

APPLIED DATABASES

Hospital Management

(An Electronic Medical Record for an Outpatient Clinic)

TEAM MEMBERS

Praneeth Bomma

Alekhya Akkinapally

Venkata Sai Vamsi Kishna Gollapudi

CONTENTS

1. Description of the Project
2. Objectives of the system
3. Scope
4. Use Cases
5. Database Design
 - UML Diagram
 - ER Diagram
 - Tables
 - Store Procedures
 - User Authentication
 - Views
 - Triggers
 - Indexes
6. Conclusion
7. Future Enhancements
8. References

1. Description of the Project :

Hospitals are the essential part of our lives, providing best medical facilities to people suffering from various ailments, which may be due to change in climatic conditions, increased workload, emotional trauma stress etc. It is necessary for the hospitals to keep track of its day-to-day activities & records of its patients, doctors, nurses, billing and payments that keep the hospital running smoothly and successfully.

But keeping track of all the activities and their records on paper is very cumbersome and error prone. It also is very inefficient and a time-consuming process. Observing the continuous increase in population and number of people visiting the hospital. Recording and maintaining all these records is highly unreliable, inefficient and error-prone. It is also not economically & technically feasible to maintain these records on paper.

Thus keeping the working of the manual system as the basis of our project. We have developed an automated version of the manual system, named as “Hospital Management System”.

The main aim of our project is to provide a paper-less hospital up to 90%. It also aims at providing low-cost reliable automation of the existing systems. The system also provides excellent security of data at every level of user-system interaction and also provides robust & reliable storage and backup facilities.

2.Objectives of the system :

The project “Hospital management system” is aimed to develop to maintain the day-to-day state of admission/discharge of patients, list of doctors, reports generation, and etc. It is designed to achieve the following objectives:

- To computerize all details regarding patient details and hospital details.
- Scheduling the appointment of patient with doctors to make it convenient for both.
- The information of the patients should be kept up to date and their record should be kept in the system.
- Scheduling the services of specialized doctors properly so that the facilities provided by hospital are fully utilized effectively and efficiently.
- The information of the patients should be kept up to date and their record should be kept in the system for historical purposes.
- The hospital will send the reminder to the patients a day before the appointment and also if they are due for their flu shot or annual health care.
- The hospital maintains a relationship with the insurance company which will generate the bill after computerizing the payments made by the insurance.

3.Scope:

The proposed software product is the Hospital Management System (HMS). The system will be used in any Hospital, Clinic to get the information from the patients and then storing that data for future usage.

The current system in use is a paper-based system. It is too slow and cannot provide updated lists of patients within a reasonable timeframe. The intentions of the system are to reduce overtime pay and increase the number of patients that can be treated accurately. Requirements statements in this document are both functional and nonfunctional. E-R Diagram.

Here we have three potential users:

1. Patients:

This user here is a patient who will use the application to get the information about the doctors in the hospital. When the patient wants to visit a doctor for some reason they can select a doctor based on the specialization or the available time. The patients can also edit their information like if their address has been changed or if their phone number has been changed.

2. Doctors:

The doctor is responsible to view the details of the patients, their allergies or the medical conditions and also the advices or the medication given by the nurse. The doctor can edit his details such as address, phone number and also there is a special attribute in the doctor's profile that if the doctors accepts the new patients or not. The doctor can edit that if he does not want to see new patients.

3. Nurse:

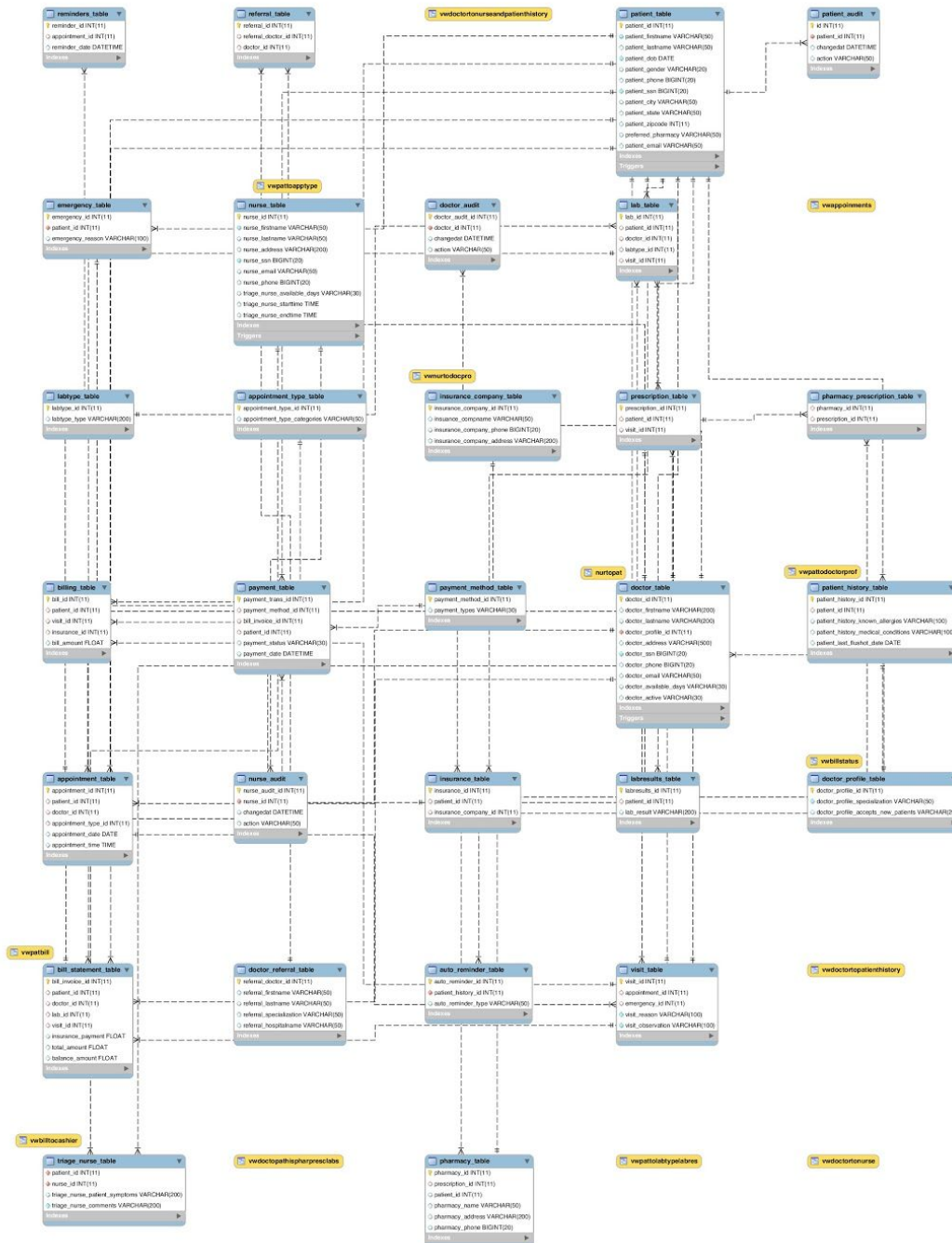
The nurse is responsible for the maintenance of the information displayed in the tables. The main functionalities of the admin include add and remove the record of any tables. For this to be done, the admin must log in into the application with valid credentials.

