Database Concepts

CS 377: Database Systems

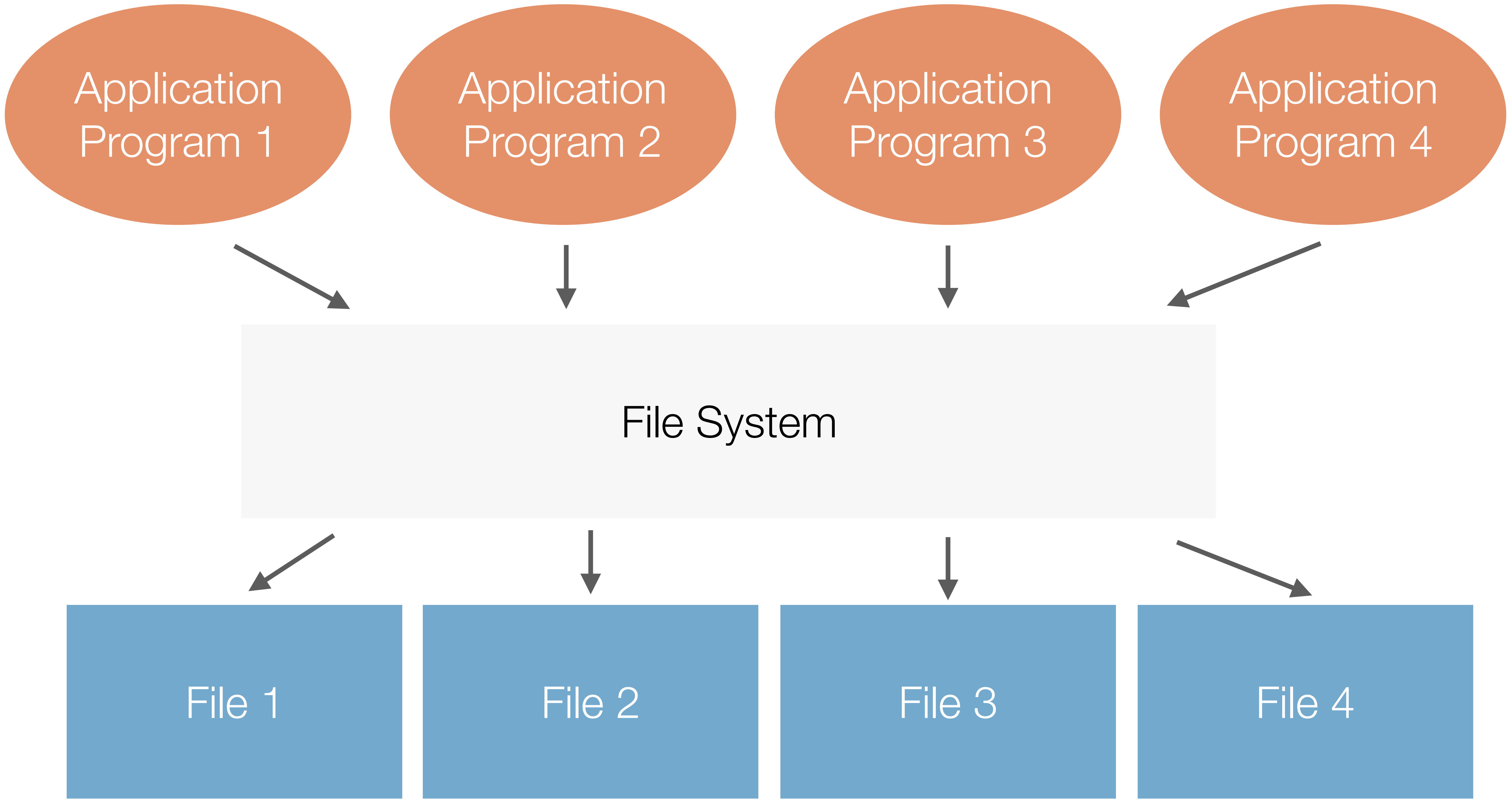
Course Logistics

- Course website with lectures and assignments
  http://joyceho.github.io/cs377_s16/index.html

- Sign up for Piazza (use OPUS name or emory email)
  http://piazza.com/emory/spring2016/cs377

- Office Hours
  Mine: Tu/Th 1-4 pm @ MSC W414
  Camilo: M 2-4 pm @ MSC N410

Traditional File Approach



Example: Traditional File Approach

ID	Name	Level
5523414	John	Soph
1234579	Alice	Jr
4235973	Bob	Sr
…	…	…

Program to store:

```
/* Write the data to file */
myFile.WriteString(x.ID, 10);
myFile.WriteString(x.name, 30);
myFile.writeString(x.level, 4);
```

Program to read:

```
/* Read one record */
x.ID = myFile.ReadString(10);
x.name = myFile.ReadString(30);
x.level = myFile.ReadString(4);
```

Downside to Traditional Files

- One program for each task can lead to duplication of information
(e.g., payroll information and employee record contains common entries such as SSN or other personal records)

- Data is not easily shared across different programs
(e.g., payroll record format program may not have access to reading the employee record format)

- Data security can be low as files can be easily accessible by different users if not careful

Downside to Traditional Files

- Rigid file format means structure of data must be encoded into each file (e.g., we add the student's address to the student record)

ID	Name	Level
5523414	John	Soph
…	…	…

⇒

ID	Name	Address	Level
5523414	John	12th Street	Soph
…	…	…	…

- Viability of programs dependent on data format is known as **Physical Data Dependence**
  - Existing programs cannot access the new data file
  - Any changes in structure of data means you need to change all programs that access the data —> system maintenance nightmare

Databases

- A coherent collection of related data
- Designed with some pre-conceived set of applications and specific group(s) of users
- Represent some aspect of real world

