Database Systems

DSE 2252

4 Credits

Reference:

Database System Concepts , 6th Edition

Authors:

Abraham Silberschatz

Henry F. Korth

S. Sudarshan

DSE 2252 DATABASE SYSTEMS [3 1 0 4]

Introduction: Database System Applications, View of data, Database languages, Database users and Administrator.
Introduction to Relational Model: database schema, keys, schema diagrams, Relational Query Languages, Relational
Introduction to SQL: Data Definition, Basic structure of SQL queries, Basic operations, Set operations, Null values, Aggregate Functions, Nested subqueries, Modification of the database. Intermediate SQL: Join expressions, Views, Transactions, Integrity Constraints, SQL Data types and schemas, Authorization,
Advanced SQL-PL/SQL, Cursors, Functions, Procedures, Triggers, recursive queries, advanced aggregation features.
Database Design and Entity-Relationship Model: Design Process, ER Model, Reduction to Relational schema.
Relational Database design: Functional dependencies, Normal forms, Closure, Canonical cover,
Lossless joins , dependency preserving decomposition
Storage and File structure, Indexing & Hashing.
Query Processing, Overview, Measure of query cost, selection, Join operation, sorting, Evaluation of expressions.
Query Optimization: Overview, Estimating statistics of expression results, Materialized Views.
Transactions: Concepts, Simple transaction model, Transaction atomicity and durability, Transaction Isolation, Serializability, Transaction Isolation Levels.
Concurrency Control- Lock based protocols, Deadlock Handling, Multiple granularity,
Recovery System: Failure classification, Storage, Recovery algorithm, Buffer Management.
Unstructured database: Introduction to NoSQL, Basics of document-oriented database, MongoDB.

DURATION

- Course Plan – 48 hours
- Per Week – 4 hours

Other References -

Fundamentals of Database System, 6th Edition BY Ramez Elmasri, Shamkant Navathe, Addison Wesley Publications Co., 2010.

Database Management System 3rd Edition By Raghu Ramakrishnan, Johannes Gehrke,, 3rd Edition, WCB/McGraw Hill Publisher, 2014.

SOME DEFINITIONS

- *Database* A shared collection of logically interrelated data (and a description of this data), designed to meet the information needs of an organization.
- DBMS(Database Management System) - Database & software system that facilitates the process of defining, constructing, manipulating and sharing databases among users and applications.
- Example: Oracle, MySQL

DATABASE APPLICATIONS

- Enterprise Information- Sales, Accounting, Human resources.
- Banking and Finance-
  - Customer, Accounts, Card details & transaction
- Telecommunication:
  - Calls, texts, and data usage, generating bills, balances, n/w information
- Social-media
- Online advertisements
- Document databases

The database systems arose in the 1960s

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 two tables Student and Student_Grade table

EXAMPLE (Cont'D)

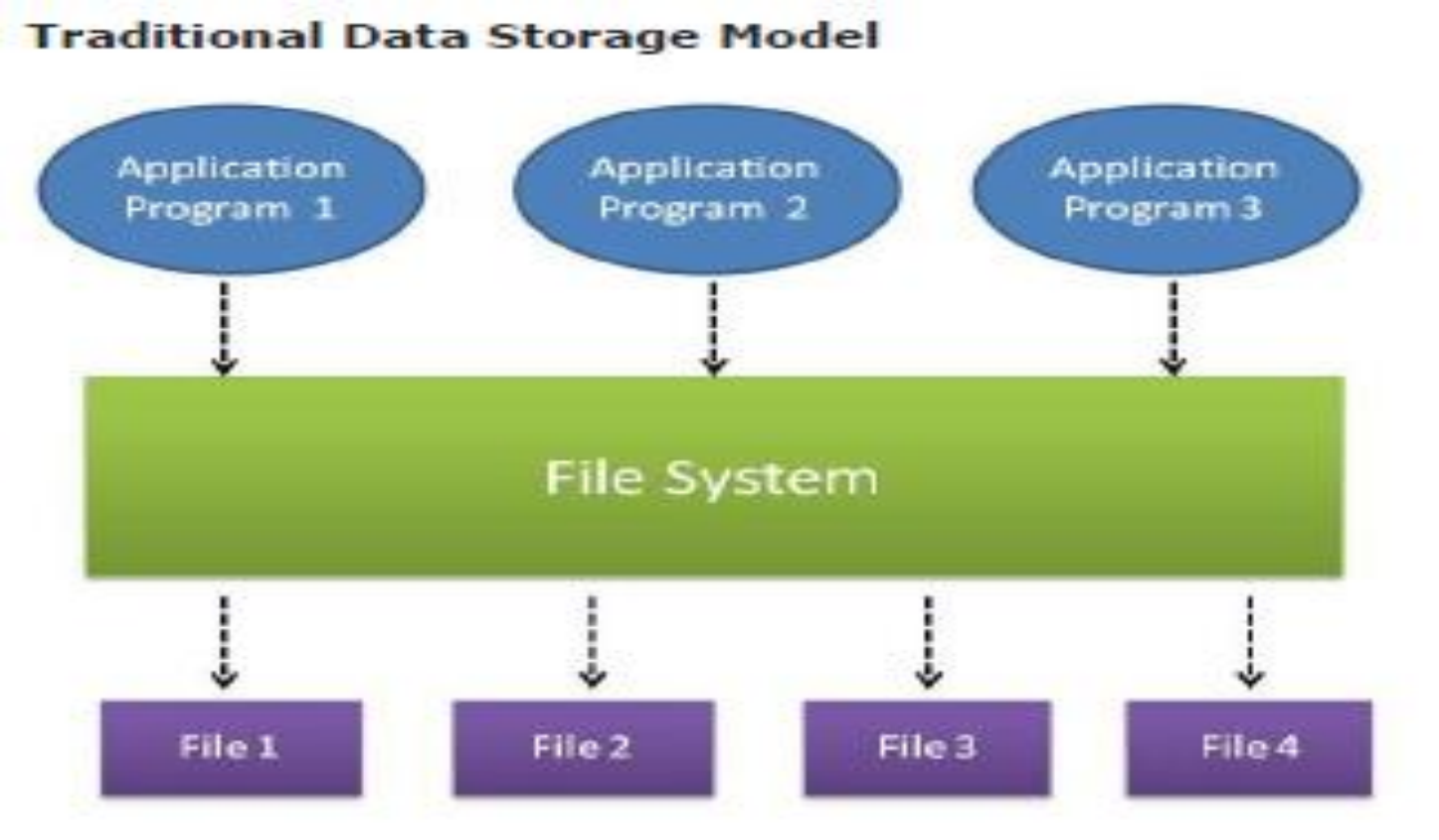
StudentID	CourseName	Year	Sem	Pctg
12345678	Web Database Applications	2004	2	72
12345121	Programming 1	2006	1	87
12345678	Computing Mathematics	2005	2	43
12345678	Computing Mathematics	2006	1	65
12345121	Web Database Applications	2006	1	65
12345876	Computing Mathematics	2005	1	75
12345876	Programming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

Student_Grade Table

StudentID	GivenNames	Surname
12345121	Sarah	Doe
12345303	Susan	Smith
12345678	John Paul	Bloggs
12345876	Susan	Smith

Student

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 access through the file system in order to access these flat files

WHY DATABASE SYSTEMS?

