Pakistan Institute
of Public Finance Accountants

Model Solutions

Database Management System (Application)

Summer Exam-2024

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

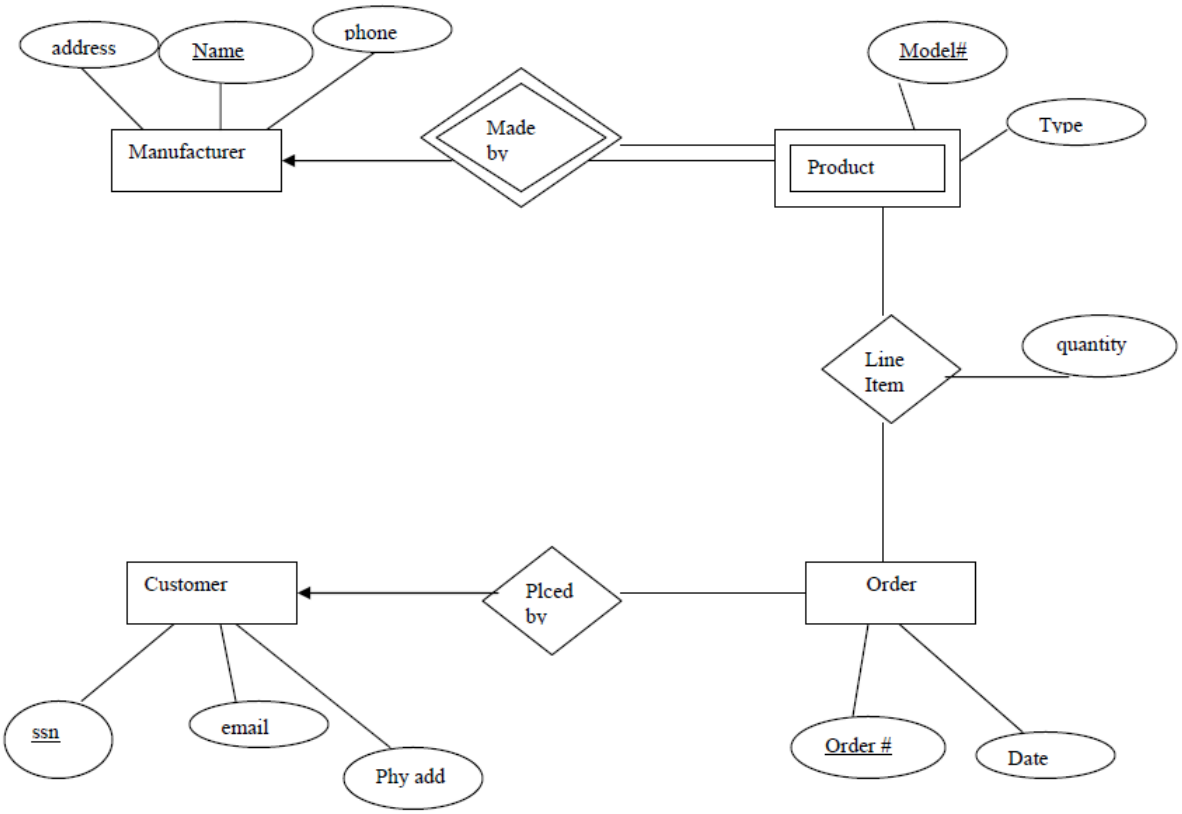
The Model Solutions have been developed by the professionals, based on standards, laws, rules, regulations, theories and practices as applicable on the date of that particular examination. No subsequent change will be applicable on the past papers solutions.

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.

Summer Exam-2024
Database Management System (Application)

Q.1. (a)



Q.1. (b) **Theoretical Foundations of Databases: Mathematical and Logical Principles**

The theoretical foundations of databases are rooted in mathematical and logical principles that provide a solid framework for their design, implementation, and functionality. These foundational concepts are essential for ensuring the integrity, consistency, and efficiency of data storage and retrieval. Let's delve into some key theoretical foundations:

Set Theory:

Set theory forms the base for understanding the mathematical model of databases. In the context of databases, a set corresponds to a collection of values, and operations such as union, intersection, and difference are employed to manipulate these sets. The fundamental notion of a relation in a relational database can be analogously linked to a set of tuples, where each tuple represents a row in a table.