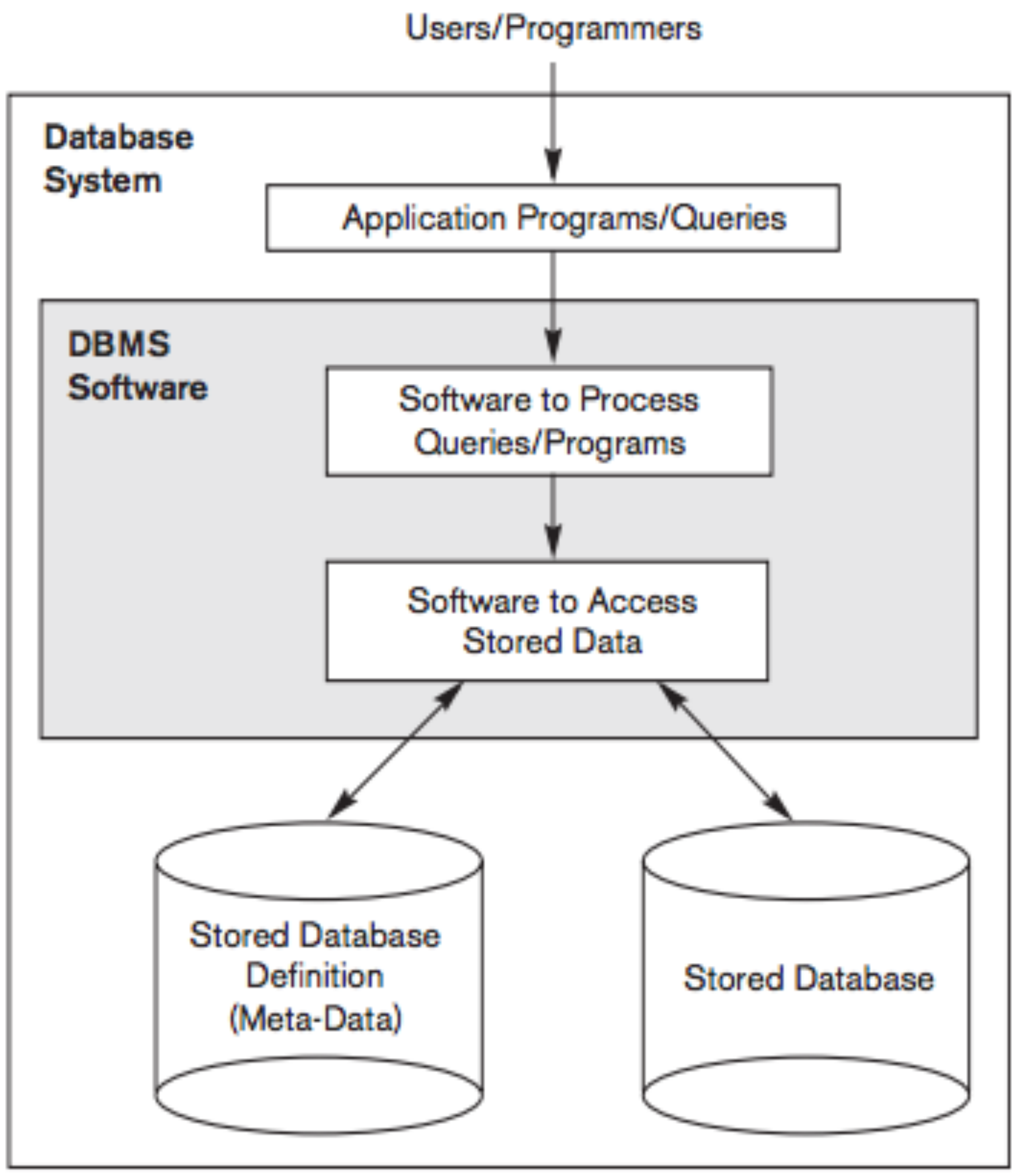

Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

Figure 1.1 from book

Database Properties

- Scalability: large amounts of information/data
- Concurrency: support multiple users
- Persistency: data is always preserved
- Security: robustness to system failures and malicious users/programs
- Data independence: computer program is independent of the physical storage structure of the data files

When DBMS Doesn’t Make Sense

- Simple, small data that is not expected to change at all
- Stringent, real-time requirements that may not be met because of the overhead of DBMS
- Embedded system limited storage capacity where a general-purpose DBMS would not fit
- Single-user access to the data

Building a Database System

- Requirements specification and analysis
  - What information to store?
  - What functionalities should be supported?
- Database design
  - Conceptual design (Entity-Relationship model)
  - Logical design (Relational data model)
  - Physical design (data storage, access structures)
- Implement the database design and populate with data

Requirement Specification & Analysis

- What data needs to be stored?
- What is the relationship between the records?
- What functionalities need to be supported? (e.g., querying, updating)
- What applications or programs will be using the database?

Data Model

- **Data Model**: collection of concepts that describe the structure of a database
  - Provides means to achieve data abstraction
- **Data Abstraction**: suppression of details of data organization and storage
  - Highlighting of the essential features for an improved understanding of data

Data Models Categories

- **High-level or conceptual data models**: close to the way many users perceive data and omits details
  - Allows non-technical users to talk about database issues
- **Low-level or physical data models**: describe the details of how data is stored on computer storage media
- **Representational or implementation data models**: in-between high-level and low-level
  - enough detail for user to see the structure of the data and how the data is stored inside the database

Database Jargon

- **Schema**: description of a database
  - Specified during the design phase
  - Rarely changed or updated unless the need arises
- **Schema diagram**: displays selected aspects of the schema
- **Database state (snapshot)**: data in database at a particular moment in time

Metadata

- Metadata: Information about the data
  - Name
  - Type
  - Size

Name of field	Type	Size
Student ID	int	10
Student Name	char	30

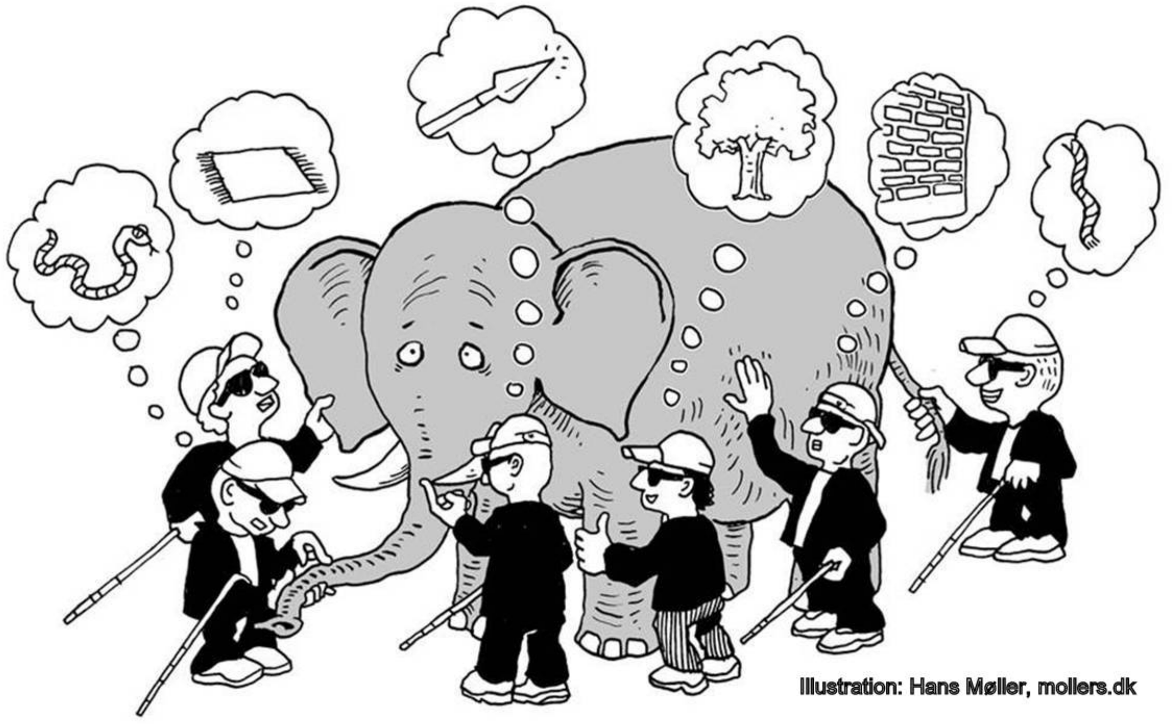
- Structure/format of the meta data does not change
  - Content can change but fields should remain the same
  - Programs that use meta data do not need to change