4.Use Cases:

The considered use cases are:

1. View of all the doctor details by the patient.
2. Taking an appointment on a specific day and on specific time.

ER Diagram:



Tables:

patient_table, patient_history_table, nurse_table, triage_nurse_table, doctor_profile_table, doctor_table, appointment_type_table, appointment_table, auto_reminder_table, emergency_table, reminders_table, doctor_referral_table, referral_table, visit_table, insurance_company_table, insurance_table, billing_table, labtype_table, lab_table, labresults_table, prescription_table, pharmacy_table, pharmacy_prescription_table, payment_method_table, payment_table.

Store Procedures:

- Stored procedure to insert new patient details

The screenshot displays a SQL IDE interface with a script editor and a results grid. The script editor shows the following SQL code:

```
360 delimiter $$
361 • create procedure spPatIns
362 (in ins_patient_firstname varchar(50) , in ins_patient_lastname varchar(50), in ins_patient_dob date ,
363 in ins_patient_gender varchar(20), in ins_patient_phone bigint , in ins_patient_email varchar(50), in ins_patient
364 in ins_patient_city varchar(50), in ins_patient_state varchar(50) , in ins_patient_zipcode int,
365 in ins_preferred_pharmacy varchar(50) )
366 begin
367 insert into
368 hospital_management.patient_table (patient_firstname,patient_lastname,patient_dob,patient_gender,patient_phone,pa
369 patient_ssn,patient_city,patient_state,patient_zipcode,preferred_pharmacy)
370 values(ins_patient_firstname,ins_patient_lastname,ins_patient_dob,ins_patient_gender,ins_patient_phone,
371 ins_patient_email,ins_patient_ssn,ins_patient_city,ins_patient_state,ins_patient_zipcode,ins_preferred_pharmacy);
372
373 select * from patient_table where patient_table.patient_ssn = ins_patient_ssn;
374 SELECT * FROM patient_audit ;
```

The results grid shows the following data:

id	patient_id	changedat	action
1	1016	2017-05-09 17:37:51	Insert
2	1015	2017-05-09 17:38:11	update
3	1016	2017-05-09 17:38:11	update
4	1017	2017-05-09 17:42:31	Insert
5	1021	2017-05-09 18:15:58	Insert

- Stored procedure for an existing user to schedule an appointment by selecting the doctor according to the doctor's specialization

```

386 delimiter $$
387 create procedure spPatApp
388 (
389     in vis_patient_id int ,
390     in vis_doctor_profile_specialization varchar(50) ,
391     in vis_appointment_type_categories varchar(50),
392     in vis_appointment_date date,
393     in vis_appointment_time time
394 )
395 begin
396     declare app_type_id int;
397     declare app_id int;
398     declare d_id int;
399     declare vis_app_id int;
400
401     set d_id=(select doctor_table.doctor_id from doctor_table , doctor_profile_table
402     where doctor_profile_specialization = vis_doctor_profile_specialization
403     and doctor_table.doctor_profile_id=doctor_profile_table.doctor_profile_id);
404
405     set app_type_id = (select appointment_type_id from appointment_type_table where
406     appointment_type_categories = vis_appointment_type_categories);
407
408     insert into appointment table(patient id,doctor id,appointment type id,appointment date,appointment time)
409
100% 1:422

```

appointment_id	patient_id	doctor_id	appointment_type_id	appointment_date	appointment_time
2012	1009	8001	9001	2017-04-13	16:00:00

