**CSA09: DATABASE MANAGEMENT SYSTEMS-ASSIGNMENT QUESTIONS**

**NAME: SARSHINI.S**

**REG NO : 192321113**

**CSA0564 - Database Management Systems for Query Processing**

**Question 1:**

**ER Diagram Question: Traffic Flow Management System (TFMS)**

**Scenario**

You are tasked with designing an Entity-Relationship (ER) diagram for a Traffic Flow Management System (TFMS) used in a city to optimize traffic routes, manage intersections, and control traffic signals. The TFMS aims to enhance transportation efficiency by utilizing real-time data from sensors and historical traffic patterns.

The city administration has decided to implement a TFMS to address growing traffic congestion issues. The system will integrate real-time data from traffic sensors, cameras, and historical traffic patterns to provide intelligent traffic management solutions. Key functionalities include:

1. **Road Network Management**:
   * **Roads**: The city has a network of roads, each identified by a unique RoadID. Roads have attributes such as RoadName, Length (in meters), and SpeedLimit (in km/h).
2. **Intersection Control**:
   * **Intersections**: These are key points where roads meet and are crucial for traffic management. Each intersection is uniquely identified by IntersectionID and has attributes like IntersectionName and geographic Coordinates (Latitude, Longitude).
3. **Traffic Signal Management**:
   * **Traffic Signals**: Installed at intersections to regulate traffic flow. Each signal is identified by SignalID and has attributes such as SignalStatus (Green, Yellow, Red) indicating current state and Timer (countdown to next change).
4. **Real-Time Data Integration**:
   * **Traffic Data**: Real-time data collected from sensors includes TrafficDataID, Timestamp, Speed (average speed on the road), and CongestionLevel (degree of traffic congestion).
5. **Functionality Requirements**:
   * **Route Optimization**: Algorithms will be implemented to suggest optimal routes based on current traffic conditions.
   * **Traffic Signal Control**: Adaptive control algorithms will adjust signal timings dynamically based on real-time traffic flow and congestion data.
   * **Historical Analysis**: The system will store historical traffic data for analysis and planning future improvements.

**ER Diagram Design Requirements**

1. **Entities and Attributes**:
   * Clearly define entities (Roads, Intersections, Traffic Signals, Traffic Data) and their attributes based on the scenario provided.
   * Include primary keys (PK) and foreign keys (FK) where necessary to establish relationships between entities.
2. **Relationships**:
   * Illustrate relationships between entities (e.g., Roads connecting to Intersections, Intersections hosting Traffic Signals).
   * Specify cardinality (one-to-one, one-to-many, many-to-many) and optionality constraints (mandatory vs. optional relationships).
3. **Normalization Considerations**:
   * Discuss how you would ensure the ER diagram adheres to normalization principles (1NF, 2NF, 3NF) to minimize redundancy and improve data integrity.

**Tasks**

**Task 1: Entity Identification and Attributes**

Identify and list the entities relevant to the TFMS based on the scenario provided (e.g., Roads, Intersections, Traffic Signals, Traffic Data).

Define attributes for each entity, ensuring clarity and completeness.

**Task 2: Relationship Modeling**

Illustrate the relationships between entities in the ER diagram (e.g., Roads connecting to Intersections, Intersections hosting Traffic Signals).

Specify cardinality (one-to-one, one-to-many, many-to-many) and optionality constraints (mandatory vs. optional relationships).

**Task 3: ER Diagram Design**

Draw the ER diagram for the TFMS, incorporating all identified entities, attributes, and relationships.

Label primary keys (PK) and foreign keys (FK) where applicable to establish relationships between entities.

**Task 4: Justification and Normalization**

Justify your design choices, including considerations for scalability, real-time data processing, and efficient traffic management.

Discuss how you would ensure the ER diagram adheres to normalization principles (1NF, 2NF, 3NF) to minimize redundancy and improve data integrity.

**Deliverables**

1. **ER Diagram**: A well-drawn ER diagram that accurately reflects the structure and relationships of the TFMS database.
2. **Entity Definitions**: Clear definitions of entities and their attributes, supporting the ER diagram.
3. **Relationship Descriptions**: Detailed descriptions of relationships with cardinality and optionality constraints.
4. **Justification Document**: A document explaining design choices, normalization considerations, and how the ER diagram supports TFMS functionalities.

**Task 1: Entity Identification and Attributes**

Roads

Attributes:

RoadID (PK)

RoadName

Length (in meters)

SpeedLimit (in km/h)

Intersections

Attributes:

IntersectionID (PK)

IntersectionName

Latitude

Longitude

Traffic Signals

Attributes:

SignalID (PK)

IntersectionID (FK)

SignalStatus (Green, Yellow, Red)

Timer (countdown to next change)

Traffic Data

Attributes:

TrafficDataID (PK)

Timestamp

RoadID (FK)

Speed (average speed on the road)

CongestionLevel (degree of traffic congestion)

**Task 2: Relationship Modeling**

Roads to Intersections

Relationship:

A road can connect to many intersections, but an intersection can be connected to many roads (many-to-many).

