is stored in the computer by representing information such as record formats, record orderings, and access paths.
- **Access path :** Structure that makes the search for particular database records efficient

Schemas, Instances, and Database State

Distinguish between desc. of db and db itself

- **Database schema**
  - Description of a database
- **Schema diagram**
  - Displays selected aspects of schema
- **Schema construct**
  - Each object in the schema
- **Database state or snapshot**
  - Data in database at a particular moment in time

Example: **Database schema** diagram for University database

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: **Database state** for University database

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

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	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	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

Database Schema Vs Database State

- **Define a new database**
  - Specify database schema to the DBMS
- **Database state – *empty state***, with no data
- **Initial state**
  - **Populated or loaded with the initial data**
- **Database state – *current state*,** with every update operation on the database
- **Valid state**
  - Satisfies the structure and constraints specified in the schema
- Database schema – **intension**, and Database state – **extension**
- **Schema evolution**
  - Changes applied to schema as application requirements change

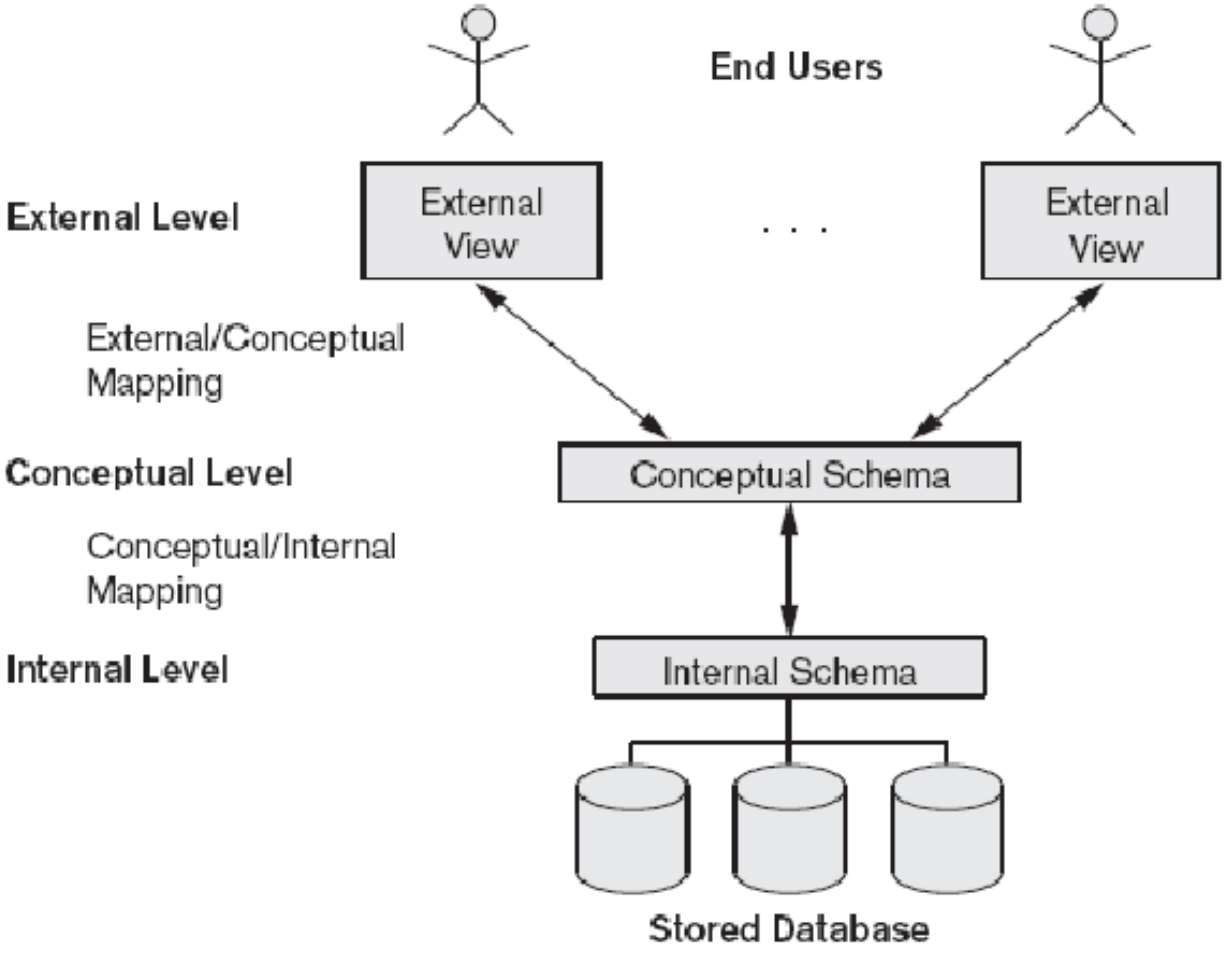
Three-Schema Architecture

- **Internal level**
  - Describes physical storage structure of the database
  - Describes complete details of data storage and access paths for the database
  - Uses physical data model
- **Conceptual level**
  - Describes structure of the whole database for a community of users
  - Hides details of physical storage structure and concentrates on describing entities, data types, relationships, user operations and constraints
  - Uses high-level data model or implementation data model
- **External or view level**
  - Describes part of the database that a particular user group is interested in
  - Uses high-level data model or implementation data model

