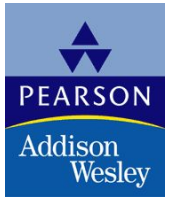
Chapter 2

Database System Concepts and Architecture



Client/Server DBMS Architecture

- A Client module is designed so that it will run on user work station
  - User Interfaces
  - Application Programs

- A server module handles data storage, access, search and other functions.

Data Abstraction

- **Data abstraction** generally refers to the suppression of details of data organization and storage, and the highlighting of the essential features for an improved understanding of data.
- One of the main characteristics of the database approach is to support data abstraction so that different users can perceive data at their preferred level of detail.
- A Data Model provides the necessary means to achieve this abstraction.

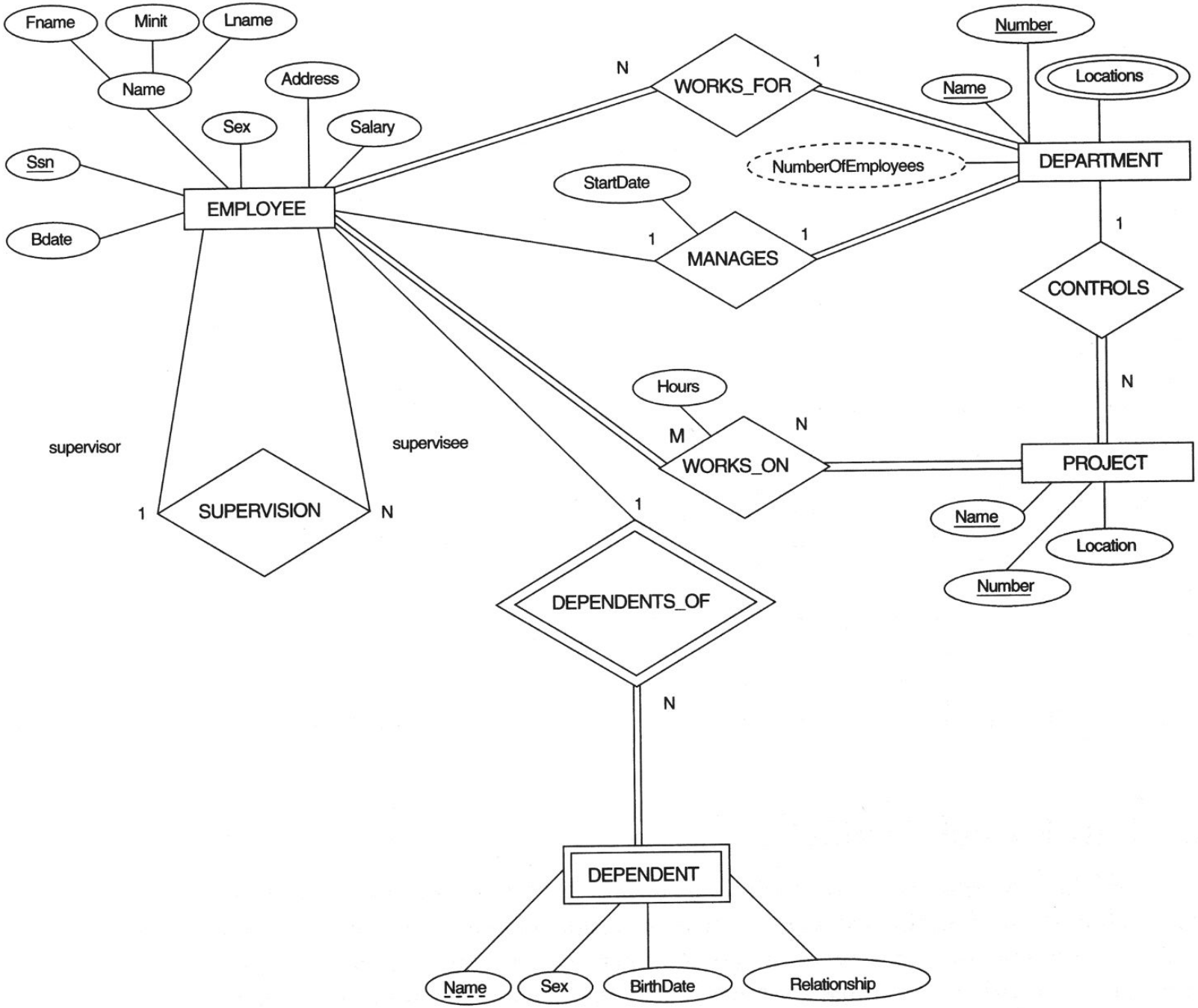
Data Model

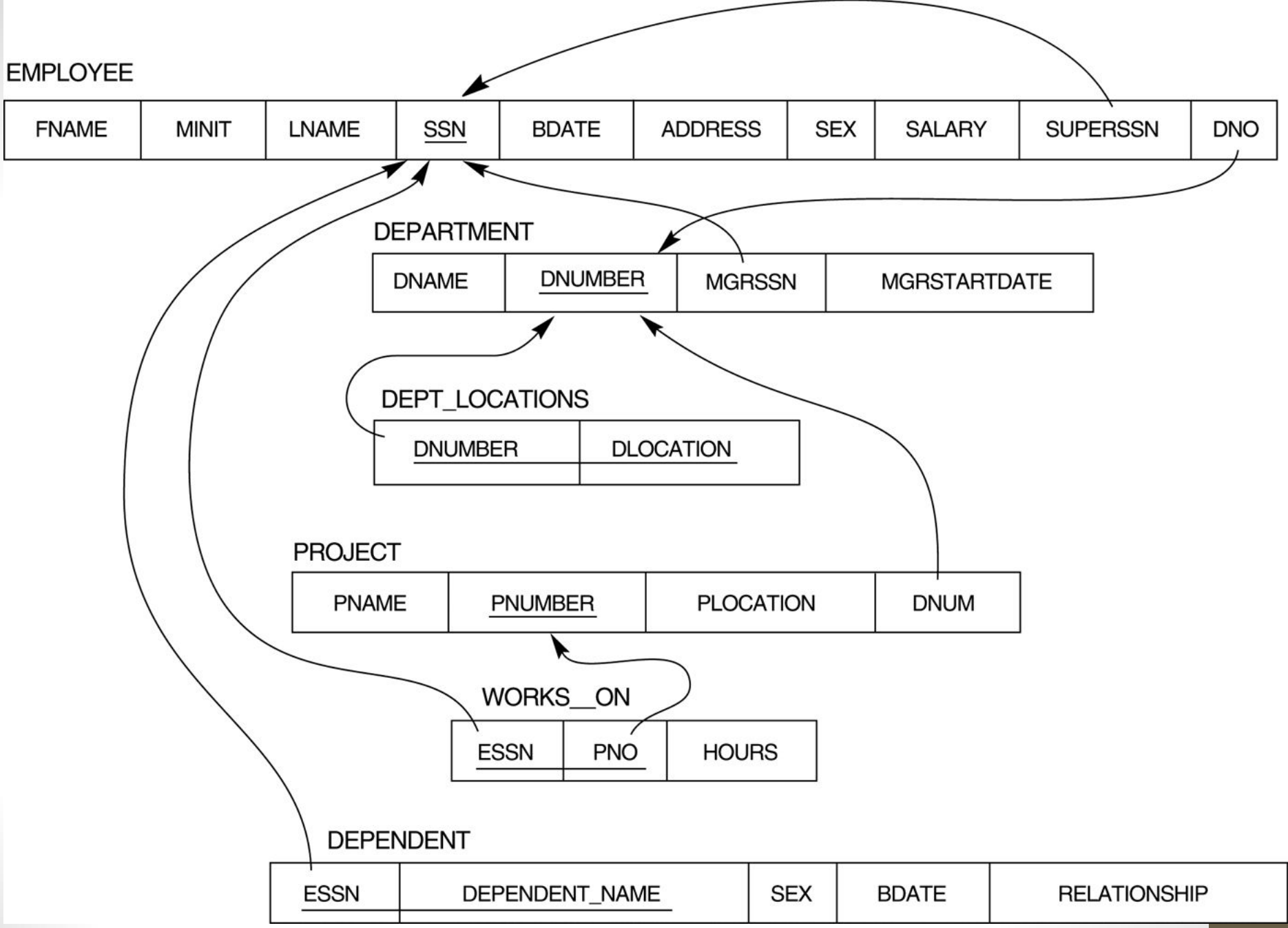
- A collection of concepts that can be used to describe structure of database.
  - **Structure** includes: data types, relationships and constraints that apply to data.
  - Some data models also include a set of basic operations for specifying retrievals and updates.
  - We can also specify User defined operations: Compute GPA

