CATHOLIC UNIVERSITY OF EASTERN AFRICA

DISASTER RELIEF MANAGEMENT SYSTEM

SAM KARUCHI – 1049382

ENOCK LUSWETI - 1050893

JULIET CHIRCHIR – 1052086

ROMEO CLAUDE – 1052022

22ND NOV 2024

INTRODUCTION

The Disaster Relief Management System (DRMS) is a useful instrument for responding to disasters. It facilitates the tracking of affected individuals, supplies, donations, volunteers, and organizations. This method uses a single database to track relief operations in real time, ensuring that aid reaches those who need it the most.DRMS enables government agencies, NGOs, and donors to collaborate more effectively. The DRMS speeds up disaster recovery by coordinating relief operations. It utilizes technology to make catastrophe response more effective and less harmful.

Objectives

The main goals of a Disaster Relief Management System (DRMS) include:

Improving the efficiency of disaster response efforts:

- Consolidating and coordinating relief efforts for a streamlined and impactful response.
- Reducing reaction times and optimizing the use of resources

Monitoring resources in real-time:

- Monitoring the real-time availability and distribution of relief supplies.
- Guaranteeing efficient distribution of resources to areas with the highest demand.

Improved Cooperation:

- Promoting teamwork in emergency response efforts.
- Promoting communication and collaboration between government entities, non-profit organizations, sponsors, and volunteers

Support that focuses on the needs and well-being of the victim.

- Facilitating specific aid and minimizing overlapping actions.
- Giving precise documentation of individuals and families that have been impacted

Enhanced Visibility and Responsibility:

- Keeping accurate records of donations, resource allocation, and volunteer engagement.
- Making sure donors and stakeholders are held accountable.

Decision-making that is influenced by technology

- Using data analytics and digital tools to improve decision-making.
- Maximizing the effectiveness of disaster management approaches.