Implement using a junction table RoadIntersections with attributes RoadID (FK) and IntersectionID (FK).

Intersections to Traffic Signals

Relationship:

An intersection can have multiple traffic signals, but each traffic signal belongs to one intersection (one-to-many).

Traffic Data to Roads

Relationship:

Real-time traffic data is associated with roads (one-to-many).

**Task 3: ER Diagram Design**

+-----------------+ +-----------------+ +-----------------+

| Roads | | Intersections | | Traffic Signals|

|-----------------| |-----------------| |-----------------|

| PK RoadID |1----------M| PK IntersectionID|1---------M| PK SignalID |

| RoadName | | IntersectionName | | FK IntersectionID|

| Length | | Latitude | | SignalStatus |

| SpeedLimit | | Longitude | | Timer |

+-----------------+ +-----------------+ +-----------------+

+-----------------+

| Traffic Data |

|-----------------|

| PK TrafficDataID|

| Timestamp |

| FK RoadID |

| Speed |

| CongestionLevel |

+-----------------+

+---------------------+

| RoadIntersections |

|---------------------|

| FK RoadID |

| FK IntersectionID |

**Task 4: Justification and Normalization**

Design Choices Justification

Scalability: The design can accommodate additional roads, intersections, traffic signals, and real-time traffic data without requiring significant changes to the database schema.

Real-Time Data Processing: The inclusion of traffic data with timestamps allows for real-time analysis and decision-making, facilitating adaptive signal control and route optimization.

Efficient Traffic Management: By linking traffic signals to intersections and traffic data to roads, the system can dynamically adjust traffic signals and suggest optimal routes based on current conditions.

Normalization Considerations

First Normal Form (1NF): All attributes have atomic values, ensuring that each column contains only one value.

Second Normal Form (2NF): All non-key attributes are fully functional dependent on the primary key. For example, in the Traffic Data entity, attributes like Speed and CongestionLevel are dependent on TrafficDataID.

Third Normal Form (3NF): There are no transitive dependencies. For example, in the Traffic Signals entity, Timer is directly dependent on SignalID, not through another attribute.

**Deliverables:**

ER Diagram: A well-drawn ER diagram reflecting the structure and relationships of the TFMS database.

Entity Definitions: Clear definitions of entities and their attributes supporting the ER diagram.

Relationship Descriptions: Detailed descriptions of relationships with cardinality and optionality constraints.

Justification Document: An explanation of design choices, normalization considerations, and how the ER diagram supports TFMS functionalities.

**Question 2:**

**Question 1: Top 3 Departments with Highest Average Salary**

**Task:**

* + - 1. Write a SQL query to find the top 3 departments with the highest average salary of employees. Ensure departments with no employees show an average salary of NULL.

**Deliverables:**

* + - 1. SQL query that retrieves DepartmentID, DepartmentName, and AvgSalary for the top 3 departments.
      2. Explanation of how the query handles departments with no employees and calculates average salary.

**Question 2: Retrieving Hierarchical Category Paths**

**Task:**

* + - 1. Write a SQL query using recursive Common Table Expressions (CTE) to retrieve all categories along with their full hierarchical path (e.g., Category > Subcategory > Sub-subcategory).

**Deliverables:**

* + - 1. SQL query that uses recursive CTE to fetch CategoryID, CategoryName, and hierarchical path.
      2. Explanation of how the recursive CTE works to traverse the hierarchical data.

**Question 3: Total Distinct Customers by Month**

**Task:**

1. Design a SQL query to find the total number of distinct customers who made a purchase in each month of the current year. Ensure months with no customer activity show a count of 0.

**Deliverables:**

* 1. SQL query that retrieves MonthName and CustomerCount for each month.
  2. Explanation of how the query ensures all months are included and handles zero customer counts.

**Question 4: Finding Closest Locations**

**Task:**

* 1. Write a SQL query to find the closest 5 locations to a given point specified by latitude and longitude. Use spatial functions or advanced mathematical calculations for proximity.

**Deliverables:**

* + - 1. SQL query that calculates the distance and retrieves LocationID, LocationName, Latitude, and Longitude for the closest 5 locations.
      2. Explanation of the spatial or mathematical approach used to determine proximity.

**Question 5: Optimizing Query for Orders Table**

**Task:**

1. Write a SQL query to retrieve orders placed in the last 7 days from a large Orders table, sorted by order date in descending order.