- Stored procedure where doctor would update the details of the patients who has visited.

```

426 delimiter $$
427 create procedure spPatVis
428 (
429     in vis_appointment_id int ,
430     in vis_emergency_id int,
431     in visit_reason varchar(100),
432     in visit_observation varchar(100)
433 )
434 begin
435     declare vis_it int;
436
437     insert into hospital_management.visit_table(appointment_id,emergency_id,visit_reason,visit_observation)
438     values (vis_appointment_id,vis_emergency_id,visit_reason,visit_observation);
439
440     set vis_it = (select visit_id from visit_table where visit_table.appointment_id=vis_appointment_id order by visit_i
441
442     select * from visit_table where visit_id=vis_it;
443
444 end $$
445 delimiter ;
446
447 call spPatVis( 2011 , null,'fever','medication for a week');
448
449
100% 1:449

```

visit_id	appointment_id	emergency_id	visit_reason	visit_observation
17015	2011	NULL	fever	medication for a week

User Authentication:

- Doctors Authentication:

Created the doctors authentication where only the doctors could login and view the details of the patients and the nurses.

- Hospital management admin:

This authenticator will have access to all the tables and view

- Nurses Authentication:

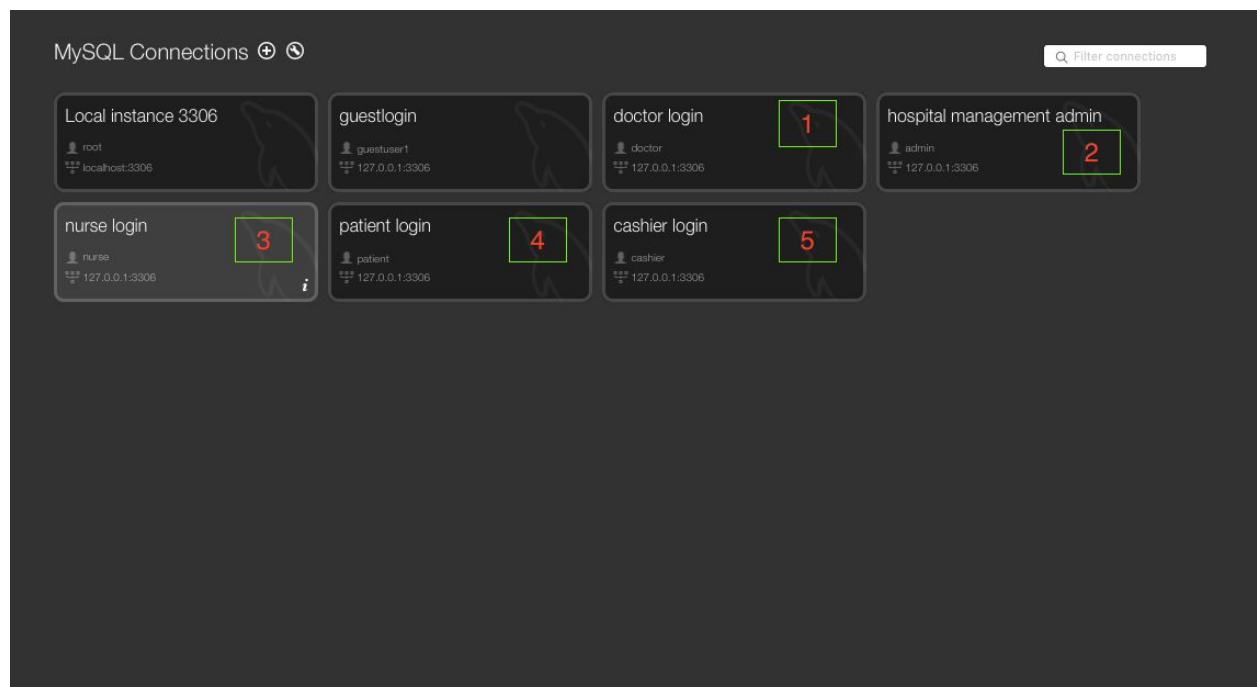
Created separate authentication for the nurse where she can edit the details of either the doctors or the patient's. This is also protected by the password which only the nurse can use.

- Patients Authentication:

Created the patient authentication where patients can only view or edit the details that they are allowed to, the things like changing their address or just viewing the details of the doctor. We have restricted the actions by giving grant.

- Cashier authentication:

This authenticator will have access to only the payment tables and billing tables and will not have access to any of the tables like the doctor or the nurse.



Views:

- Doctor to Patient:

We have created this view so that the doctor can view all the patients details at one click, so that it would save him a lot of time. Instead of writing a separate queries for each and every table, we have joined queries and wrote it in a single view.

The screenshot shows a SQL IDE with multiple tabs. The active tab is 'SQL File 22*', which contains the following SQL code:

```
1 /*patient view for doctor*/
2
3 create view vwdoctortopatienhistory
4 as
5 select patient_table.patient_id,
6 patient_firstname,patient_lastname,patient_dob,patient_gender,
7 patient_history_known_allergies,patient_history_medical_conditions,patient_last_flushot_date
8 from
9 patient_table join patient_history_table where patient_table.patient_id=patient_history_table.patient_id;
10
11 /*patient view to doctor*/
12
13 select * from vwdoctortopatienhistory;
```

Below the code editor is the 'Result Grid' showing 14 rows of patient data. The columns are: patient_id, patient_firstname, patient_lastname, patient_dob, patient_gender, patient_history_known_allergi..., patient_history_medical_conditi..., and patient_last_flushot_d....