SYSTEM DESIGN

ER DIAGRAM

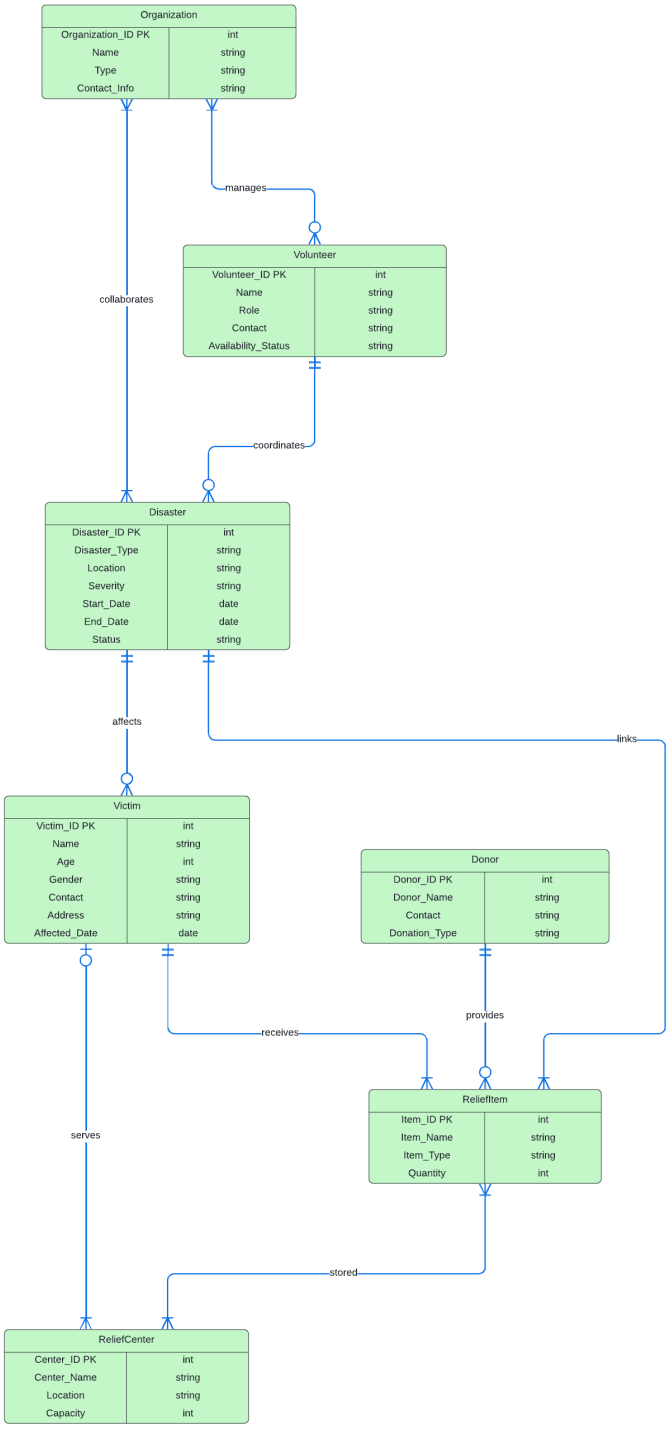

Organization
Organization_ID PK
int
Name
string
Type
string
Contact_Info
string
manages
Volunteer
Volunteer_ID PK
int
Name
string
Role
string
Contact
string
Availability_Status
string
collaborates
coordinates
Disaster
Disaster_ID PK
int
Disaster_Type
string
Location
string
Severity
string
Start_Date
date
End_Date
date
Status
string
affects
links
Victim
Victim_ID PK
int
Name
string
Age
int
Gender
string
Contact
string
Address
string
Affected_Date
date
Donor
Donor_ID PK
int
Donor_Name
string
Contact
string
Donation_Type
string
provides
receives
ReliefItem
Item_ID PK
int
Item_Name
string
Item_Type
string
Quantity
int
serves
stored
ReliefCenter
Center_ID PK
int
Center_Name
string
Location
string
Capacity
int

TABLE SCRIPTS

Table Creation Scripts

```
+ -- 1. Disasters Table
+ CREATE TABLE Disasters (
+     Disaster_ID INT PRIMARY KEY,
+     Disaster_Type VARCHAR(100) NOT NULL,
+     Location VARCHAR(255) NOT NULL,
+     Severity VARCHAR(50),
+     Start_Date DATE NOT NULL,
+     End_Date DATE,
+     Status VARCHAR(50)
+ );
+
+ -- 2. Victims Table
+ CREATE TABLE Victims (
+     Victim_ID INT PRIMARY KEY,
+     Name VARCHAR(100),
+     Age INT,
+     Gender VARCHAR(10),
+     Contact VARCHAR(50),
+     Address VARCHAR(255),
+     Affected_Date DATE,
+     Disaster_ID INT,
+     FOREIGN KEY (Disaster_ID) REFERENCES Disasters(Disaster_ID)
+ );
+
+ -- 3. Relief Items Table
+ CREATE TABLE Relief_Items (
+     Item_ID INT PRIMARY KEY,
+     Item_Name VARCHAR(100) NOT NULL,
+     Item_Type VARCHAR(50),
+     Quantity INT NOT NULL
+ );
+
+ -- 4. Relief Centers Table
+ CREATE TABLE Relief_Centers (
+     Center_ID INT PRIMARY KEY,
+     Center_Name VARCHAR(100) NOT NULL,
+     Location VARCHAR(255) NOT NULL,
+     Capacity INT
+ );
+
```

```
+
+ -- 5. Relief Center Items (Junction Table for Many-to-Many Relationship)
+ CREATE TABLE Relief_Center_Items (
+     Center_ID INT,
+     Item_ID INT,
+     Quantity INT,
+     PRIMARY KEY (Center_ID, Item_ID),
+     FOREIGN KEY (Center_ID) REFERENCES Relief_Centers(Center_ID),
+     FOREIGN KEY (Item_ID) REFERENCES Relief_Items(Item_ID)
+ );
+
+ -- 6. Volunteers Table
+ CREATE TABLE Volunteers (
+     Volunteer_ID INT PRIMARY KEY,
+     Name VARCHAR(100),
+     Role VARCHAR(50),
+     Contact VARCHAR(50),
+     Availability_Status VARCHAR(20),
+     Organization_ID INT,
+     FOREIGN KEY (Organization_ID) REFERENCES Organizations(Organization_ID)
+ );
+
+ -- 7. Organizations Table
+ CREATE TABLE Organizations (
+     Organization_ID INT PRIMARY KEY,
+     Name VARCHAR(100) NOT NULL,
+     Type VARCHAR(50),
+     Contact_Info VARCHAR(100)
+ );
+
+ -- 8. Donations Table
+ CREATE TABLE Donations (
+     Donation_ID INT PRIMARY KEY,
+     Donor_ID INT,
+     Item_ID INT,
+     Quantity INT NOT NULL,
+     Donation_Date DATE,
+     FOREIGN KEY (Donor_ID) REFERENCES Donors(Donor_ID),
+     FOREIGN KEY (Item_ID) REFERENCES Relief_Items(Item_ID)
+ );
```

```
+ -- 9. Donors Table
+ CREATE TABLE Donors (
+     Donor_ID INT PRIMARY KEY,
+     Donor_Name VARCHAR(100) NOT NULL,
+     Contact VARCHAR(50),
+     Donation_Type VARCHAR(50)
+ );
```

