Data and Applications

Kamal Karlapalem

kamal@iiit.ac.in

F35, DSAC, KRB

Course Details

Timings:

Monday, Thursday

9-10:30 AM

Lab Time – Thursday 3:30-6:30 PM

Text Book: Elmasri & Navathe,

Fundamentals of Database Systems, 7th Edition, Pearson. 2017.

Course Details (Order)

1 – Introduction/Database Concepts
2 – Data Modeling
3 – Relational Model
4 – Normalization
4– Algebra/Calculus/SQL

Course Details - Project

Database Design Project

- four phases
- group of 3 members

1 – Requirements Analysis – Due 1st Sept 2021, Wednesday.

2 – Conceptual Design – Due 14th Sept 2021, Wednesday

3 – Logical Design – Due 29th Sept 2021, Wednesday

4 – Application & SQL – Due 4th October 2021 Monday

Group members fixed. They will also submit group assignment.

Course Grading

Grade maximizer - the weight-age is decided to be student specific, to maximize the total marks of the student

Type of Evaluation	Weightage (in %)	Total Weightage
Assignments	At least 5% at most 15%	15% to 70%
Quizzes	At least 10% at most 55%	
Lab Exam	At least 5% at most 20%	35% to 50%
Project	At least 30%	

Group Assignment

- We will give a password protected pdf file of questions.
- You have limited time to answer once you start.
- The group members discuss, get the answer, and upload a pdf file of handwritten answers.
- One submission for each group.

Quiz

- Quizzes test individual learning.
- Quizzes can be multiple choice, true or false, and fill in the blanks.
- Negative marking for getting wrong answer, and negative marking for not attempting the question.
- Best of the quiz scores may be taken, depending on the performance.

Course Details

Academic Honesty:

Unless otherwise stated, all work submitted by you should be your own. Copying of assignments, help taken, or given in debugging of programs, or sharing of algorithms, data, and results would constitute cheating. For paper critiques, presentations, and project reports, any form of cut and paste from various on-line and off-line resources is prohibited. If there is any doubt about the appropriateness of your actions, please contact the instructor for explicit clarification and confirmation by an email. Cheating is an offence and will result in an appropriate disciplinary action against those involved as per institute procedures.

Motivation For This Course

Database Systems

Second Largest Software Sales….

Largest?

Most application processing jobs….

Success story for Research and Development

Three Turing Awards

Bachmann (1973), Codd (1981) & Gray (1998); Michael Stonebraker (2015)

Database and Applications - first course compulsory

Data and Integration, Data Systems, Cloud Data Systems, Data Analytics I and II, Data Visualization

What will you learn?

Relational Database Systems

SQL

Database design process

Data Models, Normalization

File Structures and Indexing

Query Processing and Optimization

Transaction Management

Concurrency Control and Recovery

What Are Database Systems?

What is a Database?

Data: factual (undoubted) information that can be recorded and have implicit meaning.

A database is a collection of related data.

What is a Database?

A database has the following implicit properties:

- **A database represents some aspect of the real world (mini-world or Universe of Discourse (UoD)).**
- **A database is a logically coherent (associated, related) collection of data with some inherent meaning.**
- **A database is designed, built and populated with data for a specific purpose.**
- **It has an intended group of users and some preconceived (already thought of) applications in which these users are interested.**

