Pakistan Institute
of Public Finance Accountants

Model Solutions

Database Management System (Application)

AGP | PG | PUBLIC Sectors

Winter Exam-2025

MODEL SOLUTIONS – DISCLAIMER

INTRODUCTION

The Model Solutions are provided to students for clear understanding of relevant subject and it helps them to prepare for their examinations in organized way.

These Model Solutions are prepared only for the guidance of students that how they should attempt the questions. The solutions are not meant for assessment criteria in the same pattern mentioned in the Model Solution. The purpose of Model Solution is only to guide the students in their future studies for appearing in examination.

The students should use these Model Solutions as a study aid. These have been prepared by the professionals on the basis of the International Standards and laws applicable at the relevant time. These solutions will not be updated with changes in laws or Standards, subsequently. The laws, standards and syllabus of the relevant time would be applicable. PIPFA is not supposed to respond to individual queries from students or any other person regarding the Model Solutions.

DISCLAIMER

The Model Solutions have been developed by the professionals, based on standards, laws, rules, regulations, theories and practices as applicable on the date of that particular examination. No subsequent change will be applicable on the past papers solutions.

Further, PIPFA is not liable in any way for an answer being solved in some other way or otherwise of the Model Solution nor would it carry out any correspondence in this regard.

PIPFA does not take responsibility for any deviation of views, opinion or solution suggested by any other person or faculty or stake holders. PIPFA assumes no responsibility for the errors or omissions in the suggested answers. Errors or omissions, if noticed, should be brought to the notice of the Executive Director for information.

Winter Exam-2025
Solutions – Database Management System (Application)

Q.1. E-R Model (Entity-Relationship Model)

Definition
The **E-R (Entity-Relationship) Model** is a **high-level conceptual data model** used to design the structure of a database. It represents real-world entities and the relationships between them.
The **E-R Model** is best suited for designing relational databases by visually modeling entities and relationships.
Components of E-R Model

- **Entity:** A real-world object (e.g., Student, Course).
- **Entity Set:** A group of similar entities (e.g., all students).
- **Attributes:** Properties of entities (e.g., Student has Name, Age).
- **Relationship:** Association between entities (e.g., Student *enrolls in* Course).
- **ER Diagram:** A visual representation using rectangles (entities), ovals (attributes), and diamonds (relationships).

Example
Designing a student-course enrollment system:

- **Entities:**
  - Student(StudentID, Name, Age)
  - Course(CourseID, Title)
- **Relationship:**
  - Enrolls connects Student and Course.

ER Diagram Representation
[Student] ----<Enrolls>---- [Course]
| |
[StudentID] [CourseID]
[Name] [Title]
[Age]

Use Case
Used during the **database design phase** to map out how data is structured and related before implementation.

Object-Oriented Model of DBMS

Definition
The **Object-Oriented Data Model** integrates **object-oriented programming concepts** into database design. It represents data as **objects**, just like in programming languages such as Java or C++.
The **Object-Oriented Model** provides a more natural way to represent complex data by combining structure and behavior, making it suitable for advanced applications.
Concept of Object-Oriented Model

- **Object:** Encapsulates both **data** (attributes) and **behavior** (methods).
- **Class:** A blueprint for creating objects.
- **Inheritance:** Allows a class to inherit properties from another class.
- **Encapsulation & Reusability:** Promotes modular, reusable database components.

Example
A Person class with a subclass Student.
Class: Person
- Attributes: Name, Age
Class: Student extends Person
- Attributes: StudentID, CourseList
- Method: enrollInCourse()

Object Representation:
```
{
 "Student": {
   "Name": "Alice",
   "Age": 20,
   "StudentID": "S101",
   "CourseList": ["Math", "Physics"]}}
```

Q.2. **I**n a **Database Management System (DBMS)**, the concepts of **entities** and **entity sets** are fundamental to understanding how data is modeled and organized, especially in the **Entity-Relationship (ER) model**.
entities are individual objects with a distinct identity and properties, while **entity sets** are collections of similar entities. Understanding these concepts is crucial for designing databases using the ER model, which forms the basis for relational database schema design.

Entity
An **entity** is a real-world object or thing that can be distinctly identified. Entities represent objects that exist physically (like a person or a car) or conceptually (like a course or a department). Each entity has **attributes**, which are the properties that describe it.

Examples
- A **student** in a university system.
- An **employee** in a company.
- A **book** in a library.

Characteristics of Entities
- **Uniqueness**: Each entity is distinguishable from other entities.
- **Representation**: In an ER diagram, entities are represented by **rectangles**.
- **Identification**: Entities are identified uniquely by **primary keys** (a set of attributes).

