

# CENTRALIZED HOSPITAL MANAGEMENT SYSTEM FOR CHRONIC ILLNESS MANAGEMENT

By Nadia Isanga

## **CENTRALIZED HOSPITAL MANAGEMENT SYSTEM FOR CHRONIC ILLNESS MANAGEMENT.**

### **Introduction**

In Uganda, managing healthcare for patients with chronic and critical illnesses remains a significant challenge due to the lack of an integrated healthcare system. Fragmented medical records, delayed referrals, and limited access to timely emergency services often result in suboptimal care, especially for conditions such as diabetes, hypertension, asthma, and heart diseases. This scenario proposes the development of a **Centralized Hospital Management System (CHMS)** tailored to address these challenges, leveraging advanced data management techniques.

The CHMS will serve as a unified platform, centralizing patient information and facilitating seamless access across hospitals nationwide. The system will allow for integration with the National ID database for consistent patient identification.

### **Objectives of the System:**

- Facilitate real-time retrieval of patient records across hospitals.
- Enhance doctor preparation through timely access to medical histories for upcoming appointments.
- Improve patient care quality and reduce appointment delays.
- Optimize emergency services by linking patients to hospitals with available resources (e.g., ICU beds) and it will cater for a smart referral system to ensure patients are matched with the nearest and most suitable healthcare facilities.
- Provide actionable public health insights through disease and region-based reporting.

This is exactly the type of scenario that would benefit from a well-organized data set. Without proper data organization, the consequences could include loss of lives, inadequate

healthcare, and ineffective government planning due to the lack of accurate information on who is ill and what conditions they face.

## Conceptual Design of the Database System (E/R Diagram)

The E/R diagram of the system as shown below consists of six main entities: **Patient**, **Hospital**, **MedicalRecord**, **Appointment**, **EmergencyService**, **Insurance**, and a derived entity **PublicHealthDashboard**. Information on the dashboard is derived from other entities.

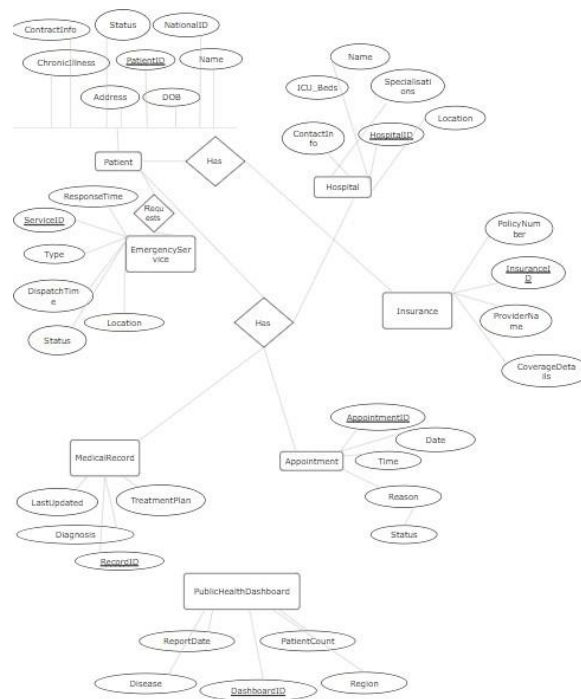


Figure 1: Conceptual Design of the E.R Diagram

### Entities and Attributes

#### 1. Patient

Attribute	Explanation
PatientID (Primary Key):	Unique identifier for each patient.
NationalID	Used as a consistent identifier across hospitals to allow retrieval of records nationwide.
Name, DOB, Address, ContactInfo	Basic patient biodata.
ChronicIllness	Tracks the patient's primary chronic condition.

Status: Indicates whether the patient is "Active" or "Archived"	Active Patients are still receiving care while archived patients have completed their treatment cycle.