Three-Schema Architecture

- **Internal level**
  - Describes physical storage structure of the database
  - Describes complete details of data storage and access paths for the database
- **Conceptual level**
  - Describes structure of the whole database for a community of users
  - Hides details of physical storage structure
- **External or view level**
  - Describes part of the database that a particular user group is interested in

Three-Schema Architecture



Data Independence

- Capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

1. Logical data independence

- The capacity to change the conceptual schema without affecting the external schemas or application programs
- Conceptual schema could be changed:
  - to expand the database – by adding a new attribute to some relation
  - to reduce the database – by deleting an attribute

Data Independence

2. Physical data independence

- The capacity to modify physical schema without affecting the logical or external schema.
- Performance tuning (retrieval or update) – modification at physical level or creating a new index etc.

DBMS Languages

- **Data definition language (DDL)**
  - Defines schema
- **Storage definition language (SDL)**
  - Specifies the internal schema
- **View definition language (VDL)**
  - Specifies user views/mappings to conceptual schema
- **Data manipulation language (DML)**
  - Allows retrieval, insertion, deletion, modification

Types of DMLs

- **High-level or nonprocedural DML**
  - Requires a user to specify what data are needed without specifying how to get those data.
  - Can be used on its own to specify complex database operations concisely
  - **Set-at-a-time or set-oriented**
- **Low-level or procedural DML**
  - Requires a user to specify what data are needed and how to get those data.
  - Must be embedded in a general-purpose programming language
  - **Record-at-a-time**