Entity Set
An **entity set** is a **collection of similar types of entities**. It refers to all entities of a particular type stored in the database.

Example
- All students in a university form a **Student** entity set.
- All employees in a company form an **Employee** entity set.

Types of Entity Sets

1. **Strong Entity Set**:
   - Exists independently.
   - Has a primary key.
   - Example: Student entity set with attributes like Roll_No, Name, Age.

2. **Weak Entity Set**:
    - Cannot exist without being related to a strong entity set.
    - Does not have a primary key of its own; depends on a **foreign key** from the strong entity set.
    - Represented with a **double rectangle**.
    - Example: Dependent entity set that depends on the Employee entity set.

Attributes of Entities
Attributes describe the properties of entities and are represented as **ellipses** in ER diagrams. Types of attributes include:

- **Simple (atomic)**: Cannot be divided further (e.g., Age, Name).
- **Composite**: Can be divided into smaller sub-parts (e.g., Name → First Name, Last Name).
- **Derived**: Can be derived from other attributes (e.g., Age from Date of Birth).
- **Multivalued**: May have more than one value (e.g., Phone numbers).

ER Diagram Representation
In an **ER diagram**:

- **Entities** are represented by **rectangles**.
- **Attributes** are shown as **ellipses** connected to entities.
- **Entity sets** are labeled rectangles representing the collection of entities.

Relationships and Relationship Sets
In a **Database Management System (DBMS)**, especially within the **Entity-Relationship (ER) model**, **relationships** and **relationship sets** are fundamental for modeling how entities interact with each other. They help in representing real-world associations between different data objects.
Relationships and relationship sets are central to modeling how different entities interact in a database. They provide structure and clarity in database design, supporting data retrieval and ensuring relational integrity. Proper understanding of relationships ensures an efficient and normalized database structure.

Relationship
A **relationship** is an **association among two or more entities**. It describes how entities in the database are connected or related to each other.
Example

- A **Student** enrolled in a **Course**.
- An **Employee** works in a **Department**.
- A **Customer** places an **Order**.

In the above examples

- "enrolled in", "works in", and "places" are **relationships**.
- They link different entities (e.g., Student ↔ Course).

Representation

- In an **ER diagram**, a relationship is represented by a **diamond shape**.
- The **entities** involved are connected to the diamond.

Relationship Set
A **relationship set** is a **collection of similar relationships**. While a relationship refers to a single association, a relationship set includes **all instances** of that association in the database.

Example

- The relationship "enrolled in" between Students and Courses:
    - If 100 students enroll in 10 courses, there will be multiple entries in the **ENROLLS** relationship set.

Degree of a Relationship
The **degree** of a relationship is the number of entities involved.

- **Unary (1-ary):** Involves one entity set.
  - Example: An **Employee** supervises another **Employee**.
- **Binary (2-ary):** Involves two entity sets. *(Most common)*
  - Example: A **Customer** buys a **Product**.
- **Ternary (3-ary):** Involves three entity sets.
  - Example: A **Doctor** prescribes a **Medicine** to a **Patient**.

Relationship Types (Cardinality)
Cardinality defines the **number of entity instances** that can be associated with another entity via the relationship.

- **One-to-One (1:1):** One entity of A is related to only one entity of B.
  - Example: One **person** has one **passport**.
- **One-to-Many (1:N):** One entity of A is related to many entities of B.
  - Example: One **department** has many **employees**.
- **Many-to-One (N:1):** Many entities of A relate to one entity of B.
  - Example: Many **students** enrolled in one **course**.
- **Many-to-Many (M:N):** Many entities of A relate to many entities of B.
  - Example: Many **students** enroll in many **courses**.

Participation of Entities

- **Total Participation:** Every entity in the entity set must participate in the relationship.
  - Represented by a **double line** in ER diagram.
  - Example: Every **employee** must work in some **department**.
- **Partial Participation:** Some entities may not be part of the relationship.
  - Represented by a **single line**.

Attributes of Relationships
Just like entities, relationships can also have **attributes**.
Example

- In a **works on** relationship between **Employee** and **Project**, an attribute could be **Hours** (how many hours an employee works on a project).

7. Importance in DBMS

- Relationships are crucial for defining **data interconnections**.
- They help in **querying meaningful information** by navigating across related tables.
- They are the basis for designing **foreign keys** in relational models.
- Relationships maintain **referential integrity** and reduce **data redundancy**.

Q.3. Consider an e-commerce database that stores customer information, their account balances, and purchase details. To ensure that customer account balances are updated automatically after each purchase, a trigger can be implemented.

