Database Management System(MCA 4122)

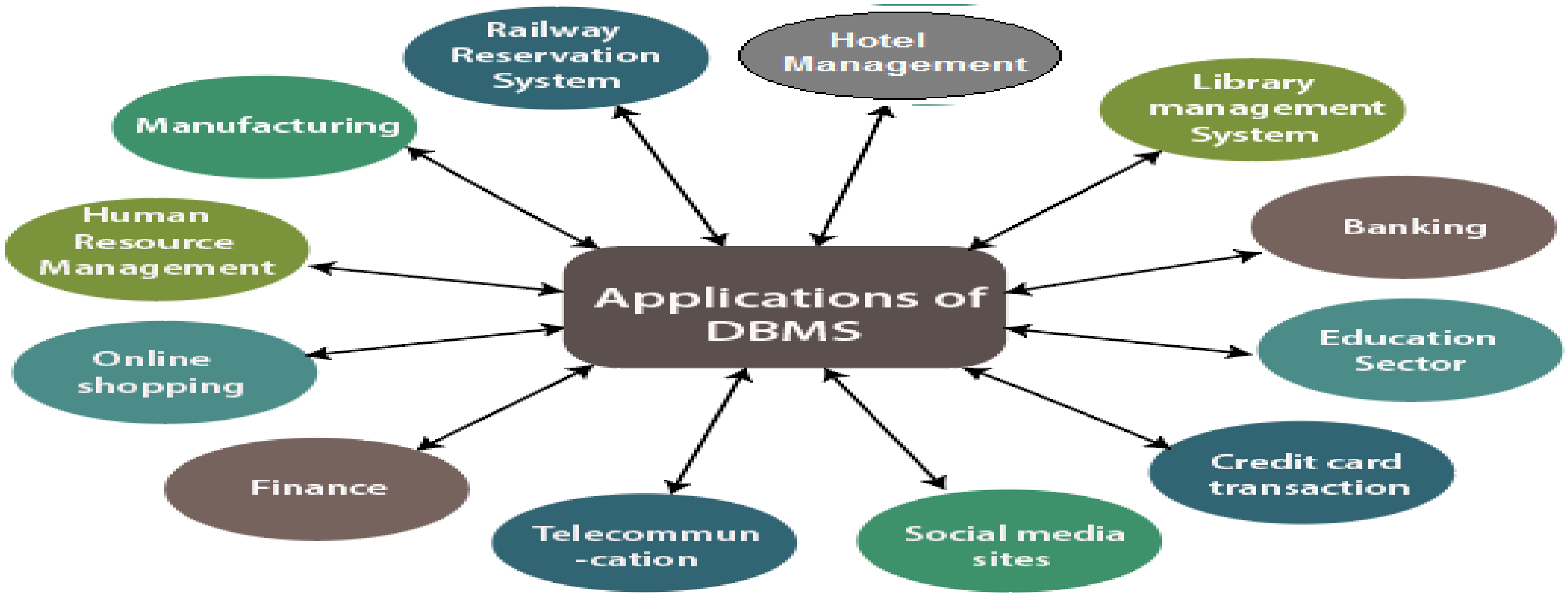
References

1. Abraham Silberschatz, Henry Korth, S. Sudarshan “Database System Concepts “, 6th Edition, McGraw Hill, 2010.
2. Ramez Elmasri, Shamkant Navathe “Fundamentals of Database System”, 6th Edition, Addison Wesley Publications Co., 2010
3. Raghu Ramakrishnan, Johannes Gehrke, “Database Management System”, 3rd Edition, WCB/McGraw Hill Publisher, 2007
4. Benjamin Rosenzweig, Elen Silvestrova, “Oracle PL/SQL by Example”,4th Edition, Addion-Wesley,2009
5. Ivan Bayross, “SQL, PL/SQL-The Programming Language of ORACLE”, 4th Edition, BPB Publications, 2009

Unit 1- Introduction to DBMS

- Purpose of Database Systems
- View of Data
- Data Models
- Data Definition Language
- Data Manipulation Language
- Transaction Management
- Storage Management
- Database Administrator
- Database Users
- Overall System Structure

Database System Applications



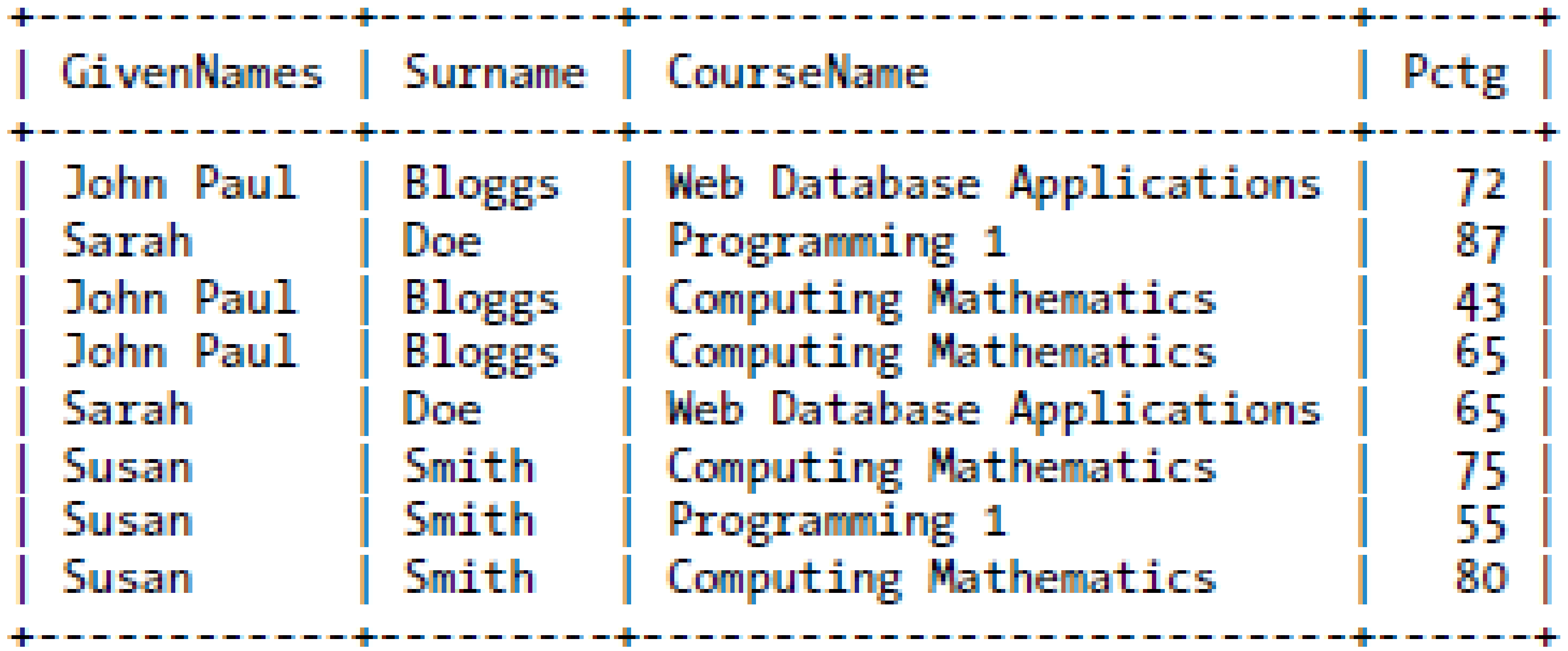
Database Management System (DBMS)

- *Database* A collection of related data (and a description of this data), designed to meet the information needs of an organization.
- DBMS is a collection of interrelated data and set of programs to access those data.
- DBMS is a general purpose software system that facilitates the process of defining, constructing, manipulating and sharing databases among users and applications.