Physical Data Independence

- Property where computer program is independent of the record format (physical storage) of the data files
- Decouple program from data format in two steps
  - Read meta data and learn structure of data
  - Use meta data to access the data itself

Logical Data Independence

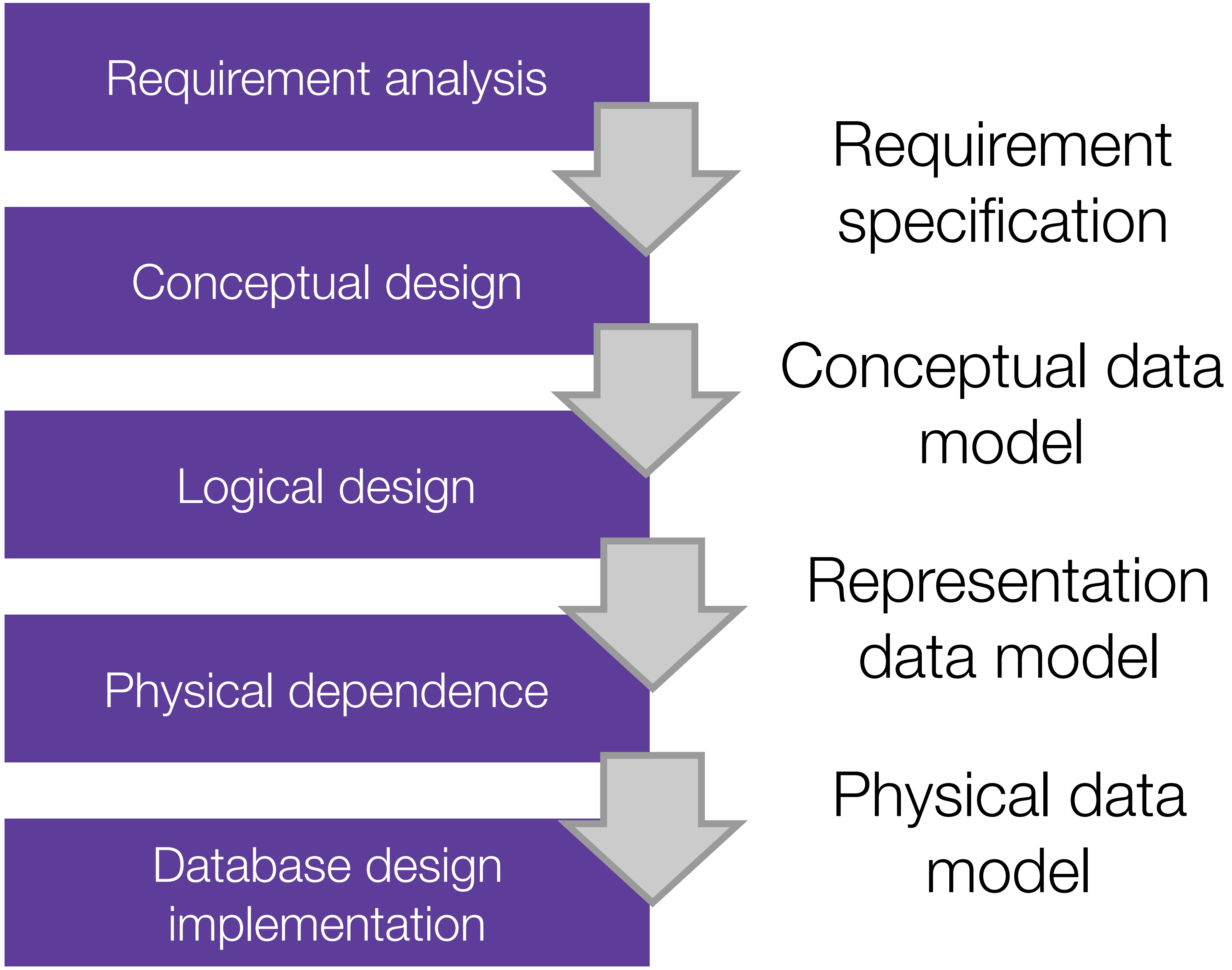
- Ability to present the stored information in different way to different users
    - Data storage format and the data presentation format are independent of one another
- Analogous to blind men examining an elephant
- Solution is to store information need for every group



Physical Data Models

- Describe how data is stored as files on the machine
- **Access path**: structure that makes the search for particular database records efficient
    - Example: **index** which allows direct access to data using an index term or keyword

Building a Database System



Example: Building a Database

- UNIVERSITY database: information to run undergraduate studies for students
  - Students
  - Courses
  - Grades
  - Instructors

Example: Requirement Analysis

- Query the system
  - Retrieve student grades for all the courses taken
  - List students who took ‘Database Systems’ in a specific semester
  - List prerequisites of the ‘Database’ course
- Update the system
  - Change student levels from freshman to sophomore
  - Create a new course for a semester
  - Enter grade for a student for a specific course

Example: Data Models

- Conceptual model: Entity-Relationship (ER) model
  - One of the more popular models
  - Covered next class
- Representational Data Models: Relational Data Model
  - Used most frequently in traditional DBMS
  - Relations or tables
  - Covered following class

Three Schema Architecture

- Three different views in system development
- Encapsulates important characteristics of database approach
    - Use of catalog to store database description
    - Insulation of programs and data
    - Multiple user view support