Database systems were developed to handle the following difficulties of a typical file-processing 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-PROCESSING SYSTEMS TO STORE DATA

1. **Data redundancy and inconsistency**
   - Multiple file formats, duplication of information in different files.
     - Ex: if a student has a double major (say, music and mathematics) , his personal information(say Phone number/ Address) may be stored in both the departments.
       - Same personal information about student stored at two different places(redundancy).
     - Inconsistency- various copies of the same data may no longer agree

DRAWBACKS OF USING FILE-PROCESSING SYSTEMS TO STORE DATA

2. **Difficulty in accessing data**
   - Need to write a new program to carry out each new task.
   - **Ex:** A program retrieves "*customer information who are in any given city*". Assume later point of time a requirement arises to "*find customers who are in a given city and balance is above 100000/-*". Now old programs so not work, so you need to put some manual effort or write another program. Both are time consuming.
   - Do not allow needed data to be retrieved in a **convenient** and **efficient** manner.

DRAWBACKS OF USING FILE-PROCESSING SYSTEMS TO STORE DATA

3. **Data isolation — multiple files and formats**
   - Over the time different developers might have developed programs and correspondingly data files(different formats) under different programming languages.
   - i.e. Data are scattered in various files with different file formats.
   - Structure of file is tightly coupled with the program.

DRAWBACKS OF USING FILE-PROCESSING SYSTEMS TO STORE DATA

4. **Integrity problems**
   - Integrity constraints (e.g. "*dept_name field value* of student must be a valid department name" **or** "*Balance* of an Account Maintained by the department must be more than 10000/-") become "buried" in program code rather than being stated explicitly.
   - Hard to add new constraints or change existing ones.

DRAWBACKS OF USING FILE-PROCESSING SYSTEMS TO STORE DATA

5. **Atomicity of updates**
   - Failures may leave database in an inconsistent state if the partial updates are executed.
     - Example: Account A has balance 1000 and B has 2000.
     - Transfer of funds 100/- from one account A to another B should either complete or not happen at all.

DRAWBACKS OF USING FILE-PROCESSING SYSTEMS TO STORE DATA

6. **Concurrent access by multiple users**
   - Concurrent accessed needed for performance
   - Uncontrolled concurrent accesses can lead to inconsistencies
     - Example: Two people **X** & **Y** read a balance (say 500) of an account A. **X** updates balance by withdrawing money (say 100). Same time Y also reads balance(500) to do withdraw 50. Result (final Balance) vary depending which person(X or Y) updates balance last.

DRAWBACKS OF USING FILE-PROCESSING SYSTEMS TO STORE DATA

7. **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.

Database systems offer solutions to all the above problems

VIEW OF DATA

- A major purpose of a database system is to provide users with an *abstract* view of the data.
- DBMS hides certain details of how the data are stored and maintained.This is *data abstraction*