Predicate Logic:

Predicate logic is a formal system for representing relationships and making logical inferences. In the context of databases, predicate logic is employed to formulate queries. Conditions in a WHERE clause of a SQL query, for instance, are expressions in predicate logic. This logical foundation enables precise and unambiguous specifications of the conditions under which data should be retrieved.

Relational Algebra:

Relational algebra is a mathematical system for manipulating relations, providing a theoretical basis for relational databases. It includes operations such as selection, projection, join, and union. These operations enable the transformation and retrieval of data in a relational database. For example, the SELECT operation corresponds to choosing certain rows based on specified conditions.

Contd…

Normalization:
Database normalization is a process rooted in set theory and relational algebra. It involves organizing data in a way that reduces redundancy and dependency, ensuring data consistency and integrity. Normal forms, such as the First Normal Form (1NF) and Third Normal Form (3NF), are manifestations of these theoretical principles. The aim is to minimize data anomalies and improve the efficiency of data storage.

ACID Properties:
ACID (Atomicity, Consistency, Isolation, Durability) properties are fundamental to transaction management in databases. These principles ensure that database transactions are reliable and maintain data integrity. For example, the atomicity property guarantees that a transaction is treated as a single, indivisible unit, either fully executed or not executed at all.

Entity-Relationship Model:
The Entity-Relationship (ER) model, based on set theory, is a graphical representation of the relationships among entities in a database. Entities are modeled as sets, and relationships are modeled as associations between these sets. This model provides a visual and conceptual framework for designing database schemas.

Graph Theory (for Non-Relational Databases):
In the context of non-relational databases (e.g., graph databases), graph theory becomes relevant. Graph databases represent data as nodes and edges in a graph, and graph theory principles are applied to traverse and query these interconnected structures efficiently.

Q.1. (c) Database redesign is necessary for two reasons. First, redesign is necessary both to fix mistakes made during the initial database design. Second, redesign is necessary to adapt the database to changes in system requirements. Such changes are common because information systems and organizations do not just influence each other they create each other. Thus, new information systems cause changes in systems requirements.

Q.2. Data abstraction in the context of database systems involves hiding complex details of the underlying data structures and operations, providing users and applications with simplified and well-defined interfaces. This abstraction is achieved through different layers, each serving a specific purpose. The primary abstraction layers in database systems are the physical, logical, and view layers.

1. Physical Layer:

Definition: The physical layer deals with how data is stored on the storage media, such as hard drives or solid-state drives. It encompasses details like data file organization, indexing methods, and storage structures.

- **Contribution to Efficiency:** Optimizations at this layer impact data retrieval speed and storage efficiency. For example, the choice of an appropriate indexing strategy or storage format can significantly enhance query performance.
- **Contribution to Security:** Security measures at this layer include access controls to physical storage and encryption methods. Ensuring that unauthorized users cannot directly access or manipulate the physical storage contributes to database security.
- **Contribution to Maintenance:** Maintenance tasks, such as optimizing storage structures or migrating to new storage technologies, are managed at this layer without affecting the logical or view layers.

Contd…

Real-world Example: In a relational database, the physical layer involves decisions about how tables are stored on disk, including choices like clustered or non-clustered indexing and partitioning strategies.

2. Logical Layer:

Definition: The logical layer abstracts the way data is perceived by users and applications. It defines the data model, schema, relationships, and constraints.

Contribution to Efficiency: Logical optimization involves query optimization, normalization, and indexing strategies that enhance overall database performance without exposing physical storage details.

Contribution to Security: Role-based access control and defining constraints (e.g., unique constraints, foreign key constraints) are part of security measures at the logical layer. This ensures data consistency and integrity.

Contribution to Maintenance: Changes in data structures, relationships, or constraints are managed at this layer without affecting the external views or physical storage. This eases the process of adapting the database to evolving requirements.

Real-world Example: In a relational database, the logical layer defines tables, their relationships, primary and foreign keys, and the normalization level of the schema.

3. View Layer:

Definition: The view layer provides customized, simplified, or abstracted representations of data to different users or applications. It involves creating virtual tables or views that present a subset of the data.

Contribution to Efficiency: Views allow users to retrieve only the necessary information, reducing the complexity of queries. Materialized views can be used to store precomputed results, improving query performance.