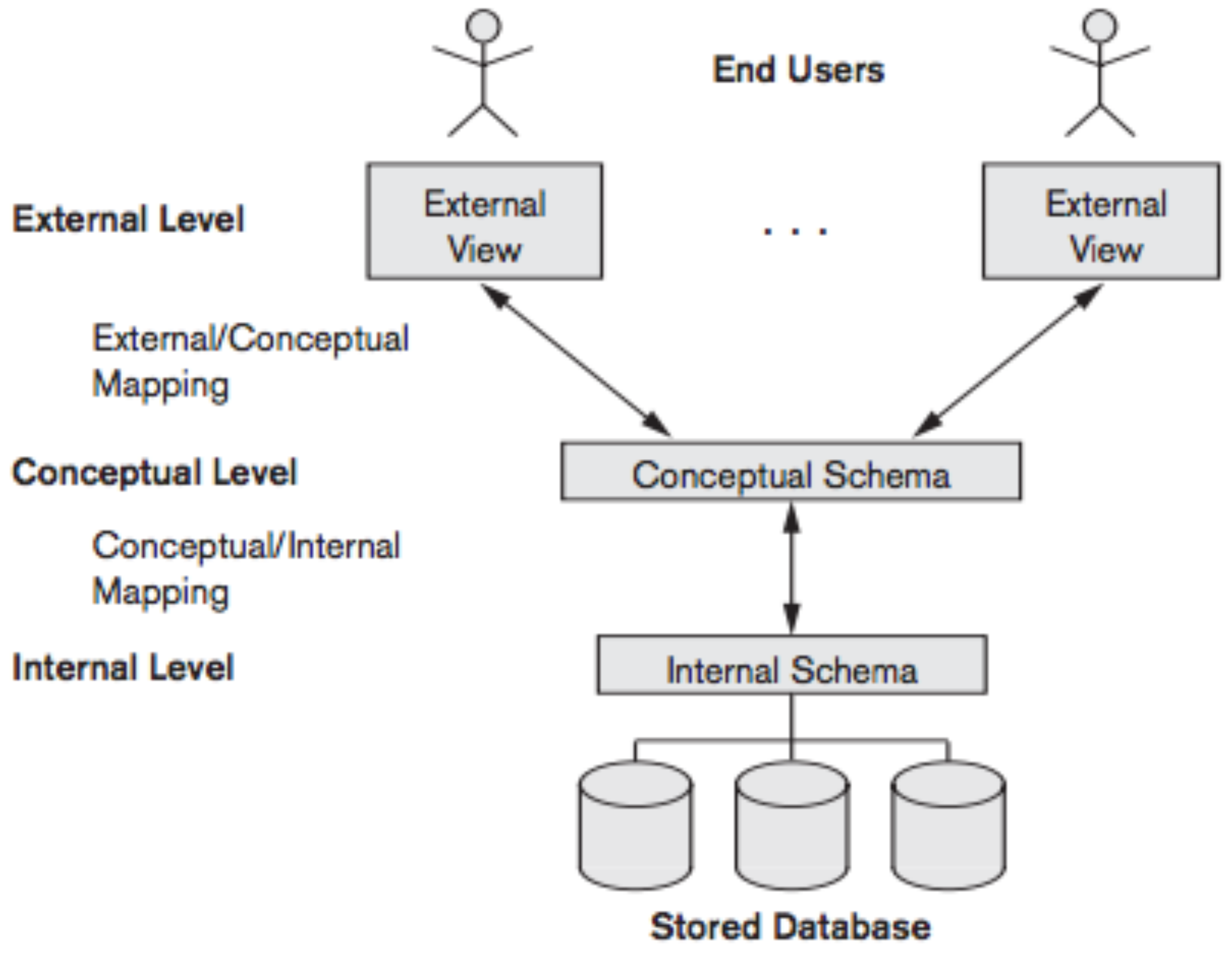


Figure 2.2 from book

Three Schema Architecture: External Level

External/View level describes the part of the database for particular class of users

- Information required for ONE particular type of user
- Multiple schemas in one database schema
- Example of a high-level data model

Three Schema Architecture: Conceptual Level

Conceptual level describes structure of whole database for a community of users

- Lists the fields in each data file (encompasses all external schemas)
- Example of implementation data model
- Most important schema in the database
- One schema for each database

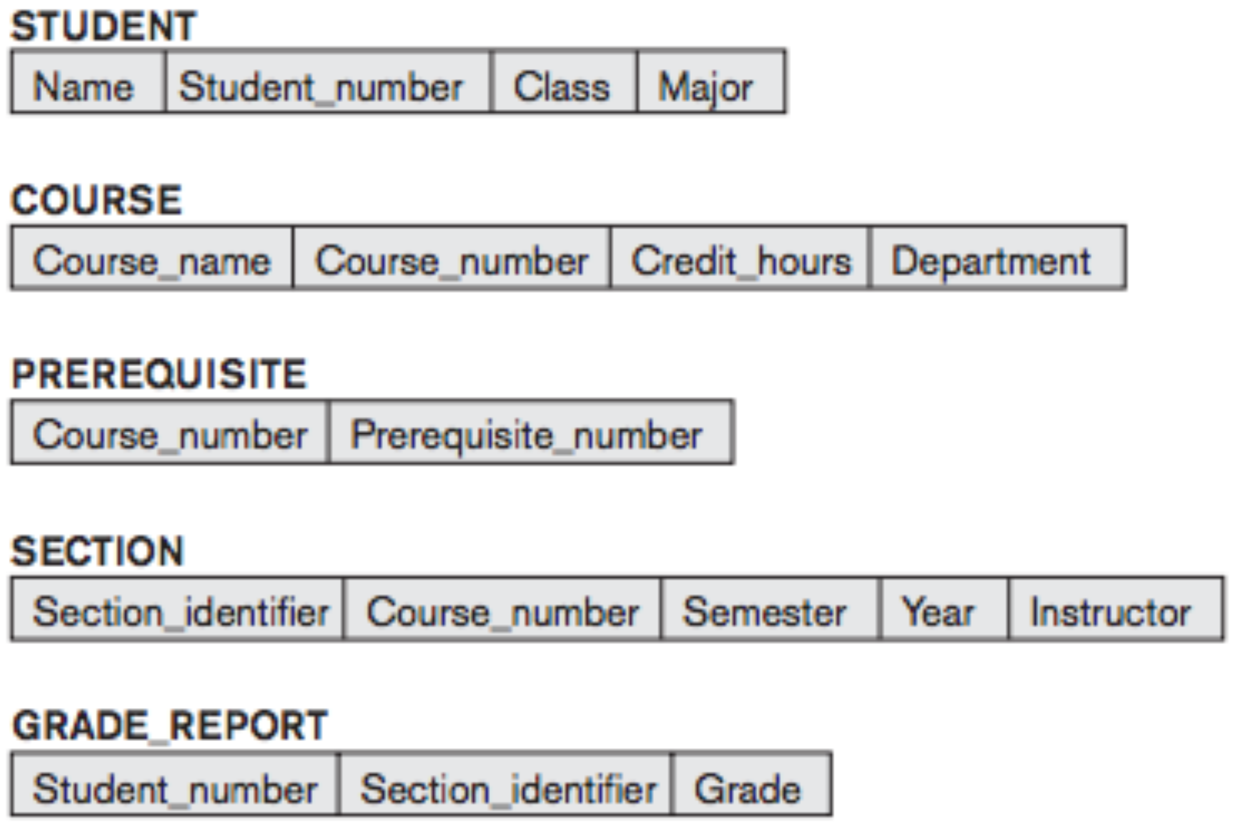
Three Schema Architecture: Internal Level

Internal level describes the physical storage structure of the database

- Details the structure of each file and how to best access each data file
- Example of the low level model
- Only one physical schema for each database

Example: Conceptual Schema

- Describes the different tables in the database
- Contains the field in each data file