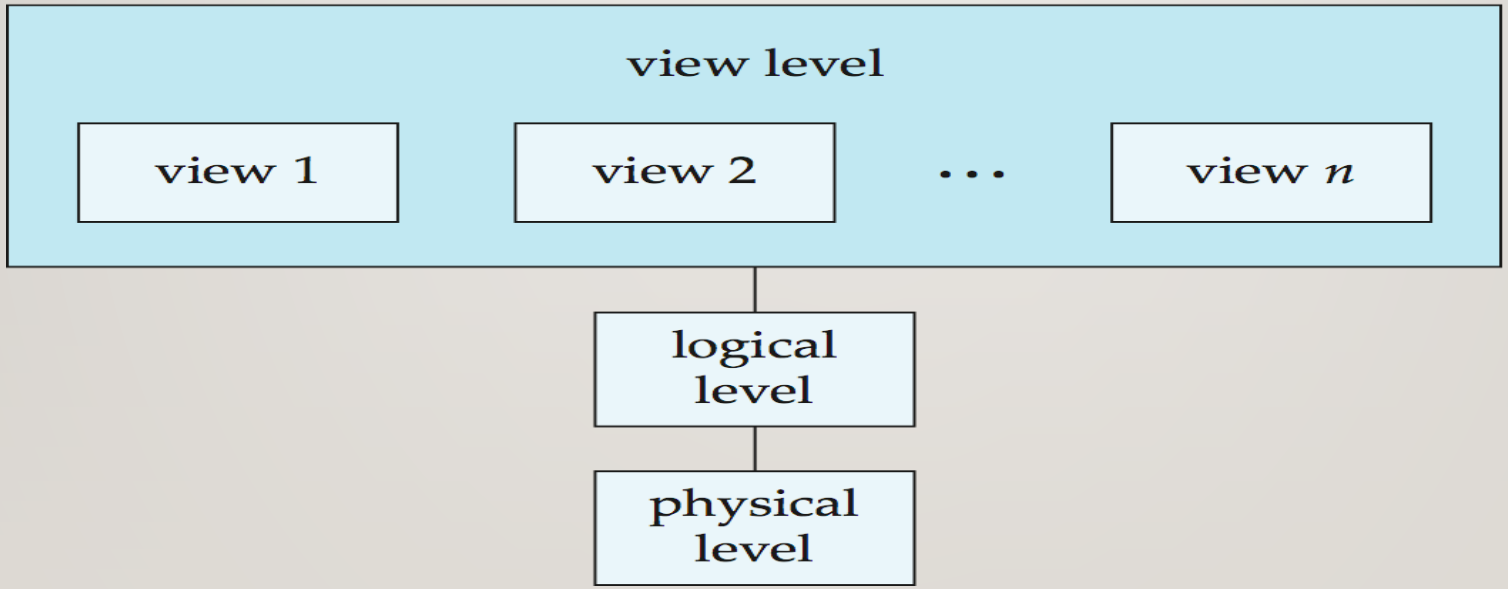


Fig 1 Three levels of data abstraction

LEVELS OF ABSTRACTION

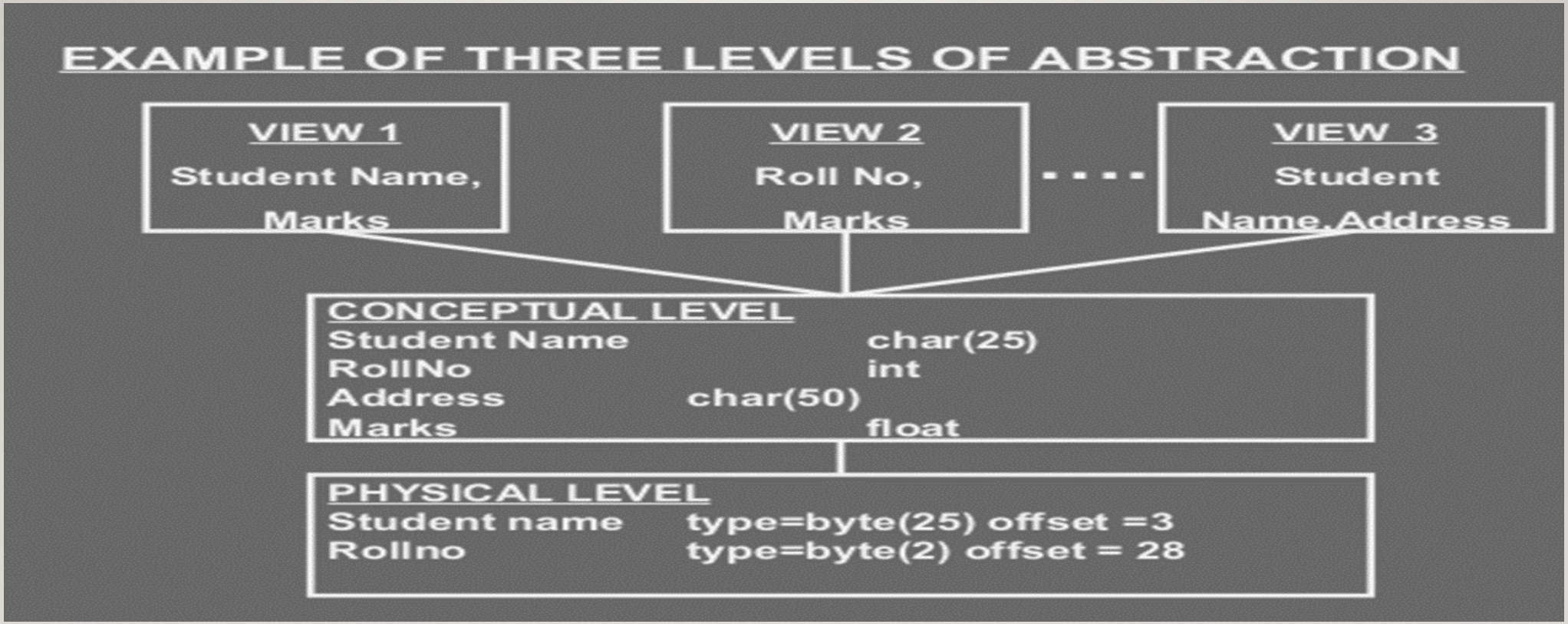
- **Physical level:** Describes how a record is actually stored.
  - (describes entire database complex low-level data structures in detail.)
- **Logical level:** Describes entire database using relatively simpler structures. Tells about kind of data is stored and the relationships among the data.

```
type customer = record
                    name: string;
                    street: char;
                    city: integer;
end;
```

- **View level:** A view describes only part of the entire database using simple structure. There may be several such views. For example Application programs hide details of data types.

  Views can also hide information (e.g. salary) for security purposes.

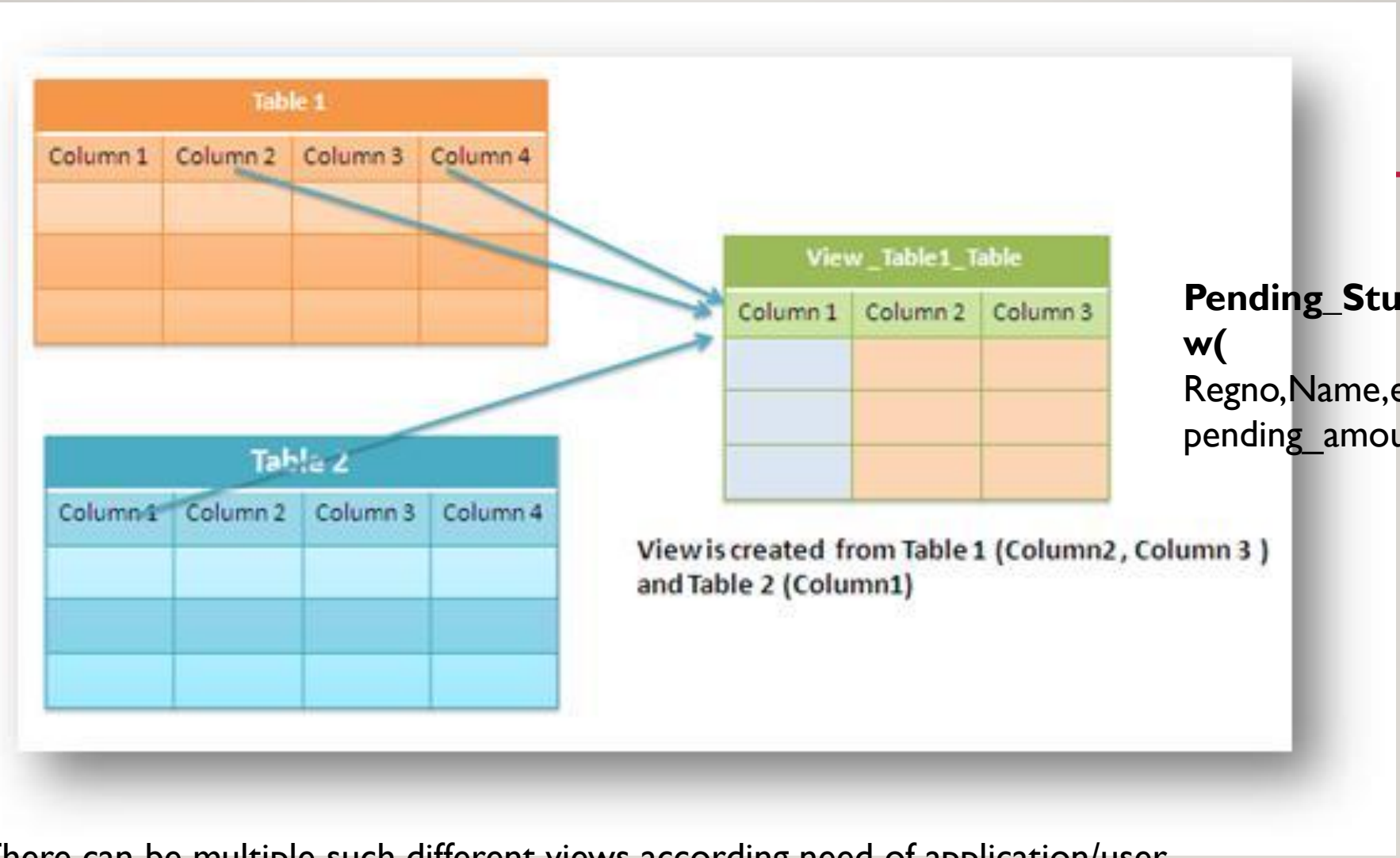
LEVELS OF ABSTRACTION



Views

Academics(
Regno, Name, email,Mark1, Mark2,Mark3, Grade)

Fees_paid
(Regno, Year1_fees, Year2_Fees,Total_pending)