## 2. Hospital

Attribute	Explanation
HospitalID (Primary Key):	Unique identifier for hospitals.
Name, Location, ContactInfo:	Basic hospital details.
Specializations:	Lists the medical areas of expertise to allow for efficient patient referral.
ICU Beds	Represents the availability of critical care resources to avoid delays, patient with a critical need is not assigned to a hospital where services are fully booked.

## 3. MedicalRecord

Attribute	Explanation
RecordID (Primary Key):	Unique identifier for records.
Diagnosis, TreatmentPlan, LastUpdated:	Tracks the patient's medical details.
Patient_PatientID (Foreign Key):	Links to Patient.
Hospital_HospitalID(Foreign Key):	Links to Hospital.
NationalID:	Ensures records are identifiable across hospitals.

**Rationale:** Captures diagnosis and treatment history, ensuring doctors have updated records for better patient care.

## Appointment

Attribute	Explanation
-----------	-------------

AppointmentID (Primary Key):	Unique identifier for appointments.
Status:	Refers to the current state of the appointment, which could either be scheduled, completed or cancelled.
Date, Time, Status, Reason:	Tracks scheduled visits.
Patient_PatientID (Foreign Key):	Links to Patient.
Hospital_HospitalID (Foreign Key):	Links to Hospital.

**Rationale:** Schedules and manages patient visits to hospitals, ensuring timely care.

### EmergencyService

Attribute	Explanation
ServicelID(Primary Key)	Unique ID for emergency responses.
Type,Location,DispatchTime, ResponseTime	Emergency details.
Status	Tracks service progress and could be active, pending or completed.
Patient_PatientID (Foreign Key):	Links to Patient.
Hospital_HospitalID (Foreign Key):	Links to Hospital.
NationalID(Foreign Key)	Helps to reference patients from different hospitals, Unique constraint removed to allow the same patient to be logged multiple times for the emergency service.

**Rationale:** Manages critical services like ambulances and airlifts for timely patient care.

### Insurance

Attribute	Explanation
InsuranceID (Primary Key)	Unique identifier for insurance policies.
ProviderName,PolicyNumber,CoverageDetails:	Financial details.
Status	Tracks service completion.
Patient_PatientID (Foreign Key):	Links to Patient.

**Rationale:** Ensures patients' financial coverage for medical services, reducing barriers to care.

### PublicHealthDashboard

Attribute	Explanation
DashboardID (Primary Key)	Unique report ID
Disease, Region, PatientCount, ReportDate:	Summarized data for public health reporting.

**Rationale:** Aggregates and reports disease trends by region to assist policymakers in healthcare planning.

### Relationships and Cardinality

The E/R diagram incorporates well-defined relationships to represent real-world data flow effectively:

- **Patient-to-MedicalRecord: One-to-Many** – A patient can have multiple medical records updated over time.
- **Patient-to-Appointment: One-to-Many** – A patient can schedule multiple appointments across different hospitals.
- **Hospital-to-Appointment: One-to-Many** – A hospital manages multiple appointments for various patients.
- **Patient-to-Insurance: One-to-One** – Each patient has one active insurance policy at a time in this scenario.
- **Patient-to-EmergencyService: One-to-Many** – A patient may require several emergency services.
- **Hospital-to-EmergencyService: One-to-Many** – One hospital can receive and respond to multiple emergency requests.

The PublicHealthDashboard entity is derived from aggregated data, such as MedicalRecord and Patient, summarizing chronic illness trends by region.

## **Normalization and Data Integrity**

- The database follows 3NF, a design principle used to eliminate duplicate data. Each table only contains relevant data for its purpose and all non-key attributes depend directly on the primary key.
- Primary keys uniquely identify records in each table, while foreign keys enforce referential integrity across relationships.
- NationalID serves as a consistent cross-hospital identifier for patients, enabling seamless record retrieval and updates.

## **Logical Design and Data Population**

The E/R diagram was drawn in MySQL Workbench and then transformed into a relational schema through Forward Engineering. Following this, the database was populated using SQL queries.