Table Population Scripts

```sql
-- Populating Disasters Table
INSERT INTO Disasters (Disaster_ID, Disaster_Type, Location, Severity, Start_Date, End_Date, Status)
VALUES
(1, 'Earthquake', 'Nairobi', 'Severe', '2024-10-10', NULL, 'Ongoing'),
(2, 'Flood', 'Kisumu', 'Moderate', '2024-09-15', '2024-10-05', 'Resolved'),
(3, 'Wildfire', 'Nyeri', 'Severe', '2024-08-01', '2024-08-15', 'Resolved');

-- Populating Victims Table
INSERT INTO Victims (Victim_ID, Name, Age, Gender, Contact, Address, Affected_Date, Disaster_ID)
VALUES
(1, 'Jane Doe', 35, 'Female', '123456789', 'Nairobi', '2024-10-10', 1),
(2, 'John Smith', 40, 'Male', '987654321', 'Kisumu', '2024-09-15', 2),
(3, 'Emily Davis', 29, 'Female', '555123456', 'Nyeri', '2024-08-01', 3);

-- Populating Relief_Items Table
INSERT INTO Relief_Items (Item_ID, Item_Name, Item_Type, Quantity)
VALUES
(1, 'Tents', 'Shelter', 200),
(2, 'Blankets', 'Clothing', 500),
(3, 'Water Bottles', 'Food & Drink', 1000),
(4, 'First Aid Kits', 'Medical', 150);

-- Populating Relief_Centers Table
INSERT INTO Relief_Centers (Center_ID, Center_Name, Location, Capacity)
VALUES
(1, 'Nairobi Relief Center', 'Nairobi', 1000),
(2, 'Kisumu Relief Hub', 'Kisumu', 700),
(3, 'Nyeri Emergency Camp', 'Nyeri', 500);

-- Populating Relief_Center_Items Table
INSERT INTO Relief_Center_Items (Center_ID, Item_ID, Quantity)
VALUES
(1, 1, 100), -- Nairobi Relief Center has 100 Tents
(1, 3, 500), -- Nairobi Relief Center has 500 Water Bottles
(2, 2, 200), -- Kisumu Relief Hub has 200 Blankets
(3, 4, 50);  -- Nyeri Emergency Camp has 50 First Aid Kits

```

```sql
-- Populating Donors Table
INSERT INTO Donors (Donor_ID, Donor_Name, Contact, Donation_Type)
VALUES
(1, 'Red Cross', 'contact@redcross.org', 'Financial'),
(2, 'Local NGO', 'contact@localngo.org', 'Relief Items'),
(3, 'John Doe', 'johndoe@example.com', 'Volunteer Service');

-- Populating Donations Table
INSERT INTO Donations (Donation_ID, Donor_ID, Item_ID, Quantity, Donation_Date)
VALUES
(1, 1, 1, 50, '2024-10-12'), -- 50 Tents donated by Red Cross
(2, 2, 3, 300, '2024-09-20'), -- 300 Water Bottles donated by Local NGO
(3, 1, 4, 20, '2024-08-10');  -- 20 First Aid Kits donated by Red Cross

-- Populating Volunteers Table
INSERT INTO Volunteers (Volunteer_ID, Name, Role, Contact, Availability_Status, Organization_ID)
VALUES
(1, 'Alice Green', 'Field Worker', 'alice@example.com', 'Available', 1),
(2, 'Bob White', 'Logistics', 'bob@example.com', 'Unavailable', 2),
(3, 'Charlie Brown', 'Medical Assistant', 'charlie@example.com', 'Available', 3);

-- Populating Organizations Table
INSERT INTO Organizations (Organization_ID, Name, Type, Contact_Info)
VALUES
(1, 'World Health Organization', 'Medical', 'info@who.org'),
(2, 'UNICEF', 'Child Support', 'contact@unicef.org'),
(3, 'Red Cross Society', 'Relief Organization', 'support@redcross.org');
```