DBMS - Most Popular Database Management Systems

Example

Student_Details Table

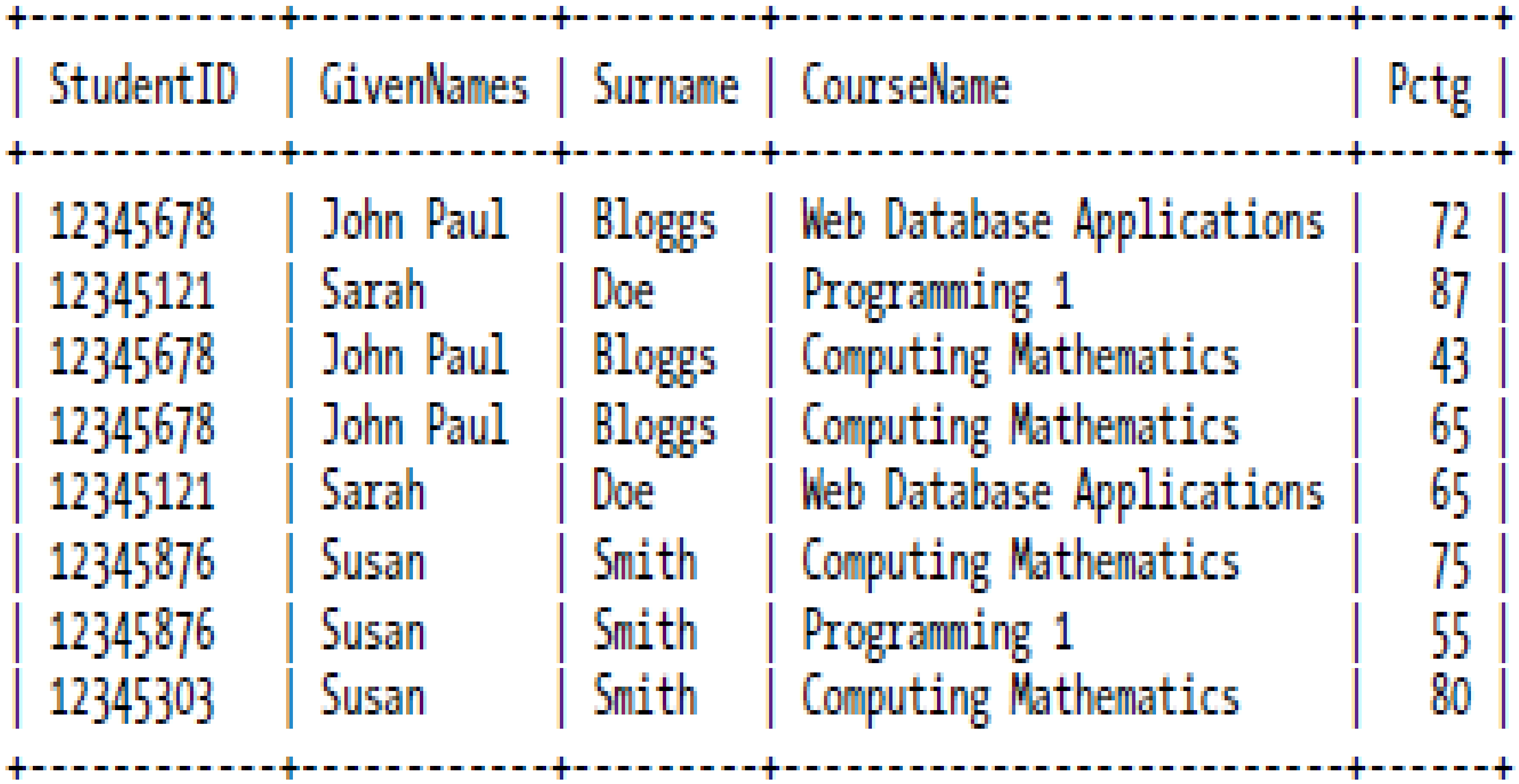


```
+------------+---------+---------------------------+------+
| GivenNames | Surname | CourseName                | Pctg |
+------------+---------+---------------------------+------+
| John Paul  | Bloggs  | Web Database Applications |   72 |
| Sarah      | Doe     | Programming 1             |   87 |
| John Paul  | Bloggs  | Computing Mathematics     |   43 |
| John Paul  | Bloggs  | Computing Mathematics     |   65 |
| Sarah      | Doe     | Web Database Applications |   65 |
| Susan      | Smith   | Computing Mathematics     |   75 |
| Susan      | Smith   | Programming 1             |   55 |
| Susan      | Smith   | Computing Mathematics     |   80 |
+------------+---------+---------------------------+------+
```

- We could have more than one student called Susan Smith; in the sample data, there are two entries for Susan Smith and the Computing Mathematics course
- Which Susan Smith got an 80?
- A common way to differentiate duplicate data entries is to assign a unique number to each entry. Here, we can assign a unique Student ID number to each student.

Example (Cont'D)

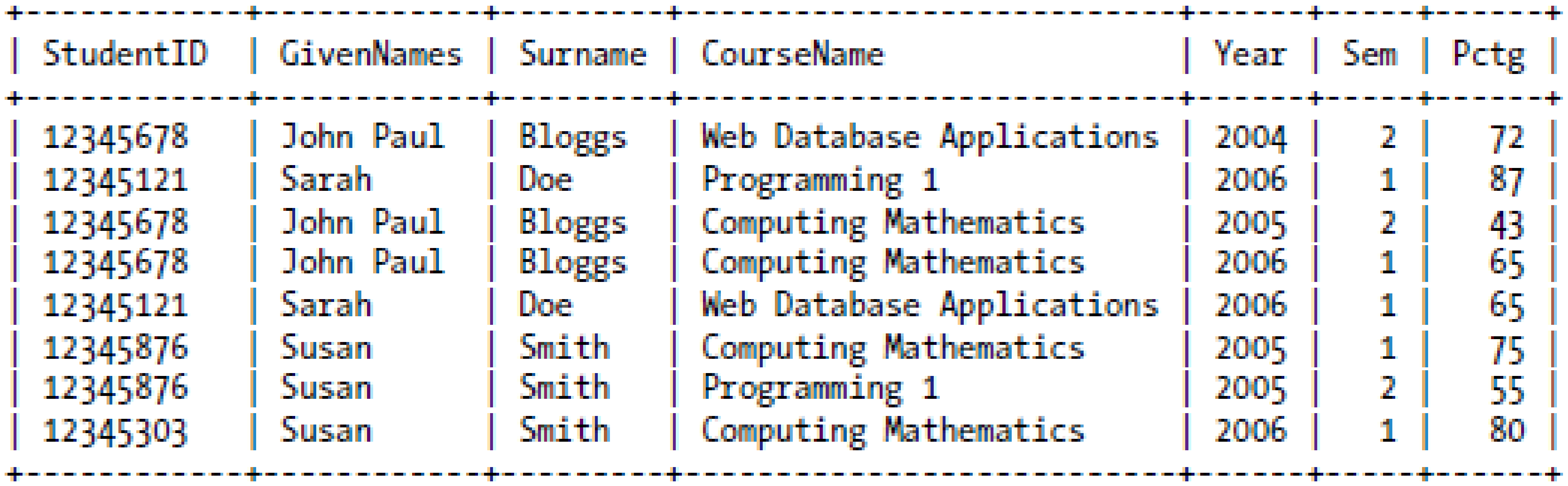
Student_Details Table



```
+------------+------------+----------+---------------------------+------+
| StudentID  | GivenNames | Surname  | CourseName                | Pctg |
+------------+------------+----------+---------------------------+------+
| 12345678   | John Paul  | Bloggs   | Web Database Applications |   72 |
| 12345121   | Sarah      | Doe      | Programming 1             |   87 |
| 12345678   | John Paul  | Bloggs   | Computing Mathematics     |   43 |
| 12345678   | John Paul  | Bloggs   | Computing Mathematics     |   65 |
| 12345121   | Sarah      | Doe      | Web Database Applications |   65 |
| 12345876   | Susan      | Smith    | Computing Mathematics     |   75 |
| 12345876   | Susan      | Smith    | Programming 1             |   55 |
| 12345303   | Susan      | Smith    | Computing Mathematics     |   80 |
+------------+------------+----------+---------------------------+------+
```

Example (Cont'D)

Student_Details Table

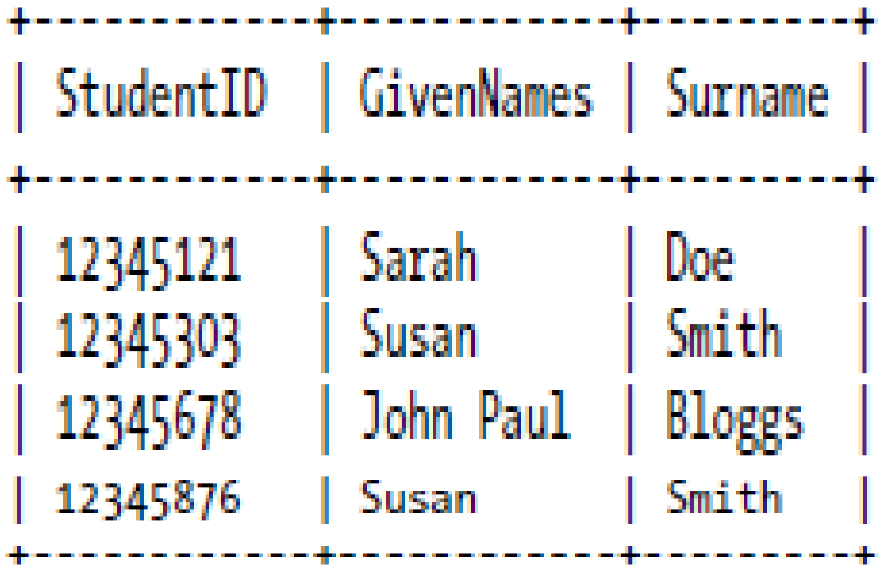


```
+--------------+-------------+----------+-----------------------------+------+-----+------+
| StudentID    | GivenNames  | Surname  | CourseName                  | Year | Sem | Pctg |
+--------------+-------------+----------+-----------------------------+------+-----+------+
| 12345678     | John Paul   | Bloggs   | Web Database Applications   | 2004 |   2 |   72 |
| 12345121     | Sarah       | Doe      | Programming 1               | 2006 |   1 |   87 |
| 12345678     | John Paul   | Bloggs   | Computing Mathematics       | 2005 |   2 |   43 |
| 12345678     | John Paul   | Bloggs   | Computing Mathematics       | 2006 |   1 |   65 |
| 12345121     | Sarah       | Doe      | Web Database Applications   | 2006 |   1 |   65 |
| 12345876     | Susan       | Smith    | Computing Mathematics       | 2005 |   1 |   75 |
| 12345876     | Susan       | Smith    | Programming 1               | 2005 |   2 |   55 |
| 12345303     | Susan       | Smith    | Computing Mathematics       | 2006 |   1 |   80 |
+--------------+-------------+----------+-----------------------------+------+-----+------+
```

