Database Management System

MCA 4151

4 Credits

Reference:

Database System Concepts , 6th Edition

Authors:

Abraham Silberschatz

Henry F. Korth

S. Sudarshan

Duration

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

Other References -

Fundamentals of Database System, 6th Edition

Ramez Elmasri, Shamkant Navathe,

Addison Wesley Publications Co., 2010.

Database Management System 3rd Eddition

Raghu Ramakrishnan, Johannes Gehrke,, 3rd Edition,

WCB/McGraw Hill Publisher, 2014.

Evaluation Pattern

- Internal Assessment(50%)
  - Two Sessional (15 marks each) - 30
  - Four Assignments (5+5+5+5) - 20

  50
- End Semester Assessment(50%)
  - Written Exam of 3 hours duration - 50

TOTAL 100

Assignments Schedule

Table 1: Schedule of In-Semester Assessment

Component	Type	Max. Marks	Schedule
MISAC 1	In-Semester Exam 1	15	September 2 - 8, 2022
MISAC 2	Quiz	5	September 19 - 24, 2022
MISAC 3	Surprise Assignment	5	September 26 – October 1, 2022
FISAC 1	From Table 3	5	October 3 – 8, 2022
MISAC 4	In-Semester Exam 2	15	October 17 – 22, 2022
FISAC 2	From Table 3	5	October 31 – November 05, 2022

Syllabus

- **I SEMSTER-MCA**
- **MCA 4151 DATABASE MANAGEMENT SYSTEM [4 0 0 4]**

Main Concepts Discussed

- Database concepts, data models and database architecture.
- Manipulate, retrieve the data from database using SQL, PL/SQL.
- Database Design – ER Model & Normalization.
- Database Query execution, Transaction Management , Concurrency & Recovery.
- Unstructured Database.

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 & software system that facilitates the process of defining, constructing, manipulating and sharing databases among users and applications.
  - Example: Oracle

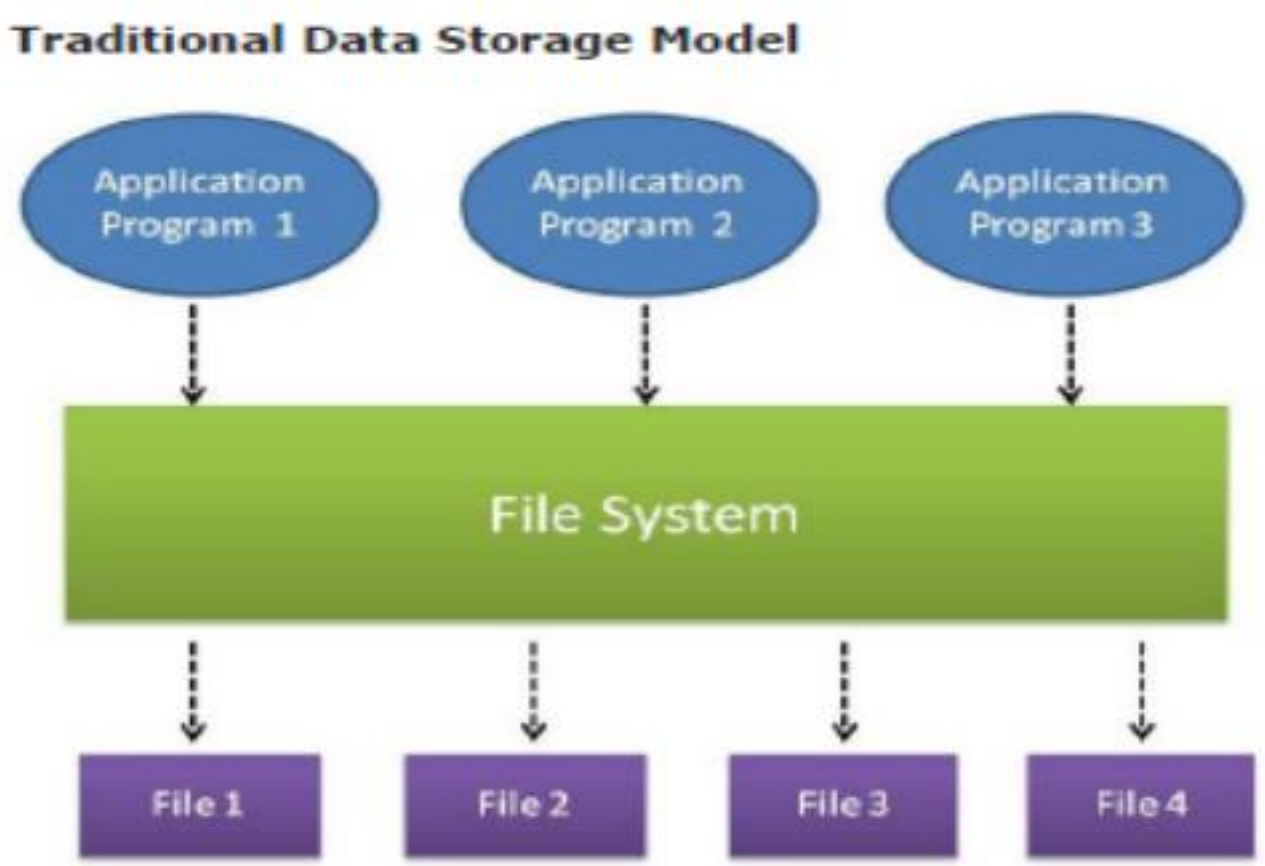
Database Applications-

Name some 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

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 access is 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.