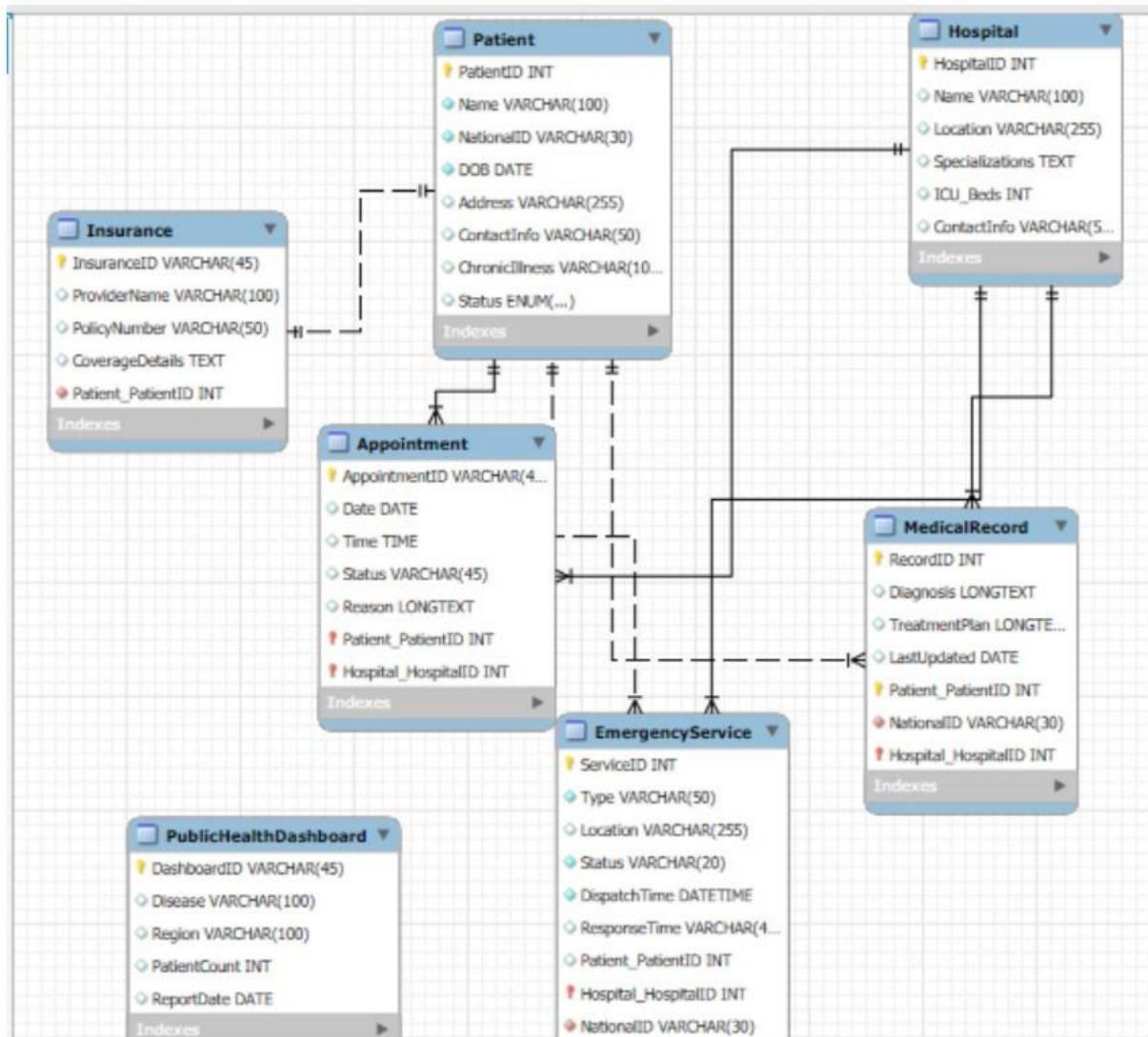


Figure 2:E.R Diagram in MYSQL workbench

Key tables include **Patient**, **Hospital**, **MedicalRecord**, **Appointment**, **EmergencyService**, **Insurance**, and **PublicHealthDashboard**. Each table aligns with its role in a real-world hospital scenario, using primary keys for unique identification and foreign keys to maintain relationships.