To get clear academic information of students we include year and semester of the students

Example (Cont’D)

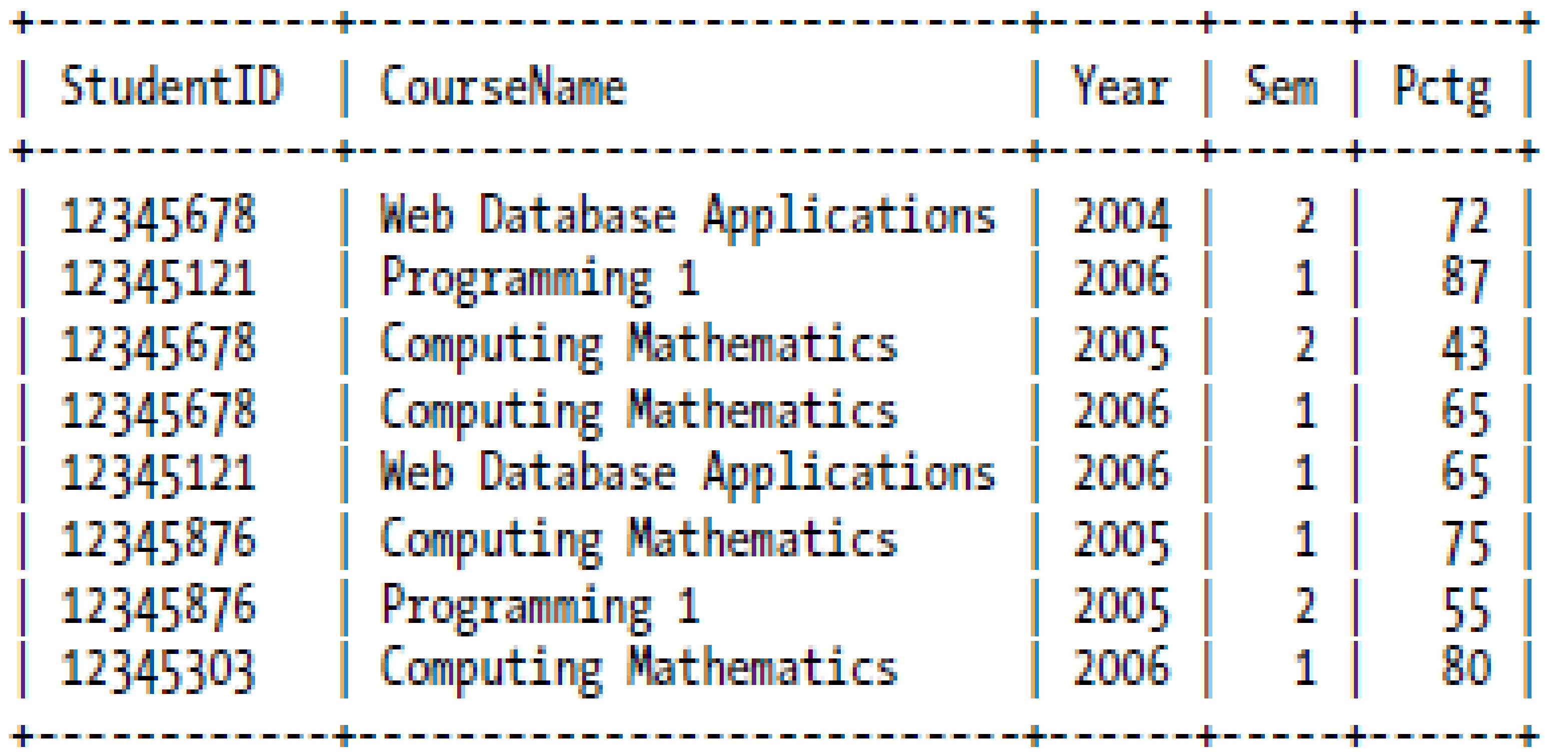
- Student_Details table has become a bit bloated: the student ID, given names, and surname are repeated for every grade
- We could split up the information and create a Student_Details table:



```
+-----------+-----------+---------+
| StudentID | GivenNames | Surname |
+-----------+-----------+---------+
| 12345121  | Sarah     | Doe     |
| 12345303  | Susan     | Smith   |
| 12345678  | John Paul | Bloggs  |
| 12345876  | Susan     | Smith   |
+-----------+-----------+---------+
```

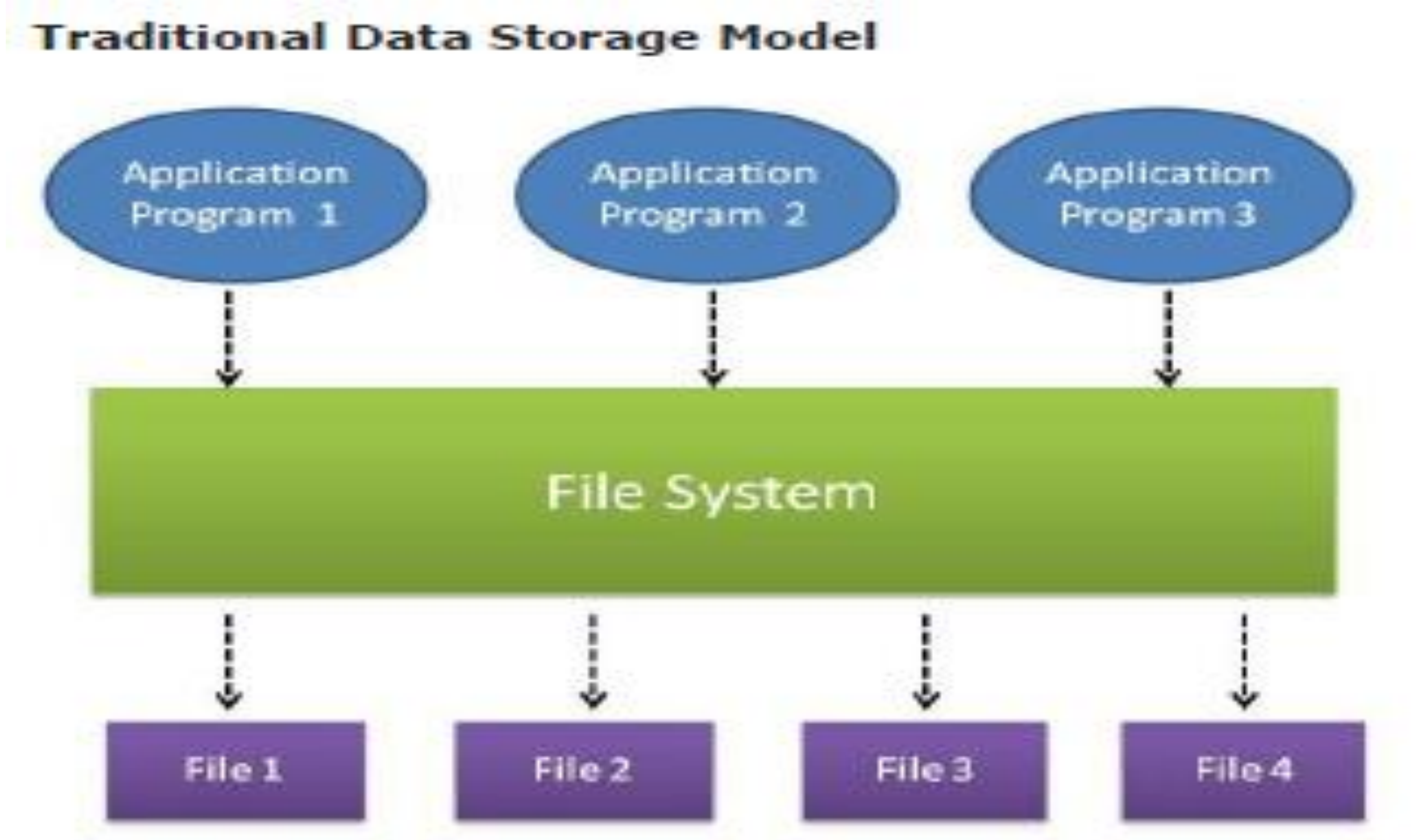
Example

Keep less information in the Student_Grades table:



```
+----------+----------------------------+------+-----+------+
| StudentID | CourseName                | Year | Sem | Pctg |
+----------+----------------------------+------+-----+------+
| 12345678 | Web Database Applications | 2004 |   2 |   72 |
| 12345121 | Programming 1              | 2006 |   1 |   87 |
| 12345678 | Computing Mathematics      | 2005 |   2 |   43 |
| 12345678 | Computing Mathematics      | 2006 |   1 |   65 |
| 12345121 | Web Database Applications | 2006 |   1 |   65 |
| 12345876 | Computing Mathematics      | 2005 |   1 |   75 |
| 12345876 | Programming 1              | 2005 |   2 |   55 |
| 12345303 | Computing Mathematics      | 2006 |   1 |   80 |
+----------+----------------------------+------+-----+------+
```