Contribution to Security: Access control at the view layer ensures that users see only the data they are authorized to access. Views act as a security mechanism by limiting the exposure of sensitive information.

Contribution to Maintenance: Changes to the underlying schema or data model can be shielded from users by modifying views. This ensures that modifications to the database structure do not disrupt applications or reporting tools.

Real-world Example: In a large enterprise database, different departments may have customized views that show only relevant information, providing a layer of abstraction over the complete dataset

Q.3. (a) Strategic Considerations and Planning in the Initiation Phase of the Database Life Cycle

The initiation phase of the database life cycle is a critical stage that sets the foundation for the entire database development process. During this phase, strategic considerations and planning activities play a pivotal role in ensuring the success of the database project. Key aspects include data requirements analysis, stakeholder involvement, and the selection of appropriate database models.

1. Data Requirements Analysis:

Definition: Data requirements analysis involves understanding and documenting the informational needs of the organization. This includes identifying the types of data required, their relationships, and the constraints that should be applied.

Contd…

a) Strategic Considerations:

Business Objectives: Align data requirements with the overall business objectives of the organization. Ensure that the database supports the core functions and goals of the business.

b) Data Quality: Establish criteria for data accuracy, completeness, consistency, and timeliness. Identify data sources and assess data quality requirements to meet organizational standards.

Scalability and Growth: Anticipate future data needs and design the database to be scalable. Consider the growth of data over time and ensure the database can accommodate increased volumes without compromising performance.

c) Real-world Example: In a healthcare organization, data requirements analysis may involve understanding the relationships between patient records, medical procedures, and billing information to support efficient healthcare delivery and financial transactions.

2. Stakeholder Involvement:

Definition: Stakeholder involvement is crucial for understanding diverse perspectives and ensuring that the database solution meets the needs of different user groups within the organization.

a) Strategic Considerations:

i. User Requirements Elicitation: Engage with end-users, administrators, and other stakeholders to elicit their requirements. Conduct interviews, surveys, and workshops to gather insights into their needs and expectations.

Communication and Collaboration: Establish clear channels of communication with stakeholders to ensure that their feedback is considered throughout the project. Foster collaboration to build a sense of ownership and commitment among stakeholders.

ii. Conflict Resolution: Address conflicting requirements or expectations early in the initiation phase. Establish mechanisms for resolving disagreements and ensuring that the final database design reflects a consensus among stakeholders.

b) Real-world Example: In a financial institution, stakeholder involvement may include collaboration with account managers, compliance officers, and IT personnel to gather requirements for a new database system that supports customer account management and regulatory reporting.

3. Selection of Appropriate Database Models:

Definition: Choosing the right database model involves selecting the most suitable data structure and architecture based on the nature of the data and the requirements of the application.

a) Strategic Considerations:

Data Complexity: Assess the complexity of the data to determine whether a relational, document-oriented, graph, or other database model is most appropriate. Consider factors such as data relationships, hierarchy, and variability.

b) Performance Requirements: Evaluate the performance characteristics required by the application. Choose a database model that aligns with the performance needs, considering factors like transaction volume, read/write patterns, and response time expectations.

c) Future Flexibility: Anticipate future changes in data structures or business requirements. Select a database model that allows for flexibility and adaptation over time.

Contd…

d) **Real-world Example:** For a content management system, the choice between a relational database for structured content and a document-oriented database for unstructured content may depend on the varying nature of data (articles, images, user comments) and the need for efficient retrieval and scalability.

Q.3. (b) To add a NOT NULL column to a table, Use ALTER TABLE statement with the ADD clause specifying the column name, data type and NOT NULL constraint. e.g ALTER TABLE table_name ADD column_name data_type NOT NULL;

Q.4. (a) **Enforcing Security Policies with Data Definition Language (DDL) in a Database Environment**

Data Definition Language (DDL) plays a crucial role in enforcing security policies and mitigating unauthorized access or modifications to the database schema. DDL statements are instrumental in defining and managing the structure of the database, and by extension, they contribute significantly to the overall security of the database environment.

Role of DDL in Enforcing Security Policies:

User Access Control:

DDL Statements: DDL statements are used to define and manage user roles, permissions, and access controls within the database. For instance, the GRANT and REVOKE statements are employed to assign specific privileges to users or roles, controlling their ability to perform certain DML and DDL operations.

