University of North Carolina at Charlotte
ITCS 6120 Applied Databases
Project 2
04th May 2020

A database for an Electronic Medical Record System based on daily Emergency Room operations

Hemanth Gaddipati.

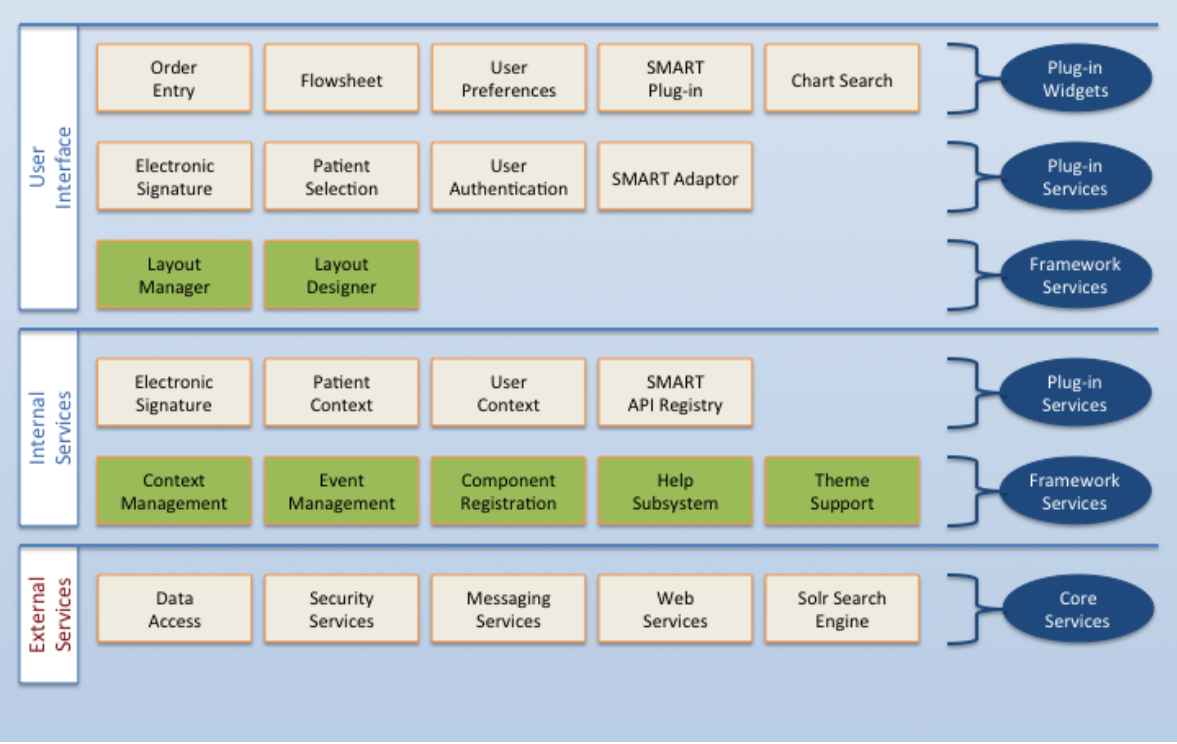
List of contents:

1. Introduction

An EMR (Electronic Medical Record)/ EHR (Electronic Health Record) provide an effective way to solve the problem of managing medical/clinical data. An EMR system is a great way to store and maintain medical data easily when compared to the conventional paper storage techniques which takes a lot of effort searching and maintaining such huge number of records that occupy a lot of physical space in the hospitals. Using EMR we can tackle this problem with a few manageable exceptions.

What is an EMR exactly?

Electronic Medical Recording – All the data in a doctor's office is usually collected in the form of charts and forms on papers. Using the paper record creates piles of papers and it is difficult to manage and utilize these records. Electronic Medical Record is the term given to medical record of patients collected in electronic format. Electronic records are easily portable and accessible anytime and from anywhere. It is easy to transfer electronic records to several different entities at a time. This helps patients and doctors who can access previous health records quickly and plan a treatment process quickly.



Comparison between paper record system and an EMR…..

Paper Office – Without EMR	Paperless Office – With EMR
1. Lot of Paper Use	1. Minimal Paper Use
2. Lot of Manpower Use	2. Minimal Manpower Use
3. Not so Efficient	3. Very Efficient
4. Not Portable	4. Portable
5. Cost Inefficient	5. Cost Efficiency to its Maximum
6. Prone to lot of Human Errors	6. Less prone to Human Errors
7. Clinical Data might get destroyed	7. Less chances of Clinical Data getting destroyed
8. Not so secure	8. Security at any extent is possible
9. Decision Making is not so quick	9. Quickest reports for Decision Making
10. Lot of dependency on staff	10. Less dependency on staff
11. Same office revenues	11. Increases office revenues
12. Less time effectiveness	12. Time saving at its maximum
13. Data collection from different office is not so effective	13. Automated data retrieval from different offices
14. The data in paper format is difficult to organize and retrieve.	14. Electronic data is very easy to organize and retrieve.

So, with the growing technology it is imperative that EMR system should be used as paper record system could be a liability which once was a blessing. In the later stages we will see the requirement analysis and DB creation for a ER operations part of a hospital and usage through a few queries.

2. Requirement Analysis

It is hard to implement an end to end EMR system so for this project I choose a part of hospital and created a data base with all the knowledge I have of the ER. Emergency room has patients walk in daily to get medical treatment for various symptoms they have. For this DB we need patient information, doctors/nurses information, patient visit information, patient admissions, patient initial vitals, tests and procedures done/ordered by providers to patients at the ER, Results of the tests, prescriptions, billing details, insurance details of the patients and discharge information of the patients.