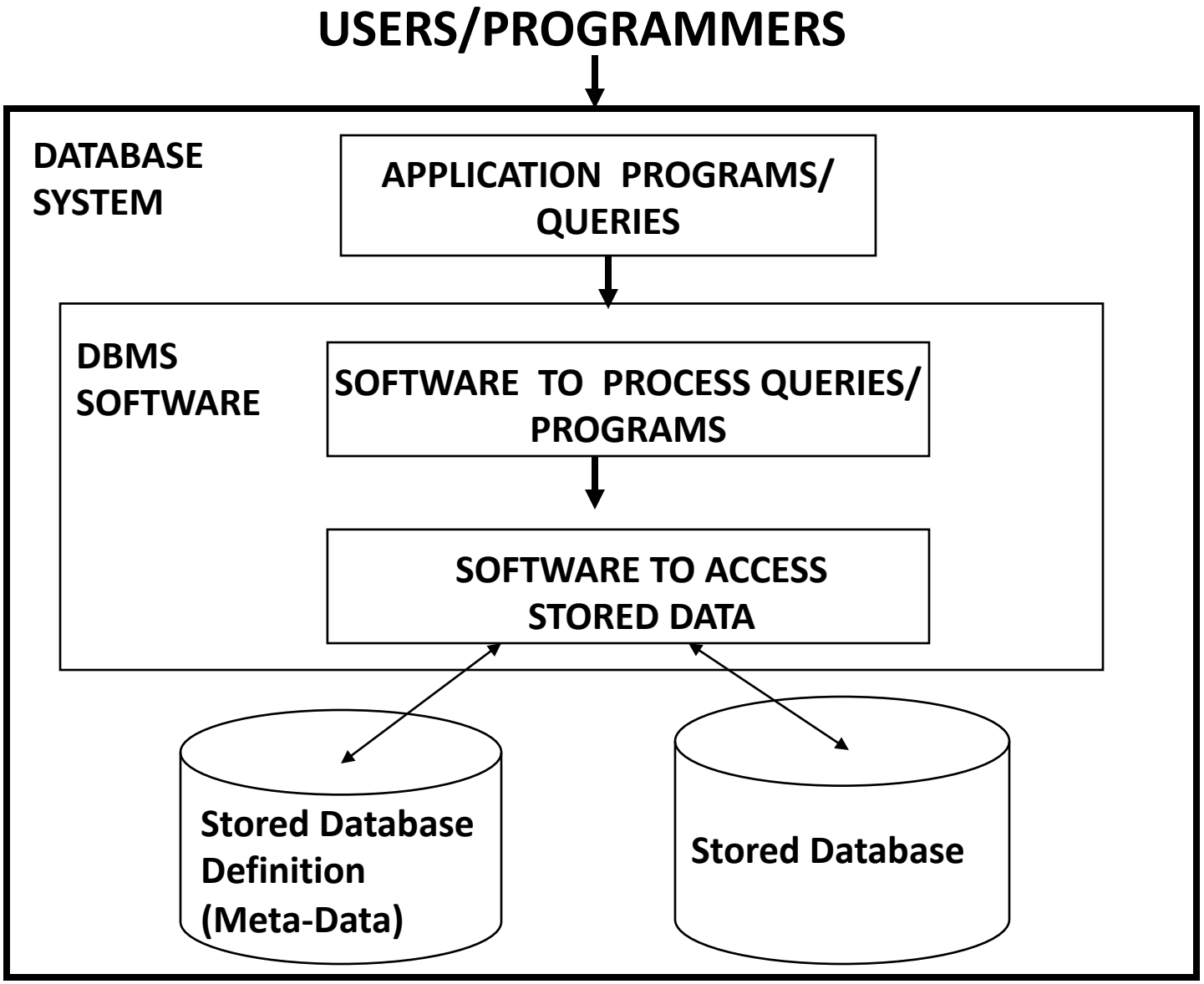
Database System

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database.

- **Defining Databases**
  - **involves specifying the data types, structures, and constraints for the data to be stored in the database.**
- **Constructing Databases**
  - **storing the data itself (populating) on some storage medium that is controlled by the DBMS.**
- **Manipulating Databases**
  - **querying the database to retrieve specific data, updating the databases to reflect changes to mini-world, and generating reports from the data.**

Database + DBMS software = Database System

Simplified Database System



Example of a Database

Consider a part of a University environment

We need data about:

- **STUDENTs**
- **COURSEs**
- **SECTIONs (of COURSEs)**
- **(academic) DEPARTMENTs**
- **INSTRUCTORs**

The above data is related as follows:

- **SECTIONs are of specific COURSEs**
- **STUDENTs take SECTIONs**
- **COURSEs have prerequisite COURSEs**
- **INSTRUCTORs teach SECTIONs**
- **COURSEs are offered by DEPARTMENTs**
- **STUDENTs major in DEPARTMENTs**

Example Database

STUDENT			
Name	Student Number	Class	Major
Smith	17	1	COSC
Brown	8	2	COSC

GRADE REPORT		
Student Number	Section-Identifier	Grade
17	85	A
18	102	B+

PREREQUISITE	
Course Number	Prerequisite Number
COSC3380	COSC3320
COSC3320	COSC1310

COURSE			
Course Name	Course Number	Credit Hours	Department
Intro to CS	COSC1310	4	COSC
Data Structures	COSC3320	4	COSC
Discrete Mathematics	MATH2410	3	MATH
Data Base	COSC3380	3	COSC

SECTION				
Section-Identifier	Course Number	Semester	Year	Instructor
85	MATH2410	Fall	91	King
92	COSC1310	Fall	91	Anderson
102	COSC3320	Spring	92	Knuth
135	COSC3380	Fall	92	Stone

Characteristics of the Database Approach

A number of characteristics distinguish the database approach from the traditional approach of programming with files.

In traditional file processing, each user defines and implements the files needed for a specific application.

In the database approach, a single repository of data is maintained that is defined once and then is accessed by various users.

Self describing nature of a Database System: the database system contains not only the database itself but also a complete definition (meta-data) of the database which is stored in the system catalog.

Characteristics of the Database Approach

Insulation between programs and data, and Data Abstraction: the structure of data files is stored separately from the access programs (program-data independence).

The characteristic that allows program-data independence is called data abstraction. A DBMS provides users with a conceptual representation of data that does not include many of the details of how the data is stored.

- **Support for multiple views of the data**
- **Sharing of Data and Multiuser Transaction Processing**

A typical database management system has all the above characteristics, whereas a typical file processing system does not.

Actors on Scene

The persons whose jobs involve the day-to-day use of a large database

- **Database administrators**
- **Database Designers**
- **End Users: Casual, Naive, Sophisticated and Stand-alone users**
- **System Analysts and Application Programmers**

Workers Behind the Scene:

- **DBMS Designers and Implementers**
- **Tool Developers**
- **Operators and Maintenance Personnel**

Intended Uses of a DBMS

- Controlling Redundancy
- Restricting Unauthorized Access
- Representing complex relationships among data
- Enforcing Integrity constraints
- Providing Backup and Recovery
- Providing multiple User Interfaces
- Persistent Storage for Program Objects and Data Structures
- Database Interfacing Using Deduction Rules

Intended Uses of a DBMS

Implications of the Database Approach

- **Reduced application development time**
- **Flexibility**
- **Available up-to-date Information**
- **Economics of Scale**

When not to use a DBMS

Main costs of using a DBMS:

- **High initial investment and possible need for additional hardware.**
- **Overhead for providing generality, security, recovery, integrity, and concurrency control.**

When a DBMS may be unnecessary

- **If the database and applications are simple, well defined, and not expected to change.**
- **If there are stringent real-time requirements that may not be met because of DBMS overhead.**
- **If access to data by multiple users is not required.**