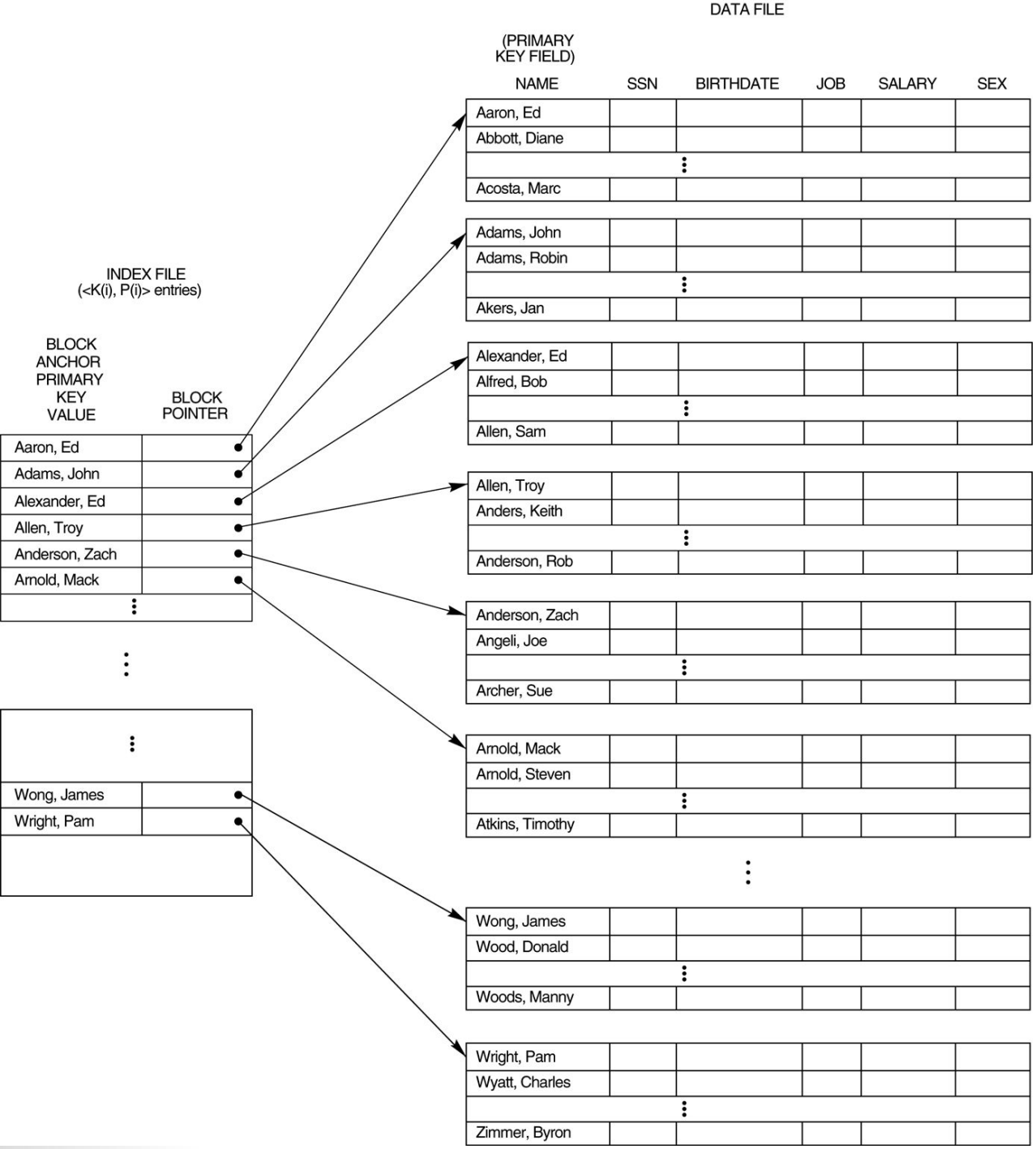
Categories of data model

(Based on types of concepts they use to describe)

- **High Level OR Conceptual Data Models (Semantic)**
  - Provide concepts that are close to the way that many users perceive data.
  - (Also called ***entity-based*** or ***object-based*** data models.)
  - Entities, Attribute, Relationship
- **Low Level OR Physical Data Models (Internal)**
  - Provide concepts that describe the details of how data is stored.
  - It provides information like the record format in which data is recorded in db.
  - Record ordering, access path etc. Example index.
  - They are meant for computer specialists and not for end users.
- **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).
  -


Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
NumberOfEmployees
StartDate
1
MANAGES
1
1
CONTROLS
N
Hours
M
N
WORKS_ON
PROJECT
Name
Location
Number
supervisor
supervisee
1
SUPERVISION
N
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship


EMPLOYEE
FNAME
MINIT
LNAME
SSN
BDATE
ADDRESS
SEX
SALARY
SUPERSSN
DNO
DEPARTMENT
DNAME
DNUMBER
MGRSSN
MGRSTARTDATE
DEPT_LOCATIONS
DNUMBER
DLOCATION
PROJECT
PNAME
PNUMBER
PLOCATION
DNUM
WORKS_ON
ESSN
PNO
HOURS
DEPENDENT
ESSN
DEPENDENT_NAME
SEX
BDATE
RELATIONSHIP


DATA FILE
(PRIMARY KEY FIELD)
NAME
SSN
BIRTHDATE
JOB
SALARY
SEX
INDEX FILE (<K(i), P(i)> entries)
BLOCK ANCHOR PRIMARY KEY VALUE
BLOCK POINTER
Aaron, Ed
Adams, John
Alexander, Ed
Allen, Troy
Anderson, Zach
Arnold, Mack
Wong, James
Wright, Pam
Aaron, Ed
Abbott, Diane
Acosta, Marc
Adams, John
Adams, Robin
Akers, Jan
Alexander, Ed
Alfred, Bob
Allen, Sam
Allen, Troy
Anders, Keith
Anderson, Rob
Anderson, Zach
Angeli, Joe
Archer, Sue
Arnold, Mack
Arnold, Steven
Atkins, Timothy
Wong, James
Wood, Donald
Woods, Manny
Wright, Pam
Wyatt, Charles
Zimmer, Byron

Conceptual Data Model

- Conceptual data models use concepts such as entities, attributes, and relationships.
- An **entity** represents a real-world object or concept, such as an employee or a project from the miniworld that is described in the database.
- An **attribute** represents some property of interest that further describes an entity, such as the employee's name or salary.
- A **relationship** among two or more entities represents an association among the entities, for example, a works-on relationship between an employee and a project.

Database Schema Vs. Database Sta

- **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(*illustrative*) display of a database schema. Example not data type and relationships. Shows Name.
- **Schema Construct**: A component of the schema or an object within the schema, e.g., STUDENT, COURSE.
- **Database Instance**: 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 state** (or **occurrence or snapshot**).