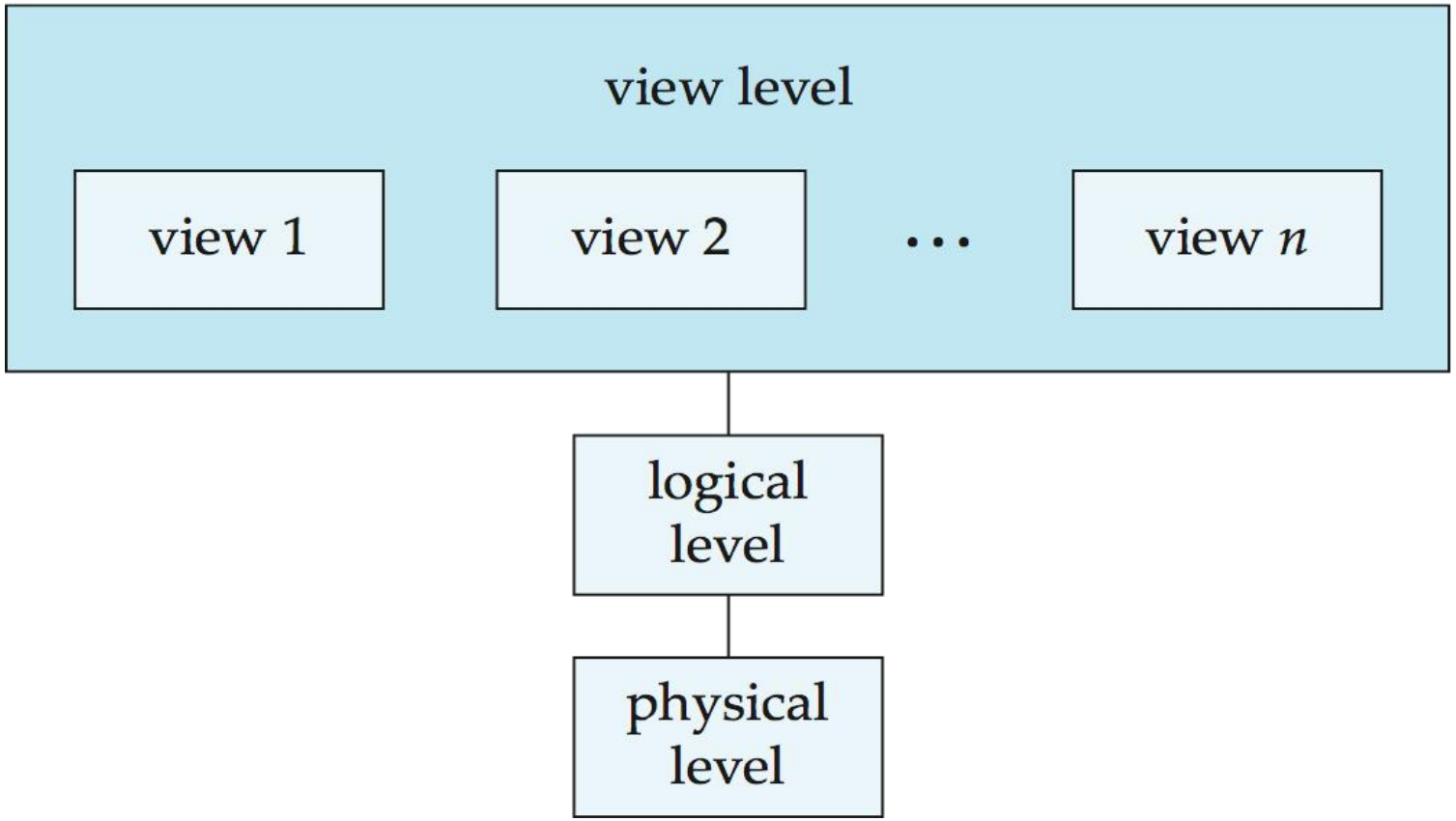


Fig 1 Three levels of data abstraction

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 what 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.

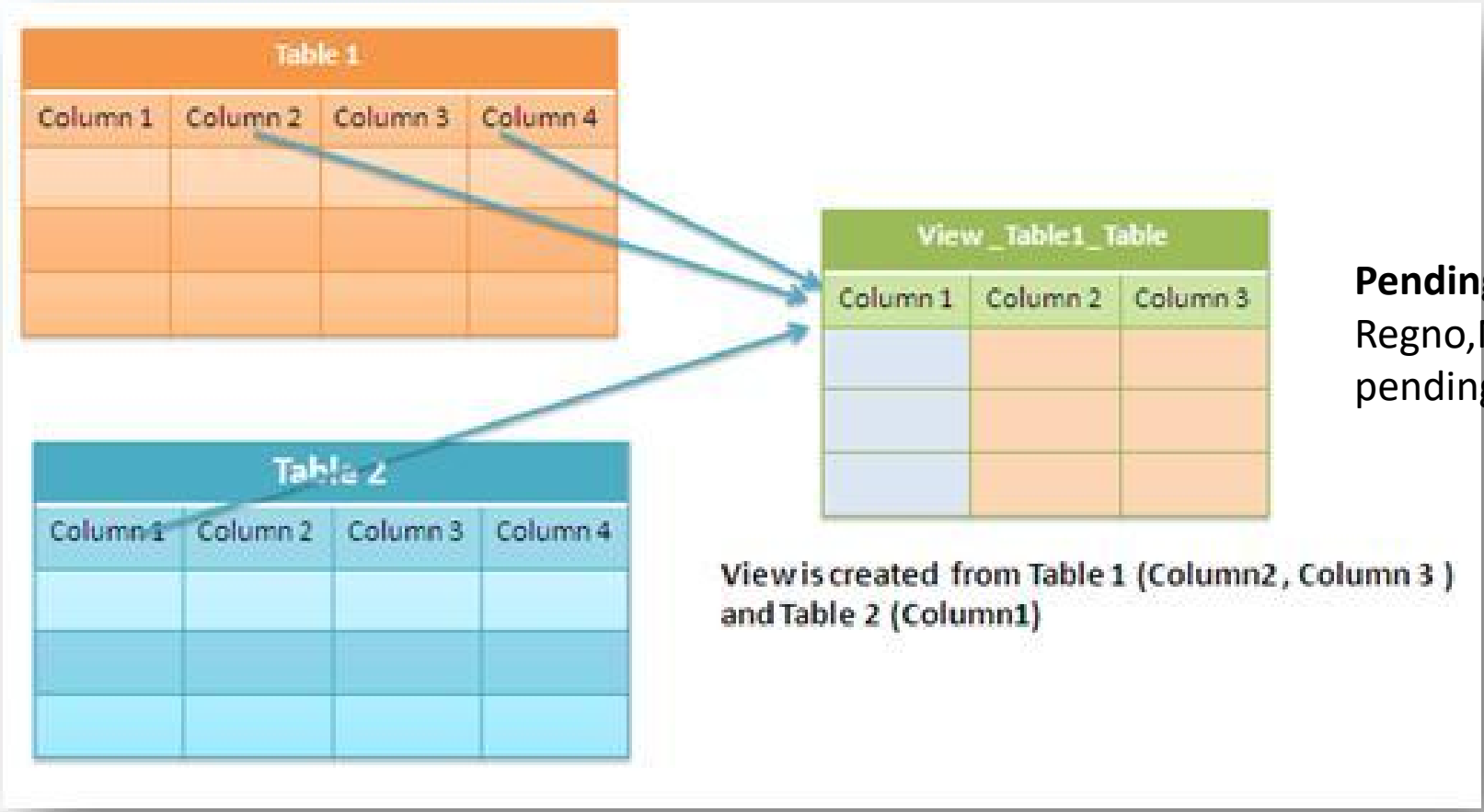
  For example Application programs hide details of data types.