Data types like **VARCHAR** (text fields), **INT** (identifiers), and **DATE/DATETIME** (time-specific data) were carefully selected. Indexes were applied to frequently queried fields, such as NationalID and HospitalID, for better performance.

The tables were populated with sample data using SQL queries. This dataset supports complex queries, and from it, a meaningful analysis could be made.

## SQL Query Implementation and Analysis

The queries used in this section address real-world healthcare challenges, such as hospital capacity and patient care continuity. They align directly with the goals of a CHMS to improve patient outcomes and optimize hospital workflows.

Advanced SQL techniques (joins, filtering, grouping) were utilized to extract meaningful insights.

### Query 1: Detect Patients with Multiple Emergency Service Requests.

#### Purpose:

Identifies patients who required multiple emergency services within a short time frame (e.g., within the last month), pointing to potential critical care needs or deteriorating conditions.

#### SQL CODE

```
SELECT e.Patient_PatientID, p.Name, COUNT(e.ServiceID) AS
EmergencyCount FROM EmergencyService e JOIN Patient p ON
e.Patient_PatientID = p.PatientID WHERE e.DispatchTime >=
CURDATE() - INTERVAL 1 MONTH GROUP BY e.Patient_PatientID,
p.Name HAVING COUNT(e.ServiceID) > 1 ORDER BY EmergencyCount
DESC;
```

#### Explanation:

This query uses **SELECT** to retrieve patient IDs, names, and a count of emergency requests. It **JOINS** the **EmergencyService** table with the **Patient** table on Patient\_PatientID. The **WHERE** clause filters services within the last month, and **GROUP BY** combines results by patient. The **HAVING** clause ensures only patients with **more than one request** are returned, **ORDER BY** sorts the results by request count in descending order, identifying patients needing frequent emergency care.

Output of the Code.

<div> <div>Result Grid</div> <div></div> <div></div> <div>Filter Rows:</div> <div></div> <div>Export:</div> </div>			
	Patient_PatientID	Name	EmergencyCount
▶	917654311	Mary Klive	4
	957654351	Kamila Kaweesa	3
	967654361	Muungi Conrad	2

Insight:

- Identifying patients requiring frequent emergency services helps hospitals prioritize their care.
- Such information can trigger doctors to review medical records to detect patterns or underlying health issues requiring intervention.

**Query 2: Analyse Chronic Disease Trends Across Regions****Purpose:**

This query provides a region-based analysis of chronic illnesses, enabling healthcare administrators to identify regions with high disease prevalence for targeted intervention.

SQL CODE

```
SELECT Disease, Region, SUM(PatientCount) AS TotalPatients
FROM PublicHealthDashboard
GROUP BY Disease, Region
ORDER BY TotalPatients DESC;
```

Explanation

This query uses **SELECT** to retrieve Disease, Region, and the total number of patients calculated with **SUM()**. The **FROM** clause specifies the PublicHealthDashboard table. **GROUP BY** organizes data by Disease and Region, allowing totals to be summed for each combination. Finally, **ORDER BY** sorts the results in descending order of TotalPatients, highlighting regions with the highest disease burden.

### Output

	Disease	Region	TotalPatients
►	Diabetes	Kampala	3
	Asthma	Jinja	2
	Hypertension	Entebbe	1
	Arthritis	Mukono	1
	Chronic Kidney Disease	Kampala	1
	Diabetes	Makindye	1
	Epilepsy	Kampala	1
	Diabetes	Wampewo St, Kampala	1
	Diabetes	Kaza	1

### Insight:

- Helps policymakers allocate resources to regions with the highest disease burden.
- Hospitals can then organize outreach programs, public health campaigns and targeted screenings based on real data.

