Subject Name Solutions

4331603 – Summer 2025

Semester 1 Study Material

Detailed Solutions and Explanations

Question 1(a) [3 marks]

Define the following terms. 1) Metadata 2) Schema 3) Data dictionary.

Solution

Term	Definition
Metadata	Data about data that describes structure, format, and characteristics of database
Schema	Logical structure describing database organization and relationships
Data Dictionary	Centralized repository storing information about database elements

- **Metadata**: Information describing data characteristics and properties
- **Schema**: Blueprint defining database structure and constraints
- **Data Dictionary**: Catalog of all database objects and their attributes

Mnemonic

"MSD - My System Dictionary"

Question 1(b) [4 marks]

Write down advantages of Database Management system.

Solution

Advantage	Description
Data Independence	Applications independent of data storage
Data Integrity	Maintains accuracy and consistency
Security Control	User authentication and authorization
Concurrent Access	Multiple users access simultaneously

- **Reduced Redundancy**: Eliminates duplicate data storage
- **Centralized Control**: Single point of data management
- **Data Sharing**: Multiple applications can use same data
- **Backup Recovery**: Automatic data protection mechanisms

Mnemonic

"DISC-RCDB - Database Is Super Cool"

Question 1(c) [7 marks]

Explain Responsibilities of DBA.

Solution

Responsibility	Tasks
Database Design	Create logical and physical structures
Security Management	Control user access and permissions
Performance Tuning	Optimize queries and database operations
Backup Recovery	Ensure data protection and restoration
User Management	Create accounts and assign privileges

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    A[DBA Responsibilities] {-{-}{} B[Database Design]}
    A {-{-}{} C[Security Management]}
    A {-{-}{} D[Performance Tuning]}
    A {-{-}{} E[Backup \& Recovery]}
    A {-{-}{} F[User Management]}
    A {-{-}{} G[System Monitoring]}
{Highlighting}
{Shaded}
```

- **Database Installation**: Setup and configure DBMS software
- **Data Migration**: Transfer data between systems safely
- **Documentation**: Maintain database schemas and procedures
- **Monitoring**: Track system performance and resource usage
- **Troubleshooting**: Resolve database issues and errors

Mnemonic

"DSPBU-DMT - DBA Solves Problems By Understanding Database Management Tasks"

Question 1(c OR) [7 marks]

What is data abstraction? Explain three level ANSI SPARC architecture in detail.

Solution

Data Abstraction: Hiding complex database implementation details from users while providing simplified interfaces.

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[External Level] {-{-}{} B[Conceptual Level]}
    B {-{-}{} C[Internal Level]}
    A1[User Views] {-{-}{} A}
    B1[Logical Schema] {-{-}{} B}
    C1[Physical Storage] {-{-}{} C}
{Highlighting}
{Shaded}
```

Level	Description	Users
External Level	Individual user views and applications	End Users
Conceptual Level	Complete logical database structure	Database Designers
Internal Level	Physical storage and access methods	System Programmers

- **External Level**: Multiple user views hiding complexity
- **Conceptual Level**: Complete database schema without storage details
- **Internal Level**: Physical file organization and indexing
- **Data Independence**: Changes at one level don't affect others

Mnemonic

"ECI - Every Computer Implements"

Question 2(a) [3 marks]

Differentiate Schema vs Instance

Solution

Aspect	Schema	Instance
Definition	Database structure blueprint	Actual data at specific time
Nature	Static logical design	Dynamic data content
Changes	Rarely modified	Frequently updated

- **Schema**: Describes database organization and constraints
- **Instance**: Snapshot of database content at particular moment
- **Relationship**: Schema defines structure, instance contains data

Mnemonic

"SI - Structure vs Information"

Question 2(b) [4 marks]

Explain Specialization with example.

Solution

Specialization: Process of creating subclasses from superclass based on specific characteristics.

```
erDiagram
    EMPLOYEE \{
        int emp\_id
        string name
        float salary
    \}
    MANAGER \{
        string department
        int team\_size
    \}
    DEVELOPER \{
        string programming\_language
        string project
    \}

    EMPLOYEE ||{-{-}|| MANAGER : specializes}
    EMPLOYEE ||{-{-}|| DEVELOPER : specializes}
```

- **Top-Down Approach**: From general entity to specific entities
- **Inheritance**: Subclasses inherit superclass attributes
- **Disjoint**: Manager and Developer are separate categories
- **Example**: Employee specialized into Manager and Developer

Mnemonic

"STID - Specialization Takes Inheritance Down"

Question 2(c) [7 marks]

What is ER diagram? Explain different symbols used in E-R diagram with example.

Solution

ER Diagram: Graphical representation showing entities, attributes, and relationships in database design.