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

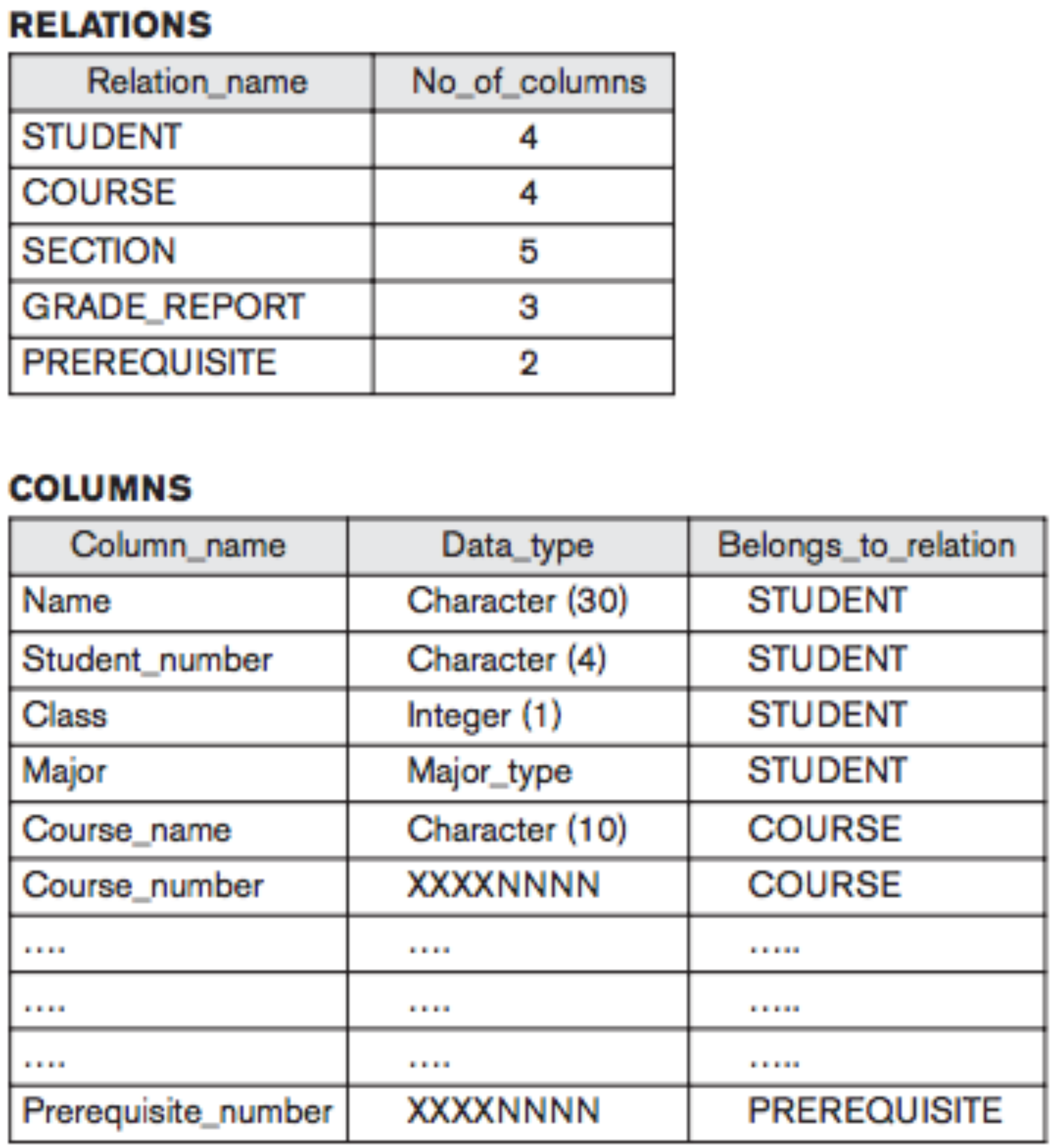
GRADE_REPORT

Student_number	Section_identifier	Grade

Figure 2.1 from book

Example: Metadata or Data Catalog

- Metadata describes the conceptual schema to access file structures of physical schema
- Metadata structure should never change



RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

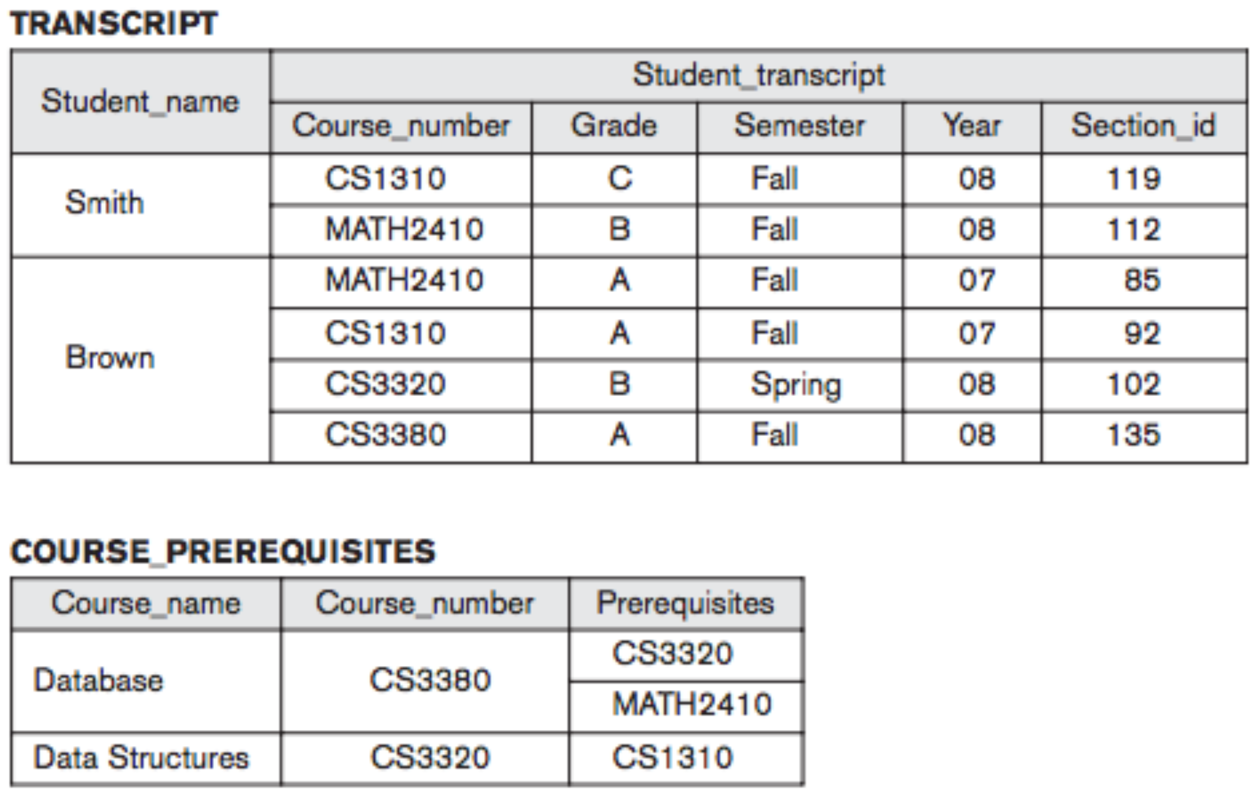
COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
....	....	.....
....	....	.....
....	....	.....
Prerequisite_number	XXXXNNNN	PREREQUISITE