Example of a Database Schema

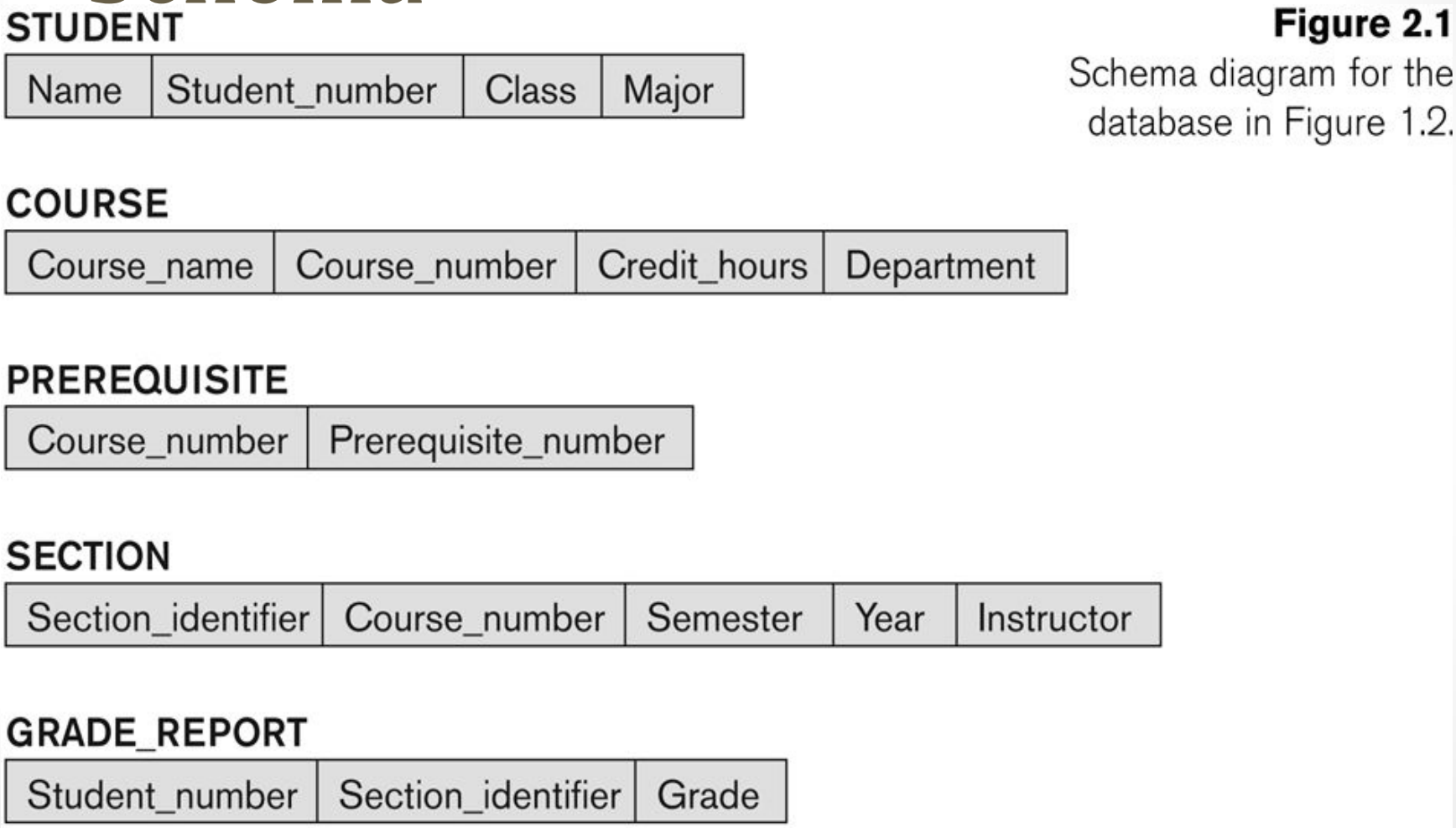


Figure 2.1
Schema diagram for the database in Figure 1.2.

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 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**.

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.

