Project 2 Report: Emergency Department Database

Heathvonn Styles

801038400

December 10, 2021

Project Overview

This project was developed based on an Emergency Department and procedures which are typically carried out in an Emergency Department. The goal of this project is to build a functional efficient medical database by which patient visits, diagnosis, doctor assignments, equipment accessibility can be appropriately managed.

In this database, patients should not be admitted without first making a visit to the emergency department. Before patients can visit, they must have demographics information entered in the database. This demographics information allows for referencing information like name, age and marital status of any given patient.

Database Setup

1. Create database: Run emr\_database\_script.sql
2. Load data: Run emr\_database\_data\_load.sql
3. Create user accounts and grant accesses: Run create\_users\_grant\_access.sql
4. Run tests (instructions in script file): Run emr\_database\_tests.sql

Scope

There are multiple functions that play a role in having a database that functions like an Emergency Department:

1. Patient must first enter their demographics information in the patient demographics table
2. Patient must be entered into the patient visits table. Note: if patients demographics information has not been entered, an error with be thrown stating to first add demographics information in patient demographics table.
3. If a patient is entered in patient visits table and their admission status is updated to YES, they must then be added to the active patient’s table. This means patients have been admitted.
4. Based on a patients’ diagnosis, there must be a decrease in room availability in the availability status table
5. If patient’s discharged status is update to ‘Yes’ their diagnosis records must be submitted to the records table and the patient must be removed from active patients table
6. Every database transaction is monitored by audit trail. There exists an audit table that captures user, time, transaction type and previous record information.

Database Design

Table count: 18

Triggers: 33

Procedures: 18

Functional Requirements

Tables

**Patient Demographics**: This table holds all patient demographics information. For example: patient id, patient name, date of birth, sex, occupation, ethnicity, marriage status, address, insurance number and insurance name.

The purpose of this table is to hold on relevant patient information when they come for a visit. This table must be filled with the patient information before a visit can take place. Just as with an actual hospital visit a patient is required to fill out a form with their information.

**Patient visits**: This table holds information relevant to the patients visit. The columns for this table are the visit id, patient id, time of visit, date of visit, day of the week, and a diagnosis id.

The patient id is important because it links back to the patient demographics table as foreign key. This is necessary in order to pull all data relevant to the patient’s id from the demographics table. Another foreign key would be the diagnosis id which allows accessing of the patient’s diagnosis information.

**Active Patients**: This table holds information relevant to the admissions table. The columns for this table are patient id, diagnosis id, assigned doctor, time presented at facility, admit start date, admit current date, discharged status, discharged date.

Once patients have filled out the patient demographics information and have been added to the patient visits table, if patient’s admitted status is changed to “YES” the patient will be added to the active patients’ table.

**Patient Records**: This table holds information relevant to previous patients’ diagnosis information. The columns in this table are the diagnosis id, patient id, signs, symptoms, test, discharge diagnosis, prescription and prescription quantity.

**Diagnosis**: This table holds the diagnosis information for patients who visit the emergency department. This table hold the diagnosis id, diagnosis name, admitted status, signs, symptoms, test, discharge diagnosis, prescription and prescription quantity.

**Diagnosis Responsibility**: This table contains information about any relevant diagnosis and the assigned specialty for that diagnosis

**Specialty Assignments**: This table holds information about which facility a staff member with a specific specialty id works. The values held in this table are the specialty id, specialty and facility.

**Provider**: This is a table of all workers involved in handling operations of the emergency department. The provider table contains the provider id, provider name, phone number, specialty id and workday.

**Availability Status**: This table is used to keep track of rooms and supplies in the emergency department. The values in this table are the date which the inventory status is being taken, beds, staffing, critical rooms, urgent rooms, stable rooms and supplies.

Audit Tables

Each audit table stores an audit\_id, name of current user, process being done on record, and the record information before a change occurs.

Patient Demographics Audit

Patient Visits Audit

Active Patients Audit

Patient Records

Diagnosis

Diagnosis Responsibility

Specialty Assignments

Provider

Availability Status

A picture containing calendar

Description automatically generated

Triggers

Visit\_before\_admit: This trigger makes sure patients have been added to the patients visits table before being admitted (added to the active patients table)

Remove\_after\_admit: This trigger is used to remove patients from patient visits table after they have been admitted (added to the active patients table)

Save\_record: This trigger saves patient information to patient records table after they have been discharged (discharged value changed to ‘Yes’ in active patients table)

Manage\_availabilities: This trigger updates beds and supplies count in availability table once diagnosis has been assigned to patient

Add admit: This trigger adds patient to active patients table once patient’s admitted status is changed to ‘Yes’

Collect\_demographics\_before\_visit: This trigger checks whether patient demographics information has been added to patient demographics table before a patient visit.