Pending_Stud_Fees_View(
Regno,Name,emailed, pending_amount**)**

There can be multiple such different views according need of application/user

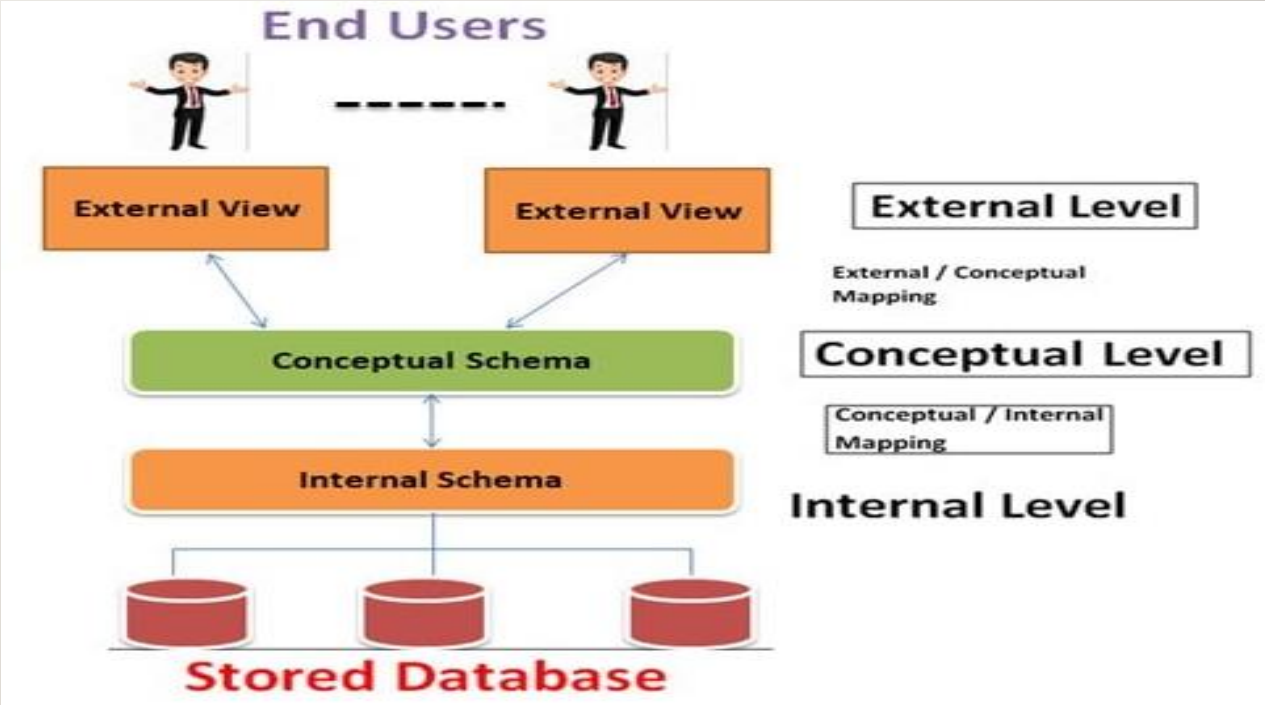
INSTANCES AND SCHEMAS

- Schema – The overall design(structure) 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 is stored in which schema ? All or any one ? Which one?

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(mappings) 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

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-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

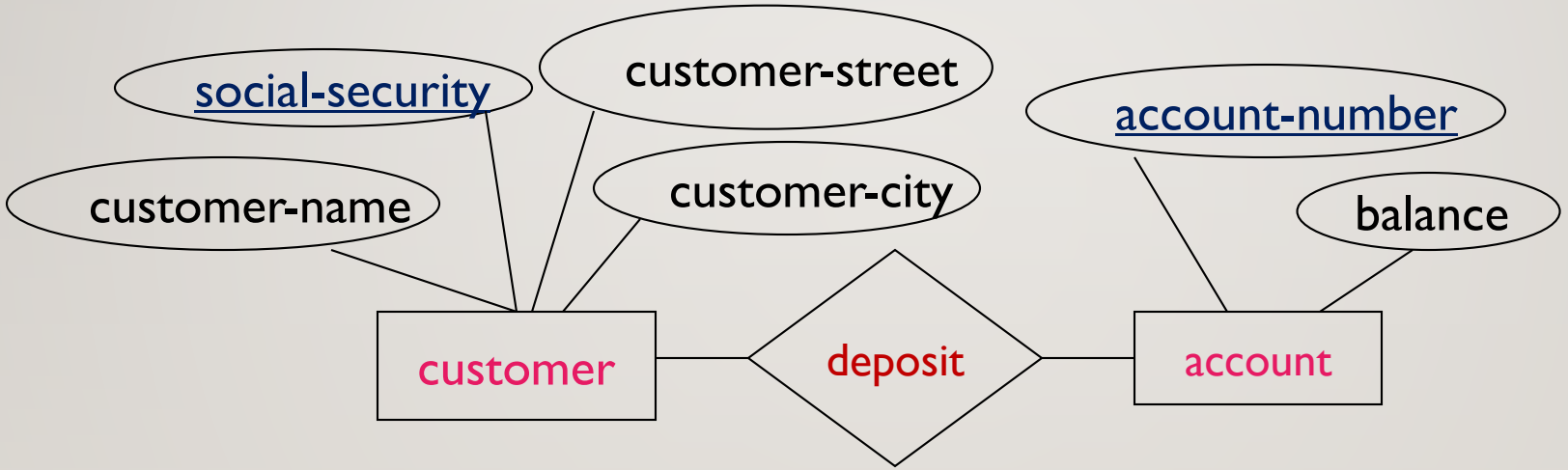


Fig 3 ER model (old notation)

widely used in database design

ENTITY RELATIONSHIP MODEL

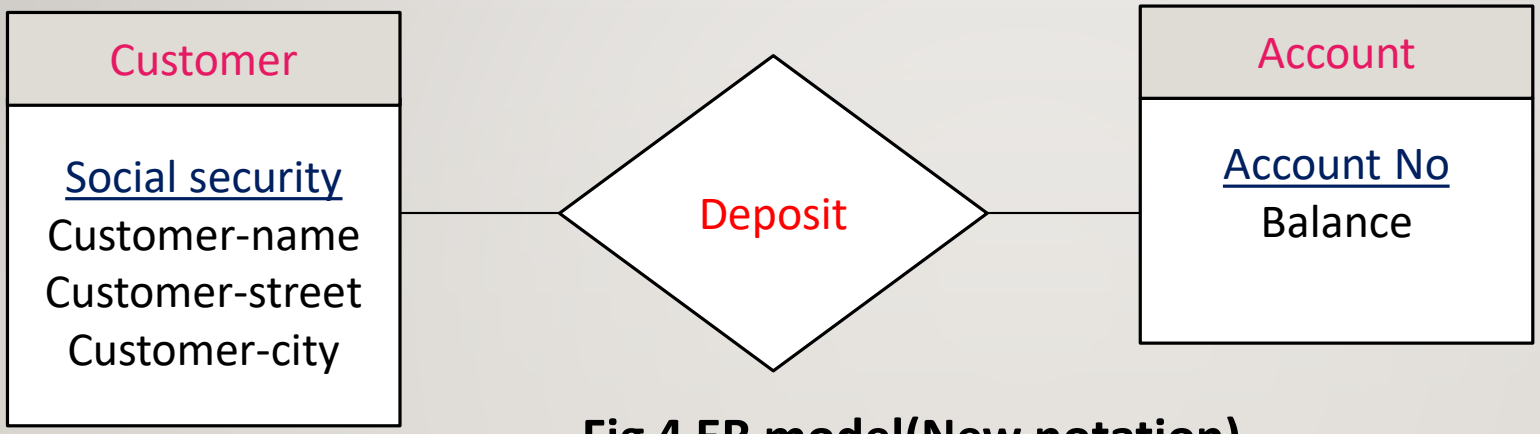


Fig 4 ER model(New notation)

Normalization

Another method for designing a relational database.

