DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

<PES2201800053> **<AYUSHI AGARWAL>**

HOSPITAL MANAGEMENT SYSTEM

Hospital management system manages a database containing data about all the Hospitals in a city, in a particular area, treatments offered at the hospital, all doctors in a city, in a particular area, their specializations, experience, ratings, affiliations with various hospitals, availability of doctors at a hospital, book appointments. This database can be used by the user to find a hospital or clinic that best fits the user's requirements and affordability. The user can compare fees charged by various hospitals and doctors.

The system also deals with complex queries so that the database can be used in cases of emergency in order to get best and nearest doctors, based on the current availability of the respective doctor. Moreover, the user can request an appointment online by looking at the available timings of the doctor. Hence, the user will be able to apply all the filters which are possible and fulfils his/her requirements and sort them in ascending and descending order according to his/her convenience.

Transactions like update and delete are also accommodated so that the doctors who reach their retirement age are automatically deleted from the database, making the database efficient. Trigger is activated automatically for the doctors who reach their retirement age.

INTRODUCTION

This project deals with the management of hospital database efficiently and effectively. This database ensures user to find the hospital that best fits their requirements and affordability.

MINI WORLD ENTITIES:

1) **Doctor**: This holds the details of different doctor available in the hospital based on their qualification, experience, and rating.
2) **Hospital**: These are the different hospitals available having name, id, address, contact no., type (govt. or private), open and close timings.
3) **Hospital area**: It consists of area and location link to different hospitals.
4) **Doctor specialization**: The different doctor specialized in various fields.
5) **Doctor treatment**: Associated with the doctors in each hospital giving patient the information about the expense level based on the different treatments available.
6) **Hospital treatment**: Different treatments and expense level in each hospital.
7) **Appointment request**: Appointment made by the patient by setting time
8) **Hospital doctors**: Associated with the doctors available in a given hospital at a given point of time.

RELATIONS

- Hospital - hospital_area → 1:n
- Hospital - appointment_request → 1:n
- Hospital - doctor_treatment → 1:1
- Doctor_treatment – doctor → n:1
- Doctor – doctor_specialization → 1:n
- Doctor – hospital_treatment → 1:1
- Appointment_request – hospital_doctors → n:1

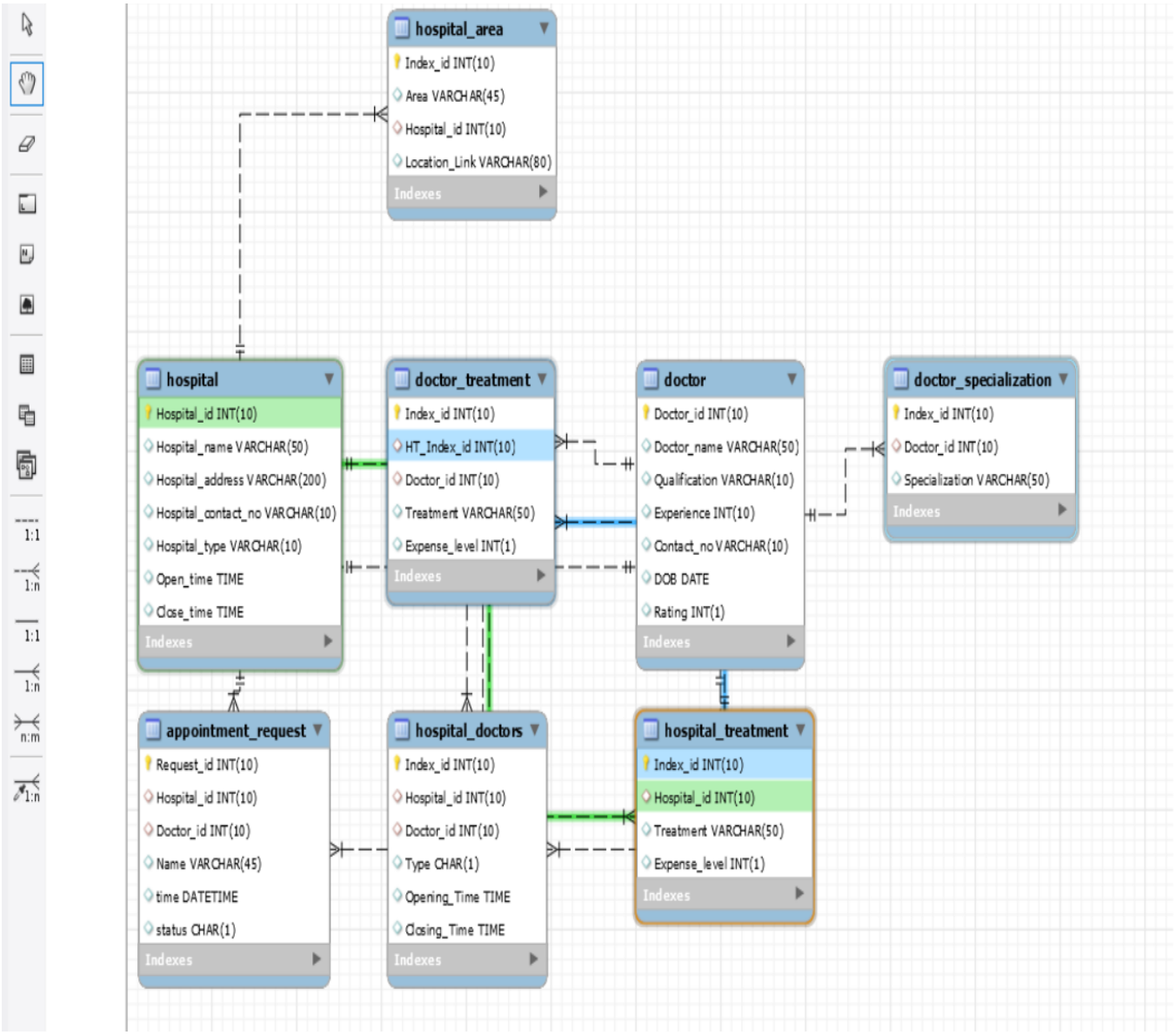
TRANSACTIONS AND TRIGGERS:

1. Remove doctors above the age of retirement from the database.
2. Remove records of the doctor having no record in the hospital management database.
3. Update the doctors from temporary to permanent.
4. Delete the doctors who resign from the hospital.
5. Triggers are generated to remove the doctors from the database at the age of retirement.

DATA TYPES:

1. **INT:** It is used to store whole numbers, that is, numbers without fractional components, of various ranges.
2. **VARCHAR:** It is used to store alphanumeric characters. It can have a varying length.
3. **TIME:** A time value may have a precision up to 6 digits. The precision specifies the number of fractional digits placed in the second field. The TIME data type requires 8 bytes and its allowed range is from 00:00:00 to 24:00:00. Current time can be retrieved with current_time function.
4. **DATETIME:** Defines a date that is combined with a time of day with fractional seconds that is based on a 24-hour clock.
5. **CHAR:** The CHAR data type stores character data.

ER DIAGRAM:


hospital_area
Index_id INT(10)
Area VARCHAR(45)
Hospital_id INT(10)
Location_Link VARCHAR(80)
Indexes
hospital
Hospital_id INT(10)
Hospital_name VARCHAR(50)
Hospital_address VARCHAR(200)
Hospital_contact_no VARCHAR(10)
Hospital_type VARCHAR(10)
Open_time TIME
Close_time TIME
Indexes
doctor_treatment
Index_id INT(10)
HT_Index_id INT(10)
Doctor_id INT(10)
Treatment VARCHAR(50)
Expense_level INT(1)
Indexes
doctor
Doctor_id INT(10)
Doctor_name VARCHAR(50)
Qualification VARCHAR(10)
Experience INT(10)
Contact_no VARCHAR(10)
DOB DATE
Rating INT(1)
Indexes
doctor_specialization
Index_id INT(10)
Doctor_id INT(10)
Specialization VARCHAR(50)
Indexes
appointment_request
Request_id INT(10)
Hospital_id INT(10)
Doctor_id INT(10)
Name VARCHAR(45)
time DATETIME
status CHAR(1)
Indexes
hospital_doctors
Index_id INT(10)
Hospital_id INT(10)
Doctor_id INT(10)
Type CHAR(1)
Opening_Time TIME
Closing_Time TIME
Indexes
hospital_treatment
Index_id INT(10)
Hospital_id INT(10)
Treatment VARCHAR(50)
Expense_level INT(1)
Indexes
1:1
1:n
1:1
1:n
n:m
1:n

SCHEMA:

1) HOSPITAL

Attributes	Datatypes	Constraints
Hospital_id	Int(10)	Primary Key
Hospital_name	Varchar(50)	Not null
Hospital_address	Varchar(200)	Not null
Hospital_contact_no	Varchar(10)	Not null
Hospital_type	Varchar(10)	Not null
Open_time	Time	Not null
Close_time	time	Not null

2) DOCTOR

Attributes	Datatypes	Constraints
Doctor_id	Int(10)	Primary Key
Doctor_name	Varchar(50)	Not null
Qualification	Varchar(40)	Not null
Experience	Int(10)	Not null
Contact_no	Varchar(10)	Not null
DOB	date	Not null
Rating	Int(1)	Not null

3) DOCTOR_SPECIALIZATION

Attributes	Datatypes	Constraints
Index_id	Int(10)	Primary Key
Doctor_id	Int(10)	Foreign Key
Specialization	Varchar(50)	Not null

4) HOSPITAL_DOCTORS

Attributes	Datatypes	Constraints
Index_id	Int(10)	Primary Key
Hospital_id	Int(10)	Foreign Key
Doctor_id	Int(10)	Foreign Key
Type	Char(1)	Not null
Opening time	time	Not null
Closing time	time	Not null

5) **HOSPITAL_TREATMENT**

Attributes	Datatypes	Constraints
Index_id	Int(10)	Primary Key
Hospital_id	Int(10)	Foreign Key
Treatment	Varchar(50)	Not null
Expense_level	Int(1)	Not null