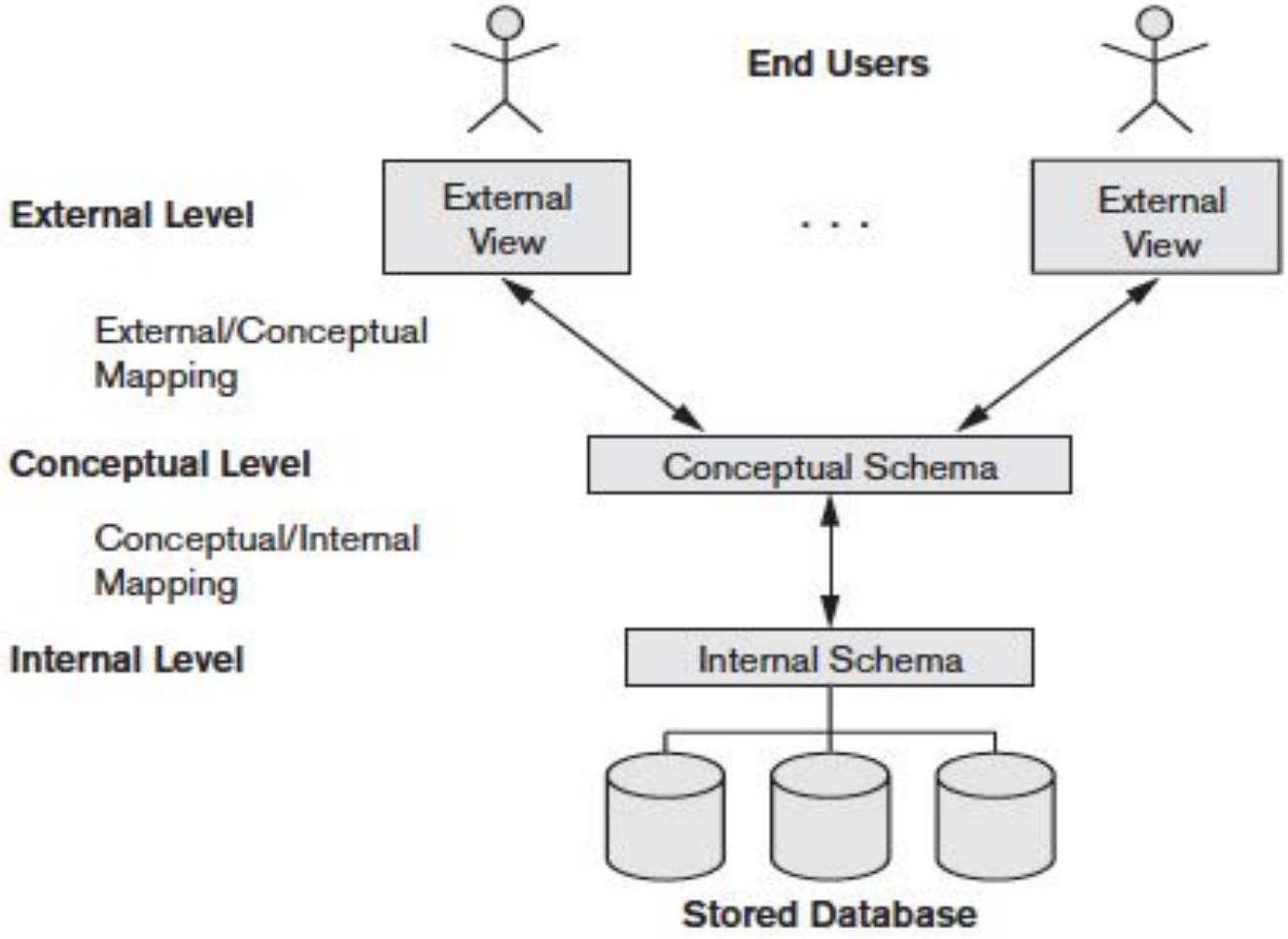
Three-Schema Architecture

- Proposed to visualize/support DBMS characteristics of:
  - **Program-data independence**.
  - Support of **multiple views** of the data.
  - Use of Catalog to store data description

  - **GOAL:**

    To separate user applications and physical Database.

Three-Schema Architecture



Three-Schema Architecture

- Defines DBMS schemas at three levels:
- **Internal 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. Hides storage details and describes entities , data types, relationships, constraints.
  - Uses a **conceptual** or an **implementation** data model.
- **External schemas** at the external level to describe the various user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

Three-Schema Architecture

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

Example:
User refer to only its own external schema.
DBMS transform request against the conceptual schema
DBMS transform request against the internal schema
If retrieval, data is extracted from the stored DB.
Reformatted again to match the users external view.

Data Independence

- **Logical Data Independence**: The capacity to change the conceptual schema without having to change the external schemas and their application programs.
- Example: adding/removing a data item , changing constraints.