Figure 1.3 from book

Example: Two Different Views

- Designed for two different set of users
- Transcript view is suitable for instructors and administrators
- Course prerequisites view is appropriate for students and administrators



TRANSCRIPT

Student_name	Student_transcript				
	Course_number	Grade	Semester	Year	Section_id
Smith	CS1310	C	Fall	08	119
	MATH2410	B	Fall	08	112
Brown	MATH2410	A	Fall	07	85
	CS1310	A	Fall	07	92
	CS3320	B	Spring	08	102
	CS3380	A	Fall	08	135

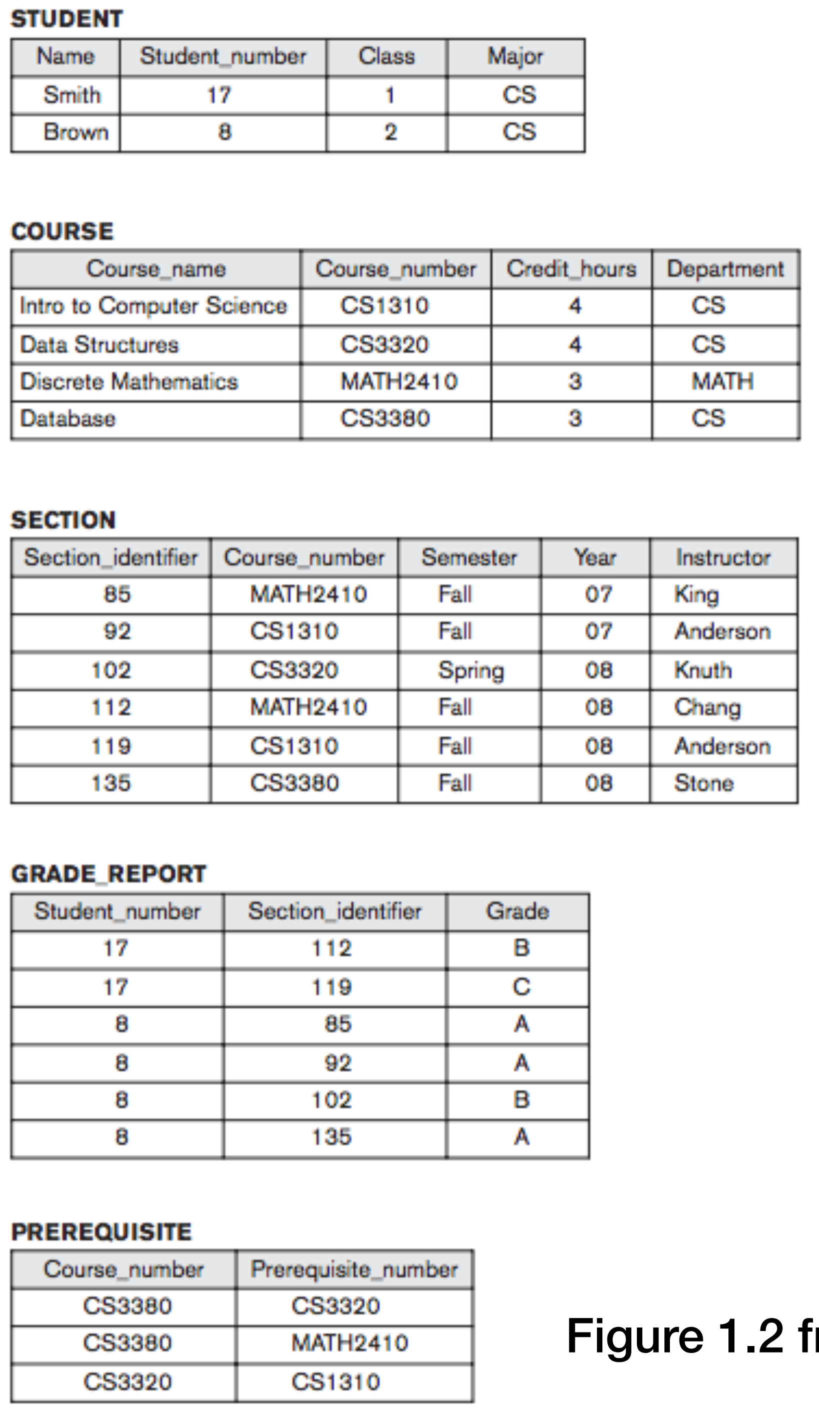
COURSE_PREREQUISITES

Course_name	Course_number	Prerequisites
Database	CS3380	CS3320
		MATH2410
Data Structures	CS3320	CS1310

Figure 1.5 from book

Example: Database Snapshot

- Encapsulates the state of the database at a particular time
- Contains the current set of occurrences or instances in the 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

Figure 1.2 from book

DBMS System

- Complex software system
- Consists of many different files and record structures
- Vendors will never give you the source code for the system