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

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

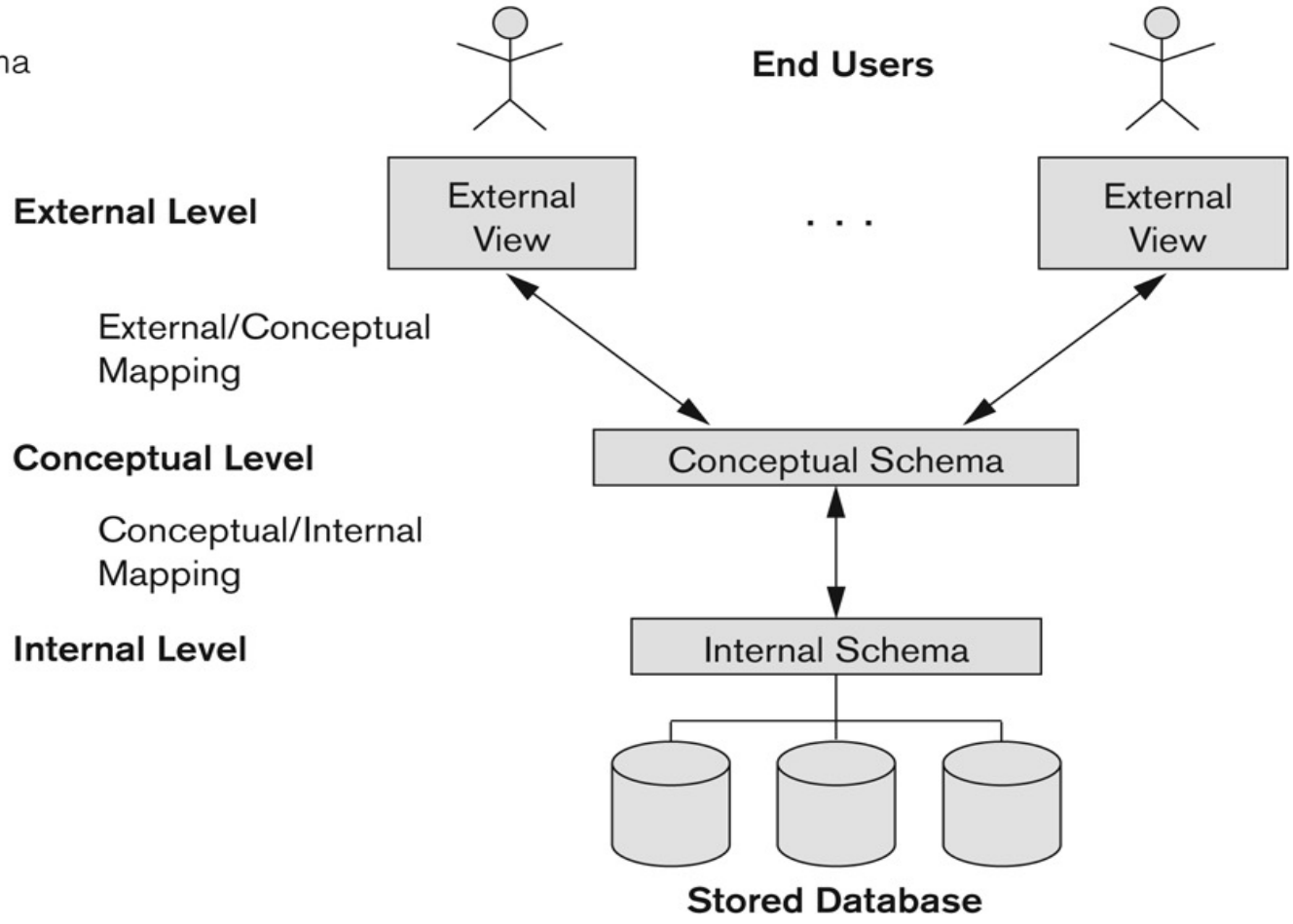
- Proposed to support DBMS characteristics of:
  - **Program-data independence.**
  - Support of **multiple views** of the data.
- Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization

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)

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.

DBMS Programming Language Interfaces

- Programmer interfaces for embedding DML in a programming languages:
  - **Embedded Approach**: e.g embedded SQL (for C, C++, etc.), SQLJ (for Java)
  - **Procedure Call Approach**: e.g. JDBC for Java, ODBC for other programming languages
  - **Database Programming Language Approach**: e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components

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

Other DBMS Interfaces

- Speech as Input and Output
- Web Browser as an interface
- Parametric interfaces, e.g., bank tellers using function keys.
- Interfaces for the DBA:
  - Creating user accounts, granting authorizations
  - Setting system parameters
  - Changing schemas or access paths

Database System Utilities

- To perform certain functions such as:
  - Loading data stored in files into a database. Includes data conversion tools.
  - Backing up the database periodically on tape.
  - Reorganizing database file structures.
  - Report generation utilities.
  - Performance monitoring utilities.
  - Other functions, such as sorting, user monitoring, data compression, etc.