Advanced SQL Queries

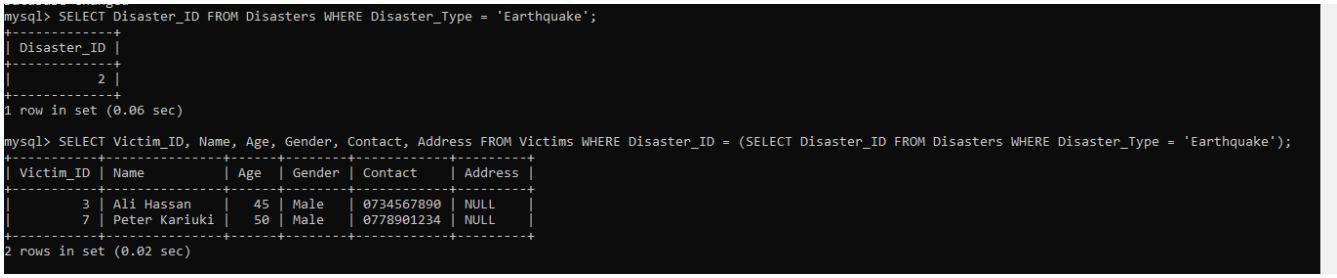
```sql
-- Total relief items distributed per disaster
SELECT D.Disaster_Type, SUM(RCI.Quantity) AS Total_Distributed
FROM Disasters D
JOIN Relief_Centers RC ON D.Location = RC.Location
JOIN Relief_Center_Items RCI ON RC.Center_ID = RCI.Center_ID
GROUP BY D.Disaster_Type;

-- Number of victims per disaster
SELECT D.Disaster_Type, COUNT(V.Victim_ID) AS Victim_Count
FROM Disasters D
JOIN Victims V ON D.Disaster_ID = V.Disaster_ID
GROUP BY D.Disaster_Type;

-- Rank relief centers by capacity
SELECT Center_Name, Location, Capacity,
       RANK() OVER (ORDER BY Capacity DESC) AS Rank_By_Capacity
FROM Relief_Centers;

-- Cumulative donation amounts by donor
SELECT Donor_Name, SUM(D.Amount) OVER (PARTITION BY Donor_ID) AS Total_Donated
FROM Donations D
JOIN Donors ON D.Donor_ID = Donors.Donor_ID;

DELIMITER //
CREATE PROCEDURE AllocateReliefItems(IN disaster_id INT, IN center_id INT)
BEGIN
    UPDATE Relief_Center_Items
    SET Quantity = Quantity - 100
    WHERE Center_ID = center_id;
    INSERT INTO Relief_Log (Disaster_ID, Center_ID, Allocated_Quantity, Allocation_Date)
    VALUES (disaster_id, center_id, 100, NOW());
END //
DELIMITER ;

```

```sql
--trigger
CREATE TRIGGER VictimUpdateLog
AFTER UPDATE ON Victims
FOR EACH ROW
BEGIN
    INSERT INTO Victim_Log (Victim_ID, Old_Contact, New_Contact, Update_Timestamp)
    VALUES (OLD.Victim_ID, OLD.Contact, NEW.Contact, NOW());
END;

--victim distribution by gender and disaster
SELECT Disaster_Type,
       SUM(CASE WHEN Gender = 'Male' THEN 1 ELSE 0 END) AS Male_Victims,
       SUM(CASE WHEN Gender = 'Female' THEN 1 ELSE 0 END) AS Female_Victims
FROM Victims
JOIN Disasters ON Victims.Disaster_ID = Disasters.Disaster_ID
GROUP BY Disaster_Type;

```

Testing involved verifying

- Database integrity through successful enforcement of constraints.
- Accuracy of queries, ensuring correct data retrieval and manipulation.

- Sample test case: Retrieving victims affected by the "Earthquake" disaster.



```
mysql> SELECT Disaster_ID FROM Disasters WHERE Disaster_Type = 'Earthquake';
+-------------+
| Disaster_ID |
+-------------+
|           2 |
+-------------+
1 row in set (0.06 sec)

mysql> SELECT Victim_ID, Name, Age, Gender, Contact, Address FROM Victims WHERE Disaster_ID = (SELECT Disaster_ID FROM Disasters WHERE Disaster_Type = 'Earthquake');
+-----------+--------------+------+--------+------------+---------+
| Victim_ID | Name         | Age  | Gender | Contact    | Address |
+-----------+--------------+------+--------+------------+---------+
|         3 | Ali Hassan   |   45 | Male   | 0734567890 | NULL    |
|         7 | Peter Kariuki |   50 | Male   | 0778901234 | NULL    |
+-----------+--------------+------+--------+------------+---------+
2 rows in set (0.02 sec)
```

Conclusion

The Disaster Relief Management System successfully achieves its objectives of organizing disaster data, tracking relief efforts, and supporting disaster recovery.

Recommendations

- Incorporate GIS mapping to visualize disaster areas.
- Add a user-friendly interface for stakeholders to interact with the system.
- Integrate predictive analytics to anticipate resource needs.

Appendices

Appendix 1: SQL Code Snippets

Include all SQL scripts used for creating tables, inserting data, and performing queries.

Appendix 2: ER Diagram

A full-page representation of the ER diagram