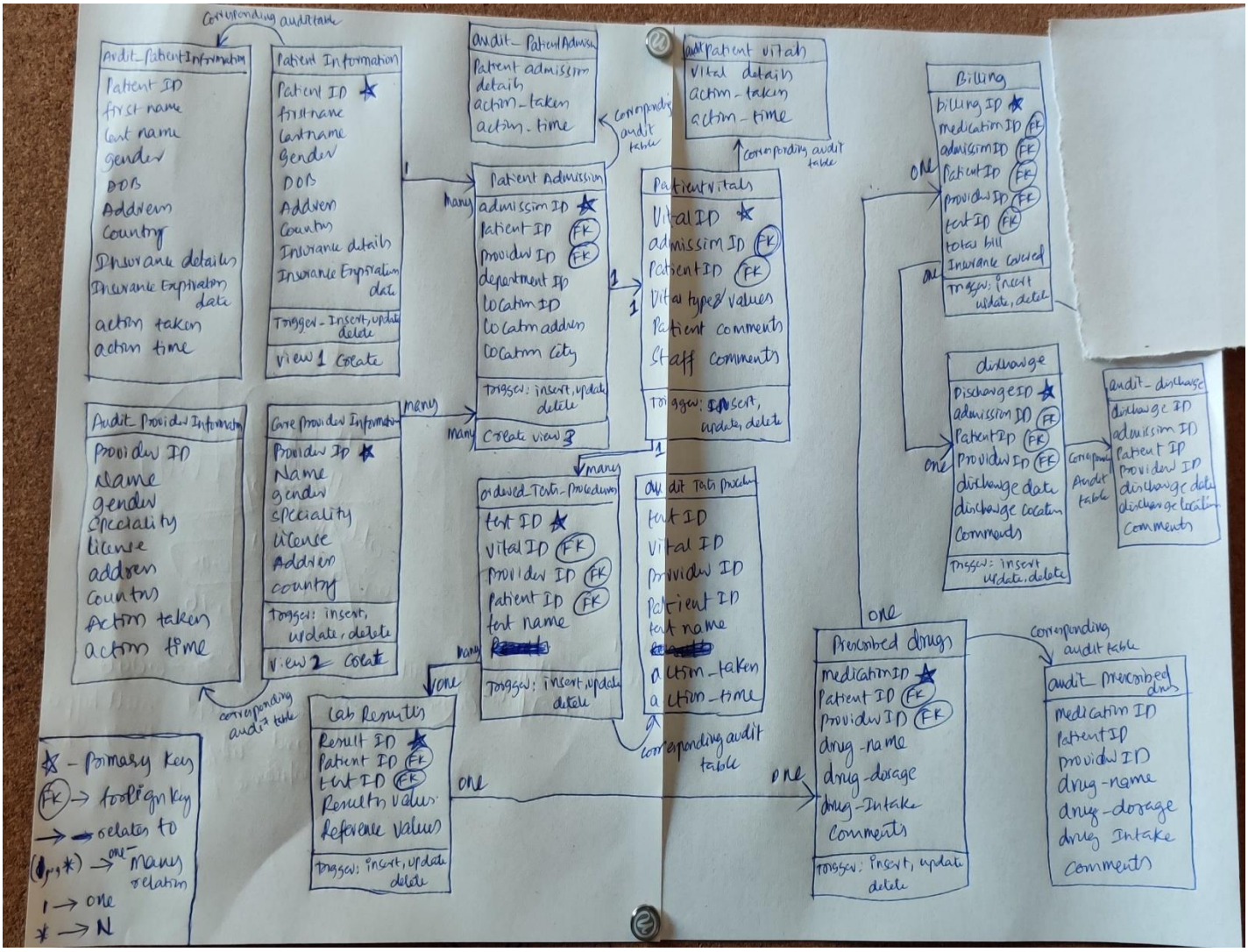
I created a Database with tables which has columns covering all the above-mentioned details. Typical scenario is a patient walks in and gets medical attention and then gets discharged from the ER or gets admitted to the hospital if the patient needs any additional treatment. ER provides primary care and then works as a initial screening tool to the hospital. Most of the cases they see are minimal severity and does not require extensive care.

Normal scenario would be a patient walks in fills out his details both personal and insurance related if he/her is new or provide a user ID to fetch the existing details then get an appointment with a care provider and then get his initial vitals checked by staff. Then the doctor would evaluate the symptoms based on the initial rests and then if any other tests are required, they'd order it and reevaluate the scenario. Once the doctor has all the information, he will prescribe medications and then the patient is discharged and now he checks out of the ER paying the bill after insurance coverage.

We have audit tables keep track of all the changes made to parent tables. The audit tables will keep track of all the updates, insert and delete queries that are and will be performed on the parent/main tables.

Stored procedures improve the time of execution and is way simpler when you need to run the same query multiple times. The stored procedure with pre-defined conditions will help the users to perform tasks with more ease with the help of MySQL workbench application programming interface.

3. UML Diagram



All the parent tables are created with primary and foreign keys are defined in the UML diagram above along with a few audit tables that will keep track of changes made to parent tables. The triggers are created on parent tables so that audit tables are updated accordingly. The stored procedures and views help users with no experience with SQL execute operations with ease.