patient_id	patient_firstname	patient_lastname	patient_dob	patient_gender	patient_history_known_allergi...	patient_history_medical_conditi...	patient_last_flushot_d...
1001	Smith	John	1974-06-23	M	null	thyroid	2016-04-22
1002	John	Wick	1980-02-27	M	pet allergy	high blood pressure	2015-10-02
1003	Jacob	Tyler	1960-06-14	M	food allergy	diabetes	2016-08-22
1004	Michael	James	1985-08-22	M	null	low blood pressure	2016-04-22
1005	Joshua	John	1982-05-12	M	null	asthma	2016-08-13
1006	Mathew	Jonathan	1970-09-30	M	skin allergy	depression	2016-10-19
1007	Ethan	Nathan	1976-08-03	M	Latex allergy	migraine	2016-11-10
1008	Emma	Mia	2000-07-23	F	mold allergy	null	2016-11-06
1009	Emily	Alisa	2005-06-15	F	null	sinus	2016-11-30
1010	Anthony	Noah	1990-07-17	M	eye allergy	null	2016-05-23
1011	Mani	Sharma	2016-06-17	M	accident	migraine	2016-08-19
1012	Jai	Wick	1980-02-27	M	food allergy	migraine	2016-10-01
1013	Jack	Tyler	1960-06-14	M	severe allergy	migraine	2016-01-11
1014	Mich	James	1985-08-22	M	high temperature	migraine	2016-12-10

- Doctor to Nurse and patient:

This is the view where the doctor can view the nurse and also the patient details, where he can view what advices did the nurse give to the patient and also he can check what medical problems or what allergies so the patients have.

The screenshot shows a SQL IDE with a view definition and its result grid. The view definition is as follows:

```

1 create view vwdoctortonurseandpatienthistory
2 as
3 select patient_history_table.patient_id,patient_firstname,patient_lastname,triage_nurse_table.nurse_id,nurse_firstname,
4 triage_nurse_patient_symptoms,triage_nurse_comments
5 from
6 patient_history_table,triage_nurse_table,patient_table,nurse_table where patient_history_table.patient_id=triage_nurse_table.patient_id
7 patient_history_table.patient_id = patient_table.patient_id and nurse_table.nurse_id=triage_nurse_table.nurse_id;
8
9 /*view for doctor to patient_history and nurse */
10
11 select * from vwdoctortonurseandpatienthistory;
12

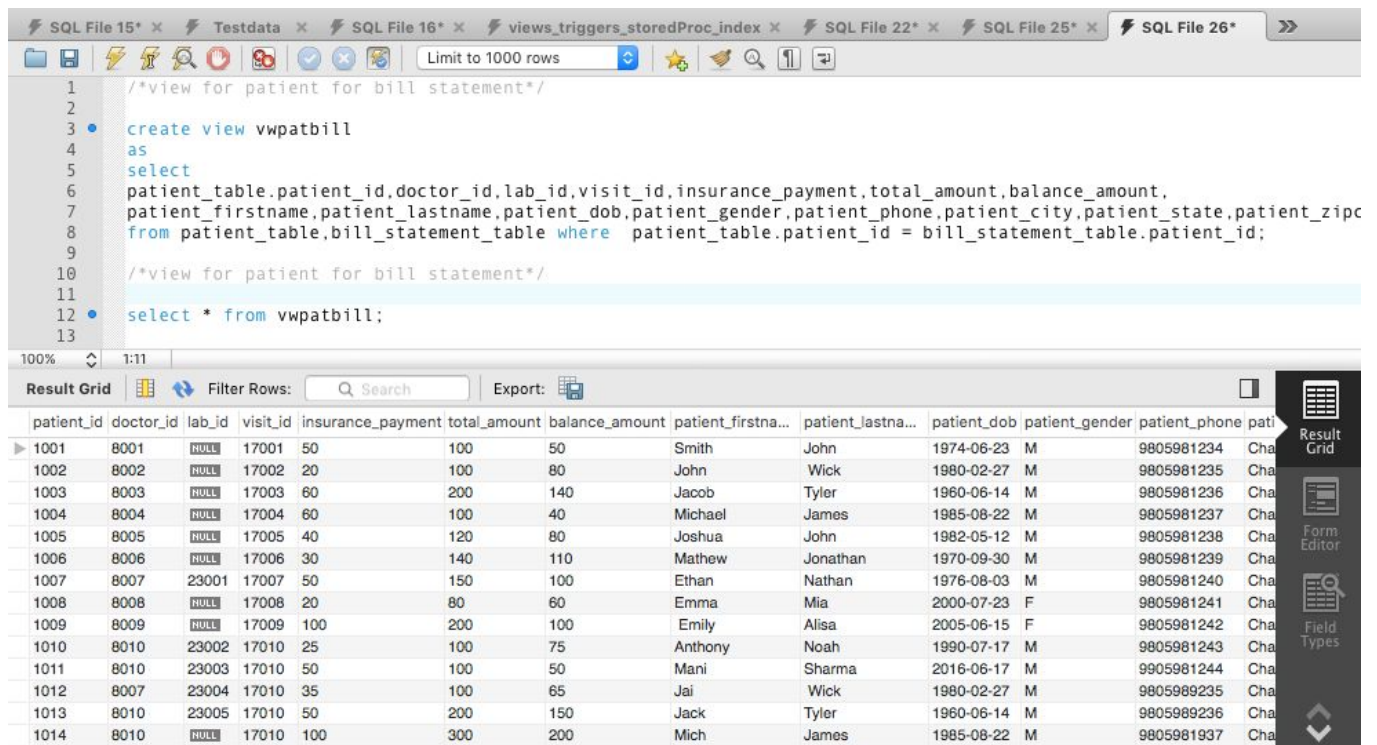
```