To design set of schema with minimum redundancy.

Ensuring schema are in appropriate normal forms. The approach is based on functional dependencies

SEMI-STRUCTURED DATA MODEL

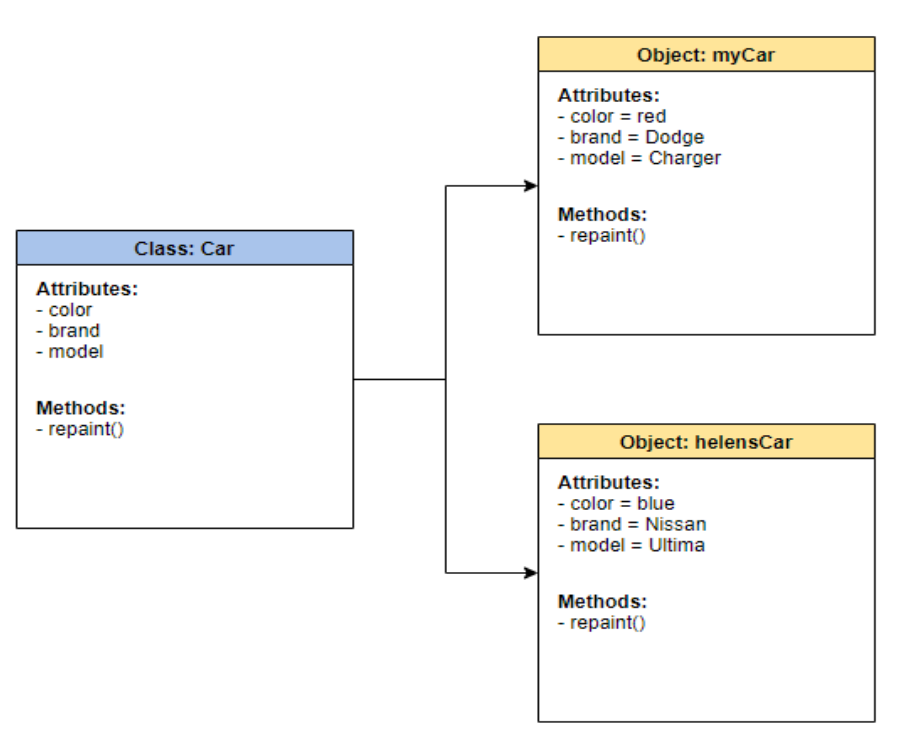
- *JSON* and *Extensible Markup Language* (*XML*) are widely used semi-structured data representations.
- The ability to specify new tags and create nested tags structures **make XML best way to exchange data**. **Tags** make data **self documenting**.
- Database schemas constrain what information can be stored, and the data types of stored values
- XML documents are not required to have an associated schema.
- JSON is a light-weight alternative to XML for data-interchange

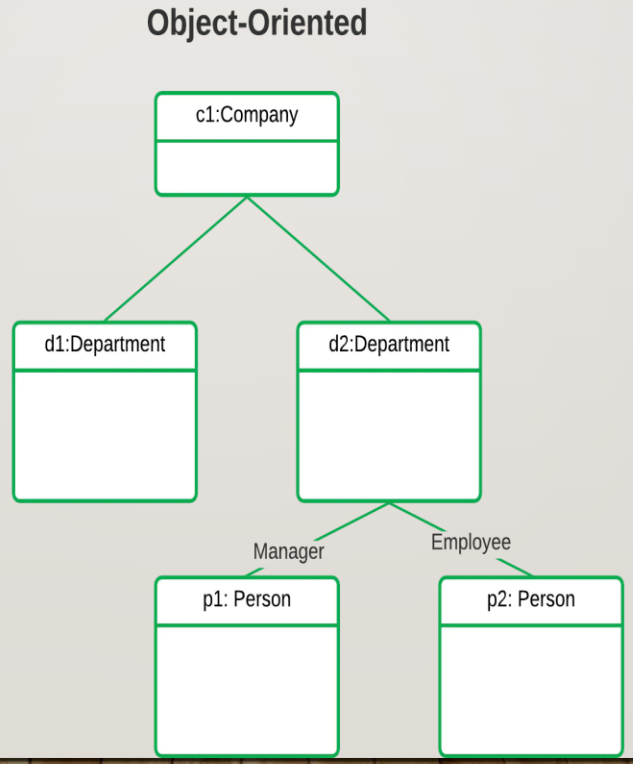
Example of Nested Elements

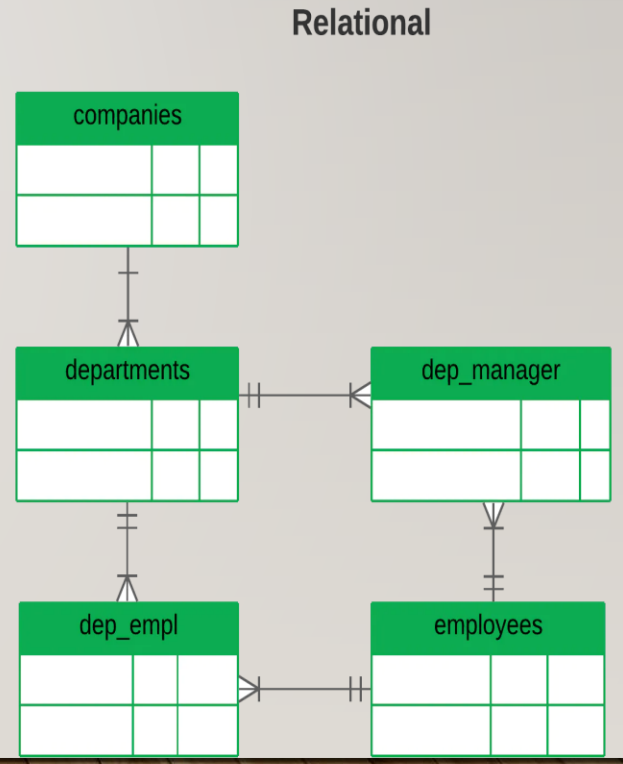
```
<?xml version = "1.0"?>
<bank-1>
   <customer>
      <customer_name> Hayes </customer_name>
      <customer_street> Main </customer_street>
      <customer_city>    Harrison </customer_city>
      <account>
         <account_number> A-102 </account_number>
         <branch_name>     Perryridge </branch_name>
         <balance>           400 </balance>
      </account>
      <account>
         ...
      </account>
   </customer>
   .
   .
</bank-1>
```

OBJECT-BASED DATA MODEL

Standards exist to store objects in relational tables along with **procedures** to be executed in the database. Relational model is extended with idea of encapsulation.







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. **SQL**
- Portion of DML that involves information retrieval is called a query language.

DATA DEFINITION LANGUAGE (DDL)…

- Specification notation for defining the database schema
- 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** –
  - **Authorization** – read, insert, update, delete.

