CSA0964: DATABASE MANAGEMENT SYSTEMS FOR QUERY PROCESSING ASSIGNMENT

SAKTHI GOVINDARAJU

192371038

**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.

**ANSWER**:

**Task 1: Entity Identification and Attributes**

Roads

RoadID (PK)

RoadName

Length (in meters)

SpeedLimit (in km/h)

Intersections

IntersectionID (PK)

IntersectionName

Latitude

Longitude

Traffic Signals

SignalID (PK)

IntersectionID (FK)

SignalStatus (Green, Yellow, Red)

Timer (countdown to next change)

Traffic Data

TrafficDataID (PK)

RoadID (FK)

Timestamp

Speed (average speed on the road)

CongestionLevel (degree of traffic congestion)

**Task 2: Relationship Modeling**

Roads and Intersections

Relationship: One-to-many (one road can connect to multiple intersections, but each intersection is associated with multiple roads)

Cardinality: One Road to Many Intersections

Optionality: Mandatory for Intersections (an intersection must be connected to at least one road)

Intersections and Traffic Signals

Relationship: One-to-many (one intersection can have multiple traffic signals)

Cardinality: One Intersection to Many Traffic Signals

Optionality: Mandatory for Traffic Signals (a traffic signal must be located at an intersection)

Roads and Traffic Data

Relationship: One-to-many (one road can have multiple traffic data entries over time)

Cardinality: One Road to Many Traffic Data entries

Optionality: Optional for Traffic Data (traffic data can exist without being associated with a specific road at times)

**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:

Scalability: The design supports scalability by allowing new roads, intersections, and traffic signals to be added without impacting existing data structures.

Real-Time Data Processing: The real-time data entity (Traffic Data) is designed to capture time-stamped traffic information, enabling real-time traffic management and analysis.

Efficient Traffic Management: Relationships between roads, intersections, and traffic signals ensure that traffic flow can be managed efficiently, with adaptive control algorithms adjusting signal timings based on real-time data.

Normalization:

1NF (First Normal Form): Ensures that each attribute contains only atomic (indivisible) values. Each table has a primary key that uniquely identifies each record.

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

3NF (Third Normal Form): Ensures that all attributes are directly dependent on the primary key. For example, in the Traffic Signals entity, SignalStatus and Timer are directly dependent on SignalID.

Deliverables

ER Diagram:

The ER diagram is drawn, reflecting the entities, attributes, and relationships.

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:

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.

Discussion of strategies used to optimize the query and improve performance

**ANSWER:**

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

SELECT DepartmentID, DepartmentName, AVG(Salary) AS AvgSalary

FROM Departments LEFT JOIN Employees

ON Departments.DepartmentID = Employees.DepartmentID

GROUP BY Departments.DepartmentID, Departments.DepartmentName

ORDER BY AvgSalary DESC

LIMIT 3;

**Explanation:**

* The LEFT JOIN ensures that all departments are included, even those with no employees.
* The AVG(Salary) function calculates the average salary for each department.
* The GROUP BY clause groups results by DepartmentID and DepartmentName.
* The ORDER BY AvgSalary DESC orders the departments by average salary in descending order.
* The LIMIT 3 restricts the results to the top 3 departments.
* Departments with no employees will show an average salary of NULL due to the LEFT JOIN.

**Question 2: Retrieving Hierarchical Category Paths**

WITH RECURSIVE CategoryCTE AS (

SELECT CategoryID, CategoryName, ParentID,

CAST(CategoryName AS VARCHAR(255)) AS HierarchicalPath

FROM Categories

WHERE ParentID IS NULL

UNION ALL

SELECT c.CategoryID, c.CategoryName, c.ParentID,

CONCAT(p.HierarchicalPath, ' > ', c.CategoryName)

FROM Categories c

INNER JOIN CategoryCTE p ON c.ParentID = p.CategoryID

)

SELECT CategoryID, CategoryName, HierarchicalPath

FROM CategoryCTE;

**Explanation:**

* The WITH RECURSIVE clause creates a Common Table Expression (CTE) that recursively fetches categories and their hierarchical paths.
* The base case selects categories with no parent (ParentID IS NULL).
* The recursive part joins the CTE with the Categories table on ParentID to build the hierarchical path.
* The CONCAT function appends the category name to the path.
* The final SELECT statement retrieves the CategoryID, CategoryName, and HierarchicalPath.

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

WITH Months AS (

SELECT DATE\_FORMAT(Date, '%Y-%m') AS Month

FROM Orders

WHERE YEAR(Date) = YEAR(CURDATE())

GROUP BY Month

UNION

SELECT DATE\_FORMAT(MAKEDATE(YEAR(CURDATE()), 1) + INTERVAL (a.a + (10 \* b.a)) MONTH, '%Y-%m') AS Month

FROM (SELECT 0 AS a UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4 UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) AS a

CROSS JOIN (SELECT 0 AS a UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4 UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9) AS b

WHERE MAKEDATE(YEAR(CURDATE()), 1) + INTERVAL (a.a + (10 \* b.a)) MONTH <= LAST\_DAY(CURDATE())

)

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

FROM Months m

LEFT JOIN Orders o ON DATE\_FORMAT(o.Date, '%Y-%m') = m.Month

GROUP BY m.Month

ORDER BY m.Month;

**Explanation:**

* The Months CTE generates a list of all months in the current year.
* The LEFT JOIN ensures all months are included, even those with no orders.
* The COALESCE function returns 0 for months with no customer activity.
* The final SELECT retrieves the Month and CustomerCount.

**Question 4: Finding Closest Locations**

SELECT LocationID, LocationName, Latitude, Longitude,

(6371 \* 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:**

* The query calculates the distance using the Haversine formula, which measures the distance between two points on a sphere.
* :lat and :lon are placeholders for the given latitude and longitude.
* The ORDER BY Distance clause sorts the locations by proximity.
* The LIMIT 5 clause restricts the results to the closest 5 locations

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

SELECT OrderID, OrderDate, CustomerID, TotalAmount

FROM Orders

WHERE OrderDate >= CURDATE() - INTERVAL 7 DAY

ORDER BY OrderDate DESC;

**Explanation:**

* An index on OrderDate can significantly improve performance.
* The query selects orders placed in the last 7 days using CURDATE() - INTERVAL 7 DAY.
* The ORDER BY OrderDate DESC clause sorts the results by order date in descending order.
* Additional optimizations could include covering indexes or partitioning the Orders table based on OrderDate for large datasets.