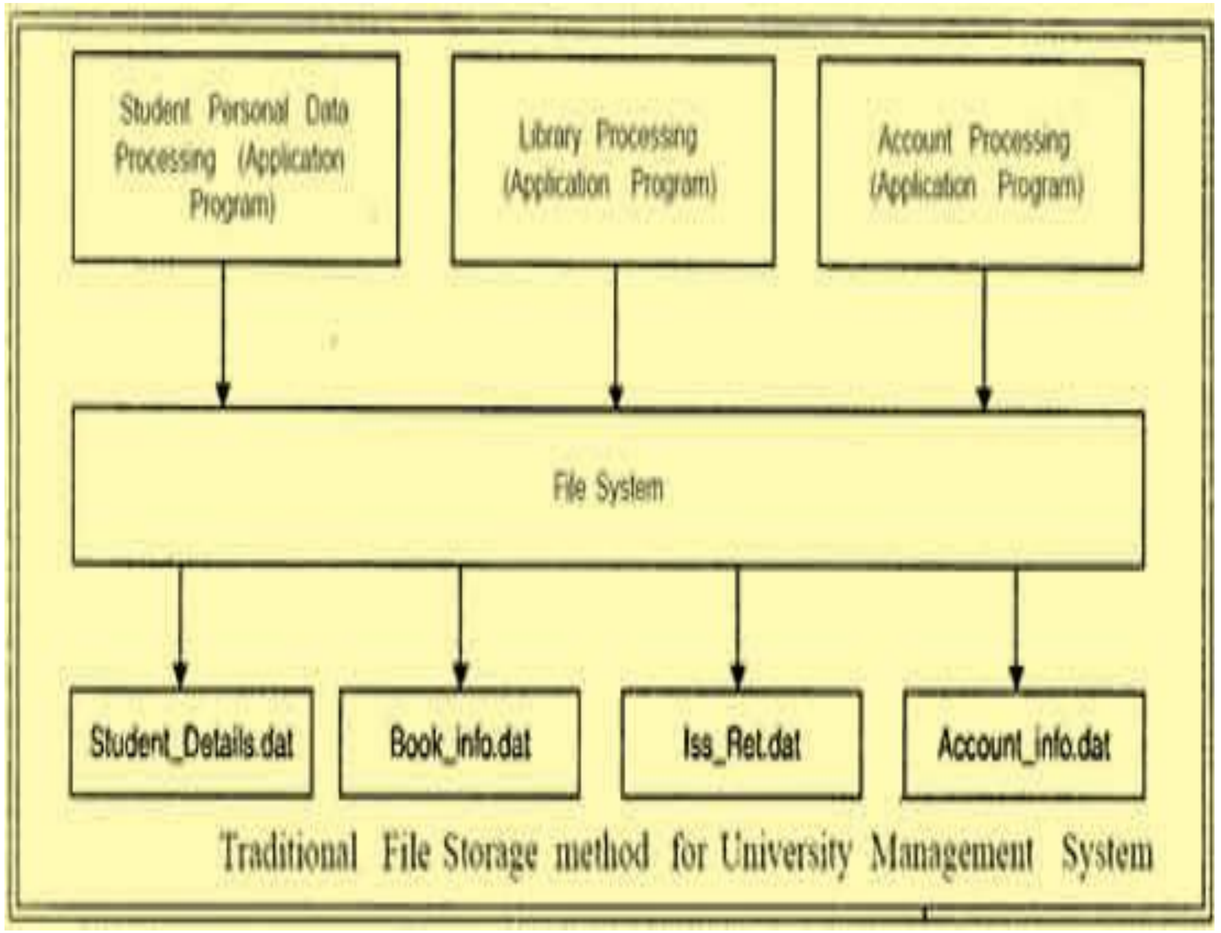
Traditional File-oriented Data Storage



- In traditional approach, information is stored in flat files which are maintained by the file system under the operating system's control.
- Application programs go through the file system in order to access these flat files

Traditional File Processing System

- Store information in flat files which are maintained by the file system under the operating system's control
- Here, flat files are files containing records having no structured relationship among them
- The file handling which we learn under C/C ++ is the example of file processing system
- The Application programs written in C/C ++ like programming languages go through the file system to access these flat. files as shown.


Student Personal Data Processing (Application Program)
Library Processing (Application Program)
Account Processing (Application Program)
File System
Student_Details.dat
Book_info.dat
Iss_Ret.dat
Account_info.dat
Traditional File Storage method for University Management System

Example for flat file (Student)

Roll No	Name	Place	Gender	Dept No	Dname
101	Ananya	Cochin	F	D1	Computer Applications
102	Gauri	Hubli	F	D2	Electronics
103	Harsha	Manipal	M	D3	Mathematics
104	John	Mysore	M	D1	Computer Applications
105	Peter	Allahabad	M	D1	Computer Applications
106	Faizal	Poona	M	D2	Electronics
107	Lisa	Kanpur	F	D2	Electronics

Purpose of Database Systems

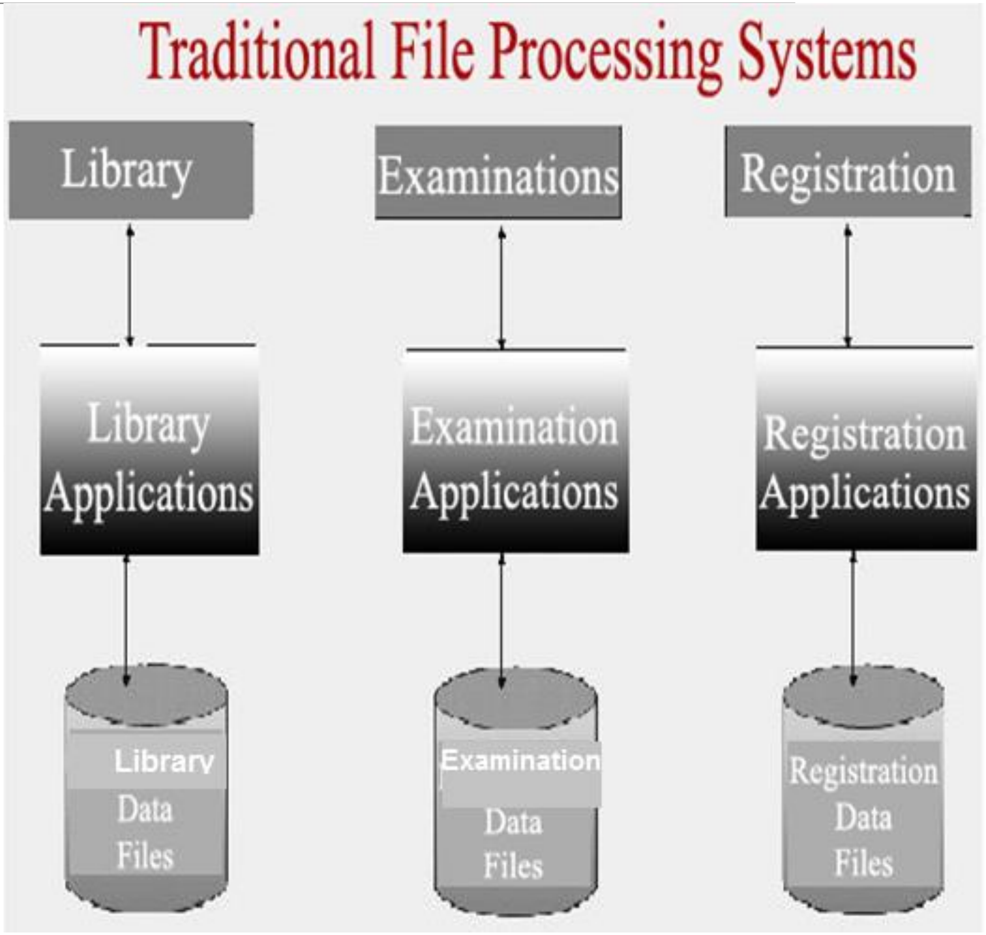
Database management systems were developed to handle the following difficulties of typical file-processing systems supported by conventional operating systems:

1. Data redundancy and inconsistency
2. Difficulty in accessing data
3. Data isolation – multiple files and formats
4. Integrity problems-consistency constraints
5. Atomicity of updates
6. Concurrent access by multiple users
7. Security problems

Drawbacks of using file systems to store data...

1) Data redundancy and inconsistency

- Multiple file formats, duplication of information in different files
  - Ex: The student information is repeated in all the three files
  - This repeated information is known as ***data redundancy*** which leads to higher storage and access cost
- It also leads to data inconsistency , various copies of the same data may not agree



Drawbacks of using file systems to store data…

- **Difficulty in accessing data**
  - ➢ Need to write a new program to carry out each new task.
  - ➢ Do not allow needed data to be retrieved in a convenient and efficient manner.
- **Data isolation — multiple files and formats**
  - Writing new application programs to retrieve the appropriate data is difficult

Drawbacks of using file systems to store data…

- **Integrity problems**
  - The data values stored in the database must satisfy certain types of consistency constraints
  - Integrity constraints (e.g. “*dept_name* of student must be a valid department name”) become “buried” in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Drawbacks of using file systems to store data…

- **Atomicity of updates**
  - Failures may leave database in an inconsistent state with partial updates carried out
    - Example: Transfer of funds from one account to another should either complete or not happen at all

Drawbacks of using file systems to store data…

- **Concurrent access by multiple users**
  - Concurrent accessed needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people A & B reading a balance (say 500) and updating it by withdrawing money (say 100 by A & 50 by B) at the same time.