- **Physical Data Independence**: The capacity to change the internal schema without having to change the conceptual schema.
- Example : new access paths for query efficiency (for fast retrievals)

- Physical data independence exist in most of the cases; but logical is hard to come. **WHY ???**

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 external schemas (views).
  - **View definition language** (**VDL**) are used to define 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**).

```
using System;
using System.Data;
using System.Data.SqlClient;

class Program
{
    static void Main()
    {
        string connectionString =
            "Data Source=(local);Initial Catalog=Northwind;"
            + "Integrated Security=true";
        // Provide the query string with a parameter placeholder.
        string queryString =
            "SELECT ProductID, UnitPrice, ProductName from
dbo.products "
                + "WHERE UnitPrice > @pricePoint "
                + "ORDER BY UnitPrice DESC;";
```

DBMS Languages -DML

- **Two classes of languages**

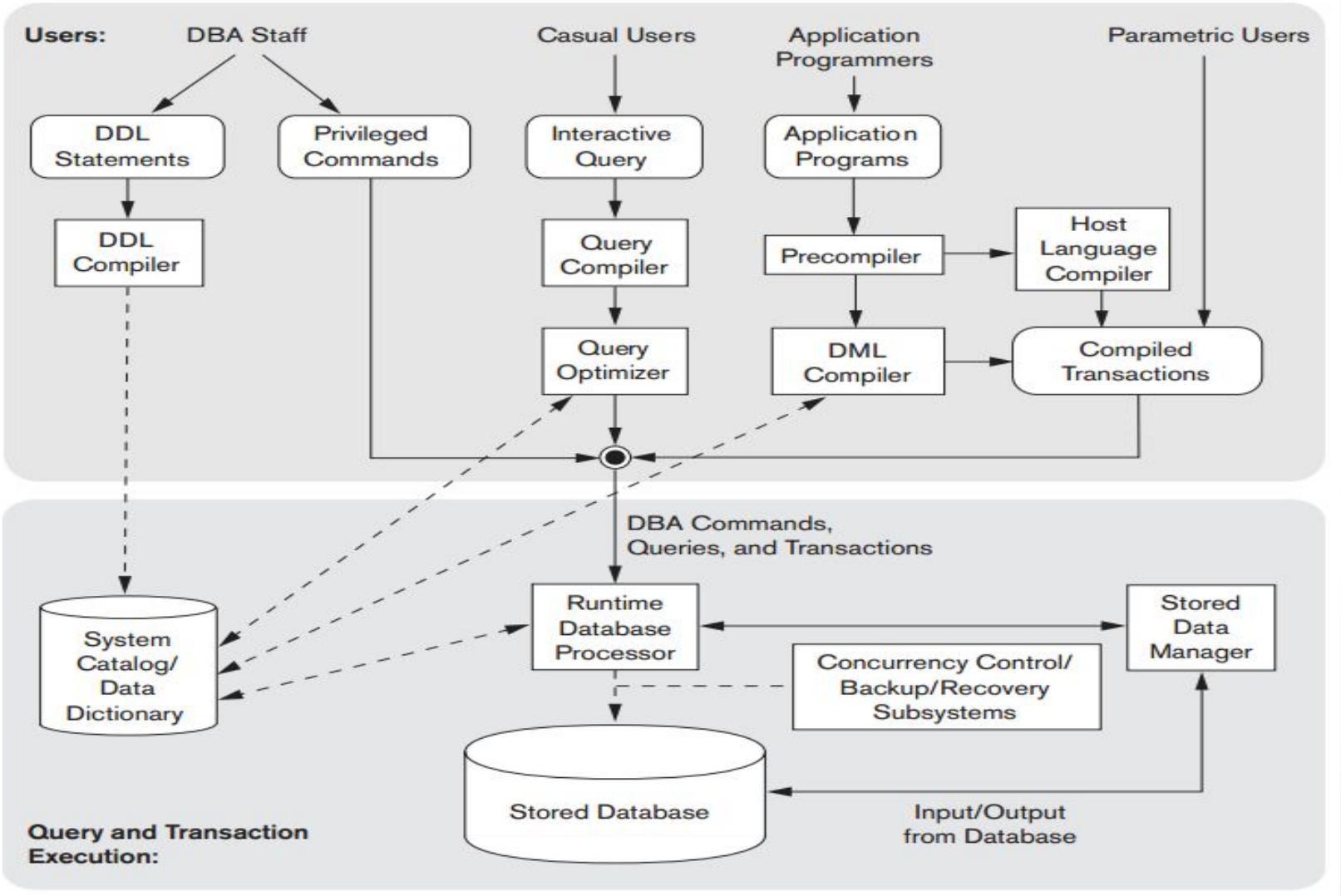
- – Procedural – user specifies what data is required and how to get those data
- –nonprocedural – user specifies what data is required without specifying how to get those data