4.Tables and Columns

I updated the tables a little bit to make it more accessible and the details are as follows

Table 1 PatientInformation

Column Name	Description
PatientId	Primary Key
Patient_LastName	Patients Last Name
Patient_FirstName	Patients First Name
Patient_MiddleName	Patients Middle Name
Patient_Gender	Sex of Patient
Patient_DoB	DOB of Patient
Patient_BloodGroup	The blood Group of a patient
Patient_AddressLine1	Address
Patient_AddressLine2	Address
City	City
State	State
ZipCode	Zip
Country	Country
County	County
Patient_PrimaryPhone	Primary Phone
Patient_Email	email of the patient
Patient_Insuranceprovider	provider name
Patient_insurancegroupid	insurance group id
Patient_insuranceuserid	user insurance id
Patient_InsuranceExpirationDate	Expiration date

Table 2 ProviderInformation

Column Name	Description
ProviderId	Primary Key
Provider_FullName	Full Name of the Provider
Provider_gender	Sex of the Provider
Provider_DoB	DOB of Provider
Provider_status	If the provider is active in service
Provider_rating	Rating of the Provider

Provider_AddressLine1	Address of the provider
Provider_AddressLine2	Addesss
City	City of Provider
State	State of Provider
ZipCode	Zip Code
Country	Country
County	Country
Provider_PrimaryPhone	Phone – Primary
Provider_Email	email of the patient
Provider_licenseid	License number
Provider_licenseExpirationDate	Expiration date
Provider_Specialty1_Name	Specialty of Provider
Provider_Specialty1_id	Specialty ID
Provider_Specialty2_Name	Second Specialty of Provider
Provider_Specialty2_id	Second Id

Table 3 patientAdmission

Column Name	Description
admissionId	Primary Key
ProviderId	foreign Key on table 4 providerId
PatientId	Foreign Key referred to Table 1 PatientId
LocationId	foreign Key
departmentId	department ID
emergency_level	emergency of the patient
admission_date	patient admission date
admin_comments	staff comments at the time of patient's admission
LocationId	location id
Location_Country	Country of facility where patient visits the Provider
Location_State	State of facility where patient visits the Provider
Location_County	County of facility where patient visits the Provider
Location_City	City of facility where patient visits the Provider
Location Address	Address of facility where patient visits the Provider

Table 4 patientVitals

Column Name	Description
vitalId	Primary Key
admissionId	foreign key admission id

patientId	foreign key patient id
locationid	foreign key location id
vital_type_and _values	vitals of the patient
patient_comments	patient_comments
staff_comments	staff comments

Table 5 ordered_tests_procedures

Column Name	Description
testId	Primary Key
vitalId	foreign Key vital id
ProviderId	foreign Key on table 4 providerId
patientId	foreign key patient id
test_name	test name
lab_staff_comments	lab staff comments

Table 6 labResults

Column Name	Description
Lab_ResultId	Primary Key
Appointment_Id	Foreign Key
Patient_Id	Foreign Key
Lab_authid	lab staff authorization id
testId	foreign key
lab_verificationid	lab staff verification id
testcomments1	lab staff comments
testcomments2	lab staff comments
lab_resultstatus	lab status on tests
result_description	description
result_values	result values of tests
reference_range	reference values
summary_comments	summary

Table 7 medicationsprescribed

Column Name	Description
medicationId	Primary Key

admissionId	foreign key admission id
patientId	foreign key patient id
ProviderId	foreign Key on table 4 providerId
drug_prescriptionId	drug id
drug_name	drug name
drug_dosage	drug dosage
drug_unitOfMeasurement	unit of measurement
drug_route	drug route
drug_startDate	drug start date
drug_endDate	drug stop date
drug_daily_intake	drug daily intake in terms of count

Table 8 Billing

Column Name	Description
billingId	Primary Key
medicationId	foreign Key
admissionId	foreign key admission id
patientId	foreign key patient id
ProviderId	foreign Key on table 4 providerId
testId	foreign key test id
totalBill	total bill of the servies
amount_paid_by_InsuranceProvider	amount covered by insurance of the patient

Table 9 discharge

Column Name	Description
Dischargeid	primary key
admissionId	foreign key admission id
patientId	foreign key patient id
ProviderId	foreign Key on table 4 providerId
discharge_date	patient discharge date
discharge_location	discharge location
Discharge_comments	staff comments on patient discharge
billingId	foreign key

Table 10 Audit_PatientInformation

Column Name	Description

PatientId	Patient ID
Patient_LastName	Patients Last Name
Patient_FirstName	Patients First Name
Patient_MiddleName	Patients Middle Name
Patient_AddressLine1	Address
Patient_AddressLine2	Address
City	City
State	State
ZipCode	Zip
Country	Country
County	County
Patient_PrimaryPhone	Primary Phone
Patient_Email	email of the patient
Patient_Insuranceprovider	provider name
Patient_insurancegroupid	insurance group id
Patient_insuranceuserid	user insurance id
Patient_InsuranceExpirationDate	Expiration date

Table 11 Audit_CareProviderInformation

Column Name	Description
ProviderId	Provider id
Provider_FullName	Full Name of the Provider
Provider_gender	Sex of the Provider
Provider_DoB	DOB of Provider
Provider_status	If the provider is active in service
Provider_rating	Rating of the Provider
Provider_AddressLine1	Address of the provider
Provider_AddressLine2	Addesss
City	City of Provider
State	State of Provider
ZipCode	Zip Code
Country	Country
County	Country
Provider_PrimaryPhone	Phone – Primary
Provider_Email	email of the patient
ProviderId	foreign Key on table 4 providerId
Provider_licenseid	License number
Provider_licenseExpirationDate	Expiration date
Provider_Specialty1_Name	Specialty of Provider

Provider_Specialty1_id	Specialty ID
Provider_Specialty2_Name	Second Specialty of Provider
Provider_Specialty2_id	Second Id