The result grid shows the following data:

patient_id	patient_firstname	patient_lastname	nurse_id	nurse_firstname	nurse_lastname	triage_nurse_available_days	triage_nurse_patient_symptoms	triage_nurse_comments
1001	Smith	John	5001	Mary	Smith	M	thyroid	visit doctor
1010	Anthony	Noah	5004	Mitchel	James	Sa	eye burning	use eye drops and
1008	Emma	Mia	5003	Honey	Tyler	F	mold allergy	visit doctor
1002	John	Wick	5010	Pamela	Rao	S	scratches of pet	visit doctor
1007	Ethan	Nathan	5009	Mounika	Reddy	S	Latex due to excessive use of gloves	avoid bananas, che
1006	Mathew	Jonathan	5008	Priyanka	Mia	S	rashes on the back	visit doctor
1003	Jacob	Tyler	5006	Roja	Jonathan	R	diarrhea, indigestion, vomiting	visit doctor
1009	Emily	Alisa	5005	Malini	John	M	sinus	avoid icecreams, co
1005	Joshua	John	5002	John	Rick	T	asthma	do yoga and visit d
1004	Michael	James	5007	Rajini	Nathan	W	low blood pressure	visit doctor

- Patient to billing view:

Here patient can view his bill where he can see how much it cost for the visits and how much the hospital has charged for the labs and how much did his insurance pay and what is the amount that he has to finally pay which will make the task easy instead of writing a query separately to check the patient's details, then check if he has undergone any lab and then check the billing table.



The screenshot shows a SQL IDE with a query editor at the top and a result grid at the bottom. The query editor contains the following SQL code:

```

1  /*view for patient for bill statement*/
2
3  • create view vwpatbill
4  as
5  select
6  patient_table.patient_id,doctor_id,lab_id,visit_id,insurance_payment,total_amount,balance_amount,
7  patient_firstname,patient_lastname,patient_dob,patient_gender,patient_phone,patient_city,patient_state,patient_zipc
8  from patient_table,bill_statement_table where patient_table.patient_id = bill_statement_table.patient_id;
9
10 /*view for patient for bill statement*/
11
12 • select * from vwpatbill;
13

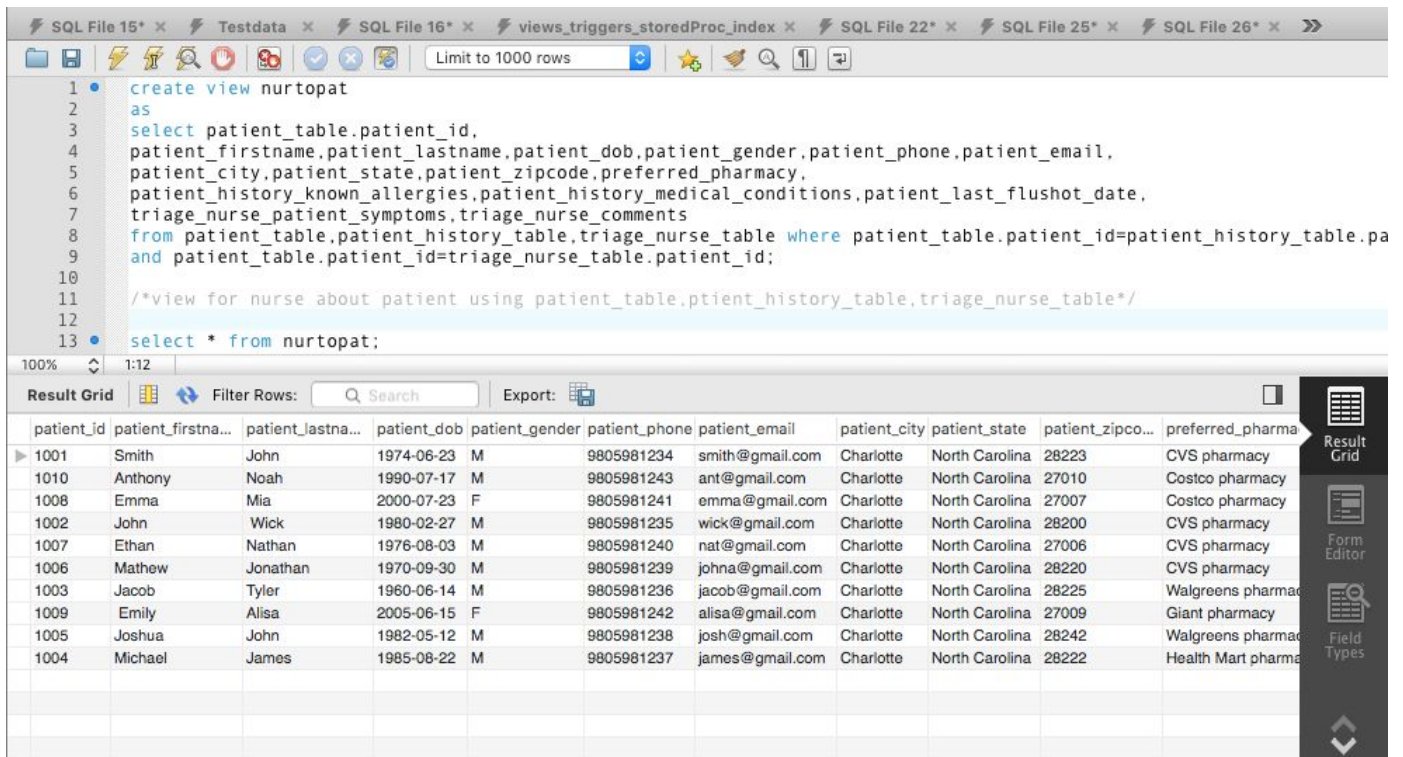
```