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

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

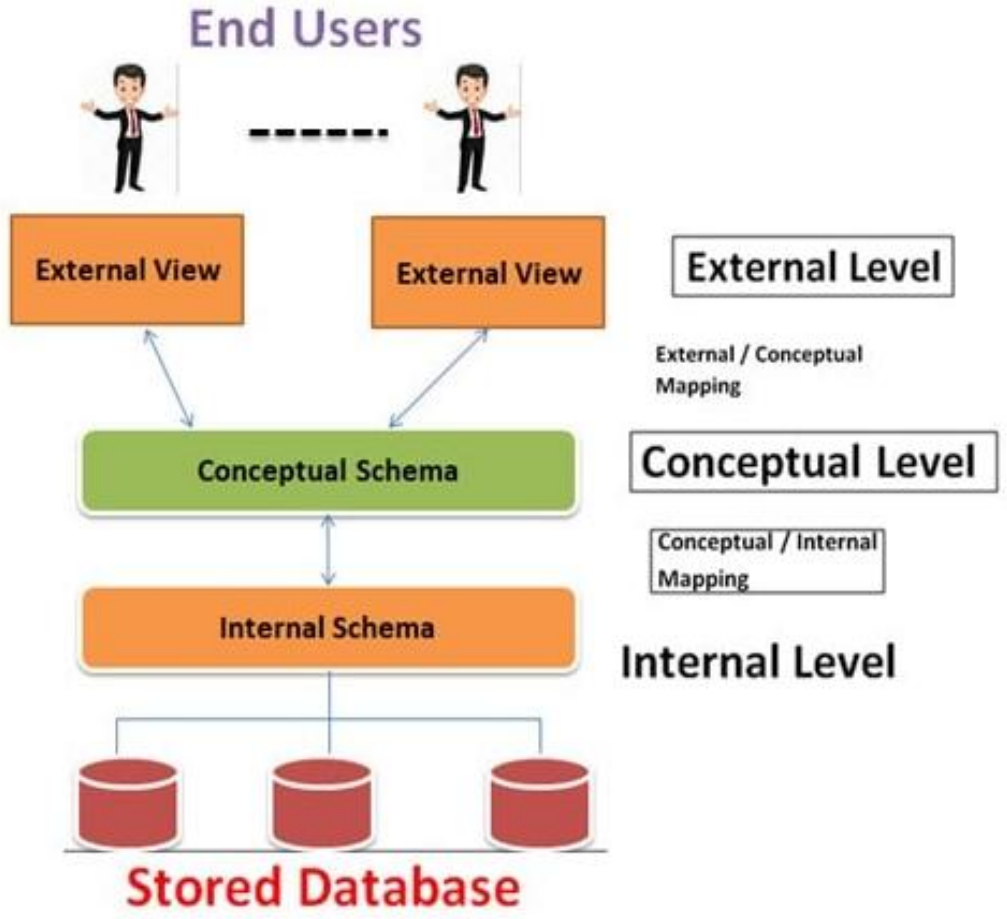
Instances and Schemas

- **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. The user of the logical level does not need to be aware of this complexity.
- **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

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

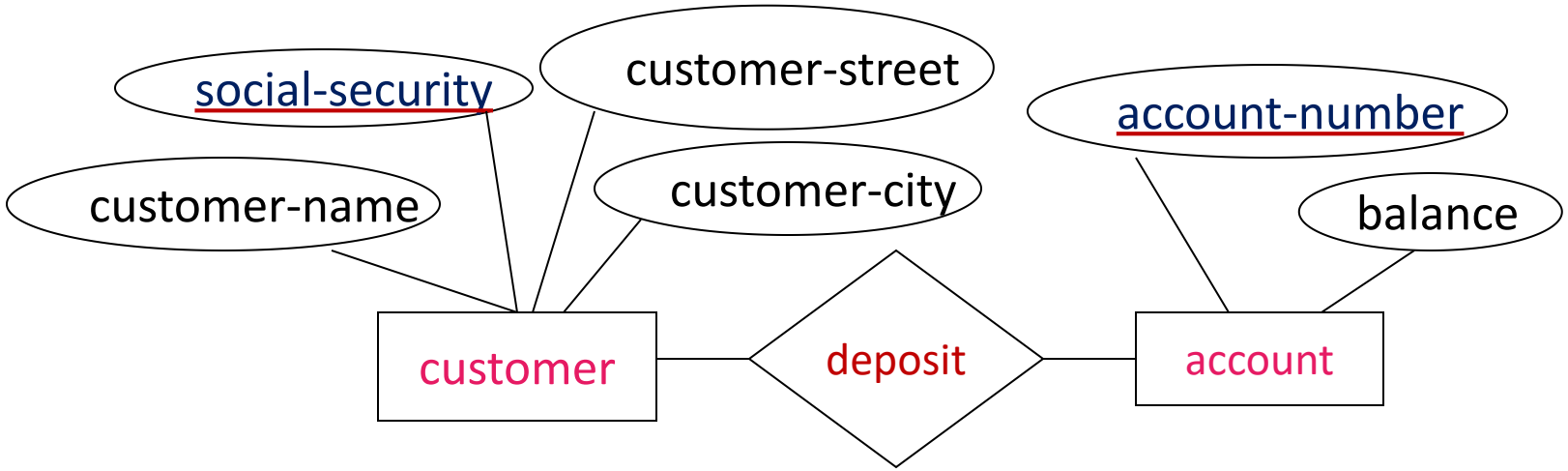


Fig 3 ER model (old notation)