Other Tools

- Data dictionary / repository:
  - Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.
  - **Active data dictionary** is accessed by DBMS software and users/DBA.
  - **Passive data dictionary** is accessed by users/DBA only.

Other Tools

- Application Development Environments and CASE (computer-aided software engineering) tools:
- Examples:
  - PowerBuilder (Sybase)
  - JBuilder (Borland)
  - JDeveloper 10G (Oracle)

Typical DBMS Component Modules

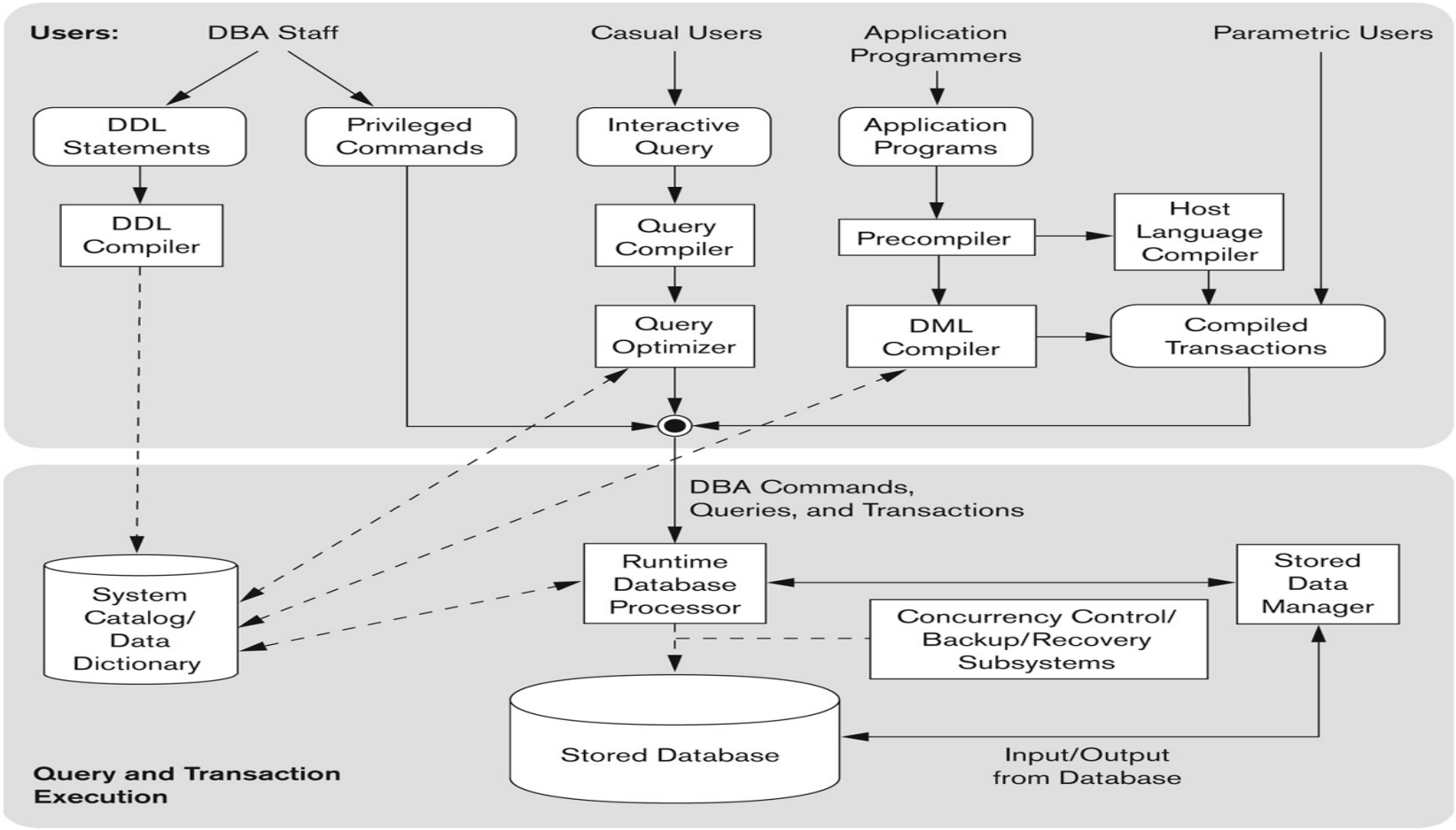


Figure 2.3
Component modules of a DBMS and their interactions.