The result grid displays 14 rows of data. The columns are: patient_id, doctor_id, lab_id, visit_id, insurance_payment, total_amount, balance_amount, patient_firstname, patient_lastname, patient_dob, patient_gender, patient_phone, and patient_city. The data is as follows:

patient_id	doctor_id	lab_id	visit_id	insurance_payment	total_amount	balance_amount	patient_firstname	patient_lastname	patient_dob	patient_gender	patient_phone	patient_city
1001	8001	NULL	17001	50	100	50	Smith	John	1974-06-23	M	9805981234	Cha
1002	8002	NULL	17002	20	100	80	John	Wick	1980-02-27	M	9805981235	Cha
1003	8003	NULL	17003	60	200	140	Jacob	Tyler	1960-06-14	M	9805981236	Cha
1004	8004	NULL	17004	60	100	40	Michael	James	1985-08-22	M	9805981237	Cha
1005	8005	NULL	17005	40	120	80	Joshua	John	1982-05-12	M	9805981238	Cha
1006	8006	NULL	17006	30	140	110	Mathew	Jonathan	1970-09-30	M	9805981239	Cha
1007	8007	23001	17007	50	150	100	Ethan	Nathan	1976-08-03	M	9805981240	Cha
1008	8008	NULL	17008	20	80	60	Emma	Mia	2000-07-23	F	9805981241	Cha
1009	8009	NULL	17009	100	200	100	Emily	Alisa	2005-06-15	F	9805981242	Cha
1010	8010	23002	17010	25	100	75	Anthony	Noah	1990-07-17	M	9805981243	Cha
1011	8010	23003	17010	50	100	50	Mani	Sharma	2016-06-17	M	9905981244	Cha
1012	8007	23004	17010	35	100	65	Jai	Wick	1980-02-27	M	9805989235	Cha
1013	8010	23005	17010	50	200	150	Jack	Tyler	1960-06-14	M	9805989236	Cha
1014	8010	NULL	17010	100	300	200	Mich	James	1985-08-22	M	9805981937	Cha

- Nurse to patients view:

Here the nurse will have access to the patient details and also the patient history based, so that when the triage nurse is suggesting a medication for the patient she should about the allergies and the medical conditions that the patient already has.



The screenshot shows a SQL IDE with a script editor and a result grid. The script editor contains the following SQL code:

```

1 • create view nurtopat
2 as
3 select patient_table.patient_id,
4 patient_firstname,patient_lastname,patient_dob,patient_gender,patient_phone,patient_email,
5 patient_city,patient_state,patient_zipcode,preferred_pharmacy,
6 patient_history_known_allergies,patient_history_medical_conditions,patient_last_flushot_date,
7 triage_nurse_patient_symptoms,triage_nurse_comments
8 from patient_table,patient_history_table,triage_nurse_table where patient_table.patient_id=patient_history_table.pa
9 and patient_table.patient_id=triage_nurse_table.patient_id;
10
11 /*view for nurse about patient using patient_table,ptient_history_table,triage_nurse_table*/
12
13 • select * from nurtopat;
  
```

The result grid displays the following data:

patient_id	patient_firstname	patient_lastname	patient_dob	patient_gender	patient_phone	patient_email	patient_city	patient_state	patient_zipcode	preferred_pharmacy
1001	Smith	John	1974-06-23	M	9805981234	smith@gmail.com	Charlotte	North Carolina	28223	CVS pharmacy
1010	Anthony	Noah	1990-07-17	M	9805981243	ant@gmail.com	Charlotte	North Carolina	27010	Costco pharmacy
1008	Emma	Mia	2000-07-23	F	9805981241	emma@gmail.com	Charlotte	North Carolina	27007	Costco pharmacy
1002	John	Wick	1980-02-27	M	9805981235	wick@gmail.com	Charlotte	North Carolina	28200	CVS pharmacy
1007	Ethan	Nathan	1976-08-03	M	9805981240	nat@gmail.com	Charlotte	North Carolina	27006	CVS pharmacy
1006	Mathew	Jonathan	1970-09-30	M	9805981239	johna@gmail.com	Charlotte	North Carolina	28220	CVS pharmacy
1003	Jacob	Tyler	1960-06-14	M	9805981236	jacob@gmail.com	Charlotte	North Carolina	28225	Walgreens pharmacy
1009	Emily	Alisa	2005-06-15	F	9805981242	alisa@gmail.com	Charlotte	North Carolina	27009	Giant pharmacy
1005	Joshua	John	1982-05-12	M	9805981238	josh@gmail.com	Charlotte	North Carolina	28242	Walgreens pharmacy
1004	Michael	James	1985-08-22	M	9805981237	james@gmail.com	Charlotte	North Carolina	28222	Health Mart pharmacy

Triggers:

We have created triggers to get notified with the changes that are made to the data. So we have used audit tables for the patient, doctor and nurse where if a new patient is inserted in the patient table so that should get reflected in the patient audit table so that we can keep a record of all the modifications that are done to the data.

- Audit table for patient table:

We have added two triggers on insert and updates which generates row inside audit table for every respective operation.