Audit trail triggers:

1. Active\_patients\_audit\_insert
2. Active\_patients\_audit update
3. Active\_patients\_audit delete
4. Availability\_status\_audit\_insert
5. Availability\_status\_audit\_update
6. Availability\_status\_audit\_delete
7. Diagnosis\_audit\_insert
8. Diagnosis\_audit\_update
9. Diagnosis\_audit\_delete
10. Diagnosis\_responsibility\_audit\_insert
11. Diagnosis\_responsibility\_audit\_update
12. Diagnosis\_responsibility\_audit\_delete
13. Patient\_demographics\_audit\_insert
14. Patient\_demographics \_audit\_update
15. Patient\_demographics\_audit\_delete
16. Patient\_records\_audit\_insert
17. Patient\_records\_audit\_update
18. Patient\_records\_audit\_delete
19. Patient\_visits\_audit\_insert
20. Patient\_visits\_audit\_update
21. Patient\_visits\_audit\_delete
22. Provider\_audit\_insert
23. Provider\_audit\_update
24. Provider\_audit\_delete
25. Specialty\_assignments\_audit\_insert
26. Specialty\_assignments\_audit\_update
27. Specialty\_assignments\_audit\_delete

Procedures

Add\_diagnosis: Adds new diagnosis to diagnosis table

Add\_visiting\_patient: Adds patient to patient visits table

Assign\_doctor: Assigns doctor to patient in active patients table

Available\_rooms\_count: Calculates total rooms currently available

Facility\_assignments\_for\_providers: Return providers/doctors and their facility assignments

Find\_patients\_for\_provider: Finds all patients be taking care of by a particular provider

Find patients with diagnosis: Finds all patients with the mentioned diagnosis

Find patients without doctor: Finds all patients who are not currently assigned a doctor

Find patients without prescriptions: Finds all patients who are not currently assigned a prescription

Find patients with prescriptions: Finds all patients who have been assigned a prescription

Find\_specialty\_count:

Fire\_employee: Removes employee from provider table

Hire\_employee: Adds employee to provider table

Patients\_demographics\_count: Returns count of patients demographics information

Patients\_with\_insurance: Finds all patients who have insurance

Patients\_without\_insurance: Finds all patients in patient demographics

Register\_patient: Add patient information to patients demographics table

Entity Relationship

Diagram

Description automatically generated with medium confidence

Updated Entity Relationship Diagram

Timeline

Description automatically generated with low confidence

UML DiagramGraphical user interface, application, Teams

Description automatically generated

* Patient\_demographics has one to one relationship with active\_patients and one to one relationship with patient\_visits
* Active\_patients has one to many relationship with patient\_records and many to one relationship with diagnosis
* Diagnosis has one to one relationship with patient\_visits
* Diagnosis\_responsibilty has many to one relationship with specialty\_assignments
* Special\_assignments has one to many relationship with provider

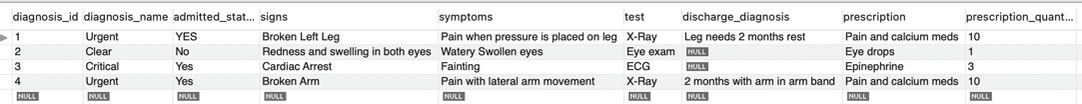
Proof of BCNF

Graphical user interface

Description automatically generated Graphical user interface, application

Description automatically generated





1NF: All columns contain only single values

2NF: - Tables are in 1NF and tables have single primary key

3NF: Table in 2NF and transitive dependencies moved to separate table

BCNF: For every functional dependency X → Y, X is the super key

Table Information

Graphical user interface

Description automatically generated Table

Description automatically generated with low confidence Graphical user interface, application

Description automatically generated Text

Description automatically generated

Table

Description automatically generated with medium confidence Graphical user interface

Description automatically generated with low confidence Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated with medium confidence

Text

Description automatically generated with low confidence

Tools Used

Draw.io

MySQL Workbench