Table 12 Audit_patientAdmission

Column Name	Description
admissionId	Primary Key
ProviderId	providerId
PatientId	PatientId
LocationId	location id
departmentId	department ID
emergency_level	emergency of the patient
admission_date	patient admission date
admin_comments	staff comments at the time of patient's admission
LocationId	location id
Location_Country	Country of facility where patient visits the Provider
Location_State	State of facility where patient visits the Provider
Location_County	County of facility where patient visits the Provider
Location_City	City of facility where patient visits the Provider
Location Address	Address of facility where patient visits the Provider

Table 13 Audit_ordered_tests_procedures

Column Name	Description
testId	testid
vitalId	vital id
ProviderId	providerId
patientId	patient id
test_name	test name
lab_staff_comments	lab staff comments

Table 14 Audit_medicationsprescribed

Column Name	Description
medicationId	medication id
admissionId	admission id
patientId	patient id
ProviderId	providerId

drug_prescriptionId	drug id
drug_name	drug name
drug_dosage	drug dosage
drug_unitOfMeasurement	unit of measurement
drug_route	drug route
drug_startDate	drug start date
drug_endDate	drug stop date
drug_daily_intake	drug daily intake in terms of count

Table 15 Audit_discharge

Column Name	Description
Dischargeid	discharge id
admissionId	admission id
patientId	patient id
ProviderId	providerId
discharge_date	patient discharge date
discharge_location	discharge location
Discharge_comments	staff comments on patient discharge
billingId	foreign key

5.Tools used for creating Database

I used MySQL workbench to write and execute queries on MySQL server using Structured Query Language

6.Triggers

A database trigger is a stored procedure that is invoked automatically when a predefined event occurs. Database triggers enable DBAs (Data Base Administrators) to create additional relationships between separate databases. In other ways, a trigger can be defined to execute before or after an INSERT, UPDATE, or DELETE operation, either once per modified row, or once per SQL statement. If a trigger event occurs, the trigger's function is called at the appropriate time to handle the event. Triggers can be assigned to tables and are called every time trigger is triggered.

Here are sample scenarios and all the rest of the scenarios are included in the submission folder

Scenario 1: Trigger to update audit table when parent table is updated

```
-- trigger to update patient information

drop trigger if exists patientInformation_after_update;

DELIMITER //

create trigger patientInformation_after_update

after update on patientinformation

for each row

begin

insert into audit_patientInformation values

(old.patientId,old.patient_LastName,old.patient_FirstName,old.patient_MiddleName,old.patient_addressLine1,old.patient_addressLine2,old.city,old.state,old.Zipcode,old.countyCode,

old.country,old.patient_primaryPhone,old.patient_emailId,old.patient_InsuranceProvider,old.patient_InsuranceGroupId,old.Patient_InsuranceUserId,old.Patient_InsuranceExpirationDate,

 'updated', now());

end//

DELIMITER ;
```

Scenario 2: Trigger to insert values to audit table when parent table is updated

```
-- trigger to insert patient information
```

```
drop trigger if exists patientInformation_after_insert;

DELIMITER //

create trigger patientInformation_after_insert

after insert on patientinformation

for each row

begin

insert into audit_patientInformation values

(new.patientId,new.patient_LastName,new.patient_FirstName,new.p
atient_MiddleName,new.patient_addressLine1,new.patient_addressL
ine2,new.city,new.state,new.Zipcode,new.countyCode,

new.country,new.patient_primaryPhone,new.patient_emailId,new.p
atient_InsuranceProvider,new.patient_InsuranceGroupId,new.Patient
_InsuranceUserId,new.Patient_InsuranceExpirationDate,

 'inserted', now());

end//

DELIMITER ;
```

Scenario 3: Trigger to delete values from parent table and update the audit table accordingly

```
-- trigger to delete patient information

drop trigger if exists patientInformation_before_delete;

DELIMITER //

 create trigger patientInformation_before_delete

before delete on patientinformation
```

```
for each row

begin

insert into audit_patientInformation values

(old.patientId,old.patient_LastName,old.patient_FirstName,old.patient_MiddleName,old.patient_addressLine1,old.patient_addressLine2,old.city,old.state,old.Zipcode,old.countyCode,

old.country,old.patient_primaryPhone,old.patient_emailId,old.patient_InsuranceProvider,old.patient_InsuranceGroupId,old.Patient_InsuranceUserId,old.Patient_InsuranceExpirationDate,

 'deleted', now());

end//

DELIMITER ;
```

7.Stored Procedures

SQL Server stored procedures are used to group one or more Transact-SQL statements into logical units. The stored procedures are stored as named objects in the SQL Server Database Server. When you call a stored procedure for the first time, SQL Server creates an execution plan and stores it in the cache. In the subsequent executions of the stored procedure, SQL Server reuses the plan so that the stored procedure can execute very fast with reliable performance.

Here are sample stored procedures and all the rest of the stored procedures are included in the submission folder