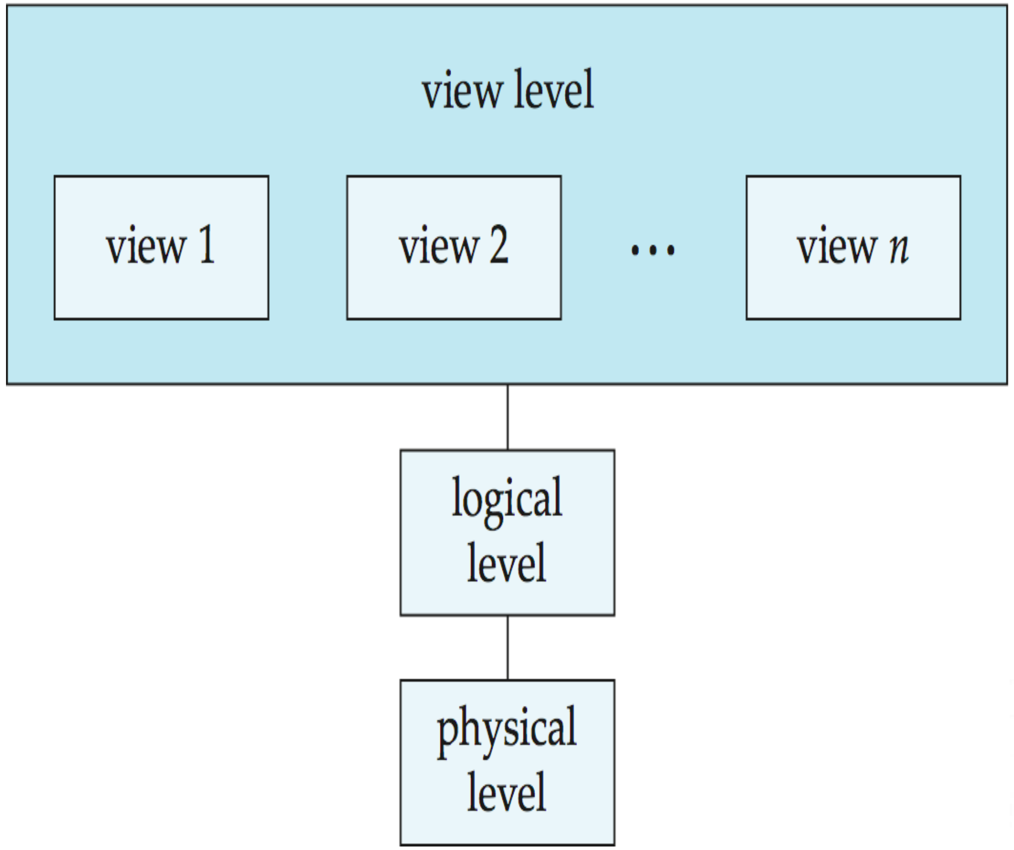
Drawbacks of using file systems to store data…

- **Security problems**
  - Application programs are added to the file-processing system in an ad hoc manner enforcing such security constraints is difficult.
  - Hard to provide user access to some, but not all, data.

View of Data

- Database System provide users with abstract view of the data.
- DBMS hides certain details of how the data are stored and maintained. This is ***data abstraction.***
- This simplifies users interactions with the system.

View of Data



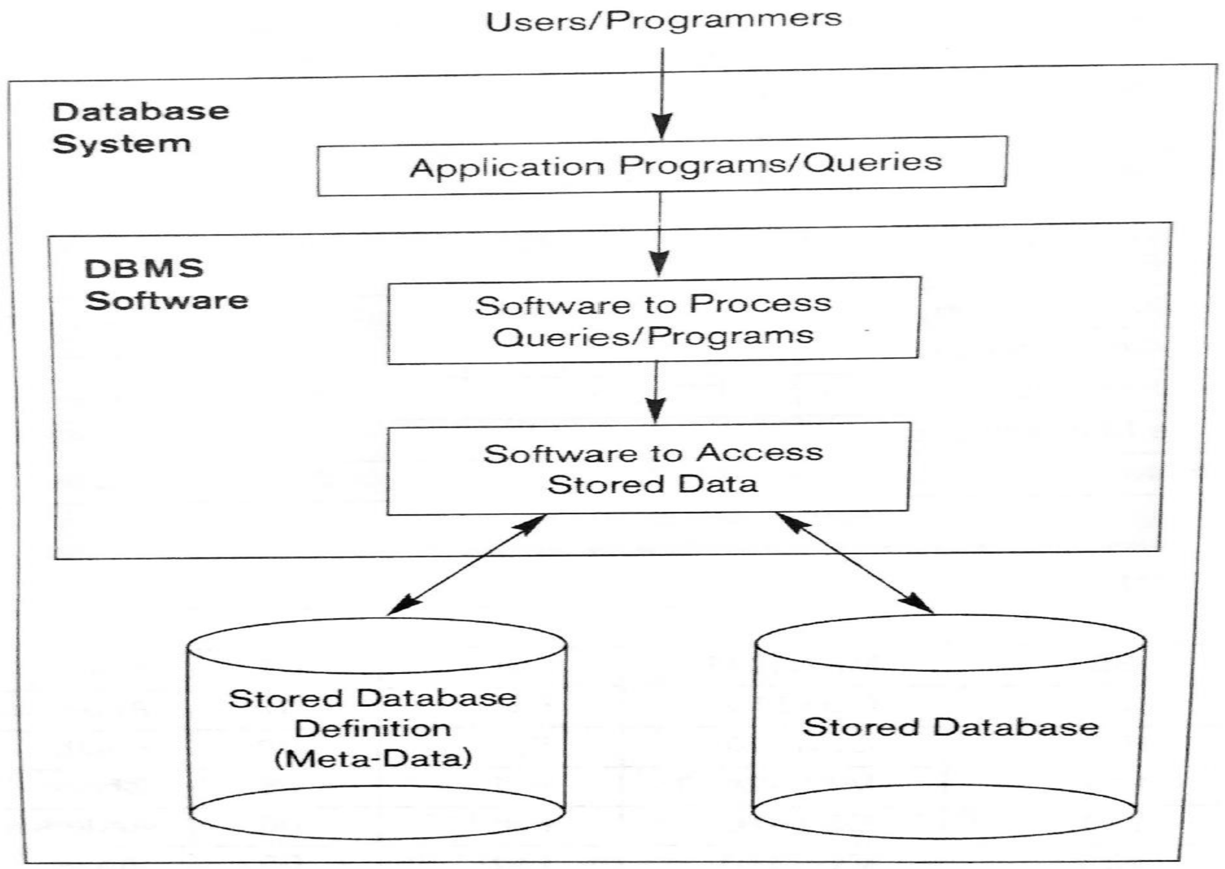

Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

Fig 1 Three levels of data abstraction

Levels of Abstraction

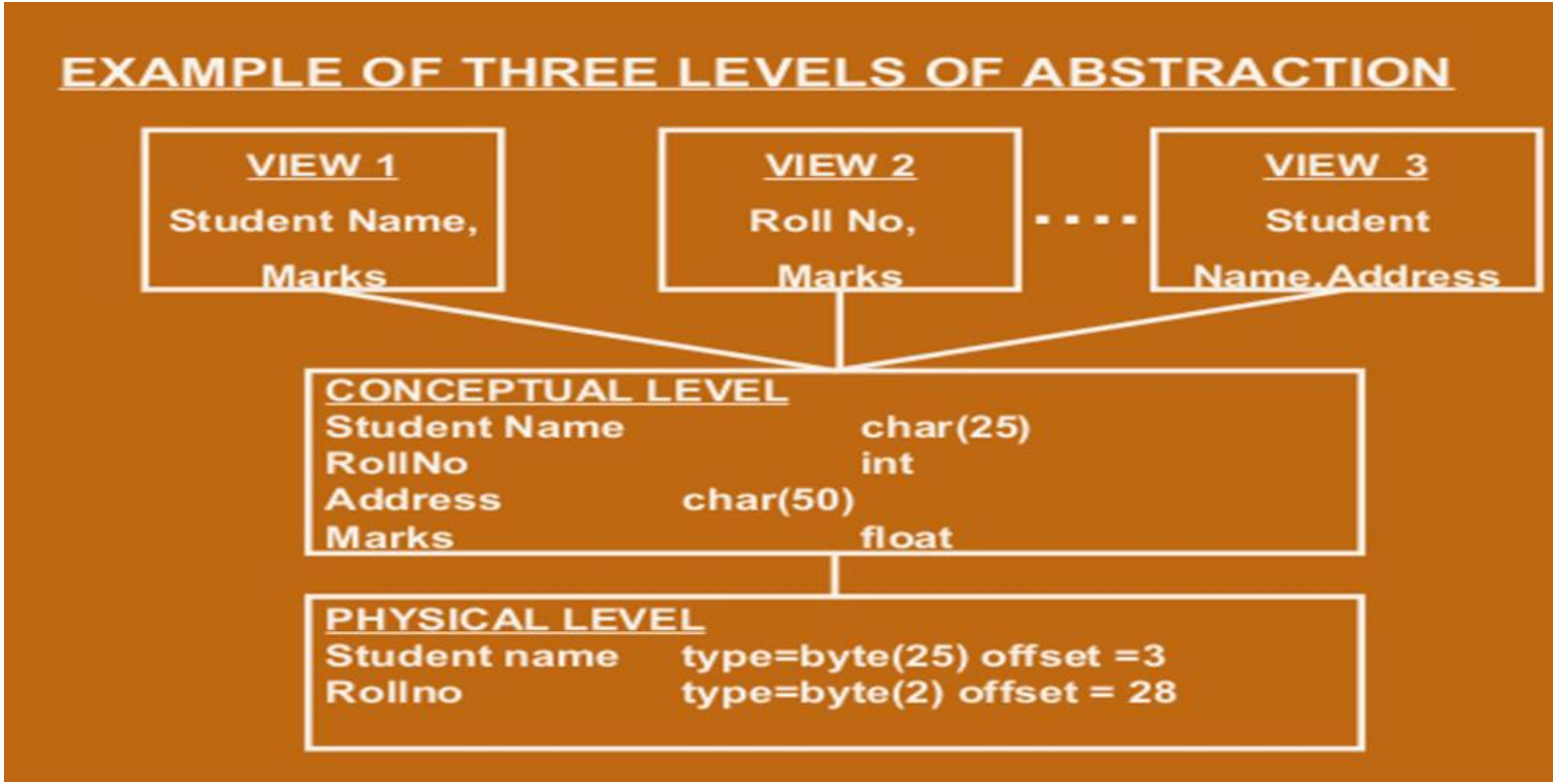
Physical level: Describes how a record (e.g. customer) is actually stored.