widely used in database design

Entity Relationship Model

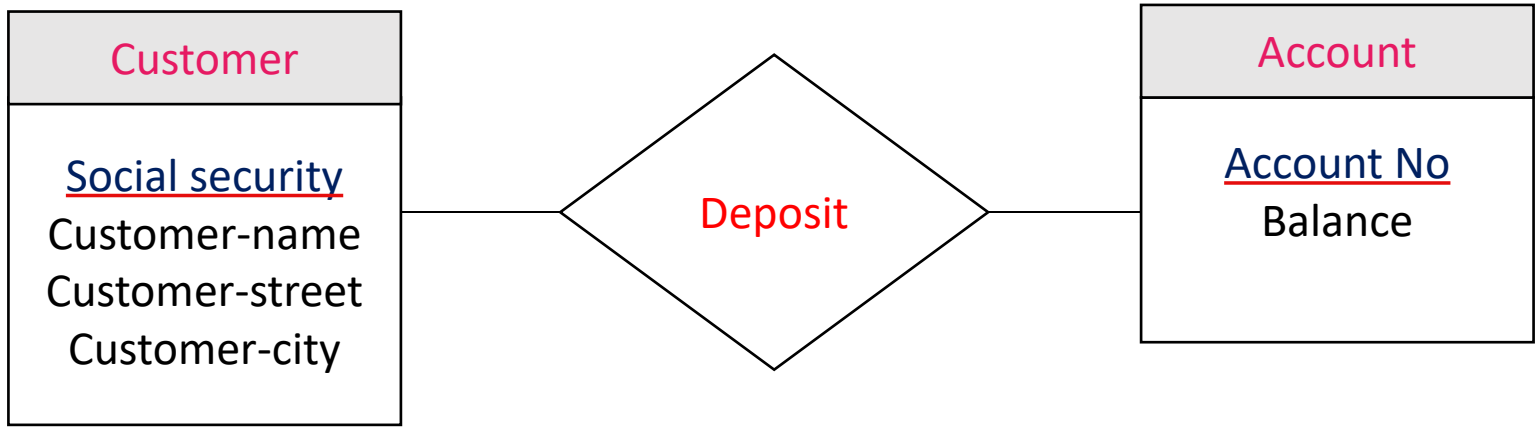


Fig 4 ER model(New notation)

Normalization

Another method for designing a relational database.

To design set of schema with minimum redundancy.

Ensuring schema are in appropriate normal

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 Started gaining importance.

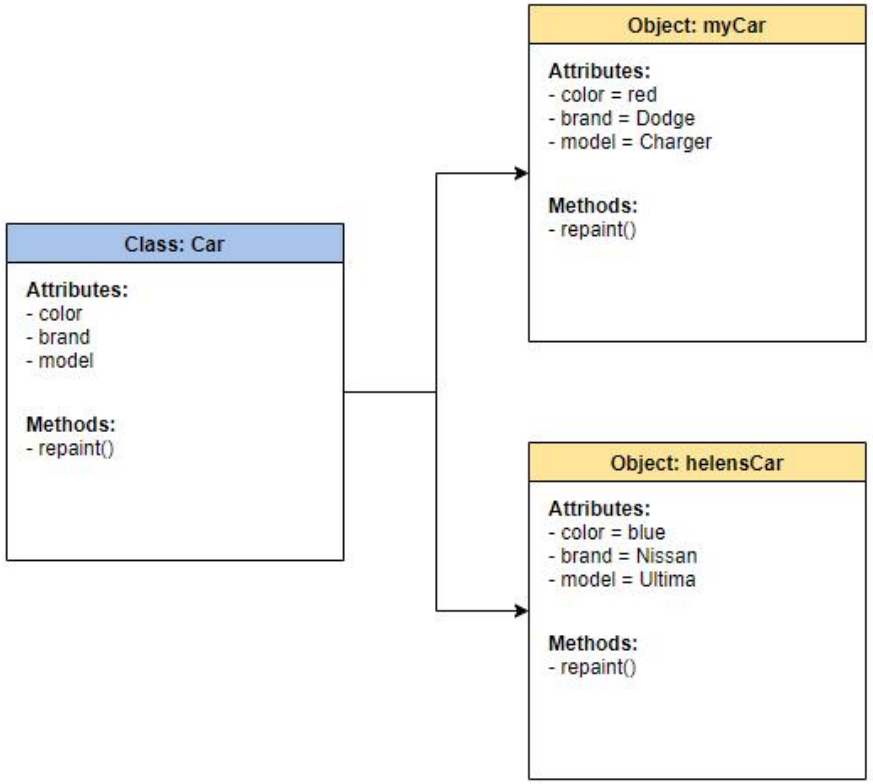
Example of Nested Elements

```
<?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.



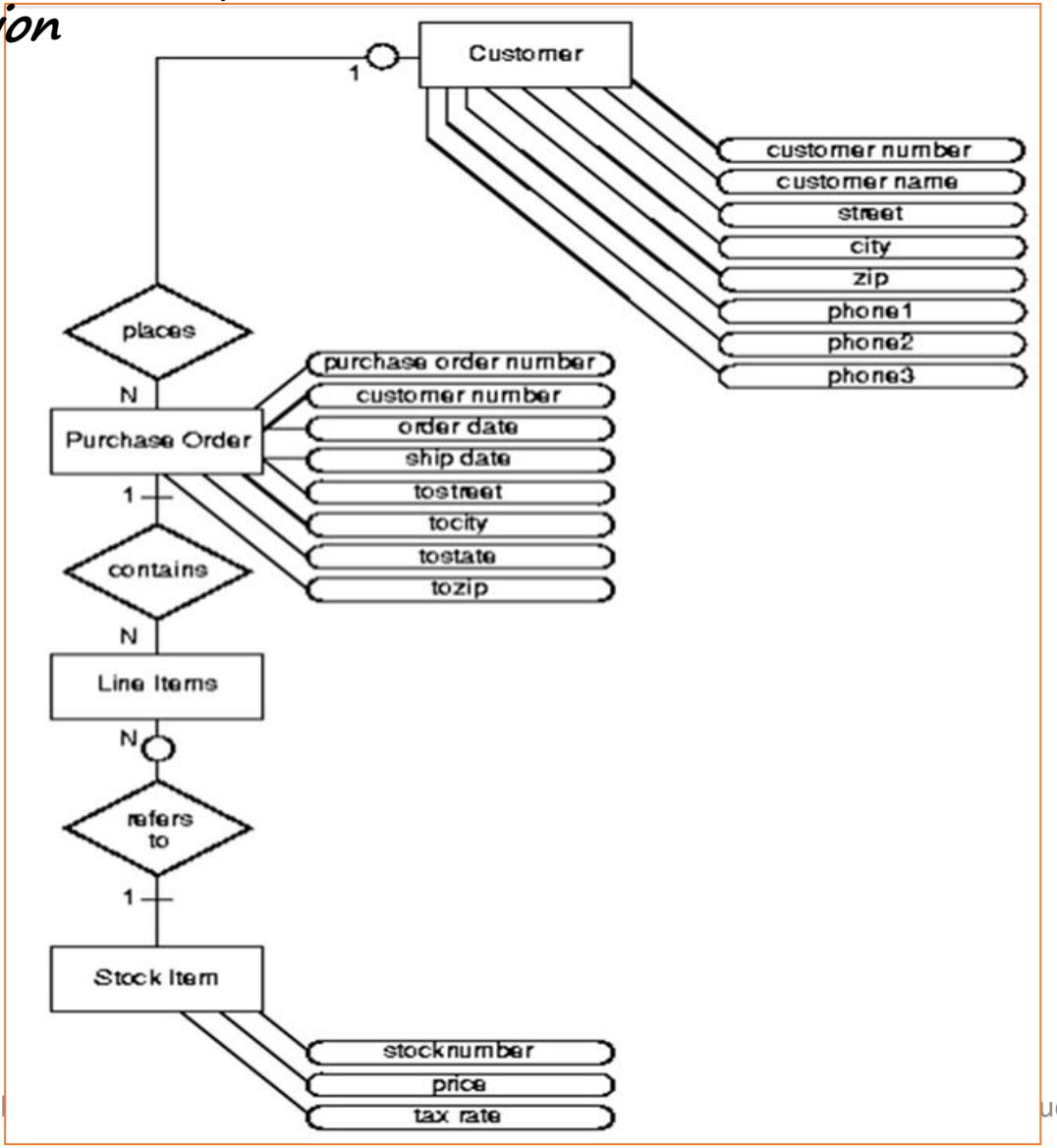