- DDL compiler takes instructions as input and generates some output. These outputs are stored in some a set of special tables in the **data dictionary**
- Data dictionary contains **metadata** (data about data)

SQL Statements

SELECT	Data retrieval
INSERT UPDATE DELETE MERGE	Data manipulation language (DML)
CREATE ALTER DROP RENAME TRUNCATE	Data definition language (DDL)
COMMIT ROLLBACK SAVEPOINT	Transaction control
GRANT REVOKE	Data control language (DCL)

SQL

- **SQL**: widely used non-procedural language
  - Example: Find the name of the instructor with ID 22222

    select *name*
    from *instructor*
    where *instructor.ID* = '22222'

- SQL is as powerful to access database, but there are some computations that are possible using a general-purpose programming language.
- SQL does not support actions such as input from users, output to displays, or communication over the network.
- Such computations and actions must be written in a *host* language, such as **C, C++, or Java**, with embedded SQL queries that access the data in the database.
- Application programs generally access databases through one of
  - Language extensions to allow **embedded SQL**
  - Application program interface (e.g., **ODBC/JDBC**) which allow SQL queries to be sent to a database.
  - DML precompiler convert DML into procedural calls in the Host Language

Example: JAVA & SQL Embeded code

```
// Importing SQL libraries to create database
import java.sql.*;

public class GFG {

    // Step1: Main driver method
    public static void main(String[] args)
    {
        // Step 2: Making connection using

        Connection con = null;
        PreparedStatement p = null;
        ResultSet rs = null;

        con = connection.connectDB();

        // Try block to catch exception/s
        try {

            // SQL command data stored in String datatype
            String sql = "select * from cuslogin";
            p = con.prepareStatement(sql);
            rs = p.executeQuery();
System.out.println("id\t\tname\t\temail");
// Condiion check
  while (rs.next()) {
      int id = rs.getInt("id");
      String name = rs.getString("name");
      String email = rs.getString("email");
      System.out.println(id + "\t\t" + name
                                          + "\t\t" + email);
                }
            }
      // Catch block to handle exception
      catch (SQLException e) {
          // Print exception pop-up on scrreen
              System.out.println(e);
          }
      }
}
```

Example: C# SQL Embeded code

```
namespace AccessingDatabase{
class Program{
 static void Main(string[] args){
  using( SqlConnection conn=new SqlConnection()){
  conn.ConnectionString = "server=ABC; database=Stud_DB; Integrated Security=True";
   try{
         SqlCommand cmd = new SqlCommand();
         cmd.CommandType = CommandType.Text;
         cmd.CommandText = "SELECT RegNo FROM StudRegistration";
         cmd.Connection = conn;
         conn.Open();
     //Execute the query
       SqlDataReader sdr=  cmd.ExecuteReader();
       while(sdr.Read()){
           int id = (int)sdr["id"];
           Console.WriteLine(id);
     }
         conn.Close();   }
  catch (Exception ex){
    Console.WriteLine("Can not open connection !");
} } } } }
```

Example: C++ SQL Embeded code

```
#include<stdio.h>
#include <SQLAPI.h> // main SQLAPI++ header
int main(int argc, char* argv[]){
      SAConnection con; // connection object to connect to database
      SACommandcmd; // create command object
      try{
        con.Connect("test", "tester", "tester", SA_Oracle_Client);
        cmd.setConnection(&con);
        cmd.setCommandText("create table tbl(id number, name varchar(20));");
        cmd.Execute();
        cmd.setCommandText("Insert into tbl(id, name) values (1,”Vinay”)");
        cmd.setCommandText("Insert into tbl(id, name) values (2,”Kushal”)");
        cmd.Execute();
              con.Commit();
        printf("Table created, row inserted!\n");
      }
```

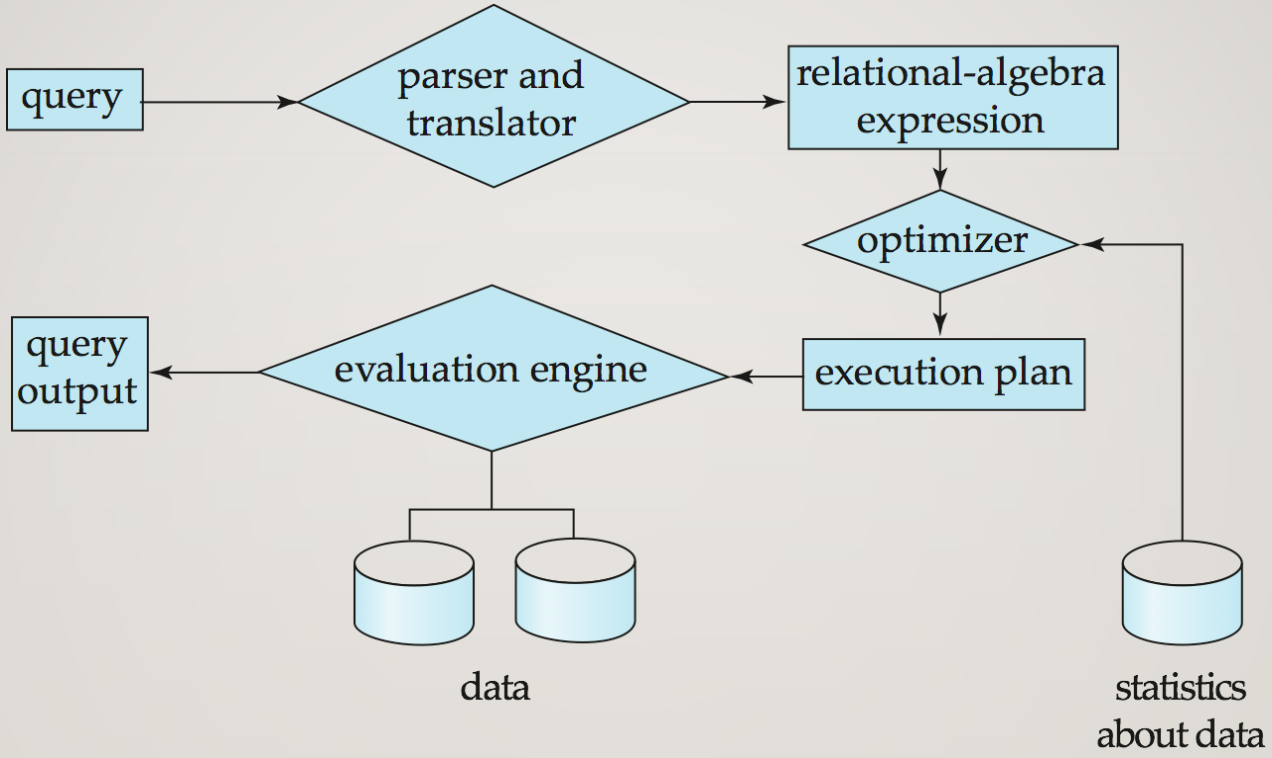
```
catch(SAException &x){
try{
      con.Rollback();
 }
  catch(SAException &) { }
  printf("%s\n", (const char*)x.ErrText());
}
  return 0;
}
```

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.

QUERY PROCESSING

1. Parsing and translation
2. Optimization
3. Evaluation



Translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level.

Ex: select salary
from instructor
where salary < 75000;

Two possible relational Algebraic expression.

- $\sigma_{salary<75000}(\Pi_{salary}(instructor))$
- $\Pi_{salary}(\sigma_{salary<75000}(instructor))$

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.

figure

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 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.

figure

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.