(describes complex low-level data structures in detail.)

Logical level: Describes what data are stored in the database, and the relationships among the data.

```
type student = record
                name: string;
                rollno: integer;
                city: string;
                gender: string;
end;
```

View level: application programs hide details of data types. Views can also hide information for security purposes.


EXAMPLE OF THREE LEVELS OF ABSTRACTION
VIEW 1
Student Name, Marks
VIEW 2
Roll No, Marks
. . . .
VIEW 3
Student Name,Address
CONCEPTUAL LEVEL
Student Name char(25)
RollNo int
Address char(50)
Marks float
PHYSICAL LEVEL
Student name type=byte(25) offset =3
Rollno type=byte(2) offset = 28

Instances and Schemas

- Schema – The overall design of the database
  - Physical schema – database design at physical level
  - Logical schema – database design at the logical level
  - Subschemas- several schemas at the view level
- Instance – The collection of information stored in the database at a particular moment.

Data Independence

- Ability to modify a schema definition in one level without affecting a schema definition in the other levels.
- Two levels of data independence
  - Physical data independence
  - Logical data independence

Data Independence Types

Physical Data Independence

- The ability to modify the physical schema without changing the logical schema.

Logical Data Independence

- The ability to modify the logical schema without changing the view level.

The interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Data Models

- A collection of conceptual tools for describing:
  - Data
  - Data relationships
  - Data semantics
  - Consistency constraints

Data Models

- Relational model
- Entity-relationship model
- Object-based Data model
- Semistructured model

Example for flat file (Student)

Roll No	Name	Place	Gender	Dept No	Dname
101	Ananya	Cochin	F	D1	Computer Applications
102	Gauri	Hubli	F	D2	Electronics
103	Harsha	Manipal	M	D3	Mathematics
104	John	Mysore	M	D1	Computer Applications
105	Peter	Allahabad	M	D1	Computer Applications
106	Faizal	Poona	M	D2	Electronics
107	Lisa	Kanpur	F	D2	Electronics

Example for Relational Model

Student

Roll No	Name	Place	Gender	Dept No
101	Ananya	Cochin	F	D1
102	Gauri	Hubli	F	D2
103	Harsha	Manipal	M	D3
104	John	Mysore	M	D1
105	Peter	Allahabad	M	D1
106	Faizal	Poona	M	D2
107	Lisa	Kanpur	F	D2

Department

Dept No	Dname
D1	Computer Applications
D2	Electronics
D3	Mathematics

Relational Model

Example of record based model

Customer

name	ssn	street	city	account-number
Johnson	192-83-7465	Alma	Palo Alto	A-101
Smith	019-28-3746	North	Rye	A-215
Johnson	192-83-7465	Alma	Palo Alto	A-201
Jones	321-12-3123	Main	Harrison	A-217
Smith	019-28-3746	North	Rye	A-201

Account

account-number	balance
A-101	500
A-201	900
A-215	700
A-217	750

Collection of tables to represent both data and the relationships among those data. Tables are also known as **relations**.

Entity-Relationship Model

Entities-collection of basic objects and relationships among these objects

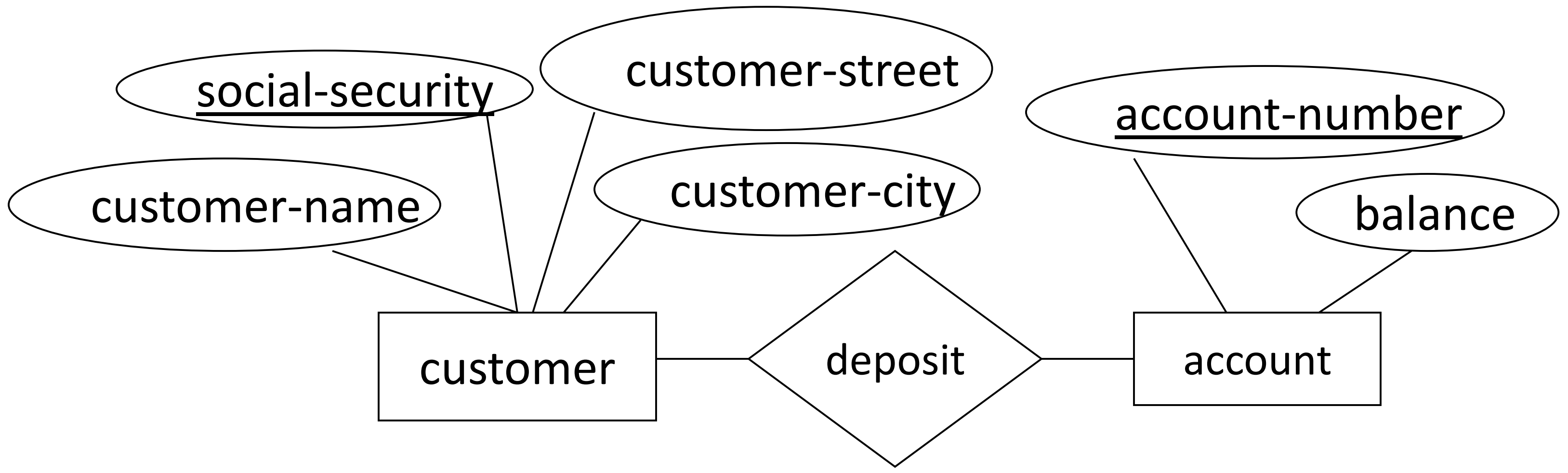


Fig 3 ER model(old notation)

Entity Relationship Model

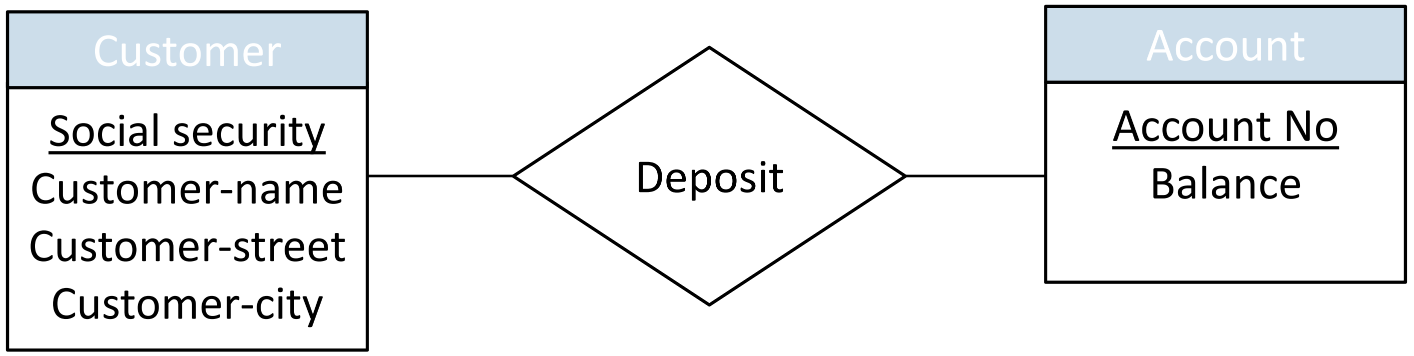


Fig 4 ER model(New notation)

Database Languages

- ➢ Data Manipulation Language
- ➢ Data Definition Language

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
- Two classes of languages
  - Procedural – user specifies what data is required and how to get those data
  - Nonprocedural(Declarative) – user specifies what data is required without specifying how to get those data
- Portion of DML that involves information retrieval is called a query language.

Data Definition Language (DDL)

- Specification notation for defining the database schema
- DDL compiler generates a set of tables stored in a data dictionary
- Data dictionary contains ***metadata*** (data about data)
- Data storage and definition language(DSL) – special type of DDL in which the storage structure and access methods used by the database system are specified.