```
CREATE TYPE Customer_objtyp AS OBJECT (
  CustNo             NUMBER,
  CustName           VARCHAR2(200),
  Address_obj        Address_objtyp,
  PhoneList_var      PhoneList_vartyp,

  ORDER MEMBER FUNCTION
    compareCustOrders(x IN Customer_objtyp)
RETURN INTEGER
  )
/
```

Historically, the **network data model** and the **hierarchical data model** preceded the relational data model.

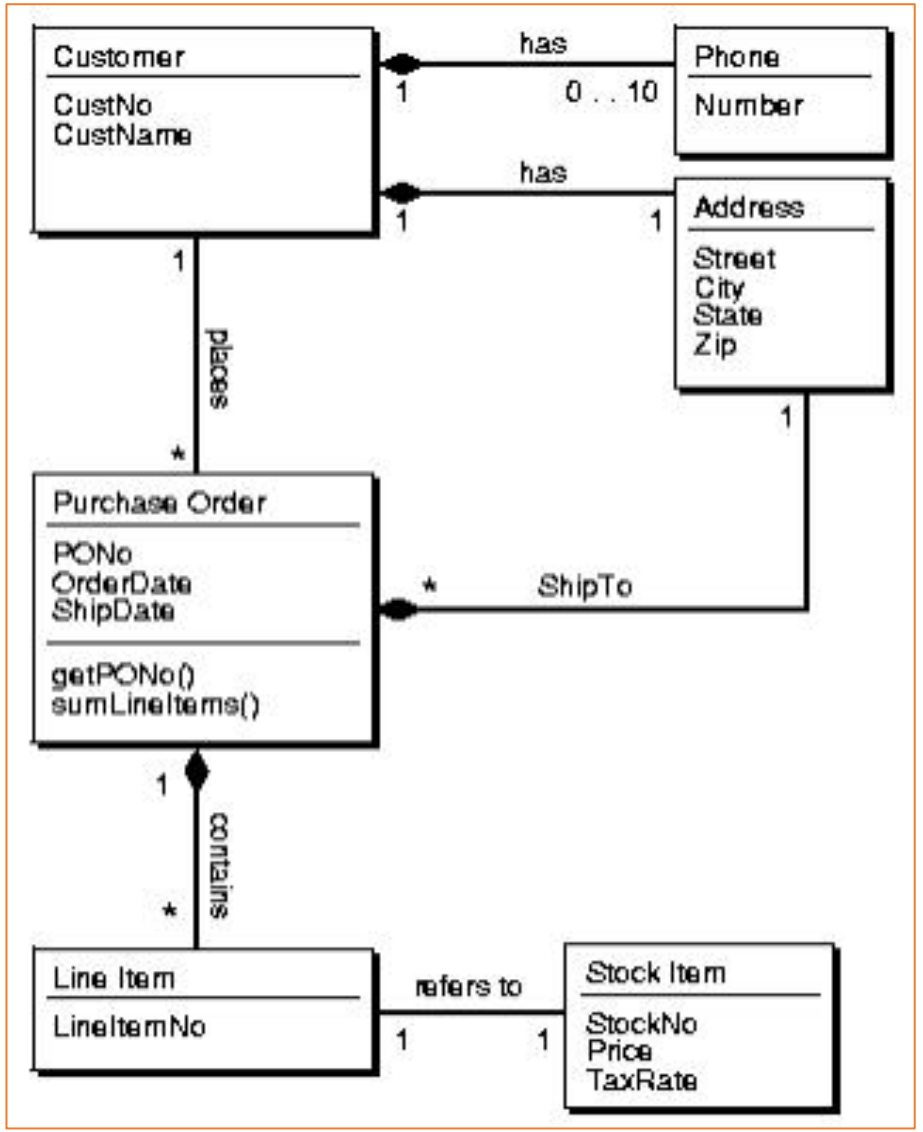
Relational Data Model

Entity-Relationship Diagram for Purchase Order Application



Object-Relational Model

Class Diagram for Purchase Order Application



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)**,**

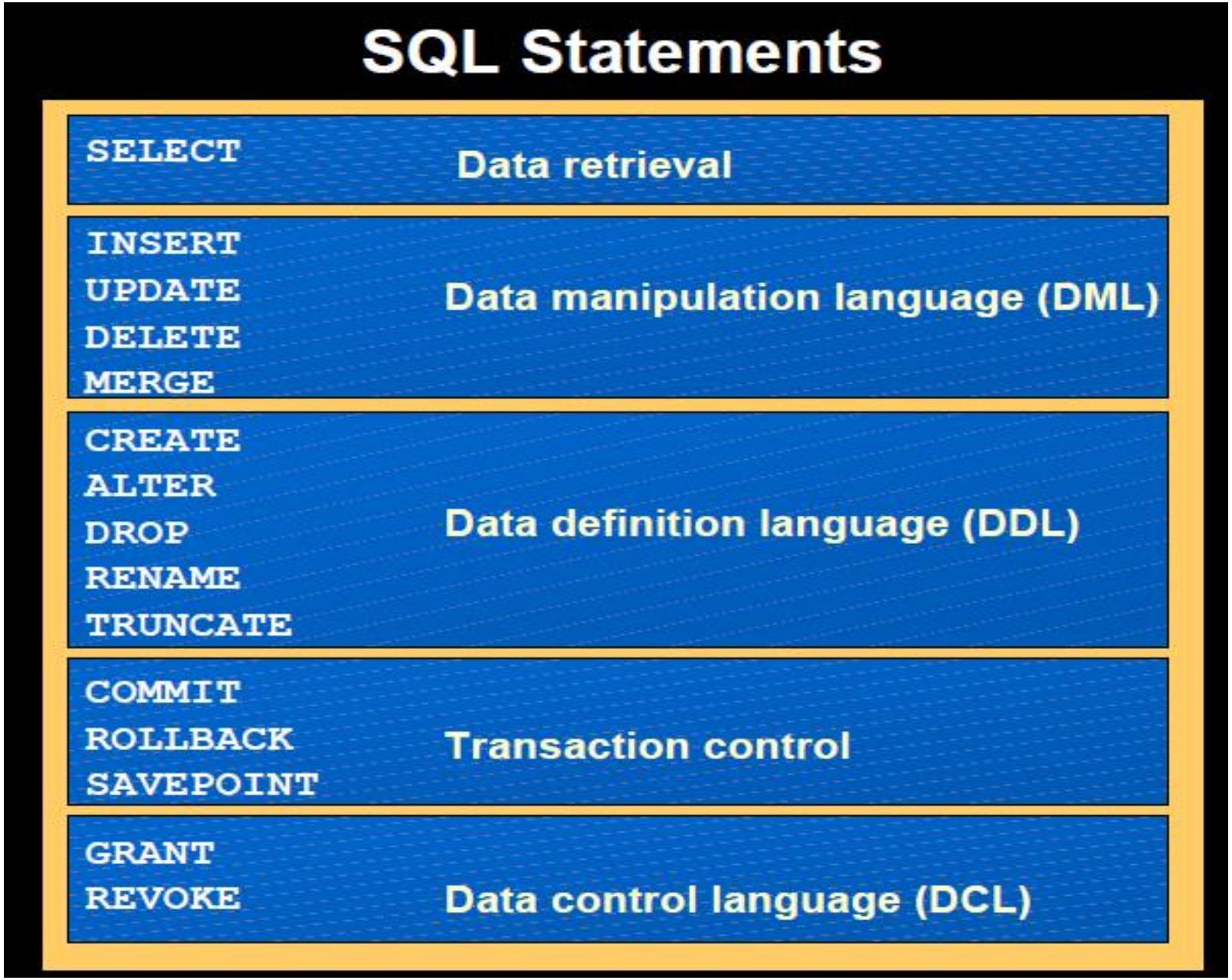
dept_name **varchar**(20),

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 also 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);
                        }
                }
}
```