Example:

```
GRANT SELECT, INSERT ON employees TO hr_user;
```

Schema Encryption and Obfuscation:

DDL Statements: DDL statements can be used to encrypt sensitive data within the database or to implement data obfuscation techniques. Encryption ensures that even if unauthorized access occurs, the data remains unintelligible without the appropriate decryption keys.

Example:

```
CREATE TABLE sensitive_data (
 id INT,
encrypted_column VARBINARY(MAX)
);
```

Auditing and Logging:

DDL Statements: DDL statements are used to configure auditing and logging settings. By enabling auditing, database administrators can track changes to the schema, monitor user activities, and detect any suspicious modifications.

Example:

```
CREATE AUDIT POLICY schema_changes;
ALTER SYSTEM SET AUDIT_POLICY = 'schema_changes=ENABLED';
```

Challenges Associated with Maintaining a Secure Database Environment:

Contd…

Granular Permission Management:

Challenge: Balancing the need for security with operational efficiency can be challenging. Ensuring that permissions are granted at a granular level without hindering day-to-day operations requires a careful analysis of user roles and responsibilities.

Dynamic Security Requirements:

Challenge: Security requirements are dynamic and evolve over time. Adapting the database schema and security policies to accommodate changing organizational needs and compliance standards poses a challenge. Frequent updates to security configurations may disrupt operations if not managed carefully.

Security Overheads:

Challenge: Implementing strong security measures, such as encryption and auditing, introduces performance overhead. Striking a balance between security and system performance requires continuous monitoring and optimization.

User Education and Training:

Challenge: Users and administrators need to be educated on security best practices and the implications of their actions within the database environment. Lack of awareness or training may lead to unintentional security vulnerabilities.

Q.4. (b) In the one-to-one relationship, there will be a constraint on EmpNumber as a foreign key in COMPUTER stating that EmpNumber must be unique.

Q.5. (a) **Transforming an Entity-Relationship (ER) Diagram into a Relational Database Schema:**

The transformation of an Entity-Relationship (ER) diagram into a relational database schema is a crucial step in the database design process. This process involves converting the visual representation of entities, relationships, and attributes into a set of tables, keys, and constraints in a relational database. Here's a comprehensive guide to the steps involved in this transformation:

1. Identify Entities and Attributes:

ER Model Aspect: Entities and attributes are represented visually in the ER diagram.

Transformation Steps:

Each entity becomes a table in the relational schema.

Each attribute becomes a column in the corresponding table.

Example:

ER Model: Employee (EmployeeID, Name, Department)

Relational Schema: Employee (EmployeeID, Name, Department)

2. Establish Relationships:

ER Model Aspect: Relationships between entities are represented in the ER diagram.

Transformation Steps:

Identify foreign keys that represent relationships between tables.

Determine cardinality (one-to-one, one-to-many, many-to-many) and participation constraints.

Create foreign key constraints based on these relationships.

Contd…

Example:

ER Model: Department (DeptID, DeptName) with a one-to-many relationship to Employee.

Relational Schema:

Department (DeptID, DeptName)

Employee (EmployeeID, Name, Department, FOREIGN KEY (Department) REFERENCES Department(DeptID))

3. Normalize the Schema:

ER Model Aspect: Normalization is addressed during the design phase.

Transformation Steps:

Apply normalization rules to eliminate redundancy and dependencies.

Create separate tables for related data to ensure each table represents a single theme or entity.

Example:

ER Model: Student (StudentID, Name, Course, Instructor)

Relational Schema (Normalized):

Student (StudentID, Name)

Course (CourseID, Instructor)

4. Handle Weak Entities:

ER Model Aspect: Weak entities are entities dependent on another entity.

Transformation Steps:

Represent weak entities as tables with a foreign key referencing the owning entity.

Include partial key attributes as part of the primary key.

Example:

ER Model: Address (Street, City) weak entity depending on Employee.

Relational Schema:

Employee (EmployeeID, Name, Address_Street, Address_City)

Address (Street, City)

5. Supertype and Subtype Entities:

ER Model Aspect: Supertype and subtype relationships represent generalization/specialization.

Transformation Steps:

Create separate tables for each subtype, with a foreign key referencing the supertype.

Use the same primary key in the subtype tables.

Example:

ER Model: Shape (ShapeID, Type), Circle (Radius) and Square (Side) subtypes.