Trigger: After Purchase Update Account Balance

Trigger Event: After a new record is inserted into the 'Purchases' table

Trigger Action:
Retrieve the customer ID associated with the purchase from the 'Purchases' table.
Fetch the current account balance for the customer from the 'Customers' table using the customer ID.
Calculate the updated account balance by subtracting the purchase amount from the current balance.
Update the customer's account balance in the 'Customers' table with the calculated updated balance.

Benefits of using a trigger for this business rule:
Accuracy: Automatically updates account balances, eliminating the risk of human error or manual intervention.

Real-time Balance Tracking: Provides up-to-date account balances, ensuring transparency and accurate customer information.

Compliance with Business Logic: Enforces the business rule of updating account balances after each purchase, maintaining data integrity.

Reduced Development Effort: Offloads the task of updating balances from application code, streamlining development and maintenance.

Q.4. (a) From storage point of view, the files ae divided into the following types:

- Sequential files
- Direct or random files
- Indexed sequential files

Sequential files
In sequential data files, records are stored on the storage media in a sequence. No locations are specified for individual records. Records are stored in an order, one after the other. Similarly, records are also retrieved in sequential order. For example, to access a particular record from a data file, all records are searched one by one until the specified record is found. Therefore, accessing records in sequential files is very slow. It is the major disadvantage of sequential access.

Direct or random files
Direct files are also referred to as random files. In these files, records are not stored in a sequence. Each record is stored by specifying a particular address or location within a file. The address is calculated against the value of the key field of the record. Sometimes, the same address is calculated, which creates a problem to store record. This problem is known as ***synonym***.
Data accessing from direct files is very fast. A record is accessed directly by specifying its address.

Indexed sequential files
Indexed Sequential files can be processed sequentially as well as randomly. In these files, the location of individual record is also stored along with data of record. For this purpose, an index is created to keep the track of locations of individual records.

Index refers to the location on the storage media where record is stored. The key fields of the records are stored separately into the index along with the address of each record. Usually, an index is created in a new file called ***index file***. The index file is updated whenever a new record is added or an existing record is deleted.

Indexed file organization is more effective and fast in accessing data from data file than sequential file organization. These files occupy more space on the storage media but accessing speed of records is same as random access files.

Q.4. (b) **Backup File**
A type of file that is used to keep a copy of data is known as ***backup file***. It is permanent file. It is used to take the backup of important data (It means to make the duplicate copy of data).
Backup files are mostly created for the protection of important data or files of an organization. The data can be recovered from backup files if any data file is lost or damaged. Backup files are created by using specific software (utility program).

Winter Exam-2025

Solutions – Database Management System (Application)

Q.5. Data Definition Language (DDL)

a Definition

Data Definition Language (DDL) is a set of SQL commands used to define and manage **database structures** such as tables, schemas, indexes, and views.

Key Functions

- Create, alter, and delete database objects.
- Define structure but not the actual data.

Common DDL Commands

- CREATE: Creates a new database object (e.g., table).
- ALTER: Modifies the structure of an existing object.
- DROP: Deletes an object.
- TRUNCATE: Removes all records from a table without logging individual deletions.

Example

```
CREATE TABLE Students (  StudentID INT PRIMARY KEY,
  Name VARCHAR(50),
  Age INT);
ALTER TABLE Students ADD Email VARCHAR(100);

DROP TABLE Students;
```

Q.5. Data Manipulation Language (DML)

b Definition

Data Manipulation Language (DML) deals with **retrieving and modifying data** stored in the database.

Key Functions

- Add, update, delete, and retrieve records.
- Operates on data **within** database objects defined by DDL.

Common DML Commands

- INSERT: Adds new data.
- UPDATE: Modifies existing data.
- DELETE: Removes data.
- SELECT: Retrieves data.

Example

```
INSERT INTO Students (StudentID, Name, Age)
VALUES (101, 'Alice', 20);


UPDATE Students
SET Age = 21
WHERE StudentID = 101;

DELETE FROM Students
WHERE StudentID = 101;

SELECT * FROM Students;
```

Q.5. Instances in DBMS

c Definition

An **instance** refers to the **actual content (data)** in the database at a specific moment. It is a **snapshot of the database**.

Key Points

- Instances change frequently as data is inserted, updated, or deleted.
- It represents the **current state** of the database.

Example
If a table Students has 3 rows today, and tomorrow it has 5, then each day's data is a different instance.
-- Current instance
SELECT * FROM Students;
Assume output:

```
+-----------+--------+-----+
| StudentID | Name   | Age |
+-----------+--------+-----+
| 101       | Alice  | 20  |
| 102       | Bob    | 22  |
+-----------+--------+-----+
```

This data represents one instance of the Students table.