Centralized and Client-Server DBMS Architectures

- Centralized DBMS:
  - Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.
  - User can still connect through a remote terminal – however, all processing is done at centralized site.

A Physical Centralized Architecture

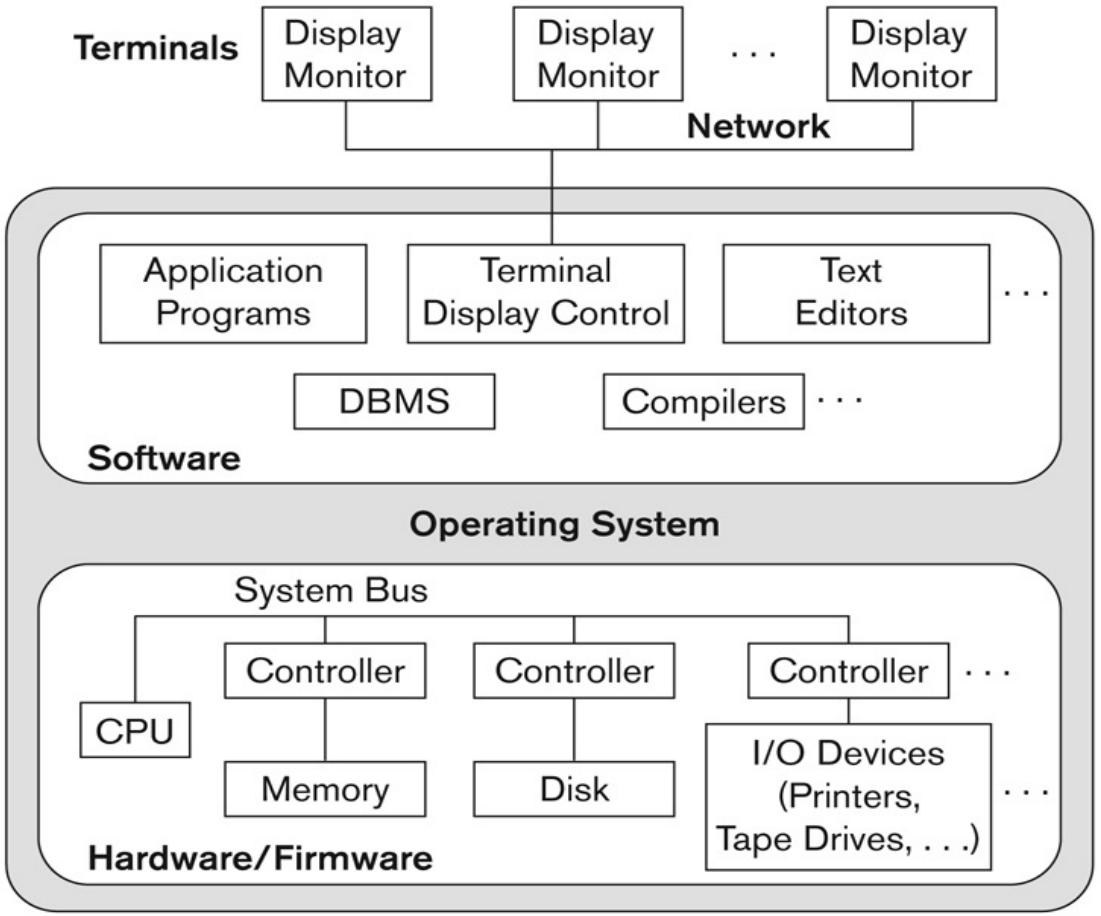


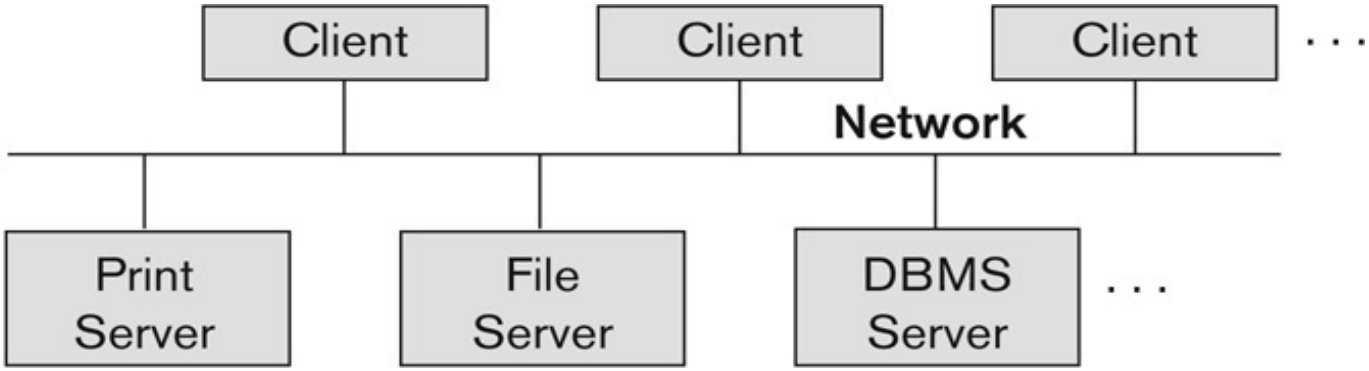
Figure 2.4
A physical centralized architecture.

Basic 2-tier Client-Server Architectures

- Specialized Servers with Specialized functions
  - Print server
  - File server
  - DBMS server
  - Web server
  - Email server
- Clients can access the specialized servers as needed

Logical two-tier client server architecture

Figure 2.5
Logical two-tier client/server architecture.



Clients

- Provide appropriate interfaces through a client software module to access and utilize the various server resources.
- Clients may be 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
- Relational DBMS servers are often called SQL servers, query servers, or transaction servers
- Applications running on clients utilize an Application Program Interface (**API**) to access server databases via standard interface such as:
  - ODBC: Open Database Connectivity standard
  - JDBC: for Java programming access
- Client and server must install appropriate client module and server module software for ODBC or JDBC
- See Chapter 9