- **Naive users**: (unsophisticated user) invoke one of the permanent application programs that have been written previously.
- **Application programmers:** are computer professionals who write application programs.
- **Sophisticated users**: form requests in a database query language or data analysis software
- **Specialized users**: write specialized database applications that do not fit into the traditional data processing framework.
  - Ex: Expert system/Knowledge database(that stores complex data types-graphics/audio data etc.)

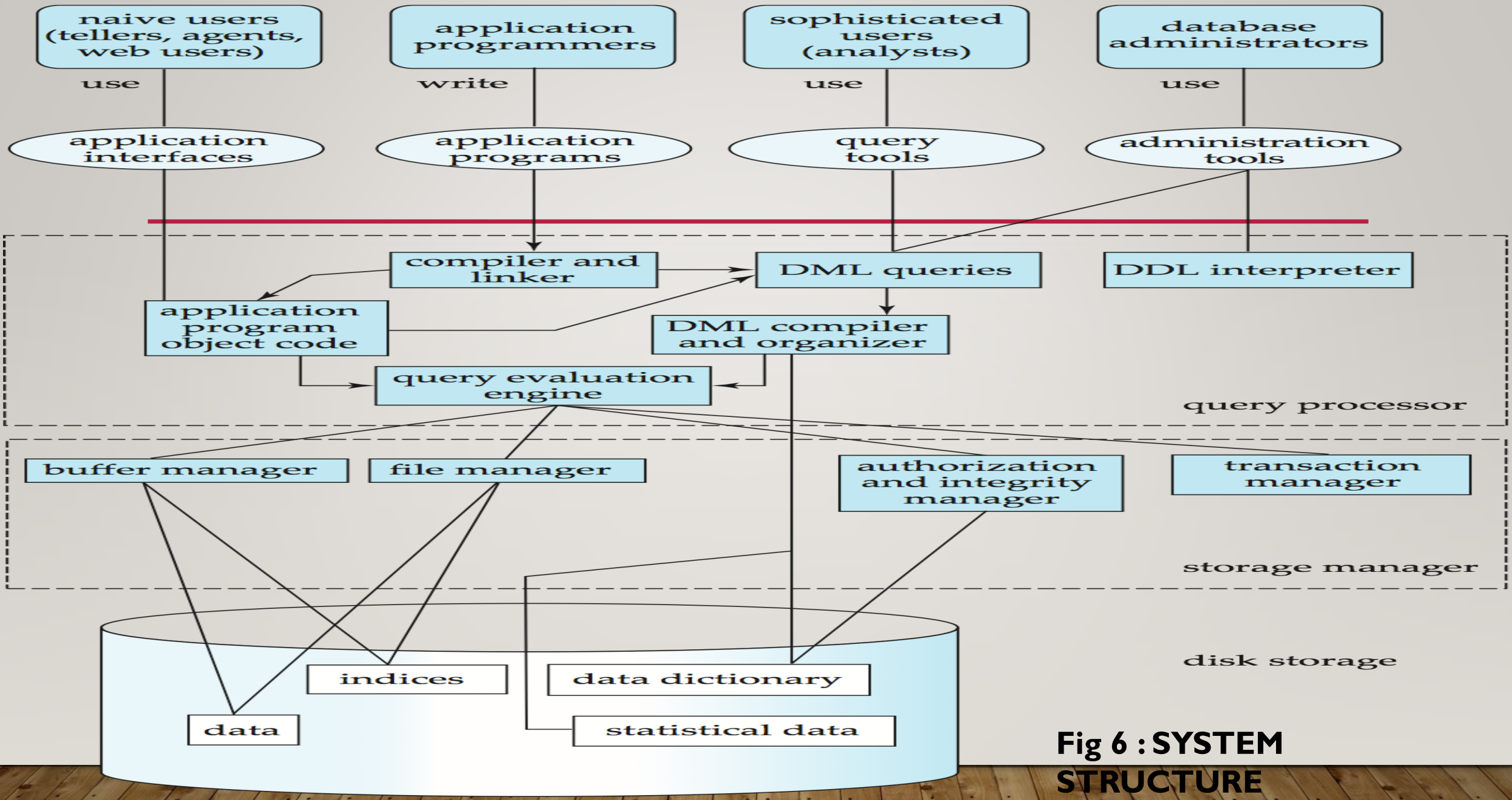
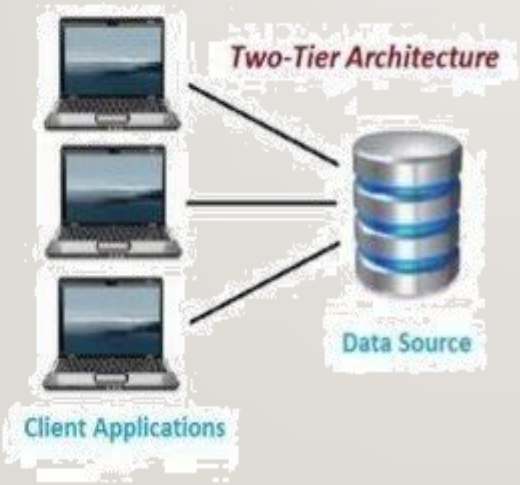