SQL File 22* x

SQL File 26* x

SQL File 27* x

views_triggers_storedProc_index x

views_triggers_storedProc_index (2) x

SQL File 13*

Limit to 1000 rows

```

1  /*creating a trigger for inserting into patient table*/
2
3  DELIMITER $$
4  CREATE TRIGGER after_patient_insert
5    after insert ON patient_table
6    FOR EACH ROW
7  BEGIN
8    INSERT INTO patient_audit
9    SET action = 'Insert',
10     patient_id = new.patient_id,
11     changedat = NOW();
12  END$$
13  DELIMITER ;
14  /*inserting values into patient_table*/
15
16  Insert into patient_table(patient_firstname,patient_lastname,patient_dob,patient_gender,patient_phone,patient_email
17  values('Hemanth','Patel','1982-08-22','M','9805981000','jam@gmail.com','123456001','Charlotte','North Carolina','28922
18
19  /*Check whether the changes are reflected in the patient_audit table;*/
20  SELECT * FROM patient_audit;

```

100%

1:20

Result Grid

Filter Rows:

Q Search

Edit:

Export/Import:

id	patient_id	changedat	action
1	1016	2017-05-08 23:21:35	Insert
NULL	NULL	NULL	NULL

Result Grid

Form Editor

- Audit table for nurse table:

We have used this audit table for nurse so that we can keep track if some data from the nurse has been updates or deleted or inserted.

The screenshot shows a SQL IDE with multiple tabs. The active tab displays the following SQL code:

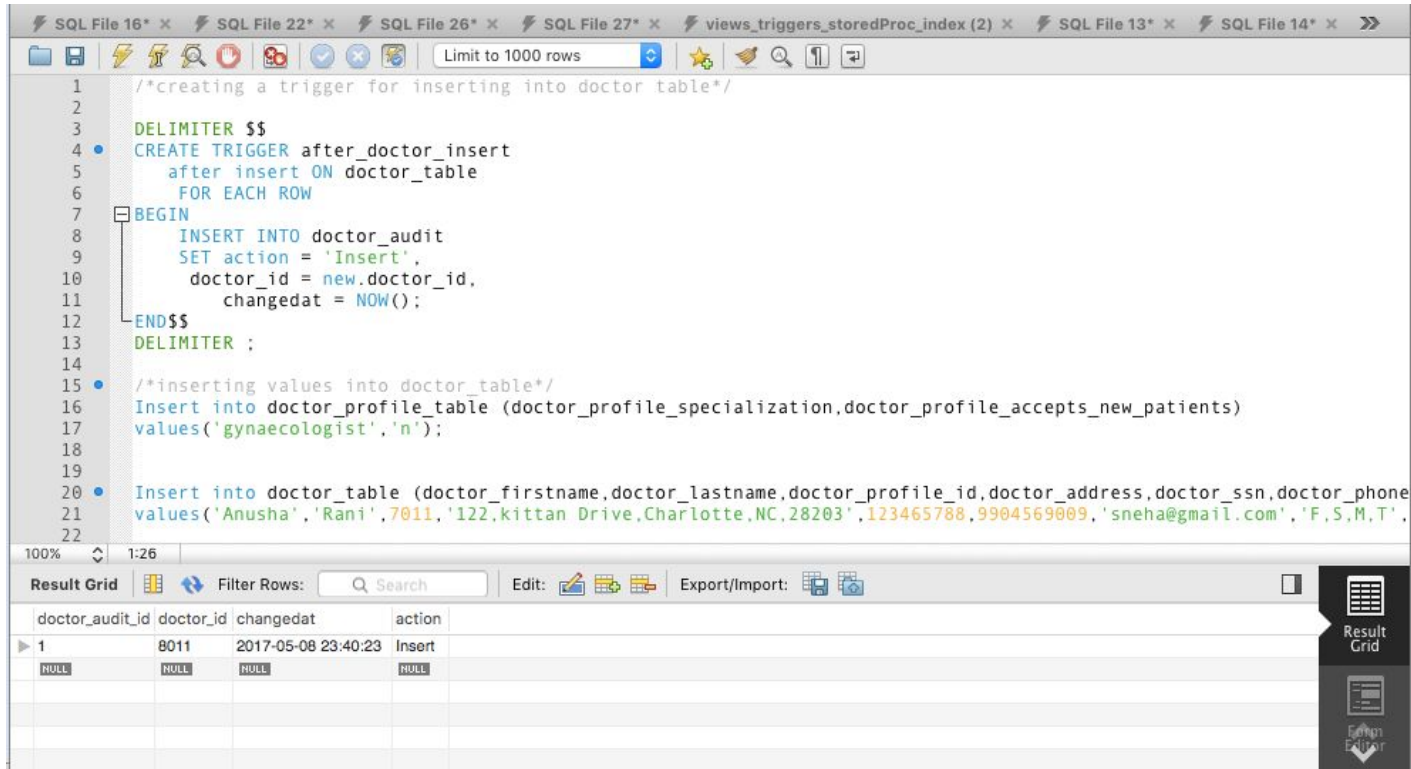
```
25 DELIMITER $$
26 • CREATE TRIGGER before_nurse_update
27   BEFORE UPDATE ON nurse_table
28   FOR EACH ROW
29 BEGIN
30   INSERT INTO nurse_audit
31   SET action = 'update',
32   nurse_id = OLD.nurse_id,
33   changedat = NOW();
34 END$$
35 DELIMITER ;
36
37
38
39 • /*updating values from nurse_table */
40 UPDATE nurse_table
41 SET
42   nurse_lastname = 'Reddy'
43 WHERE
44   nurse_firstname = 'Rajeev';
45
46 • SELECT * FROM nurse_audit;
47
```

Below the code editor, the 'Result Grid' is visible, showing the output of the SQL execution. The grid has four columns: nurse_audit_id, nurse_id, changedat, and action. It contains two rows of data.

nurse_audit_id	nurse_id	changedat	action
1	5011	2017-05-08 23:36:37	Insert
2	5011	2017-05-08 23:36:50	update
NULL	NULL	NULL	NULL

- Audit table for Doctors table:

When we have tried updating the details in the doctor's table it has been notified in the audit table.