### Query 3: Predict Resource Requirements Based on Upcoming Appointments

#### Purpose:

Analyzes the number of scheduled appointments by hospital and chronic illness type to estimate future resource demand, such as doctors, specialists, and equipment.




#### SQL CODE

```
SELECT h.Name AS HospitalName, p.ChronicIllness,
COUNT(a.AppointmentID) AS AppointmentCount FROM Appointment a
JOIN Patient p ON a.Patient_PatientID = p.PatientID JOIN
Hospital h ON a.Hospital_HospitalID = h.HospitalID WHERE
MONTH(a.Date) = 12 AND YEAR(a.Date) = YEAR(CURDATE()) GROUP BY
h.Name, p.ChronicIllness ORDER BY AppointmentCount DESC;
```

### Explanation

This query uses **SELECT** to retrieve the hospital name, chronic illness, and count of scheduled appointments. It **JOINS** the **Appointment** table with **Patient** and **Hospital** to link patient and hospital details. The **WHERE** clause filters appointments scheduled in **December** of the current year using `MONTH(Date) = 12` and `YEAR(Date) = YEAR(CURDATE())`. **GROUP BY** organizes data by hospital name and illness, while **ORDER BY** sorts results by appointment count in descending order, helping identify hospitals with high patient traffic for specific illnesses.

### Output

Result Grid     Filter Rows: <input type="text"/>   Export:    Wrap			
	HospitalName	ChronicIllness	AppointmentCount
▶	Soroti Regional Hospital	Asthma	2
	Mulago Hospital	Diabetes	2
	Masaka Hospital	Hypertension	1
	Jinja Regional Hospital	Asthma	1
	Entebbe Hospital	Diabetes	1
	Jinja Regional Hospital	Arthritis	1
	Kampala International Hospital	Diabetes	1
	Lara Regional Hospital	Diabetes	1
	Masaka Hospital	Diabetes	1
	Kampala Hospital	Epilepsy	1
	Mukono Hospital	Chronic Kidn...	1

### Insight:

- Hospitals can prepare resources such as staffing (specialists for specific illnesses) and medical equipment.
- Improves operational efficiency by predicting demand for services.

#### **Query 4: Identify Hospitals Near Full Capacity to Prioritize ICU Bed Management**

##### Purpose:

This query provides a quick and efficient way to determine which hospitals have ICU availability, helping prioritize patient transfers or admissions based on capacity.

##### SQL CODE

```
SELECT Name AS HospitalName, Location, ICU_Beds  
  
FROM Hospital  
  
WHERE ICU_Beds > 0  
  
ORDER BY ICU_Beds DESC;
```

##### Explanation

**SELECT:** Retrieves the hospital name (Name), location (Location), and number of ICU beds (ICU\_Beds). **FROM Hospital:** Specifies the Hospital table as the data source. **WHERE ICU\_Beds > 0:** Filters hospitals that have at least **one ICU bed available**. **ORDER BY ICU\_Beds DESC:** Sorts hospitals in descending order of available ICU beds, with those having the most ICU capacity appearing first.

##### Output

	HospitalName	Location	ICU_Beds
►	Lara Regional Hospital	Jinja	50
	Kampala Hospital	Kampala	45
	Mulago Hospital	Kampala	20
	Entebbe Hospital	Entebbe	15
	Mediline Hospital	Entebbe	15
	Mukono Hospital	Mukono	10
	Jinja Regional Hospital	Jinja	10
	Soroti Regional Hospital	Soroti	10
	Masaka Hospital	Masaka	10
	Kampala International Hospital	Kampala	5

:

**Insight:**

- Hospitals with limited ICU capacity can trigger alerts for emergency services to direct critical patients elsewhere.
- Administrators can plan for resource allocation or equipment transfers to address hospital capacity shortages.