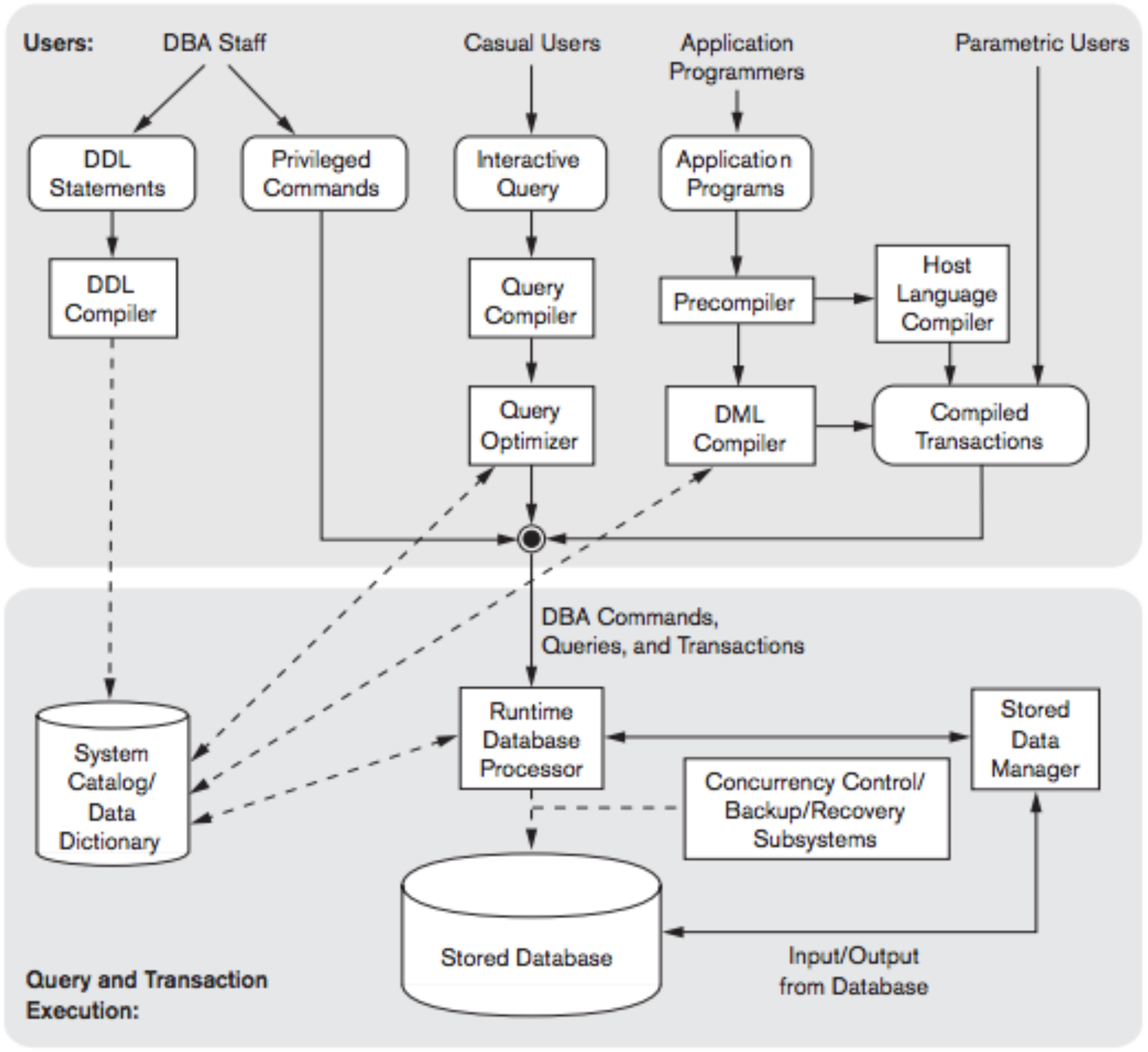


Figure 2.3 from book

Centralized DBMS Architecture

- Single machine where all DBMS functionality, application program execution, and user interface processing occurred
- Earlier architectures where user machines did not have good processing power

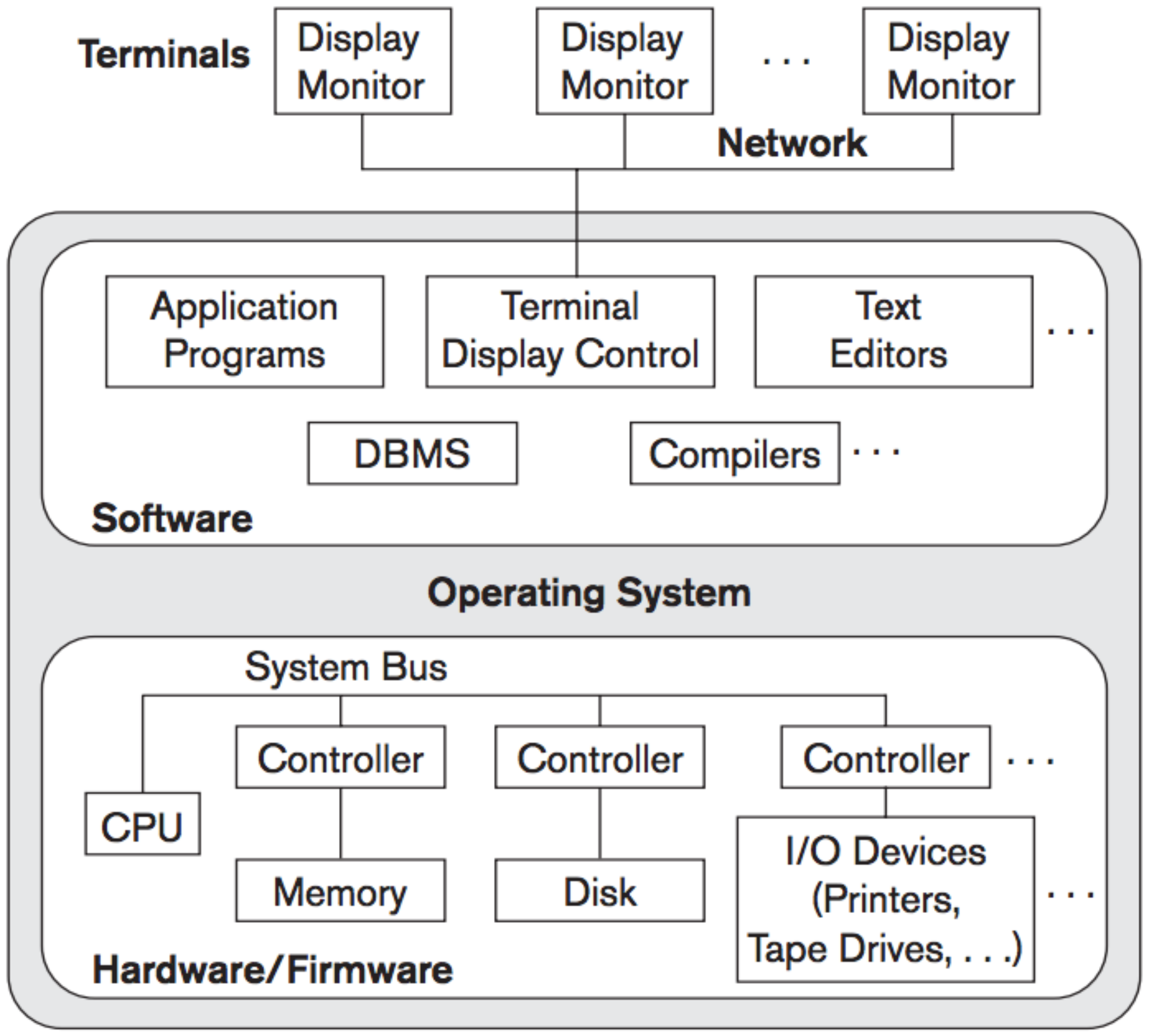


Figure 2.4 from book

Basic Client/Server Architectures

- Server: machine with specific functionalities
- Client: user interface capabilities and local processing