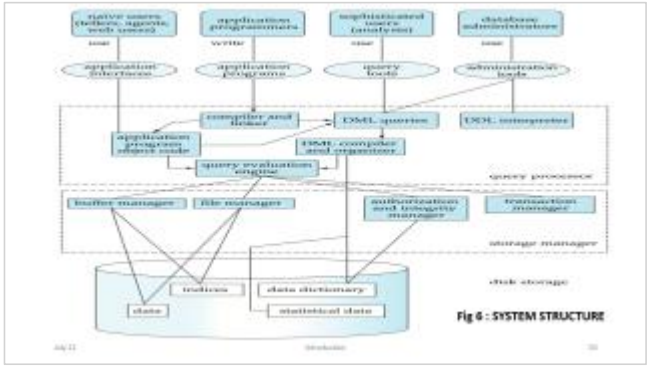
Example:
C# SQL Embeded code

```
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;
        }
```

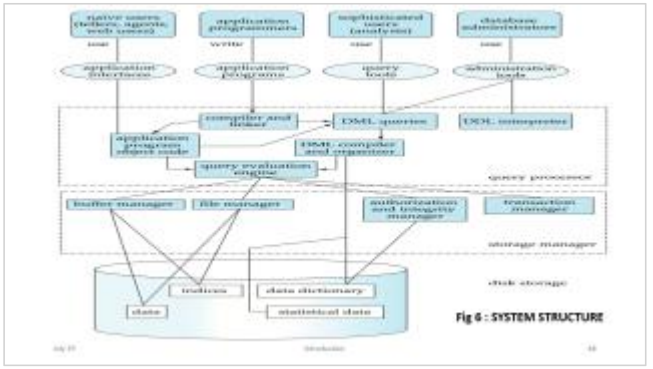
SYSTEM STRUCTURE of DBMS



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 query processing and execution to retrieve the required data efficiently.

Storage Manager



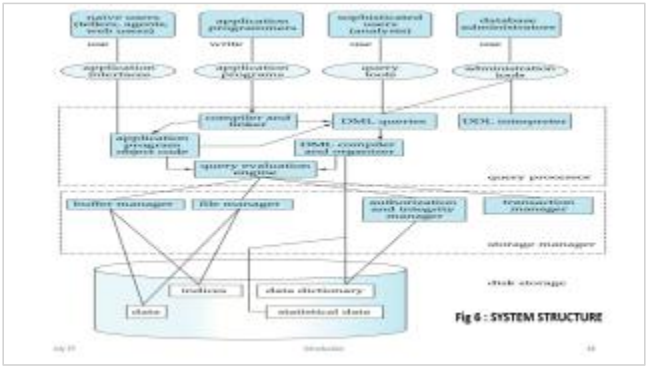
- A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

- The storage manager is responsible for the following tasks:

  - Interaction with the file manager.
  - translates the various DML statements into low-level file-system commands.
  - 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.

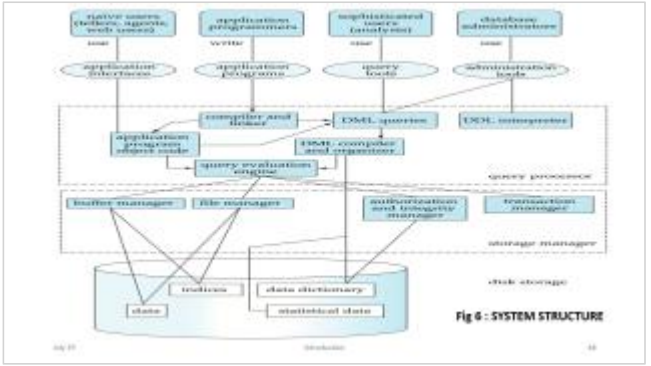
➢Transaction manager – ensures that database remains in a consistent state despite system failures and that concurrent transaction execution proceeds without conflicting.

➢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.