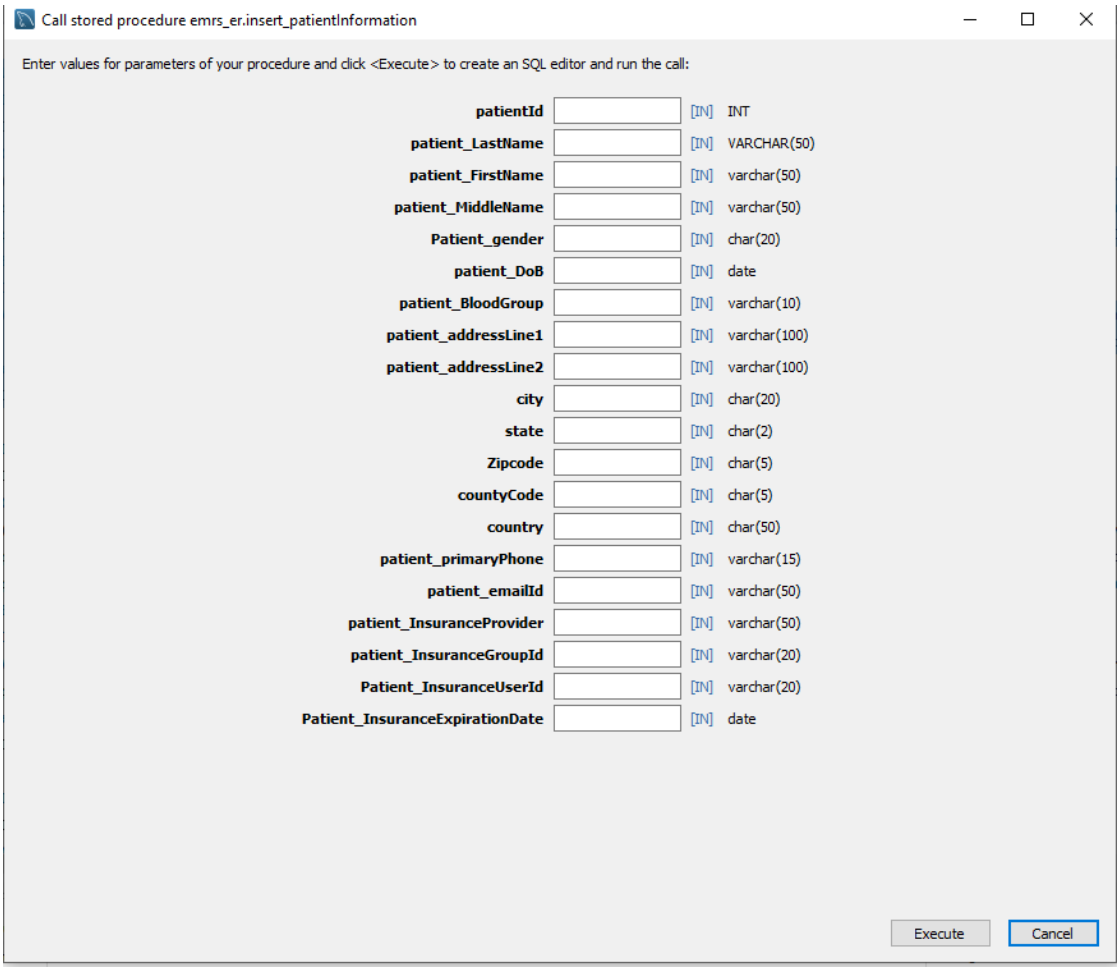
Scenario 1:

Stored Procedure that inserts the data into a table.

```
USE emrs_er;

DROP PROCEDURE IF EXISTS insert_patientInformation;

DELIMITER //

CREATE PROCEDURE insert_patientInformation

(patientId INT ,patient_LastName
VARCHAR(50),patient_FirstName
varchar(50),patient_MiddleName varchar(50),Patient_gender
char(20),patient_DoB date,patient_BloodGroup varchar(10),

patient_addressLine1 varchar(100),patient_addressLine2
varchar(100),city char(20),state char(2),Zipcode char(5),countyCode
char(5),country char(50),patient_primaryPhone varchar(15),

patient_emailId varchar(50),patient_InsuranceProvider
varchar(50),patient_InsuranceGroupId
varchar(20),Patient_InsuranceUserId
varchar(20),Patient_InsuranceExpirationDate date)

BEGIN

INSERT INTO patientInformation

VALUES (
patientId,patient_LastName,patient_FirstName,patient_MiddleNam
e,Patient_gender,patient_DoB,patient_BloodGroup,patient_address
Line1,patient_addressLine2,city,


state,Zipcode,countyCode,country,patient_primaryPhone,patient_e
mailId,patient_InsuranceProvider,patient_InsuranceGroupId,Patie
nt_InsuranceUserId,Patient_InsuranceExpirationDate);
```

```
end//
DELIMITER ;
```

Screenshot when the procedure is called via MySQL workbench API:



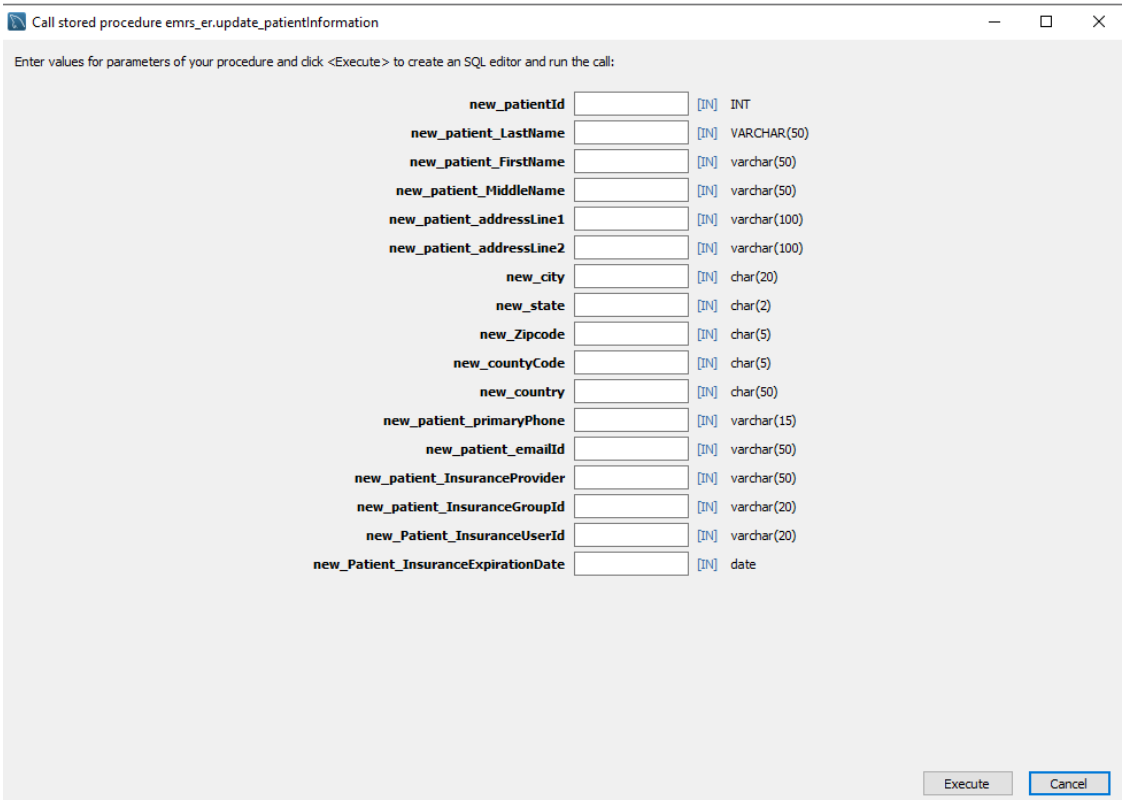
Scenario 2:

Stored Procedure that update the data in a table.

```
USE emrs_er;
DROP PROCEDURE IF EXISTS update_patientInformation;
DELIMITER //
CREATE PROCEDURE update_patientInformation(new_patientId INT
,new_patient_LastName VARCHAR(50),new_patient_FirstName
```

```
varchar(50),new_patient_MiddleName
varchar(50),new_patient_addressLine1 varchar(100),

new_patient_addressLine2 varchar(100),new_city char(20),new_state
char(2),new_Zipcode char(5),new_countyCode char(5),new_country
char(50),new_patient_primaryPhone varchar(15),

new_patient_emailId varchar(50),new_patient_InsuranceProvider
varchar(50),new_patient_InsuranceGroupId
varchar(20),new_Patient_InsuranceUserId
varchar(20),new_Patient_InsuranceExpirationDate date)

BEGIN

update patientInformation

set
patient_LastName=new_patient_lastname,patient_FirstName=new_p
atient_firstname,patient_MiddleName=new_patient_middlename,

patient_addressLine1=new_patient_addressline1,patient_addressLine
2=new_patient_addressline2,city=new_city,state=new_state,Zipcode
=new_zipcode,countyCode=new_countycode,

country=new_country,patient_primaryPhone=new_patient_primaryP
hone,patient_emailId=new_patient_emailId,patient_InsuranceProvid
er=new_patient_InsuranceProvider,

patient_InsuranceGroupId=new_patient_InsuranceGroupId,Patient_I
nsuranceUserId=new_Patient_InsuranceUserId,Patient_InsuranceExpi
rationDate=new_Patient_InsuranceExpirationDate

where patientid = new_patientid;

end//

DELIMITER ;
```

Screenshot when the procedure is called via MySQL workbench API:



Scenario 3:

Stored Procedure that deletes data in a table.

```
DROP PROCEDURE IF EXISTS delete_patientInformation;

DELIMITER //

CREATE PROCEDURE delete_patientInformation

(new_patientId INT)

BEGIN

delete from patientinformation where patientid= new_patientid;

end//

DELIMITER ;
```

Screenshot when the procedure is called via MySQL workbench API:

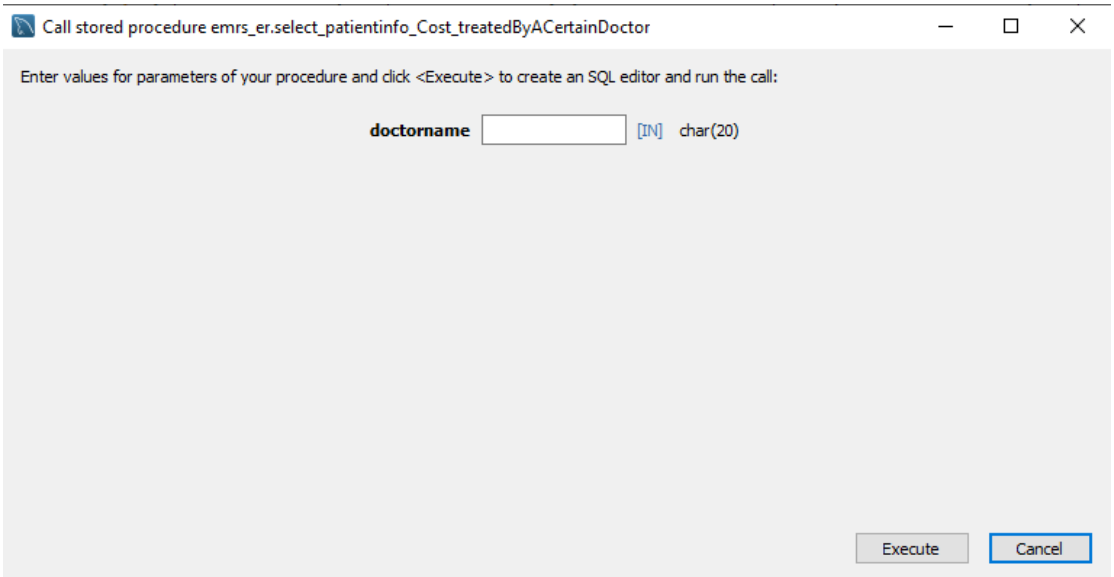
Scenario 4:

Stored Procedure that queries data from a table using select statement.

```
-- procedure that fetch patient information and cost of services who are treated by particular doctor

DROP PROCEDURE IF EXISTS select_patientinfo_Cost_treatedByACertainDoctor;

DELIMITER //

CREATE PROCEDURE select_patientinfo_Cost_treatedByACertainDoctor

(doctorname char(20))

BEGIN

select pi.patientid, concat(patient_Firstname,', ', patient_lastname) as patient_fullName, cpi.provider_fullname,(totalbill - amount_paid_by_InsuranceProvider) as patient_pays
```

```
from patientInformation as pi join billing on pi.patientid =
billing.patientid ,careproviderInformation as cpi

where pi.patientid in (select patient_id from patientAdmission as
pa,careproviderinformation as cpi  where
pa.provider_id=cpi.providerid and provider_fullname like
concat('%',doctorname,'%'))

group by patient_fullname order by patient_pays desc;

end//

DELIMITER ;
```

Screenshot when the procedure is called via MySQL workbench API:



8. User profiles and Access permissions

I created three user roles namely “nursingstaff, careprovider and supportstaff” and they have their own appropriate tables access along with stored procedures and triggers access once logged in using user name and user password.

After running the below query please proceed to root user and give permissions and privileges to the below users.

Step 1: please login to root user and go to management and click on "users and privileges"

Step 2: select each user role separately and proceed to "Administrative roles"

Step 3: select (delete, index, insert, select, show view, trigger and update) for each role then save and now login to respective users and execute the stored procedures.

SQL code:

```
Flush privileges;
-- drop user 'supportstaff'@'localhost';
-- drop user 'nursingstaff'@'localhost';
-- drop user 'careprovider'@'localhost';
create user 'supportstaff'@'localhost' identified by 'supportstaff';
create user 'nursingstaff'@'localhost' identified by 'nursingstaff';
create user 'careprovider'@'localhost' identified by 'careprovider';

grant all privileges on emrs_er.patientInformation to
'supportstaff'@'localhost';
grant all privileges on emrs_er.patientAdmission to
'supportstaff'@'localhost';
grant select on emrs_er.careproviderinformation to
'supportstaff'@'localhost';
```

grant update,select on emrs_er.discharge to 'supportstaff'@'localhost';

grant update,select on emrs_er.audit_discharge to 'supportstaff'@'localhost';

grant select on emrs_er.billing to 'supportstaff'@'localhost';

grant select on emrs_er.audit_patientInformation to 'supportstaff'@'localhost';

grant select on emrs_er.audit_careproviderinformation to 'supportstaff'@'localhost';

GRANT execute ON PROCEDURE emrs_er.insert_patientInformation TO 'supportstaff'@'localhost';

GRANT execute ON PROCEDURE emrs_er.update_patientInformation TO 'supportstaff'@'localhost';

GRANT execute ON PROCEDURE emrs_er.delete_patientInformation TO 'supportstaff'@'localhost';

GRANT execute ON PROCEDURE emrs_er.insert_patientadmission TO 'supportstaff'@'localhost';

GRANT execute ON PROCEDURE emrs_er.delete_patientadmission TO 'supportstaff'@'localhost';

GRANT execute ON PROCEDURE emrs_er.select_AdmittedPatients_requireUrgentCareRequired TO 'supportstaff'@'localhost';

grant execute on procedure emrs_er.select_patients_discharged_OnGivenDay to 'supportstaff'@'localhost';

```
grant execute on procedure
emrs_er.select_count_patients_admitted_on_a_given_Day to
'supportstaff'@'localhost';

grant execute on procedure
emrs_er.select_patients_admitted_between_two_Dates to
'supportstaff'@'localhost';


-- nursing staff grants


grant all privileges on emrs_er.discharge to
'nursingstaff'@'localhost';

grant all privileges on emrs_er.audit_discharge to
'nursingstaff'@'localhost';

grant select on emrs_er.patientinformation to
'nursingstaff'@'localhost';

grant select on emrs_er.audit_patientinformation to
'nursingstaff'@'localhost';

grant select on emrs_er.careproviderinformation to
'nursingstaff'@'localhost';

grant select on emrs_er.audit_careproviderinformation to
'nursingstaff'@'localhost';

grant update, select on emrs_er.patientvitals to
'nursingstaff'@'localhost';

grant select on emrs_er.medications_prescribed to
'nursingstaff'@'localhost';

grant select on emrs_er.audit_medications_prescribed to
'nursingstaff'@'localhost';
```

grant select on emrs_er.ordered_tests_procedures to 'nursingstaff'@'localhost';

grant select on emrs_er.audit_ordered_tests_procedures to 'nursingstaff'@'localhost';

GRANT execute ON PROCEDURE emrs_er.insert_discharge TO 'nursingstaff'@'localhost';

grant execute on procedure emrs_er.delete_discharge to 'nursingstaff'@'localhost';

grant execute on procedure emrs_er.insert_patientvitals to 'nursingstaff'@'localhost';

GRANT execute ON PROCEDURE emrs_er.select_concerning_patientvitals TO 'nursingstaff'@'localhost';

GRANT execute ON PROCEDURE emrs_er.select_AdmittedPatients_requireUrgentCareRequired TO 'nursingstaff'@'localhost';

grant execute on procedure emrs_er.select_careproviders_whoareactive_inactive to 'nursingstaff'@'localhost';

grant execute on procedure emrs_er.select_labresults_status to 'nursingstaff'@'localhost';

grant execute on procedure emrs_er.select_patientinfo_cost_treatedbyacertaindoctor to 'nursingstaff'@'localhost';