Logical two-tier architecture

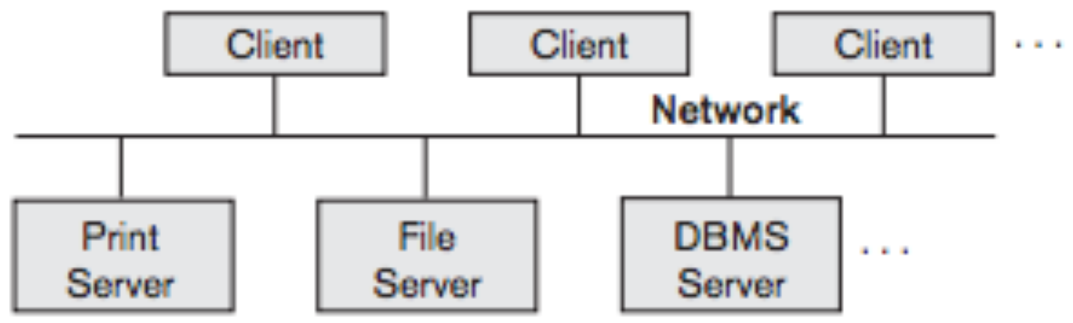


Figure 2.5 from book

Physical two-tier architecture

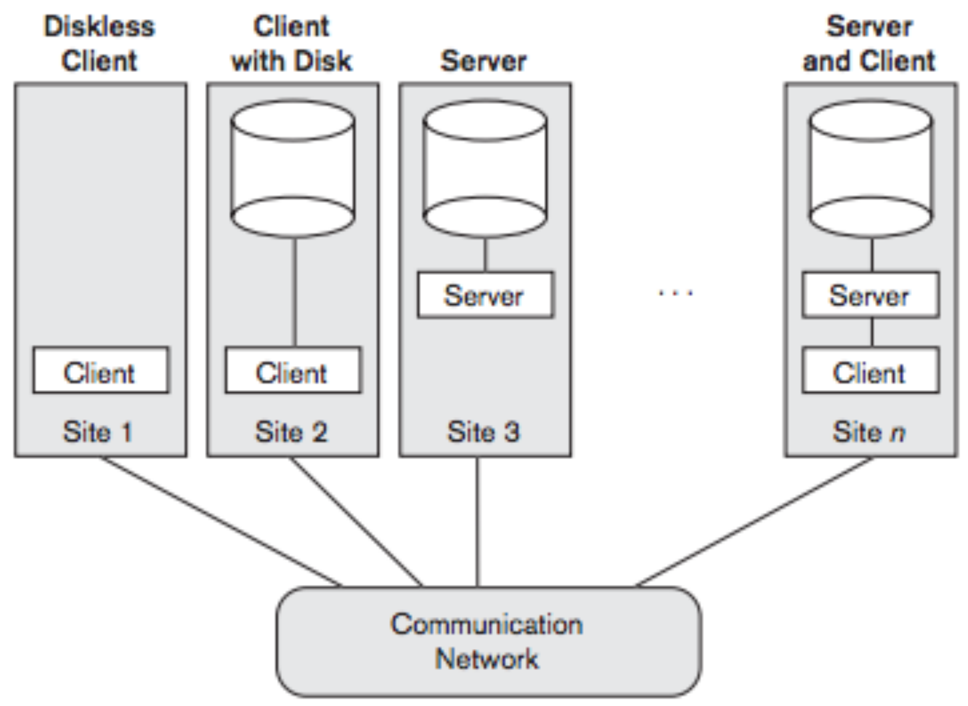
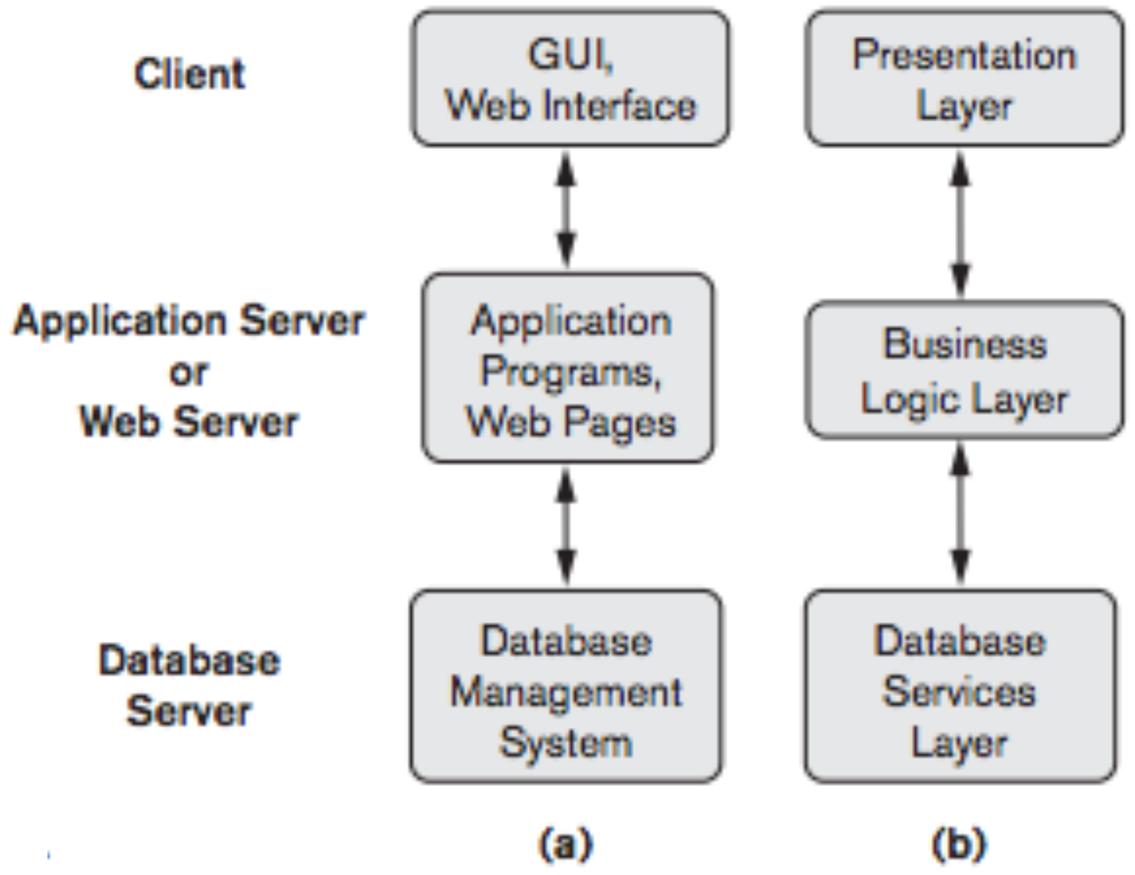


Figure 2.6 from book

n-Tier Architectures for Web Applications

- Divide the layers between the user and stored data into finer components
- Standard in web is a 3-tier structure with an application server/web server to provide intermediate layer between client and database server



Database Concepts: Recap

- Use of DBMS for large amounts of data and multiple users
- Steps for building a DBMS
- Categories of data models
- Three-Schema Architecture