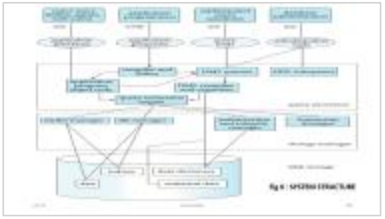
figure

Storage Manager



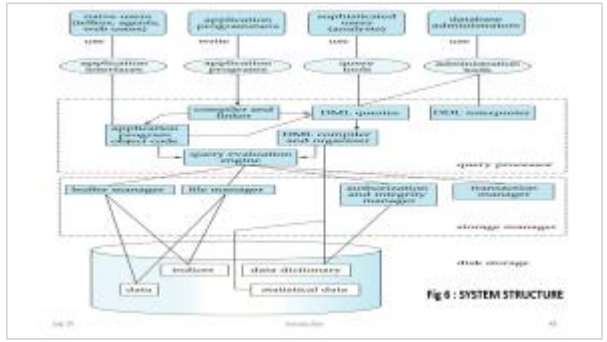
- Data structures used are:
  - Data files – store the database
  - Data dictionary – stores metadata
  - Indices – provide fast access to data items

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. ACID-properties.
- **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



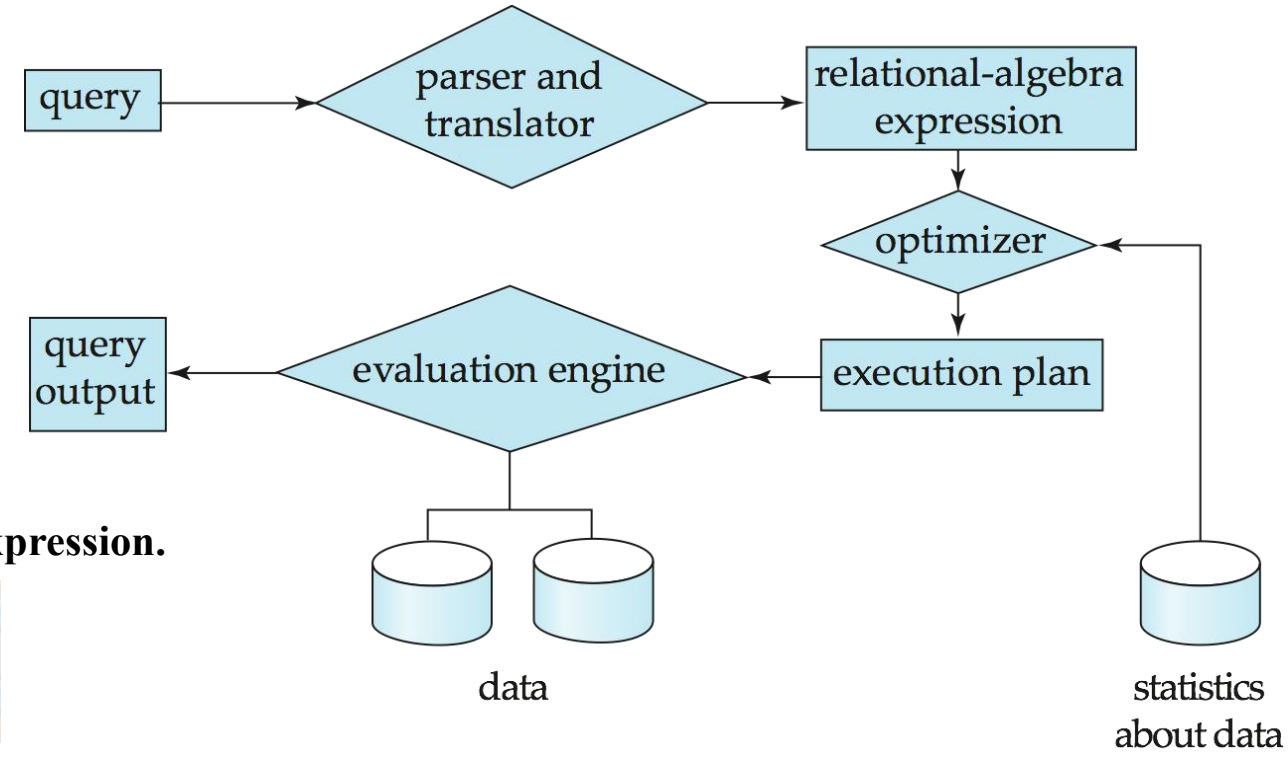
Components:

- **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

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.

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

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**: Sophisticated users who 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.)

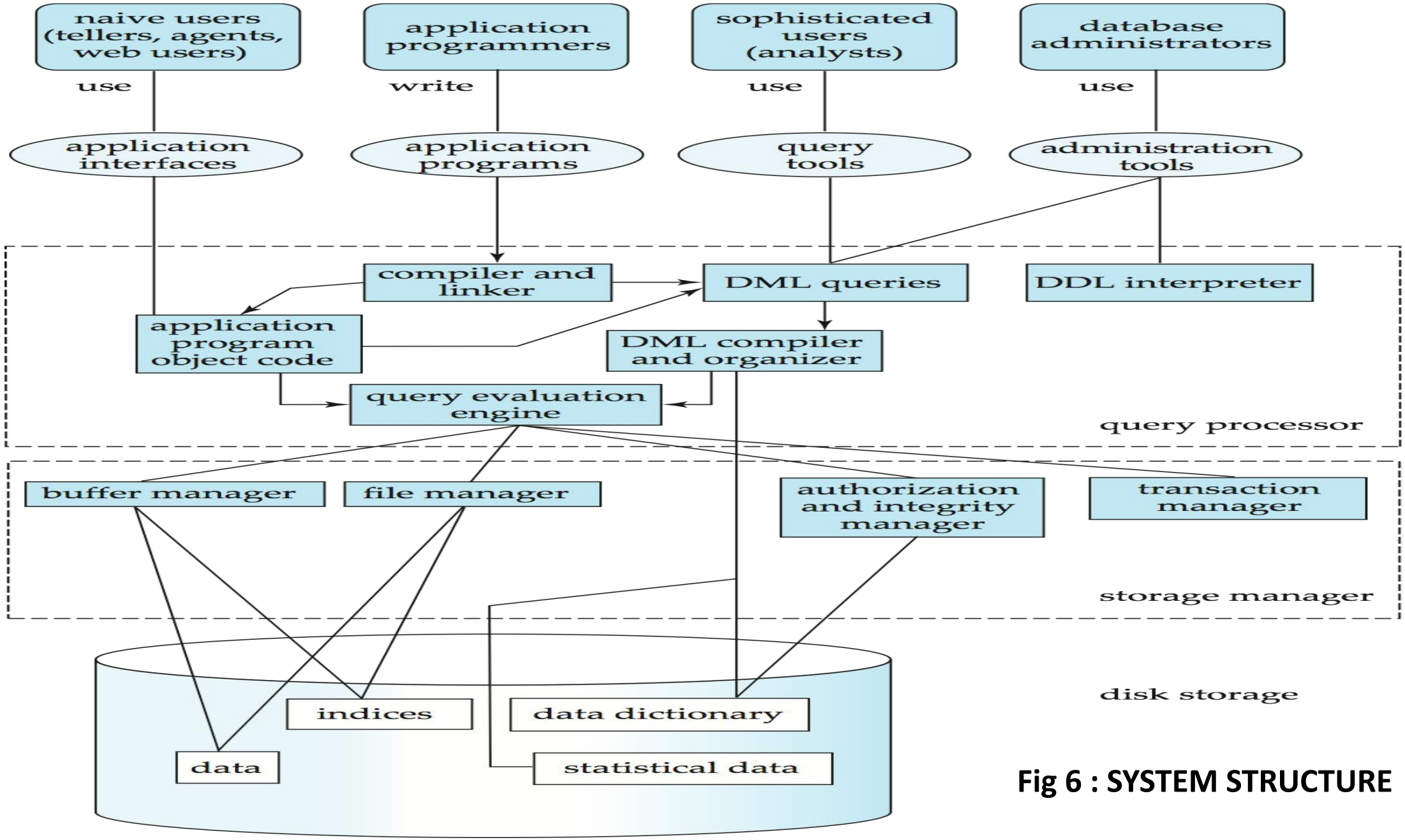
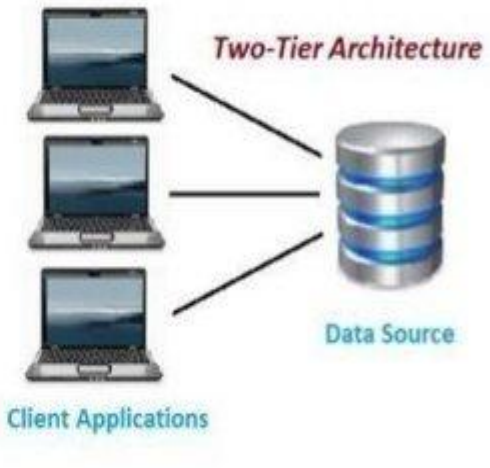