-- care provider grants

grant all privileges on emrs_er.ordered_tests_procedures to 'careprovider'@'localhost';

grant all privileges on emrs_er.audit_ordered_tests_procedures to 'careprovider'@'localhost';

grant select on emrs_er.patientinformation to 'careprovider'@'localhost';

grant select on emrs_er.audit_patientinformation to 'careprovider'@'localhost';

grant select on emrs_er.patientvitals to 'careprovider'@'localhost';

grant all privileges on emrs_er.medications_prescribed to 'careprovider'@'localhost';

grant all privileges on emrs_er.audit_medications_prescribed to 'careprovider'@'localhost';

grant select on emrs_er.labresults to 'careprovider'@'localhost';

GRANT execute ON PROCEDURE emrs_er.insert_ordered_tests_procedures TO 'careprovider'@'localhost';

GRANT execute ON PROCEDURE emrs_er.update_ordered_tests_procedures TO 'careprovider'@'localhost';

GRANT execute ON PROCEDURE emrs_er.delete_ordered_tests_procedures TO 'careprovider'@'localhost';

GRANT execute ON PROCEDURE emrs_er.insert_medications_prescribed TO 'careprovider'@'localhost';

GRANT execute ON PROCEDURE emrs_er.update_medications_prescribed TO 'careprovider'@'localhost';

GRANT execute ON PROCEDURE emrs_er.delete_medications_prescribed TO 'careprovider'@'localhost';

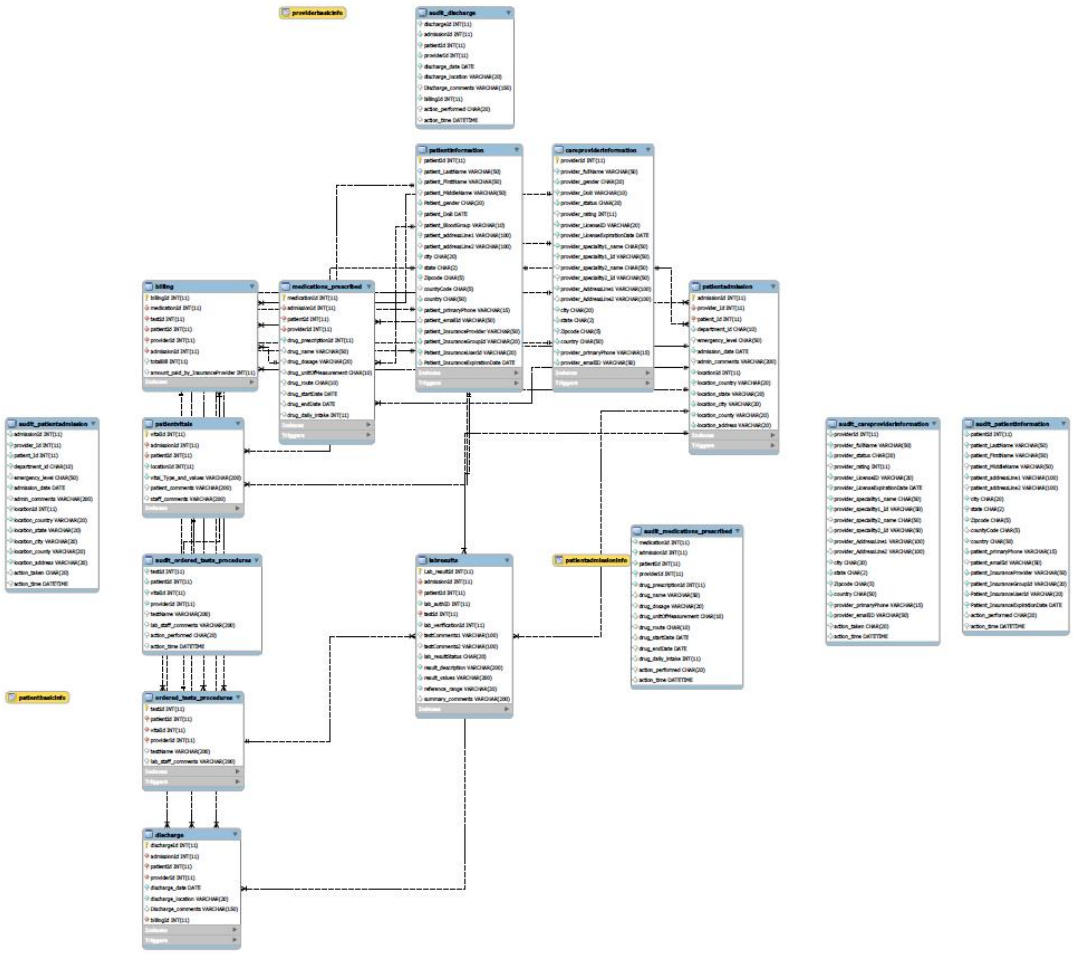
GRANT execute ON PROCEDURE emrs_er.select_AdmittedPatients_requireUrgentCareRequired TO 'careprovider'@'localhost';

grant execute on procedure emrs_er.select_medicationsprescribed to 'careprovider'@'localhost';

grant execute on procedure emrs_er.select_careproviders_whoareactive_inactive to 'careprovider'@'localhost';

grant execute on procedure emrs_er.select_concerning_patientvitals to 'careprovider'@'localhost';

9. ER diagram generated by MySQL workbench



ER diagram as PDF has been saved and can be found in the project submission folder with name "ER Model"

10. References

1) Murach's MySQL version 2

2) W3 schools-SQL

3) CodeProject.com/articles/Overview-of-SQL