The screenshot shows a SQL IDE with a script editor and a result grid. The script editor contains the following SQL code:

```

1  /*creating a trigger for inserting into doctor table*/
2
3  DELIMITER $$
4  • CREATE TRIGGER after_doctor_insert
5    after insert ON doctor_table
6    FOR EACH ROW
7  BEGIN
8    INSERT INTO doctor_audit
9    SET action = 'Insert',
10      doctor_id = new.doctor_id,
11      changedat = NOW();
12  END$$
13  DELIMITER ;
14
15  • /*inserting values into doctor table*/
16  Insert into doctor_profile_table (doctor_profile_specialization,doctor_profile_accepts_new_patients)
17  values('gynaecologist','n');
18
19
20  • Insert into doctor_table (doctor_firstname,doctor_lastname,doctor_profile_id,doctor_address,doctor_ssn,doctor_phone
21  values('Anusha','Rani',7011,'122,kittan Drive,Charlotte,NC,28203',123465788,9904569009,'sneha@gmail.com','F,S,M,T',
22

```

The result grid shows the following data:

doctor_audit_id	doctor_id	changedat	action
1	8011	2017-05-08 23:40:23	Insert
NULL	NULL	NULL	NULL

Indexes:

We have also added index on patients table, doctor table and appointment table to enhance the performance of the search query. Below are the screenshots where we have implemented the search query without index and with index and noticed the difference.

The first screenshot shows the execution statistics for the query `select * from doctor_table where doctor_firstname= 'Grace';` without an index. The statistics indicate a full table scan and no index usage.

Category	Value
Timing (as measured at client side):	Execution time: 0:00:0.00065613
Timing (as measured by the server):	Execution time: 0:00:0.00048977 Table lock wait time: 0:00:0.00024600
Errors:	Had Errors: NO Warnings: 0
Rows Processed:	Rows affected: 0 Rows sent to client: 1 Rows examined: 11
Temporary Tables:	Temporary disk tables created: 0 Temporary tables created: 0
Joins per Type:	Full table scans (Select_scan): 1 Joins using table scans (Select_full_join): 0 Joins using range search (Select_full_range_join): 0 Joins with range checks (Select_range_check): 0 Joins using range (Select_range): 0
Sorting:	Sorted rows (Sort_rows): 0 Sort merge passes (Sort_merge_passes): 0 Sorts with ranges (Sort_range): 0 Sorts with table scans (Sort_scan): 0
Index Usage:	No Index used
Other Info:	Event Id: 397 Thread Id: 29

The second screenshot shows the execution statistics for the same query after creating an index `create index ix_doctor_firstname on doctor_table (doctor_firstname);`. The statistics indicate that the index was used, significantly improving performance.

Category	Value
Timing (as measured at client side):	Execution time: 0:00:0.00050688
Timing (as measured by the server):	Execution time: 0:00:0.00035286 Table lock wait time: 0:00:0.00010300
Errors:	Had Errors: NO Warnings: 0
Rows Processed:	Rows affected: 0 Rows sent to client: 1 Rows examined: 1
Temporary Tables:	Temporary disk tables created: 0 Temporary tables created: 0
Joins per Type:	Full table scans (Select_scan): 0 Joins using table scans (Select_full_join): 0 Joins using range search (Select_full_range_join): 0 Joins with range checks (Select_range_check): 0 Joins using range (Select_range): 0
Sorting:	Sorted rows (Sort_rows): 0 Sort merge passes (Sort_merge_passes): 0 Sorts with ranges (Sort_range): 0 Sorts with table scans (Sort_scan): 0
Index Usage:	At least one Index was used
Other Info:	Event Id: 401 Thread Id: 29

6.CONCLUSION:

The project Hospital Management System (HMS) is for computerizing the working in a hospital. It is a great improvement over the manual system. The computerization of the system has speed up the process. In the current system, the front office managing is very slow. The hospital managing system was thoroughly checked and tested with dummy data and thus is found to be very reliable. The software takes care of all the requirements of an average hospital and is capable to provide easy and effective storage of information related to patients that come up to the hospital. It generates test reports and also provides the facility for viewing the details of the patient's history by a doctor for prescribing any medicine or giving any medication. Patient can also view the doctor's profile and also his available days and time so that the patient can take an appointment. It also provides billing facility.

7.Future Enhancements:

- The proposed system is Hospital Management System. We can enhance this system by including more facilities like pharmacy system for the stock details of medicines in the pharmacy.
- Providing such features enable the users to include more comments into the system.
- We could include inpatient facility to our system.
- We will try and add more employees and manage their salaries.

8.References:

- <https://www.novanthealth.org/home/patients--visitors/locations.aspx>
- PragimTechvideos: <https://www.youtube.com/channel/UCCTVrRB5KpliK6V2GGVsR1Q>
- Lagunita Videos: <https://lagunita.stanford.edu/dashboard>