Contd…

Relational Schema:

Shape (ShapeID, Type)

Circle (ShapeID, Radius, FOREIGN KEY (ShapeID) REFERENCES Shape(ShapeID))

Square (ShapeID, Side, FOREIGN KEY (ShapeID) REFERENCES Shape(ShapeID))

6. Include Constraints:

ER Model Aspect: Constraints like unique, primary key, and check constraints.

Transformation Steps:

Translate unique constraints into UNIQUE constraints in the relational schema.

Establish primary keys for each table.

Apply check constraints as needed.

Example:

ER Model: Unique constraint on EmployeeID.

Relational Schema: Employee (EmployeeID PRIMARY KEY, ...)

Q.5. (b) The hierarchical model each node can have multiple children but a child node must have only one parent. So in this particular example if a person LET say 'C' is subordinate of two Persons then we it is not easy to represent this information. To handle this scenario, we will have to duplicate C record. This will result in space wastage.
The solution is to use Relational Model.

Q.6. a Set operations in SQL, namely UNION, INTERSECT, and EXCEPT, draw inspiration from set theory concepts, offering powerful tools for combining and comparing data from multiple sources. Let's delve into the advanced applications of these operations and discuss scenarios where they prove particularly useful.

1. UNION Operator:

The UNION operator combines the results of two or more SELECT statements, removing duplicates.

Use Cases:

Merging data from similar tables with compatible columns.

Consolidating data from different regions or branches.

Example:

SELECT employee_id, employee_name FROM branch_A

UNION

SELECT employee_id, employee_name FROM branch_B;

2. INTERSECT Operator:

The INTERSECT operator returns common rows from two SELECT statements.

Contd…

Use Cases:

Identifying shared records between different datasets.

Finding common elements in different categories.

Example:

```
SELECT customer_id FROM online_orders
INTERSECT
SELECT customer_id FROM in_store_orders;
```

3. EXCEPT (or MINUS) Operator:

The EXCEPT operator returns rows that exist in the first SELECT statement but not in the second.

Use Cases:

Identifying differences between two datasets.

Finding records that exist in one dataset but not in another.

Example:

```
SELECT product_id FROM online_inventory
EXCEPT
SELECT product_id FROM in_store_inventory;
```

Advanced Applications:

Combining Multiple Set Operations:

```
SELECT employee_id, employee_name FROM branch_A
UNION
SELECT employee_id, employee_name FROM branch_B
INTERSECT
SELECT employee_id, employee_name FROM branch_C;
;
```

Using Set Operations in Subqueries:

```
SELECT product_id, product_name
FROM products
WHERE product_id IN (
   SELECT product_id FROM low_stock_items
   UNION
   SELECT product_id FROM discontinued_items
);
```

Contd…

);

Conditional Set Operations with CASE Statements:

```
SELECT customer_id,
      CASE


 WHEN customer_type = 'Premium' THEN 'Exclusive'
          ELSE 'Standard'
      END AS customer_category
FROM customer_data
INTERSECT
SELECT customer_id, 'VIP' AS customer_category
FROM vip_customers;
;
```

Scenarios:

Employee Management:

Scenario: Combining and identifying common employees in different branches.

SQL:

```
SELECT employee_id, employee_name FROM branch_A
UNION
SELECT employee_id, employee_name FROM branch_B
INTERSECT
SELECT employee_id, employee_name FROM branch_C;
C;
```

Inventory Discrepancies:

Scenario: Finding products that are in online inventory but not in the in-store inventory.

SQL:

```
SELECT product_id FROM online_inventory
EXCEPT
SELECT product_id FROM in_store_inventory;
;
```

Customer Segmentation:

Scenario: Identifying VIP customers and their exclusive features.

SQL:

Contd…

```
SELECT customer_id, 'VIP' AS customer_category
FROM vip_customers
INTERSECT
SELECT customer_id, 'Exclusive' AS customer_category
FROM exclusive_customers;
```

Q.6. b He does not plan to share his data with anyone so he probably does not need all these features. He should not be concern with security facility and concurrency control as their will not be multiple users. However, he should go for crash recovery. One can define external schema using view mechanism; therefore, it provides support for logical data independence in the relational model. The view mechanism helps us to tailor how user sees the data. So, Hassan can opt for view mechanism to simplify some database table's view (this will ease the query writing) and get data independence

****