Assignment-3

Name of the Student :- A. Manoj Reddy

Name of the Course :- Data base Managment System

Reg No :- 192311171

Course ID :- CSA0653

Submission date :- 26-07-24.

Question 1:

ER Diagram Question : Traffic Flow Managment System:

Scenario:

You are tasked with designing an Entity - Relationship (ER) diagram for a Traffic Flow Managment System (TFMS)

Task1: Entity Identification and Attributes:

Roads	Intersections	Traffic Signals	Traffic Data
RoadID (PK)	Intersectionid (PK)	Signal ID (PK)	TrafficDataID (PK)
RoadName	InsectionName	IntersectionID (FK)	RoadID (FK)
length (m)	latitude	Signalstatus	Timestamp
Speed limit(km)	longitude	Timer	Speed
			congestionLevel

Task2 : Relationship Modeling:

Relationships:

1. Roads to Intersections
   * one Road up Can connect to multiple intersections
   * An intersection Can be Connected by multiple roads.
2. Intersection to Traffic Signals:
   * one intersection can host multiple traffic data entites.

Cardinality and optionality:

1. Roads to Intersection:
   * one road can connect to zero or more intersections
   * one intersection can connect to one or more roads.
2. Intersections to Traffic Signals:
   * one intersection can have zero or more traffic signals
   * one traffic signal must be associated with one intersection.
3. Roads to Traffic Data:
   * one road can have zero or more traffic data entries
   * one traffic data enty must be associated with one road.

Task3:- Justification and Normalization

1. Scalability
   * The design allows for easy addition of new roads, intersections, traffic signals, and traffic data entries without modifying the stru
2. Real-Time Data Procesing
   * Real-time traffic data integration is facilitated by the Traffic data
3. Efficient Traffic managment
   * The clear seperation of entities

Deliverables:

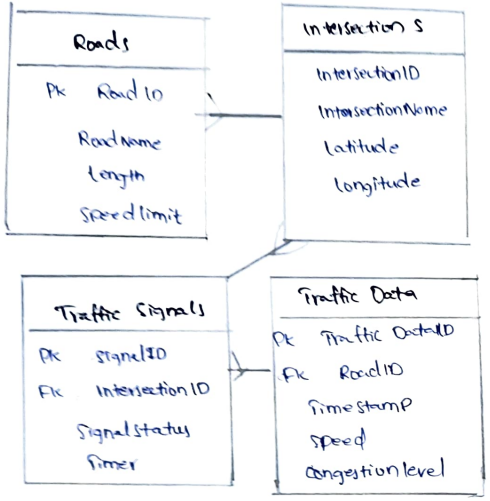
ER diagram: Provided above in plain text format

entity Definition: listed in task1

Reltionship Descriptions

Justification Document

Task 3: ER Diagram Design



Question 2:-

Question 1: Top 3 Departments with Highest Average Salary.

SQL Query

```
WITH AvgSalaries AS (
  SELECT
    d.department ID,
    d.DepartmentName,
    AVG (e.salary) As Avg Salary
FROM Departments d
  LEFT JOIN Employees e IN d.department ID =
                           e. Department ID
GROUP BY
    d.department ID,
    d. DepartmentName
  )
SELECT
    Department ID,
    Department Name.
  )
SELECT
    Department ID
    DepartmentName,
    Avg Salary
FROM
    Avg Salaries
ORDER BY
    Avg Salary DESC NULLS LAST
LIMIT 3;
```

Question 2: Retreiving Hierarchical Category Paths

SQL query.

```
WITH RECURSIVE CategoryPath AS (
  SELECT
    C. category ID,
    C.Category Name,
    C. ParentCategory ID,
  CAST (C.CategoryName As VARCHAR (255)) AS Path
FROM
    categories C
```

2.5. Optimizing Query for order table

```
Select * from orders
Where order date ≥ Date - Sub( cur date (),
    Interval 7 day)
  Order By
Order Date Desc;
```

3.