Specification notation for defining a table.

Example: **create table** *instructor* **(**

```
ID          char(5),
name        varchar(20),
dept_name   varchar(20),
salary      numeric(8,2) )
```

Data Definition Language (DDL)

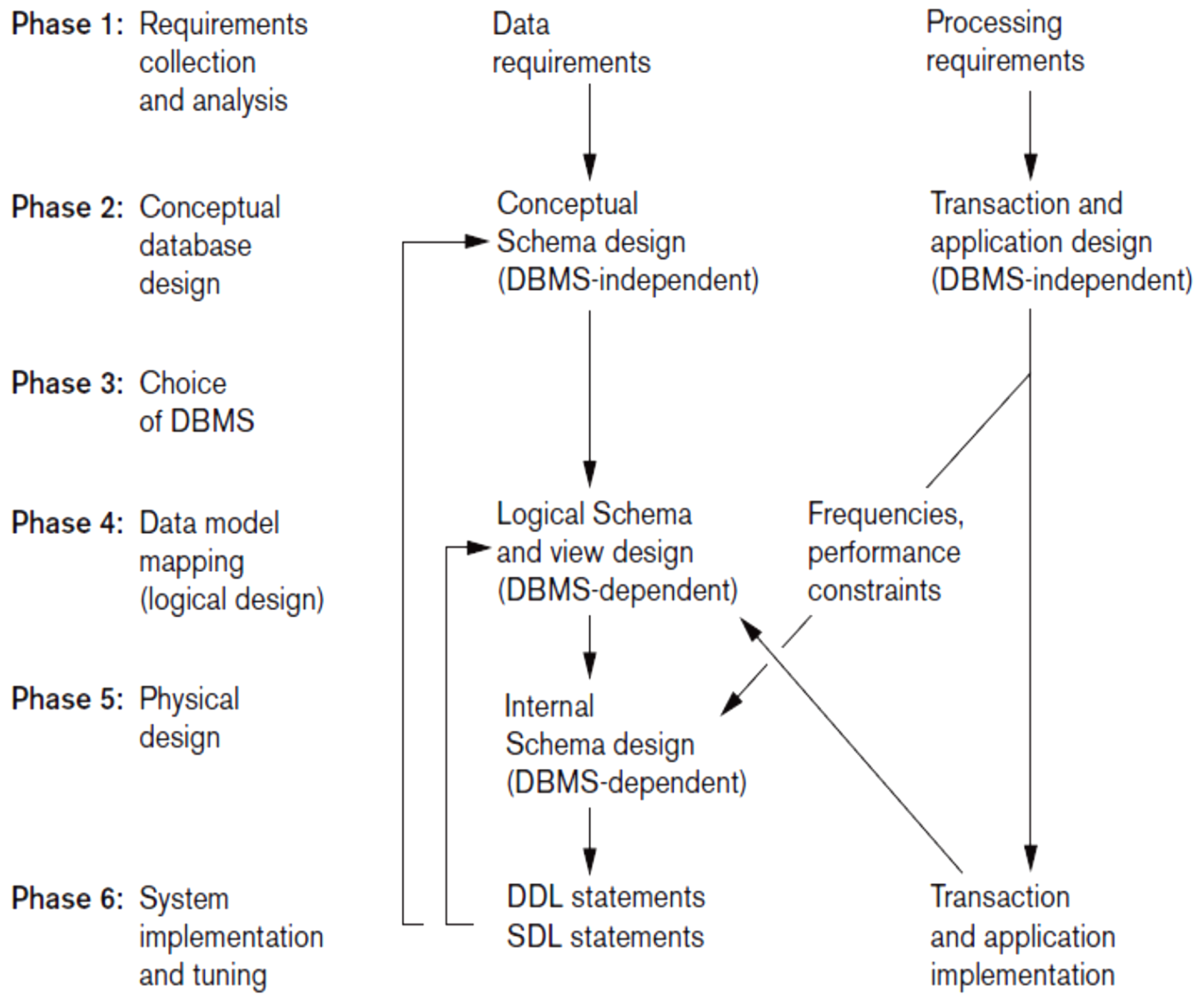
- The data values stored in the database must satisfy certain consistency constraints.
  - Domain constraints- simplest is data type , size, range & pattern etc
  - Referential Integrity – A value that appears in one relation for a given set of attributes also appears in a certain set of attributes in another relation
  - Assertions – An assertion is any condition that the database must always satisfy

    ```
    CREATE ASSERTION [ assertion_name ] CHECK ( [ condition ] );
    ```

  - Authorization – read, insert, update, delete

Database Design

- ➢Database design is required to produce an efficient, high-quality and minimum cost database



Phase 1: Requirements collection and analysis
Phase 2: Conceptual database design
Phase 3: Choice of DBMS
Phase 4: Data model mapping (logical design)
Phase 5: Physical design
Phase 6: System implementation and tuning
Data requirements
Processing requirements
Conceptual Schema design (DBMS-independent)
Transaction and application design (DBMS-independent)
Logical Schema and view design (DBMS-dependent)
Frequencies, performance constraints
Internal Schema design (DBMS-dependent)
DDL statements SDL statements
Transaction and application implementation

Requirement Collection and Analysis

- It is the process of knowing and analyzing the expectations of the users for the new database application done by a team of analysts or requirement experts
- The requirement specification such as Object Oriented Analysis, Data Flow Diagram are used to transform the initial requirement into a better structured form

Conceptual Database Design

- Database designer selects a suitable data model and translates the data requirements resulting from the previous phase into a conceptual database schema
- ER Diagram is generally used to represent the conceptual database design
- Designer reviews the schema to confirm that all data requirements are indeed satisfied and are not conflict with one another
- Designer can also examine the design to remove any redundant features-NORMALIZATION

Choice of a DBMS

- The Choice of a DBMS depends on many factors such as *cost*, *DBMS features and tools*, underlying *model, portability* and *DBMS hardware* requirements
- The technical factors that affect the choice of a DBMS are the *type of DBMS*, *storage structures* and *access path* that *DBMS supports*, the *interfaces* available, the types of *high-level query languages* and the *architecture* it supports

Logical Database Design

- Once an appropriate DBMS is chosen, the next step is to map the high-level conceptual schema onto the implementation data model of the selected DBMS
- In this phase, the database designer moves from an abstract data model to the implementation of the database

Physical Database Design

- In this phase, the physical features such as storage structures, file organization and access paths for the database files are specified to achieve good performance
- The various options for file organization and access paths include various types of indexing, clustering of records, hashing techniques etc

Database system implementation

- Here the database system can be implemented
- DDL statements of the selected DBMS are used and compiled to create the database schema and database files, and finally the database is loaded with the data

Testing and Evaluation

- The database is tested and fine tuned for the performance , integrity, concurrent access and security constraints
- This is carried out in parallel with application programming
- If the testing fails, various actions are taken such as modification of physical design, logical design or upgrading or changing DBMS software or hardware

SYSTEM STRUCTURE of DBMS

Data Storage and Querying

- The functional components of a database system can be broadly divided into the storage manager and the query processor
- Storage manager – Manages large amount of space
- Query Processor – Simplify and facilitate access to the data.

Storage Manager

- A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible for the following tasks:
  - Interaction with the file manager
  - Efficient storing, retrieving, and updating of data

figure

Storage Manager

The storage manager component include:

- **Authorization and integrity manager** – tests for the satisfaction of integrity constraints and checks the authority of the users to access data
- **Transaction manager** – ensures that database remains in a consistent state despite system failures and that concurrent transaction execution proceeds without conflicting
- **File Manager** – allocation of space in the disk storage and manages the data structures used to represent information stored on disk
- **Buffer Manager** – Fetching data from disk storage into main memory and deciding what data to cache in main memory

Storage Manager

Data structures used are:

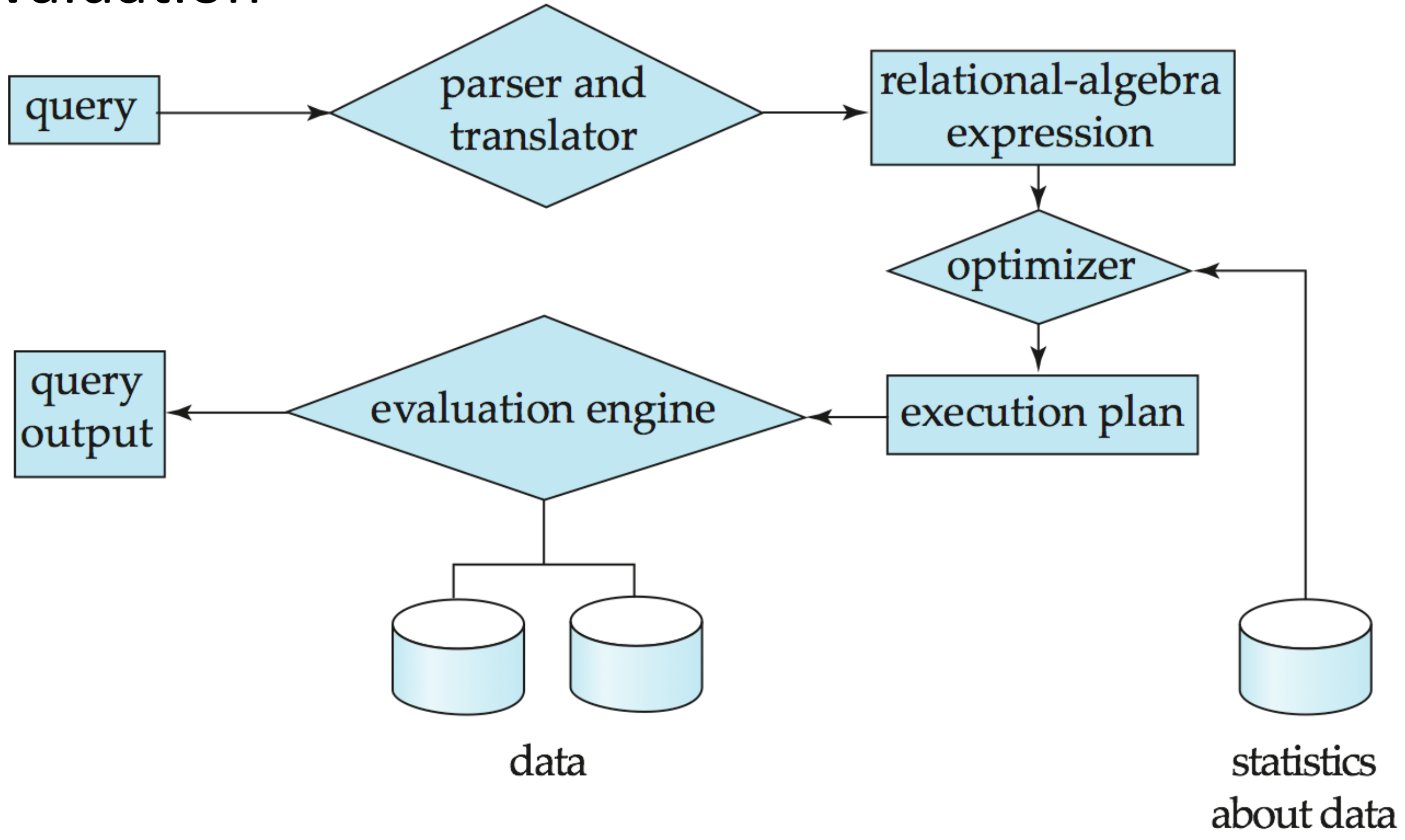
- Data files – store the database
- Data dictionary – stores metadata
- Indices – provide fast access to data items

Transaction Management

- A transaction is a collection of operations that performs a single logical unit in a database application.
- Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g. power failures and operating system crashes) and transaction failures.
- Transaction manager consists of concurrency control manager and the recovery manager
- Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.
- Recovery manager - Ensuring the atomicity and durability property

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Query Processor

Components:

- **DDL interpreter** – interprets DDL statements and records the definitions in the data dictionary
- **DML Compiler** – translates DML statements in a query language into an evaluation plan. It also performs **query optimization**.
- **Query Evaluation Engine** – executes low level instructions generated by the DML compiler

Database Administrator

- Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs
- Database administrator's duties include:
  - Schema definition
  - Storage structure and access method definition
  - Schema and physical organization modification
  - Granting user authority to access the database
  - Periodically backing up the data
  - Ensuring enough free disk space is available
  - Monitoring the jobs that may degrade performance and responding to changes in requirements

Database Users

Users are differentiated by the way they expect to interact with the system.

Application programmers: are computer professionals who write application programs

Specialized users: write specialized database applications that do not fit into the traditional data processing framework.
Ex: Expert system/Knowledge database(needs complex data types)

Sophisticated users: form requests in a database query language or data analysis software

Naive users: invoke one of the permanent application programs that have been written previously

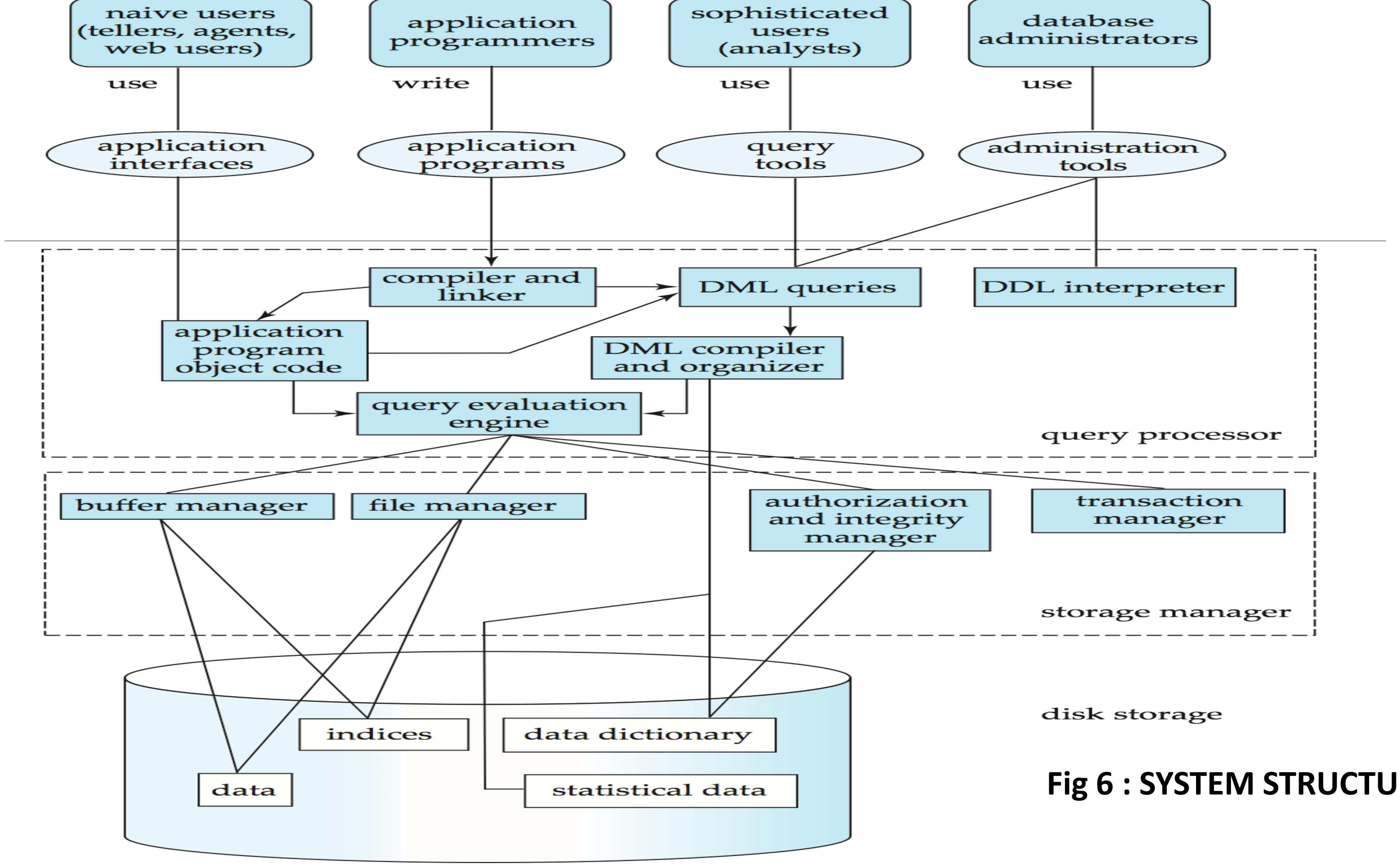
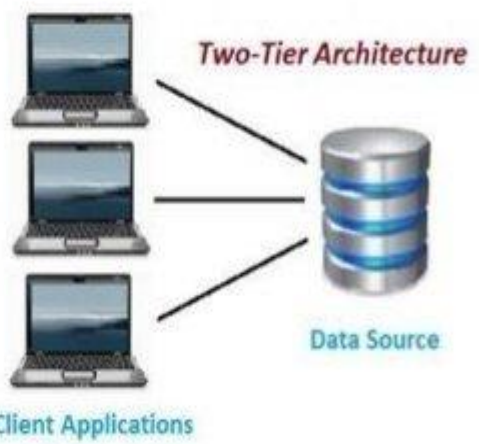


Fig 6 : SYSTEM STRUCTURE

Database Architecture

2-TIER ARCHITECTURE

- It is client-server architecture
- Direct communication
- Run faster(tight coupled)



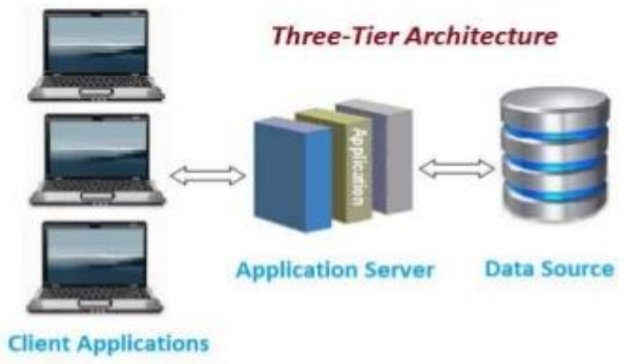
3-TIER ARCHITECTURE

- Web based application
- Three layers:

  1) Client layer

  2) Business layer

  3) Data layer



INTRODUCTION
END OF UNIT 1