**Deliverables:**

* 1. SQL query optimized for performance, considering indexing, query rewriting, or other techniques.
  2. Discussion of strategies used to optimize the query and improve performance.

**Question 1: Top 3 Departments with Highest Average Salary**

SELECT TOP 3

d.DepartmentID,

d.DepartmentName,

AVG(e.Salary) AS AvgSalary

FROM

Departments d

LEFT JOIN

Employees e ON d.DepartmentID = e.DepartmentID

GROUP BY

d.DepartmentID,

d.DepartmentName

ORDER BY

AvgSalary DESC;

**Explanation:**

Handling Departments with No Employees: The query uses a LEFT JOIN to include all departments, even those with no employees. For departments with no employees, the AVG function will return NULL.

Calculating Average Salary: The AVG function calculates the average salary for each department, and the results are grouped by DepartmentID and DepartmentName. The ORDER BY clause sorts the departments by average salary in descending order to find the top 3.

**Question 2: Retrieving Hierarchical Category Paths:**

WITH CategoryCTE AS (

SELECT

CategoryID,

CategoryName,

ParentCategoryID,

CAST(CategoryName AS VARCHAR(MAX)) AS Path

FROM

Categories

WHERE

ParentCategoryID IS NULL

UNION ALL

SELECT

c.CategoryID,

c.CategoryName,

c.ParentCategoryID,

CAST(cte.Path + ' > ' + c.CategoryName AS VARCHAR(MAX)) AS Path

FROM

Categories c

INNER JOIN

CategoryCTE cte ON c.ParentCategoryID = cte.CategoryID

)

SELECT

CategoryID,

CategoryName,

Path

FROM

CategoryCTE;

**Explanation:**

Recursive CTE: The WITH clause defines a recursive CTE named CategoryCTE.

Base Case: The initial query selects root categories (where ParentCategoryID is NULL) and initializes their path.

Recursive Step: The recursive part joins the CTE with the Categories table to append each subcategory to its parent's path.

Path Construction: The path is built by concatenating the parent path with the current category name.

**Question 3: Total Distinct Customers by Month:**

WITH Months AS (

SELECT 1 AS MonthNum, 'January' AS MonthName

UNION ALL SELECT 2, 'February'

UNION ALL SELECT 3, 'March'

UNION ALL SELECT 4, 'April'

UNION ALL SELECT 5, 'May'

UNION ALL SELECT 6, 'June'

UNION ALL SELECT 7, 'July'

UNION ALL SELECT 8, 'August'

UNION ALL SELECT 9, 'September'

UNION ALL SELECT 10, 'October'

UNION ALL SELECT 11, 'November'

UNION ALL SELECT 12, 'December'

)

SELECT

m.MonthName,

COALESCE(COUNT(DISTINCT o.CustomerID), 0) AS CustomerCount

FROM

Months m

LEFT JOIN

Orders o ON m.MonthNum = MONTH(o.OrderDate) AND YEAR(o.OrderDate) = YEAR(GETDATE())

GROUP BY

m.MonthNum,

m.MonthName

ORDER BY

m.MonthNum;

**Explanation:**

Ensuring All Months: The Months CTE lists all months of the year.

Handling Zero Customer Counts: The LEFT JOIN ensures that even months with no orders are included. The COALESCE function replaces NULL counts with 0.

Date Filtering: The join condition filters orders by the current year using YEAR(GETDATE()).

**Question 4: Finding Closest Locations:**

SELECT

LocationID,

LocationName,

Latitude,

Longitude,

(3959 \* ACOS(COS(RADIANS(@Lat)) \* COS(RADIANS(Latitude)) \* COS(RADIANS(Longitude) - RADIANS(@Lon)) + SIN(RADIANS(@Lat)) \* SIN(RADIANS(Latitude)))) AS Distance

FROM

Locations

ORDER BY

Distance

LIMIT 5;

**Explanation:**

Proximity Calculation: The query uses the Haversine formula to calculate the great-circle distance between two points on the Earth's surface.

Parameters: @Lat and @Lon represent the given latitude and longitude.

Ordering and Limiting: The query orders results by distance and retrieves the closest 5 locations.

**Question 5: Optimizing Query for Orders Table:**

SELECT

OrderID,

CustomerID,

OrderDate,

TotalAmount

FROM

Orders

WHERE

OrderDate >= DATEADD(DAY, -7, GETDATE())

ORDER BY

OrderDate DESC;

**Explanation:**

Indexing: Ensure an index exists on the OrderDate column to speed up the filtering and sorting operations.

Query Rewriting: The query uses DATEADD to dynamically calculate the date 7 days ago and filters orders accordingly.

Performance Strategies: Using indexing on OrderDate and possibly a covering index including other selected columns (OrderID, CustomerID, TotalAmount) to avoid lookups and improve query performance.