3.1. Handling Division operation:

```
Declare
  dividend number := 100;
    divisor number;
    result number;
Begin
   divisor := &divisor;
Begin
  result: = dividend / divisor;
DBMS - output - line ('Result: ' || result);
Exception
  is not allowed.');
  End;
  End;
```

3.2. Updating rows with For All

```
Declare
  emp_ids DBMS - SQL. Number - Table :=
  DBMS - sql. number - Table (101, 102, 103);
 Salary-line DBMS - Sql. number - Table; :=
  DBMS - sql.. Number - Table (1000, 2000, 3000);
Begin
  For All in emp_ids . First .. emp_ids . LAST
update employees
  Set Salary = Salary * Salary - inc (i)
  where employee ID = emp_ids (i);
      END;
```

3.3. Implementing Nested - Table procedure.

```
Create Type emp-table -type is
Table of employees %. Row Type,
Create (or) replace procedure
 get - department - employee(
   p - department - id IN number,
   p - employees Out emp-table - type )
 is BEGIN
```

2.2

```
WHERE
    C.Current Category ID  IS NULL

    UNION ALL

SELECT
    C.category ID,
    C.category Name,
    C.Parent Category ID,
CAST (CP.Path || '>' || C.category Name AS VARCHAR(255))
                                        AS Path
FROM
    Categories C
    INNER JOIN Category Path CP ON C.Parent Category ID
                                = CP.Category ID
)
SELECT
    Category ID,
    Category Name,
    Path
FROM
    CategoryPaths;
```

Final Query:-

* selects 'CategoryID', 'CategoryName', and the hiearchical 'Path' from the 'CategoryPaths' CTE
* This query effectively tranverses the hierachical Category structure and builds the full for each category.

2.3. Total Distinct Customers by month

```
Select
    Date-Format (Order date, '%Y-%m')
As month name,
    COUNT (Distinct CUSTOMER ID) AS
CUSTOMER COUNT
FROM
    Orders
Where Order date > Date-Sub (Curdate(), Interval 1 Year)
GROUP by
    Month Name
GROUP By
    Month Name;
```

2.4. Finding Closest locations:

```
Select
    location ID,
    location Name,
    latitude,
    longitude,
    (637* ASOS (Radians
        (34.7145)* COS
(Radians (latitude))* COS (Radians (-122.4194 -
Radians (longitude) * SIN (Radians (latitude)))
    As distance.
```

2.5. Optimizing Query for order table

```
Select * from orders
Where Order date ≥ Date -Sub( cur date(),
  Interval 7 day)
  Order By
Order Date Desc;
```

3.

3.1. Handling Division operation:

```
Declare
 dividend number: =100;
    divisor number;
    result number;
Begin
   divisor: = &divisor;
Beg in
  result: = dividend /divisor;
DBMS- output -line ('Result: ' || result');
Exception
   is not allowed.');
End;
End;
```

3.2. Updating rows with For All

```
Declare
  emp-ids DBMS -Sql. Number -Table; =
  DBMS -sql. number -Table (101, 102, 103);
Salary-line DBMS - Sql. number -Table; =
  DBMS - sql. Number - Table (1000, 2000, 3000);
Begin
  for All in emp-ids . First .. emp.ids .LAST
update Employees
  Set Salary = Salary * Salay -inc (i)
  where employee ID =emp-ids (i)';
    END;
```

3.3. Implementing Nested Table Procedure.

```
Create TYPE emp-table -type is
Table of employees %. Row Type',
Create (or) replace Procedure
get - department -employee(
   p- department -id IN number,
   p- employees out emp-table -type )
is BEGIN
```

```
Select *
BULK COLLECT INTO p_employees
FROM employees
where Department ID=
    p_department_ID;
END;
```

3.4. USING CURSOR Variables and Dynamic Sql

```
DECLARE
    TYPE emp_cursor IS REF CURSOR;
    emp_ref emp_cursor;
    emp_id
Employees.employee ID %. TYPE;
    firstname
Employees.First Name %. TYPE;
    last_Name
Employees.last name %. TYPE;
Salary_threshold Number := 5000;
SQL_Stmt varchar2(500);
Begin
    SQL_Stmt := 'SELECT employee id, Firstname,
last name From employees
    where salary > : Salary';
open emp_ref for SQL_Stmt using Salary.
threshold;
Loop
    Fetch emp_ref into empid; Firstname
last Name';
Exit when emp_ref %. NOT FOUND';
DBMS_output.put_line (emp_id || " - " || first_name || " " ||
last name';
    END loop;
close emp. ref;
END;
```

5.5 Designing pipelined function for sales data.

```
Create (or) Replace function get_sales_data(
P_month . number,
P_year. number)
Return Sales_data_type
Pipelined IS
Cursor Sales_cursor IS
SELECT order_ID, Customer ID, Order Amount
FROM orders
where extract (month from order date) = P_month
AND extract (year from order date) = P_year
Sales record
Sales_data_type %. row type;
BEGIN
For Sales_record IN Sales_cursor_loop
  Pipe row (Sales record);
END LOOP;
```