- **FIND EXAMPLES OF BOTH.**

DBMS Interfaces

- Stand-alone query language interfaces.
- Speech input / output
- Interfaces for dba
  - commands for creating accounts, granting account authorization
- User-friendly 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

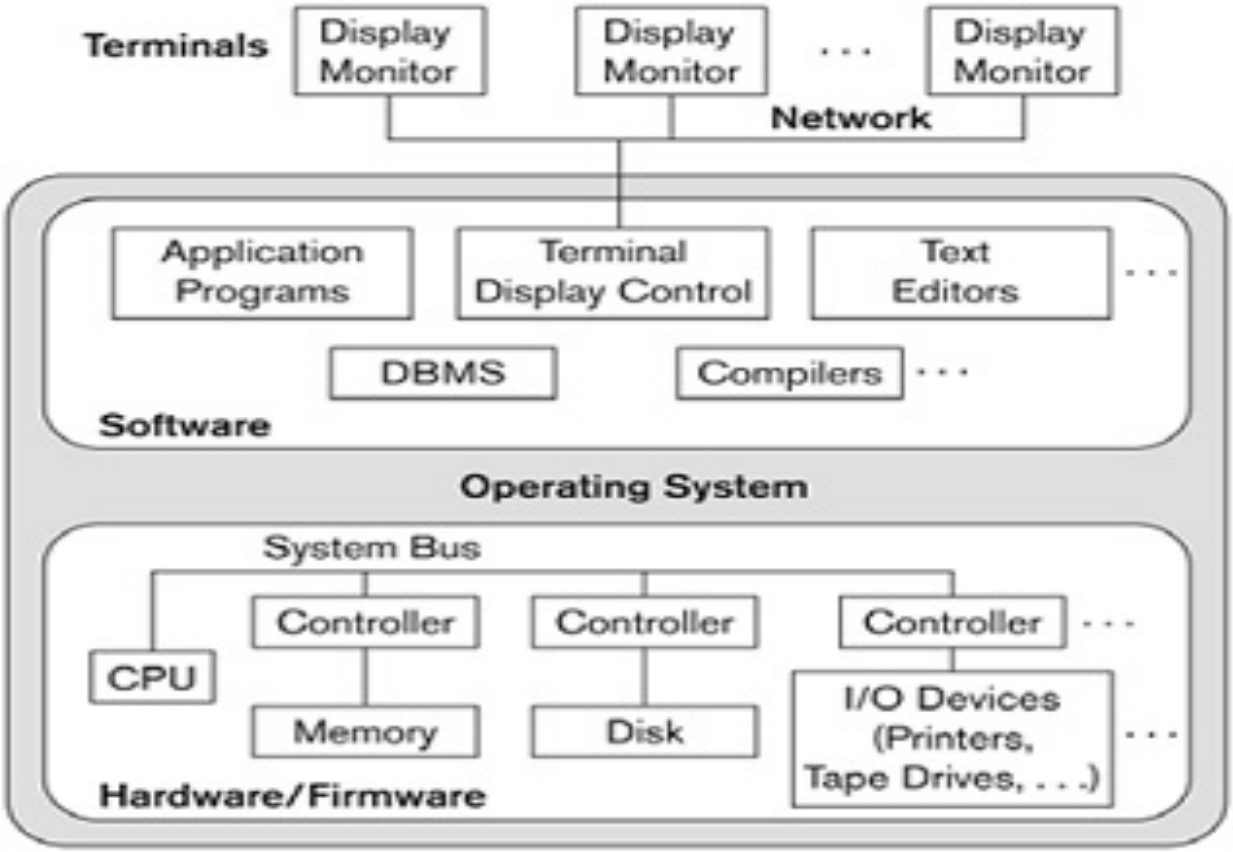
Database System Environment



Centralized DBMS Architectures

- **Centralized DBMS:** combines everything into single system including- DBMS software, user application programs and user interface programs.
- Most of the people replaced their terminals with pcs and hardware prices went low.