Schemas in DBMS
Definition
A **schema** is the **structure or blueprint** of a database. It defines how data is organized — tables, relationships, views, constraints, etc.
Key Points

- Describes the design of the database.
- Typically changes rarely.
- Logical view of the database structure.

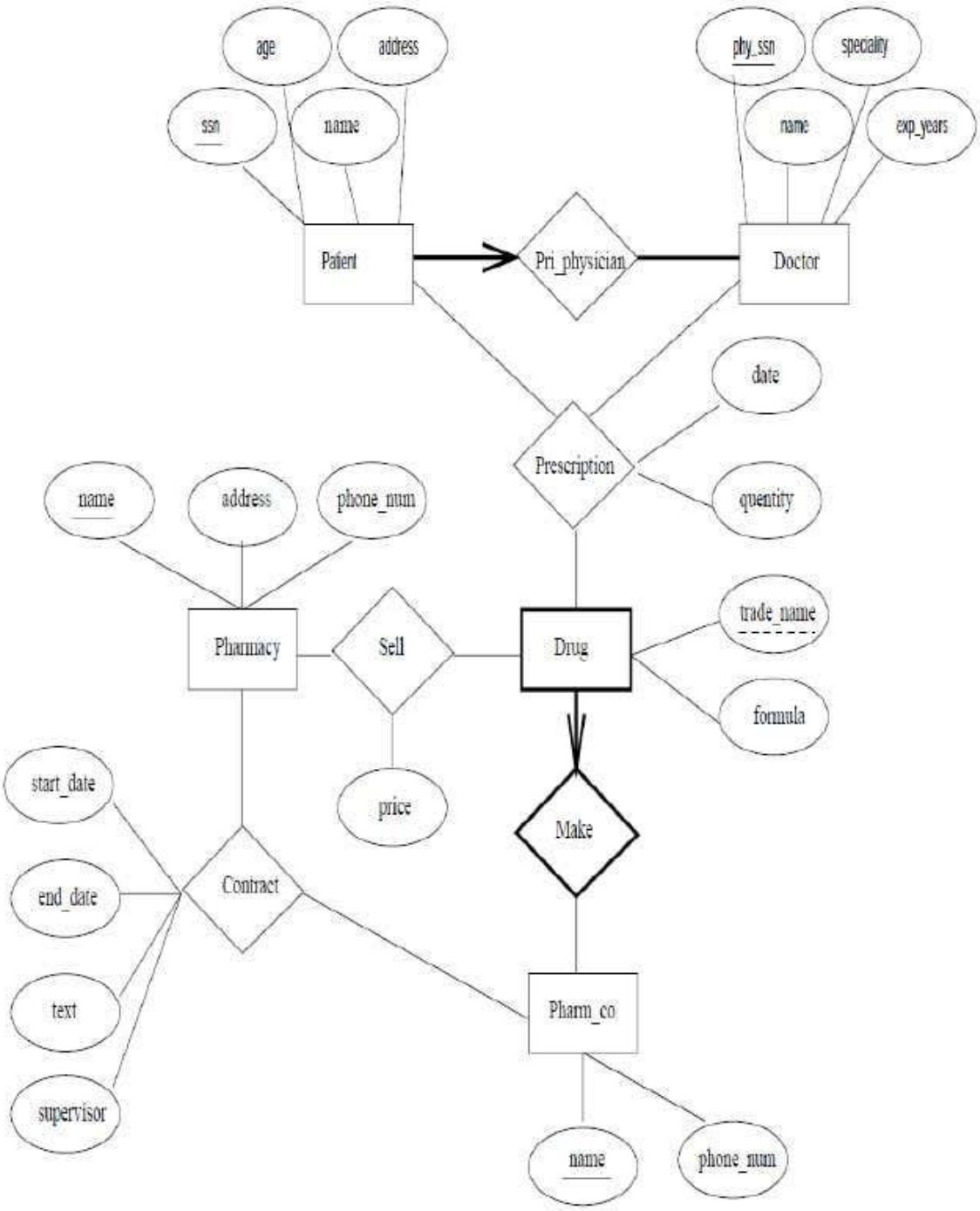
Example
Schema for a University database might include:

- Students(StudentID, Name, Age)
- Courses(CourseID, Title)
- Enrollments(StudentID, CourseID)

```
CREATE TABLE Courses (
CourseID INT PRIMARY KEY,
Title VARCHAR(100));
```

The above command defines part of the **schema** of the database.

Q.6.


age
address
ssn
name
Patient
Pri_physician
phy_ssn
speciality
name
exp_years
Doctor
date
Prescription
quantity
name
address
phone_num
Pharmacy
Sell
Drug
trade_name
formula
start_date
price
Make
end_date
Contract
text
supervisor
Pharm_co
name
phone_num

****