Fig 6 : SYSTEM STRUCTURE

Database Application Architecture

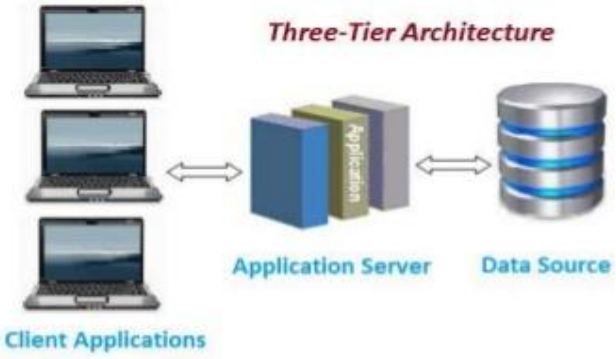
2-TIER ARCHITECTURE

- 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

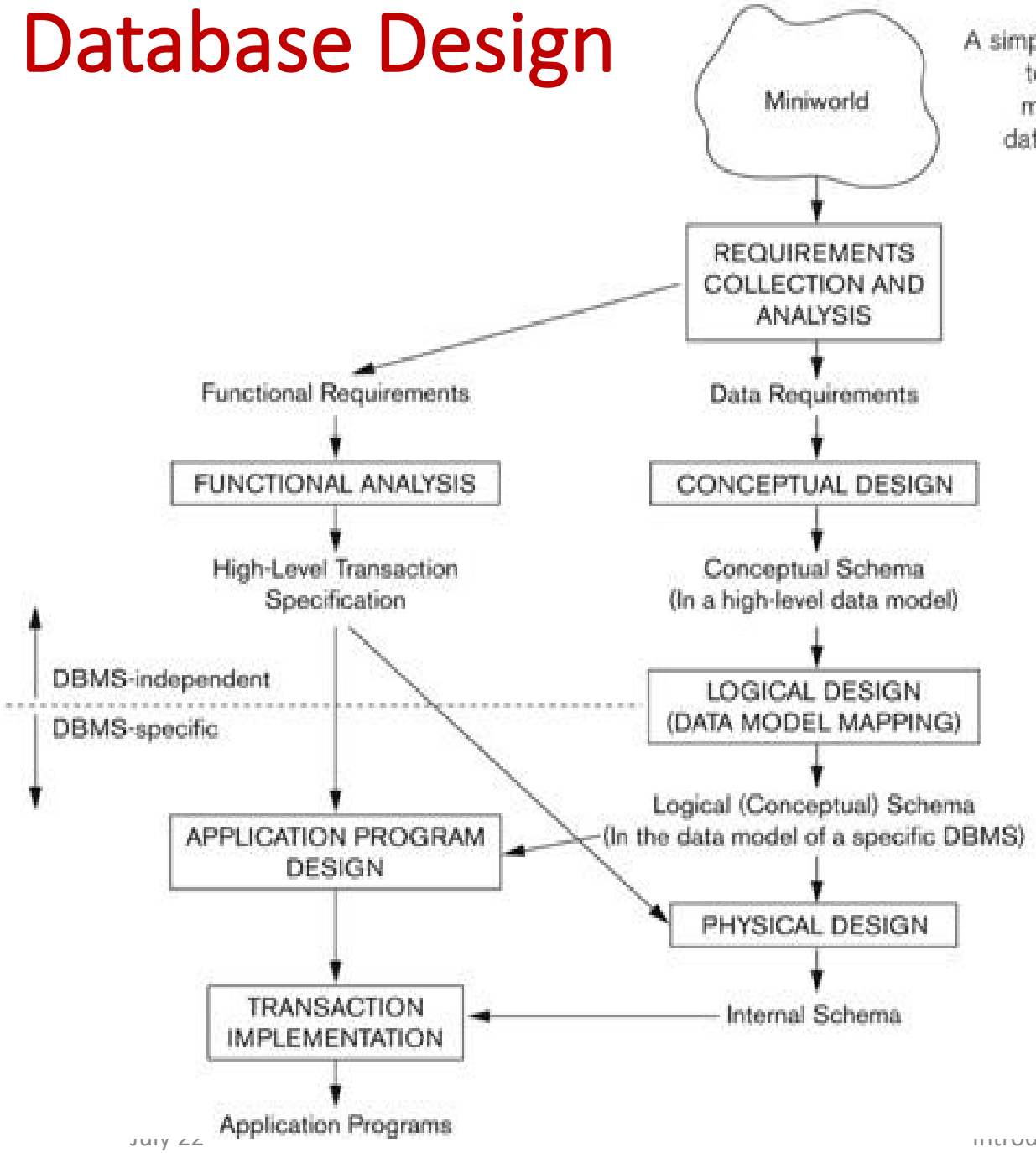


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

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** index.

END of the CHAPTER