**Query 5: Identify Hospitals with the Highest Patient Load by Chronic Illness****Purpose:**

To find the hospitals treating the most patients for a specific chronic illness, such as diabetes, hypertension, etc. This helps identify facilities under strain and allocate resources effectively.

**SQL CODE**

```

SELECT h.Name AS HospitalName, p.ChronicIllness,
COUNT(p.PatientID) AS PatientCount

FROM Patient p

JOIN MedicalRecord m ON p.PatientID = m.Patient_PatientID

JOIN Hospital h ON m.Hospital_HospitalID = h.HospitalID

WHERE p.Status = 'Active'

```

```
GROUP BY h.Name, p.ChronicIllness
```

```
ORDER BY PatientCount DESC;
```

### Explanation

**JOIN** joins the Patient, MedicalRecord, and Hospital tables to link patients, their medical records, and hospitals. Includes filters for active patients (Status = 'Active'). **GROUP BY** groups data by hospital name and chronic illness to count patients per hospital per illness.

**ORDER BY** orders by patient count to prioritize hospitals with the highest load.

### Output

	HospitalName	ChronicIllness	PatientCount
►	Kampala Hospital	Epilepsy	1
	Jinja Regional Hospital	Arthritis	1
	Lara Regional Hospital	Diabetes	1
	Entebbe Hospital	Diabetes	1
	Masaka Hospital	Hypertension	1
	Mulago Hospital	Diabetes	1
	Kampala International Hospital	Diabetes	1

### Insight

Provides actionable insights into hospital strain and specific illness burdens, aiding resource allocation and planning.

### Query 6: List Patients Without Follow-Up Appointments Scheduled

#### Purpose:

Identify active patients with chronic illnesses who do not have any future appointments, ensuring they are contacted for continued care.

#### SQL CODE



```

SELECT p.PatientID, p.Name, p.ChronicIllness

FROM Patient p

LEFT JOIN Appointment a ON p.PatientID = a.Patient_PatientID AND
a.Date >= CURDATE()

WHERE a.AppointmentID IS NULL AND p.Status = 'Active';

```

### Explanation:

**LEFT JOINS** the Patient table with Appointment to include all patients, even those without future appointments. Filters for appointments scheduled on or after today (`a.Date >= CURDATE()`). Identifies patients where no corresponding appointment exists (`a.AppointmentID IS NULL`). Focuses only on active patients (`Status = 'Active'`).

### Output

	PatientID	Name	ChronicIllness
▶	153456789	Winnie Maka	Hypertension
	917654311	Mary Klive	Diabetes
	927654321	Julia Roberts	Diabetes
	937654331	Lillie Mukasa	Arthritis
	947654341	Peter Musiime	Diabetes
	957654351	Kamila Kaweesa	Diabetes
	967654361	Muungi Conrad	Epilepsy

### Insight

Ensures patients with chronic illnesses are not overlooked, reducing gaps in care and improving health outcomes.

## Conclusion and Limitations

The Centralized Hospital Management System successfully addresses critical healthcare challenges, including chronic illness management, medical record retrieval, and emergency coordination. By integrating multiple processes into one platform, it enhances decision-making, improves resource allocation, and supports patient care.

### Limitations:

- The system assumes accurate and real-time data entry, which may not always be feasible.
- Emergency services may need integration with GPS-based systems for real-time dispatch.

### Future Improvements:

- Implementing predictive analytics to forecast patient trends, resource shortages and patient outbreaks.
- Use advanced security and data privacy measures to protect sensitive patient data from breaches and unauthorized access.
- Incorporating telemedicine features for remote consultations.

This system lays the foundation for a more efficient and data-driven healthcare ecosystem in a country like Uganda.