Fig 6 : SYSTEM STRUCTURE

DATABASE APPLICATION ARCHITECTURE

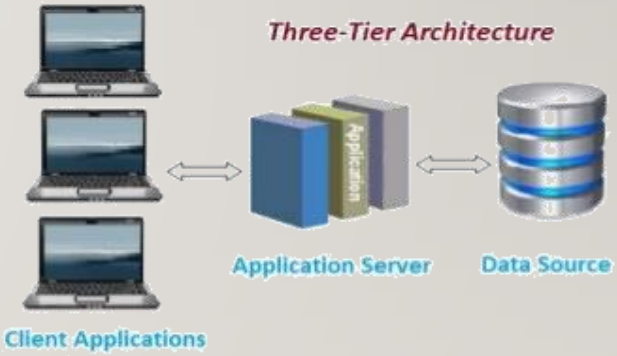
2-TIER ARCHITECTURE

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



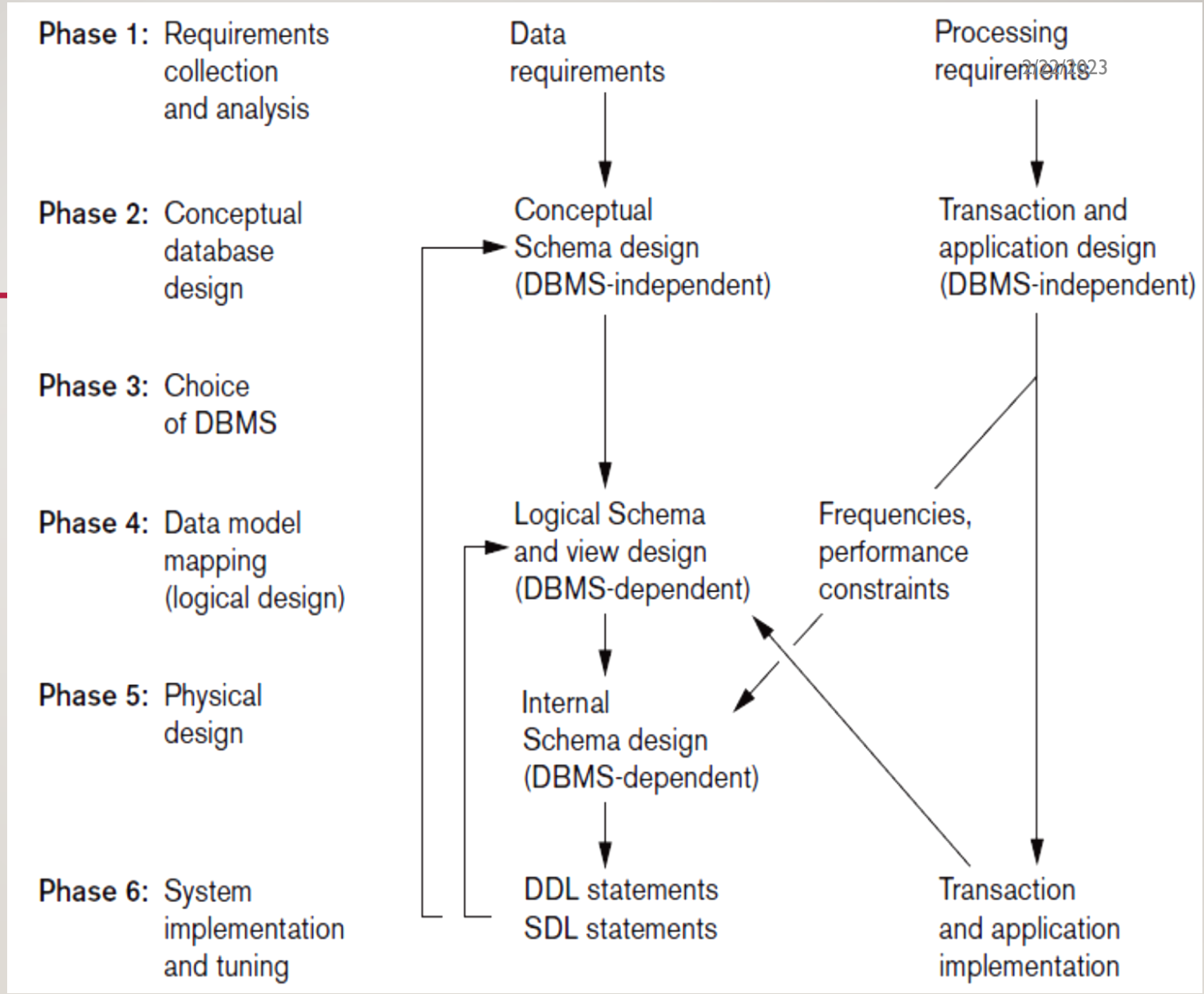
3-TIER ARCHITECTURE

- Web based application
- Three layers:
  1) Client layer
  2) Business layer
  3) Data layer



DATABASE DESIGN

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



DATABASE DESIGN

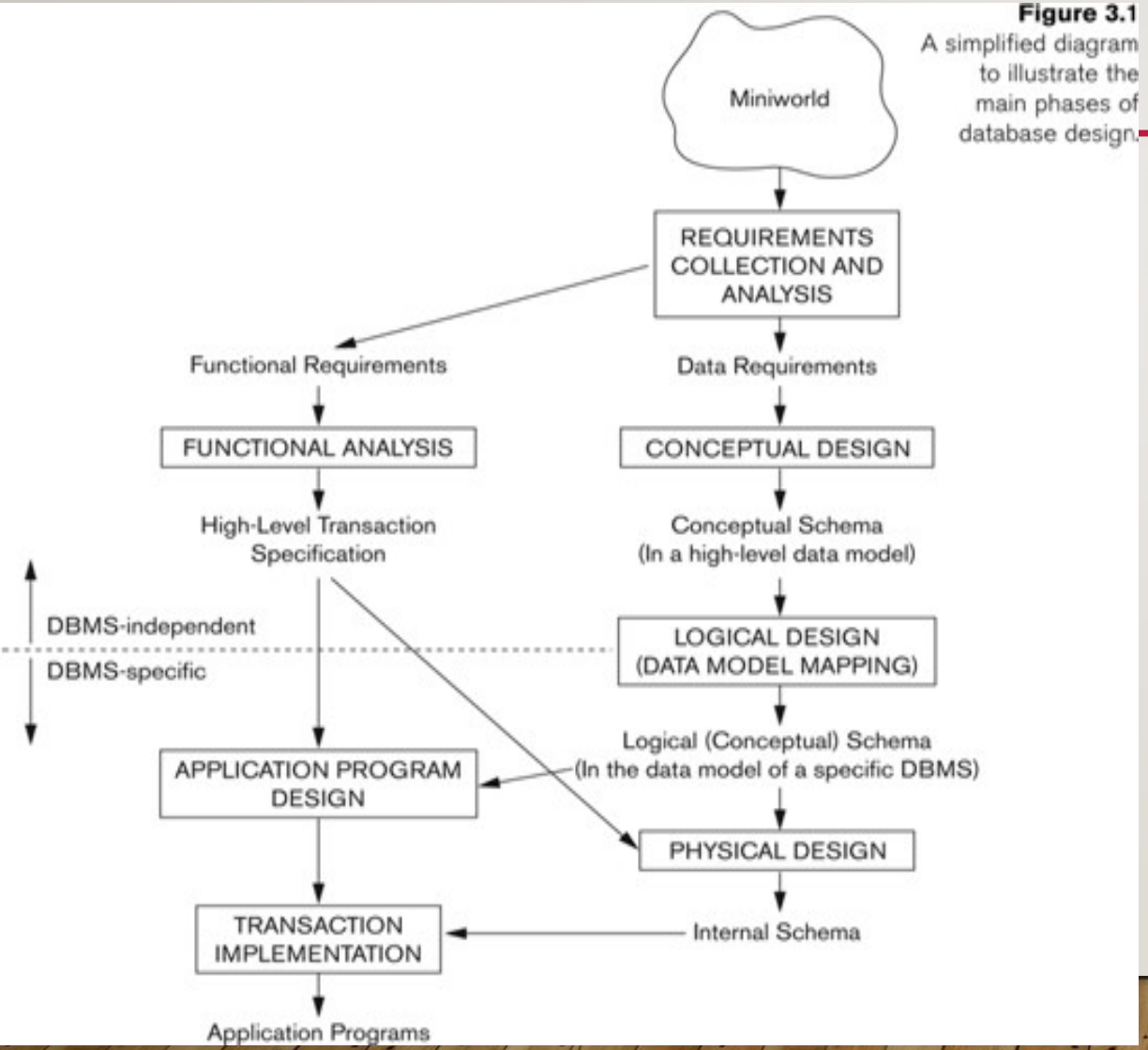


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

User Requirement Specifications

Translate into **Conceptual Schema** of the Database.
ER-Model- entity, attributes, relationship

More details about logical structure of database such as tables, columns, primary key, foreign key constraints etc.
Database Requirement Specification

Choose a **DBMS** and implement database along with physical features such as **file organization, access methods** in

END OF THE CHAPTER