Two Tier Client-Server Architecture

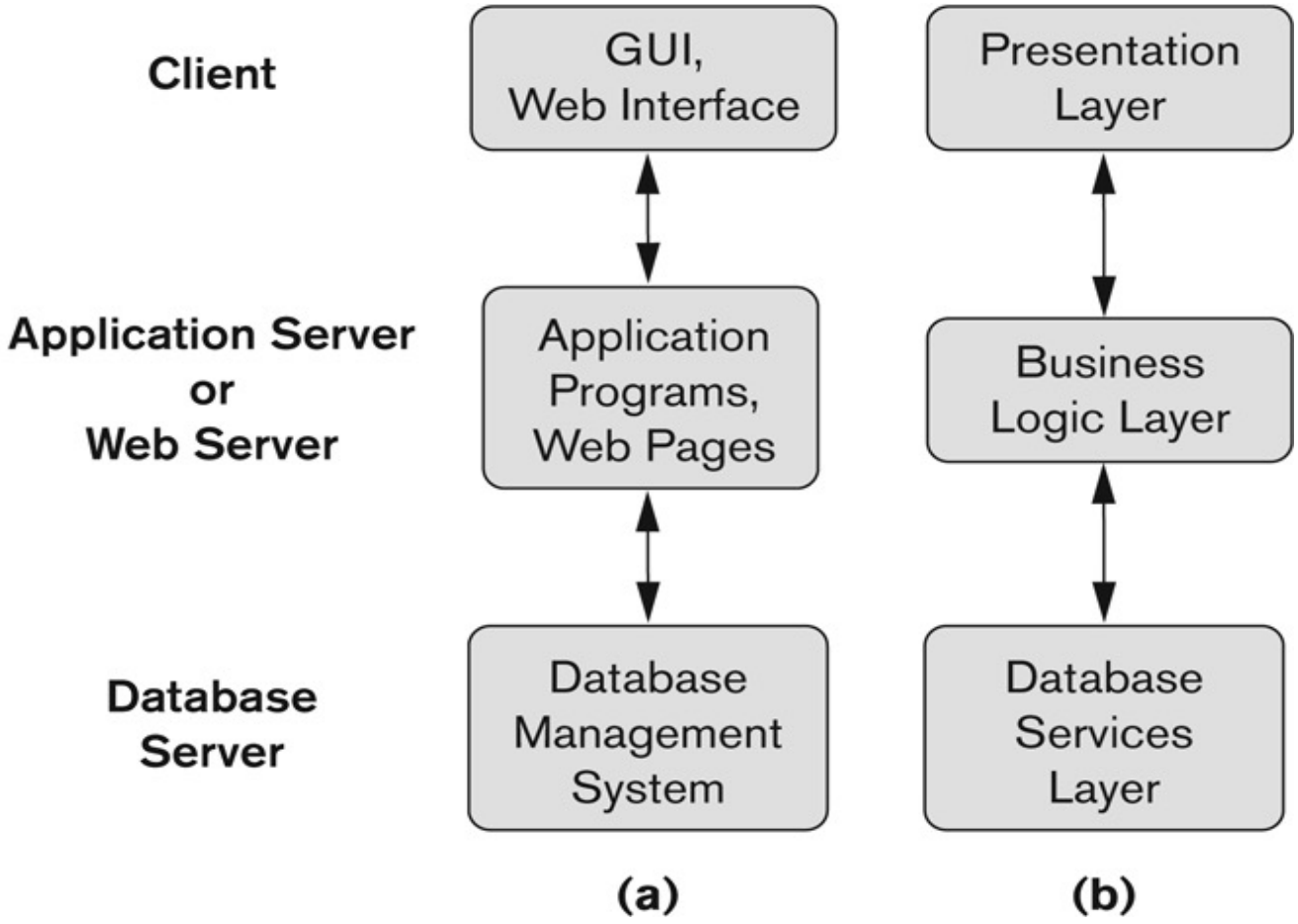
- A client program may connect to several DBMSs, sometimes called the data sources.
- In general, data sources can be files or other non-DBMS software that manages data.
- Other variations of clients are possible: e.g., in some object DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc.

Three Tier Client-Server Architecture

- Common for Web applications
- Intermediate Layer called Application Server or Web Server:
  - Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server
  - Acts like a conduit for sending partially processed data between the database server and the client.
- Three-tier Architecture Can Enhance Security:
  - Database server only accessible via middle tier
  - Clients cannot directly access database server

Three-tier client-server architecture

Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.



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 personal computers) vs. multi-user (most DBMSs).
  - Centralized (uses a single computer with one database)
    vs. distributed (uses multiple computers, multiple databases)

Variations of Distributed DBMSs (DDBMSs)

- Homogeneous DDBMS
- Heterogeneous DDBMS
- Federated or Multidatabase Systems
- Distributed Database Systems have now come to be known as client-server based database systems because:
  - They do not support a totally distributed environment, but rather a set of database servers supporting a set of clients.

Cost considerations for DBMSs

- Cost Range: from free open-source systems to configurations costing millions of dollars
- Examples of free relational DBMSs: MySQL, PostgreSQL, others
- Commercial DBMS offer additional specialized modules, e.g. time-series module, spatial data module, document module, XML module
  - These offer additional specialized functionality when purchased separately
  - Sometimes called cartridges (e.g., in Oracle) or blades
- Different licensing options: site license, maximum number of concurrent users (seat license), single user, etc.