Symbol	Shape	Purpose	Example
Entity	Rectangle	Real-world object	Student, Course
Attribute	Oval	Entity properties	Name, Age, ID
Relationship	Diamond	Entity connections	Enrolls, Takes
Primary Key	Underlined oval	Unique identifier	Student_ID

```
erDiagram
    STUDENT \{
        int student\_id PK
        string name
        string email
        date birth\_date
    \}
    COURSE \{
        string course\_id PK
        string course\_name
        int credits
    \}
    ENROLLMENT \{
        date enrollment\_date
        string grade
    \}

    STUDENT ||{-{-}o\{ ENROLLMENT : enrolls}
    COURSE ||{-{-}o\{ ENROLLMENT : includes}
```

- **Entity Sets**: Collection of similar entities with same attributes
- **Weak Entity**: Depends on strong entity for identification
- **Cardinality**: Defines relationship participation (1:1, 1:M, M:N)
- **Participation**: Total (double line) or Partial (single line)

Mnemonic

"EARP - Entities And Relationships Program"

Question 2(a OR) [3 marks]

Differentiate DA vs DBA.

Solution

Aspect	Data Administrator (DA)	Database Administrator (DBA)
Focus	Data policies and standards	Technical database operations
Level	Strategic planning	Operational implementation
Scope	Organization-wide data	Specific database systems

- **DA**: Manages data as organizational resource
- **DBA**: Handles technical database maintenance and performance
- **Collaboration**: DA sets policies, DBA implements them

Mnemonic

"DA-DBA: Design Authority - Database Builder Administrator"

Question 2(b OR) [4 marks]

Explain Generalization with example.

Solution

Generalization: Bottom-up process combining similar entities into common superclass.

```
erDiagram
    VEHICLE \{
        string vehicle\_id
        string brand
        int year
        string color
    \}
    CAR \{
        int doors
        string fuel\_type
    \}
    MOTORCYCLE \{
        int engine\_cc
        string bike\_type
    \}

    VEHICLE ||{-{-}|| CAR : generalizes}
    VEHICLE ||{-{-}|| MOTORCYCLE : generalizes}
```

- **Bottom-Up Approach**: From specific entities to general entity
- **Common Attributes**: Shared properties moved to superclass
- **Specialization Reverse**: Opposite of specialization process
- **Example**: Car and Motorcycle generalized into Vehicle

Mnemonic

"GBCS - Generalization Brings Common Superclass"

Question 2(c OR) [7 marks]

What is attribute? Explain different types of attributes with example.

Solution

Attribute: Property or characteristic that describes an entity.

Attribute Type	Description	Example
Simple	Cannot be divided further	Age, Name
Composite	Can be subdivided	Address (Street, City, ZIP)
Single-valued	One value per entity	Student_ID
Multi-valued	Multiple values possible	Phone_numbers
Derived	Calculated from other attributes	Age from Birth_date

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    A[Attributes] {-{-}{} B[Simple]}
    A {-{-}{} C[Composite]}
    A {-{-}{} D[Single{-}valued]}
    A {-{-}{} E[Multi{-}valued]}
    A {-{-}{} F[Derived]}
    A {-{-}{} G[Key Attributes]}

    C {-{-}{} C1[Address: Street, City, ZIP]}
    E {-{-}{} E1[Phone: Mobile, Home, Work]}
    F {-{-}{} F1[Age calculated from DOB]}
{Highlighting}
{Shaded}
```

- **Key Attribute**: Uniquely identifies entity instances
- **Null Values**: Attributes that may have no value
- **Default Values**: Predetermined values when not specified
- **Constraints**: Rules governing attribute values

Mnemonic

"SCSMD-K - Simple Composite Single Multi Derived Key"

Question 3(a) [3 marks]

Explain the GRANT and REVOKE statement in SQL.

Solution

Statement	Purpose	Syntax Example
GRANT	Provides privileges to users	GRANT SELECT ON table TO user
REVOKE	Removes user privileges	REVOKE INSERT ON table FROM user

```
{-{-} Grant privileges}
GRANT SELECT, INSERT ON employees TO john;
GRANT ALL PRIVILEGES ON database TO admin;

{-{-} Revoke privileges  }
REVOKE DELETE ON employees FROM john;
REVOKE ALL ON database FROM user;
```

- **Privileges**: SELECT, INSERT, UPDATE, DELETE, ALL
- **Objects**: Tables, views, databases, procedures
- **Security**: Controls data access and modification rights

Mnemonic

"GR - Grant Rights, Remove Rights"

Question 3(b) [4 marks]

Explain following Character functions. 1) INITCAP 2) SUBSTR

Solution

Function	Purpose	Syntax	Example
INITCAP	Capitalizes first letter of each word	INITCAP(string)	INITCAP('hello world') = 'Hello World'
SUBSTR	Extracts substring from string	SUBSTR(string, start, length)	SUBSTR('Database', 1, 4) = 'Data'

```
{-{-} INITCAP examples}
SELECT INITCAP({database management}) FROM dual; {-{-} Database Management}
SELECT INITCAP({gtu university}) FROM dual; {-{-} Gtu University}

{-{-} SUBSTR examples  }
SELECT SUBSTR({Programming}, 1, 7) FROM dual; {-{-} Program}
SELECT SUBSTR({Database}, 5) FROM dual; {-{-} base}
```

- **INITCAP**: Converts string to proper case format
- **SUBSTR**: Parameters are string, starting position, optional length
- **Usage**: Text formatting and string manipulation operations

Mnemonic

"IS - Initialize String, Split String"

Question 3(c) [7 marks]

Consider following tables and write answers for the given queries. stud_master (enroll_no, name, city, dept)

Solution