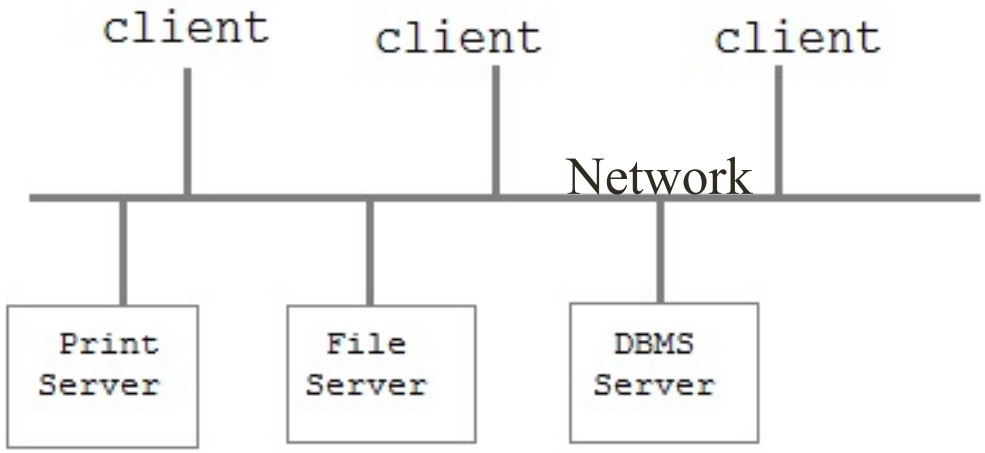
Centralized DBMS Architectures



A physical centralized architecture.

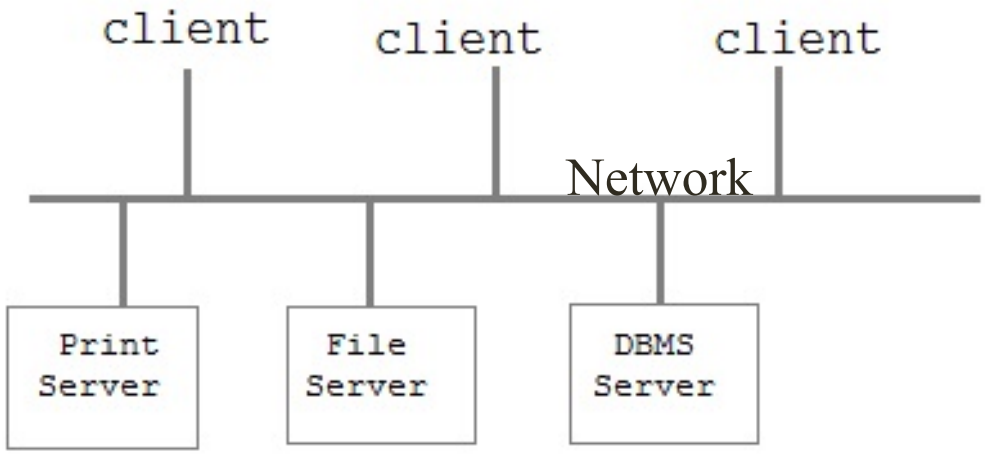
Basic Client-Server Architectures

- **Specialized Servers with Specialized functions**
- **Clients connected to servers via network**
- **Client machine provides user interfaces**
- **They have local processing power to run the applications**



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
- Connected to the servers via some form of a network.
  (LAN: local area network, wireless network, etc.)

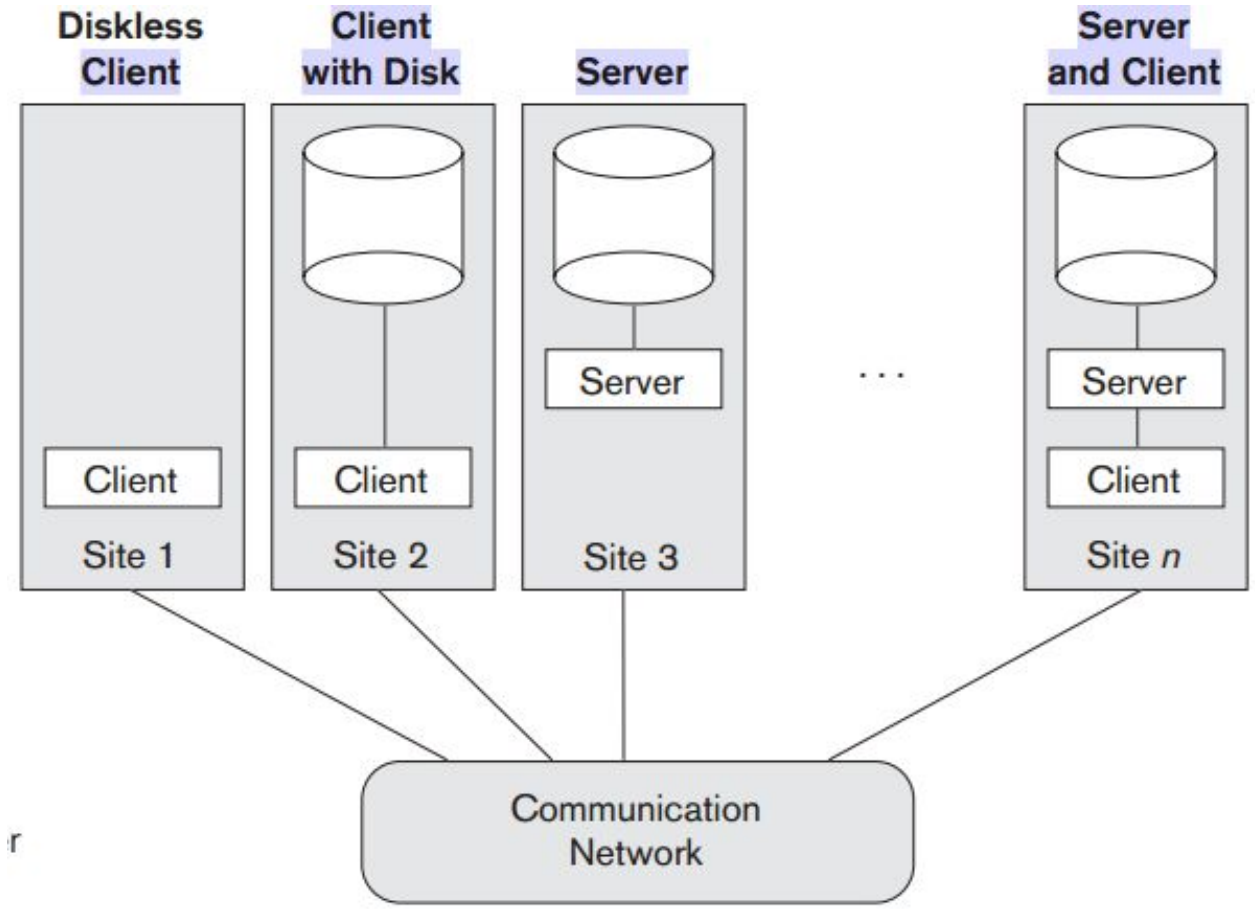
Two tier client/server Architecture - RDBMS