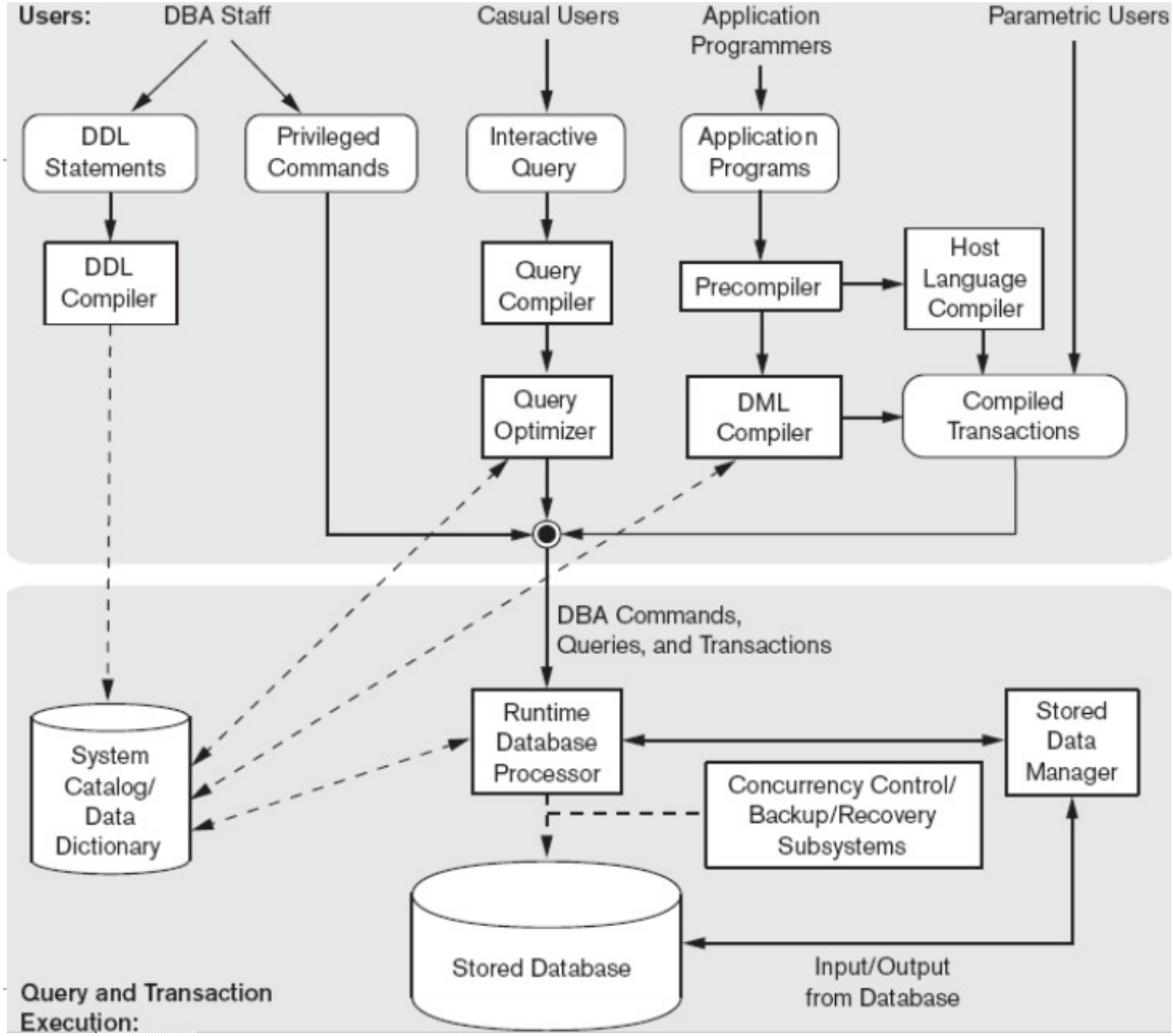
DBMS Interfaces

- Menu-based interfaces for Web clients or browsing
- Forms-based interfaces
- Graphical user interfaces
- Natural language interfaces
- Interfaces for parametric users
- Interfaces for the DBA

Database System Environment

- **DBMS component modules**
  - Stored data manager
  - DDL compiler
  - Runtime database processor
  - Interactive query interface
    - Query compiler
    - Query optimizer
  - Precompiler
  - System catalog
  - Concurrency control system
  - Backup and recovery system

Component modules of a DBMS and their interactions



Database System Utilities

- **Loading**
  - Load existing data files into the database
- **Backup**
  - Creates a backup copy of the database
- **Database storage reorganization**
  - Reorganize a set of database files into different file organizations
- **Performance monitoring**
  - Monitors database usage and provides statistics to the DBA

Tools

- **CASE Tools**
- **Data dictionary (data repository) system**
  - Stores design decisions, usage standards, application program descriptions, and user information

Extras

Basic Client/Server Architectures

- **Client**
  - User machine that provides user interface capabilities and local processing
- **Server**
  - System containing both hardware and software
  - Provides services to the client machines
    - Such as file access, printing, archiving, or database access

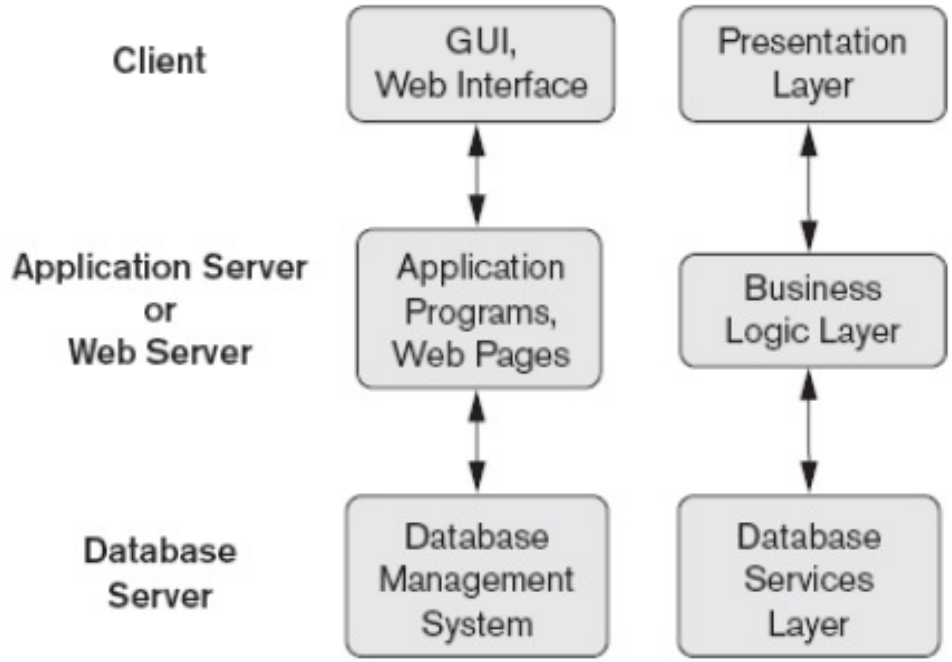
Two-Tier Client/Server Architectures for DBMSs

- Server handles
  - Query and transaction functionality related to SQL processing
- Client handles
  - User interface programs and application programs
- Open Database Connectivity (ODBC)
  - Provides application programming interface (API)
  - Allows client-side programs to call the DBMS
    - Both client and server machines must have the necessary software installed
- JDBC
  - Allows Java client programs to access one or more DBMSs through a standard interface

Logical Three-Tier Client/Server Architectures for DBMSs

Application server or Web server

- Adds intermediate layer between client and the database server
- Runs application programs and stores business rules



Resources

- Chapter 2 (Sec. 2.1 to Sec 2.4) of Fundamentals of Database Systems (FODS), 6th Edition.
- Internet Surf

- (Most of the slides adapted from Ramez Elmasri and Shamkant Navathe , FODS)