```
{-{-} 1. Display all student details who study in IT dept}
SELECT * FROM stud\_master
WHERE dept = {IT};

{-{-} 2. Retrieve all information about name where name begins with p}
SELECT * FROM stud\_master
WHERE name LIKE {p\%};

{-{-} 3. Insert new student to table}
INSERT INTO stud\_master (enroll\_no, name, city, dept)
VALUES ({202501}, {John Smith}, {Mumbai}, {CS});

{-{-} 4. Add new column gender to table stud\_master}
ALTER TABLE stud\_master
ADD gender VARCHAR(10);

{-{-} 5. Count number of rows for stud\_master table}
SELECT COUNT(*) FROM stud\_master;

{-{-} 6. Display all student details in descending order of enroll\_no}
SELECT * FROM stud\_master
ORDER BY enroll\_no DESC;

{-{-} 7. Destroy table stud\_master along with data}
DROP TABLE stud\_master;
```

Query Type	SQL Command	Purpose
SELECT	Retrieves data	Display records

INSERT	Adds new data	Create records
ALTER	Modifies structure	Add columns
COUNT	Aggregate function	Count rows

Mnemonic

"SIAC-DOC - SQL Is A Complete Database Operations Collection"

Question 3(a OR) [3 marks]

Explain equi join with example in SQL.

Solution

Equi Join: Join operation using equality condition to combine tables based on common columns.

```
{-{-} Equi Join example}
SELECT s.name, c.course\_name
FROM students s, courses c
WHERE s.course\_id = c.course\_id;

{-{-} Using JOIN syntax}
SELECT s.name, c.course\_name
FROM students s
JOIN courses c ON s.course\_id = c.course\_id;
```

- **Equality Operator**: Uses = to match column values
- **Common Columns**: Tables must have related attributes
- **Result**: Combined data from multiple tables based on matches

Mnemonic

"EJ - Equal Join"

Question 3(b OR) [4 marks]

Explain following Aggregate functions. 1) MAX 2) SUM

Solution

Function	Purpose	Syntax	Example
MAX	Returns maximum value	MAX(column)	MAX(salary) = 50000
SUM	Returns total of values	SUM(column)	SUM(marks) = 450

```
{-{-} MAX examples}
SELECT MAX(salary) FROM employees; {-{-} Highest salary}
SELECT MAX(age) FROM students; {-{-} Oldest student age}

{-{-} SUM examples}
SELECT SUM(credits) FROM courses; {-{-} Total credits}
SELECT SUM(price * quantity) FROM orders; {-{-} Total order value}
```

- **Aggregate Functions**: Operate on multiple rows, return single value
- **NULL Handling**: Ignore NULL values in calculations
- **GROUP BY**: Can be used with grouping for category-wise results

Mnemonic

"MS - Maximum Sum"

Question 3(c OR) [7 marks]

Write SQL queries for the following table: PRODUCT_Master: (prod_no, prod_name, profit, quantity, sell_price, cost_price)

Solution

```
{-{-} 1. Create table PRODUCT\_Master}
CREATE TABLE PRODUCT\_Master (
    prod\_no VARCHAR(10) PRIMARY KEY,
    prod\_name VARCHAR(50),
    profit NUMBER(10,2),
    quantity NUMBER,
    sell\_price NUMBER(10,2),
    cost\_price NUMBER(10,2)
);

{-{-} 2. Insert one record in this table}
INSERT INTO PRODUCT\_Master VALUES
({P001}, {Laptop}, 15000, 10, 45000, 30000);

{-{-} 3. Find product having profit greater than 20000}
SELECT * FROM PRODUCT\_Master
WHERE profit {} 20000;

{-{-} 4. Delete product having quantity less than 5}
DELETE FROM PRODUCT\_Master
WHERE quantity {} 5;

{-{-} 5. Add 2\% profit in product having sell price greater than 5000}
UPDATE PRODUCT\_Master
SET profit = profit * 1.02
WHERE sell\_price {} 5000;

{-{-} 6. Add new field total\_price to PRODUCT\_Master}
ALTER TABLE PRODUCT\_Master
ADD total\_price NUMBER(10,2);

{-{-} 7. Find product name having no duplicate data}
SELECT DISTINCT prod\_name FROM PRODUCT\_Master;
```

Mnemonic

"CIDFAUD - Create Insert Delete Find Add Update Distinct"

Question 4(a) [3 marks]

Explain fully functional dependency with example.

Solution

Fully Functional Dependency: Attribute is fully functionally dependent if it depends on complete primary key, not on partial key.

Dependency Type	Definition	Example
Full FD	Depends on entire key	(Student_ID, Course_ID) $\rightarrow$ *Grade*
Partial FD	Depends on part of key	(Student_ID, Course_ID) $\rightarrow$ *Student_Name*