- **RDBMS Server**
  - Provides database query and transaction services to the clients
  - Sometimes called query or transaction servers or SQL Server

Client

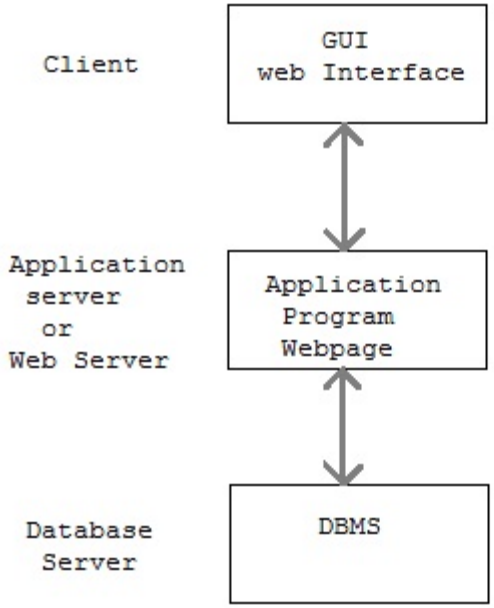
- User interface and application programs run on clients
- When client needs DBMS access , a connection is established to DBMS , then the client can communicate with DBMS.

Two Tier Client/Server Architecture



Three-Tier client/server Architecture – Web Applications

- Emergence of web lead to 3 –tier architecture.
- Intermediate Layer called **Application Server** or **Web Server:**



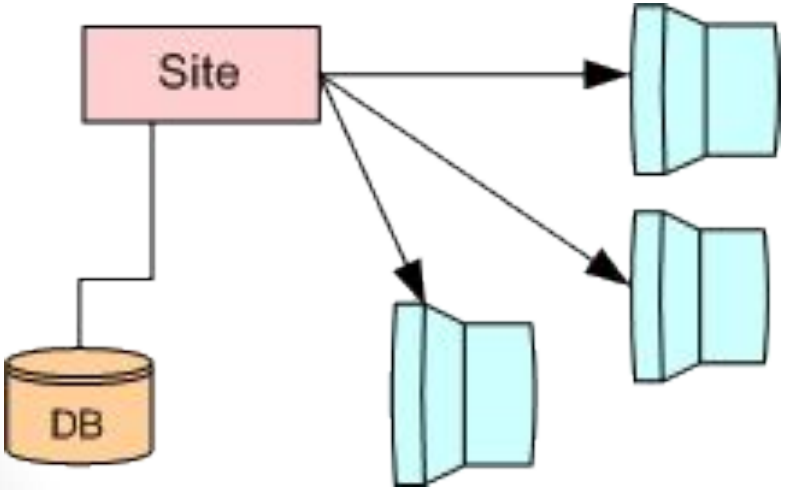
Three-Tier client/server Architecture – Web Applications

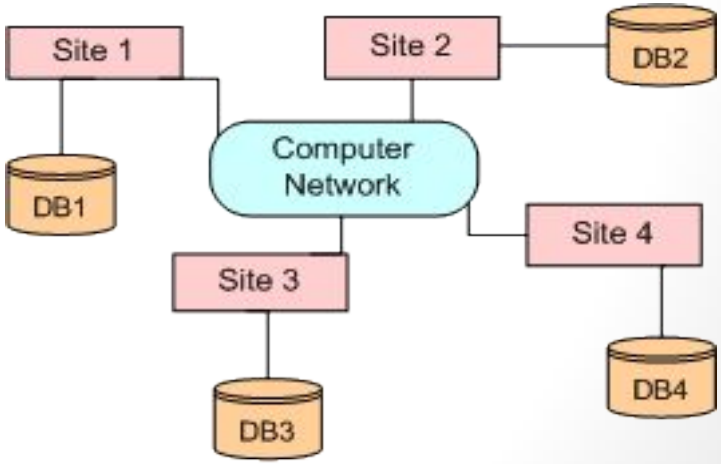
- stores the web connectivity software and **the rules and business logic (constraints)** part of the application used to access the right amount of data from the database server
- acts like a conduit for sending partially processed data between the database server and the client.

- **Additional Features- Security:**
  - encrypt the data at the server before transmission
  - decrypt data at the client
- Example ceaser cipher

Classification of DBMSs

- Single-user (typically used with micro- computers) vs. multi-user (most DBMSs).
- Centralized (db and dbms reside at one site) vs. distributed (uses multiple computers, multiple databases at different sites)
- Many DDBMS use a client server architecture.





Distributed Database System

Types of Distributed Environments:

- **Homogeneous DDBMS**
- **Heterogeneous DDBMS**
- **Federated or Multi-database Systems (Research over and tell me in next session)**

- **Homogeneous**, if all sites use the same DBMS product

- **Heterogeneous**, if sites may run different DBMS products

Some terms to get familiarized with

- Consistency
- Concurrency
- Integrity
- Persistent
- Coherent
- Atomic

- In context to Databases.