6) **DOCTOR_TREATMENT**

Attributes	Datatypes	Constraints
Index_id	Int(10)	Primary Key
HT_index_id	Int(10)	Foreign Key
Doctor_id	Int(10)	Foreign Key
Treatment	Varchar(50)	Not null
Expense_level	Int(1)	Not null

7) **HOSPITAL_ID**

Attributes	Datatypes	Constraints
Index_id	Int(10)	Primary_key
Area	Varchar(40)	Not null
Hospital_id	Int(10)	Foreign Key
Location_Link	Varchar(50)	Not null

8) **APPOINTMENT_REQUEST**

Attributes	Datatypes	Constraints
Request_id	int(10)	Primary_key
Hospital_id	int(10)	Foreign key
Doctor_id	Int(10)	Foreign Key
Name	Varchar(50)	Not null
Time	datetime	Not null
Status	char(1)	Not null

FUNCTIONAL DEPENDENCY AND NORMALIZATION:

Normalized Form of Tables:

1. **Hospital:**

   (Hospital_id, Hospital_name, Hospital_address, Hospital_contact_no, Hospital_type, Open_time, Close_time)

   - Primary key -> Hospital_id
   - Candidate key -> {Hospital_address}

2. **Doctor:**

   (Doctor_id, name, qualification, contact_no, rating, experience, DOB)

   - Primary key -> Doctor_id
   - Candidate key -> {name, contact_no}

3. **Hospital_treatment:**

   (HTindex_id, treatment, Hospital_id, expense level)

   - Primary key -> HTindex_id
   - Candidate key -> {treatment, Hospital_id}

4. **Doctor_specialization:**

   (DSindex_id, Doctor_id, specialization)

   - Primary key -> DSindex_id
   - Candidate key -> { Doctor_id, specialization}

5. **Hospital_treatment_doctor:**

   (HTDindex_id, HTindex_id, Doctor_id, expense level)

   - Primary key -> HTDindex_id
   - Candidate key -> {HTindex_id, Doctor_id}

6. **Hospital_doctor:**

   (HDindex_id, Hospital_id, Doctor_id, type, open_time, close_time)

   - Primary key -> HDindex_id
   - Candidate key -> { Hospital_id, Doctor_id}

7. **Hospital_area:**

   (HAindex_id, area, Hospital_id, location_link)

   - Primary key -> HAindex_id

- Candidate key -> {area, Hospital_id}

8. **Appointment_request:**

(Request_id, Hospital_id, Doctor_id, name, time, status)

- Primary key -> Request_id
- Candidate key -> {Hospital_id, Doctor_id, time}

→The above tables are in BCNF as all the non-candidate keys of all tables are functional dependencies of only a single candidate key. There may be cases where primary key are reference variable to the whole set of candidate key.

→In some of the tables the primary key is just the index id which may be of no importance right now other than linking the columns in with the other tables but when there comes a need to expand the database and there comes some dependencies of that table those index id will be used to further connect them.

→The above relations are in 1st NF since it does not contain mutlivalued attributes.

→If we remove the index id then 1st and 2nd forms will be violated since it is acting as the primary key in our case.

Testing for lossless join → When a natural join is performed on both the tables , it yields the original table without any loss of data .It has a lossless join .

DDL

1. **Hospital**

```
create table Hospital(
Hospital_id int(10),
Hospital_name varchar(50),
Hospital_address varchar(200),
Hospital_contact_no varchar(10),
Hospital_type varchar(10),
Open_time time,
Close_time time,
primary key (Hospital_id));
```

2. **Doctor**

```
create table Doctor(
Doctor_id int(10),
Doctor_name varchar(50),
Qualification varchar(10),
Experience int(10),
Contact_no varchar(10),
DOB date,
Rating int(1),
primary key(Doctor_id));
```

3. **Doctor Specialization**

```
create table Doctor_Specialization(
Index_id int(10),
Doctor_id int(10),
Specialization varchar(50),
primary key (Index_id),
foreign key(Doctor_id) references Doctor(Doctor_id));
```

4. **Hospital Doctors**

```
create table Hospital_Doctors(
Index_id int (10),
Hospital_id int(10),
Doctor_id int(10),
Type char(1),
Opening_Time time,
Closing_Time time,
primary key (Index_id),
foreign key(Doctor_id) references Doctor(Doctor_id),
foreign key(Hospital_id) references Hospital(Hospital_id));
```

5. **Hospital Treatment**

```
create table Hospital_Treatment(
Index_id int(10),
```

```
Hospital_id int(10),
Treatment varchar(50),
Expense_level int(1),
primary key (Index_id),
foreign key(Hospital_id) references Hospital(Hospital_id));
```

6. **Doctor Treatment**

```
create table Doctor_Treatment(
Index_id int(10),
HT_Index_id int(10),
Doctor_id int(10),
Treatment varchar(50),
Expense_level int(1),
primary key (Index_id),
foreign key(Doctor_id) references Doctor(Doctor_id),
foreign key(HT_Index_id) references Hospital_Treatment(Index_id));
```

7. **Hospital Area**

```
create table Hospital_Area(
Index_id int(10) primary key,
Area varchar(45),
Hospital_id int(10),
Location_Link varchar(80),
foreign key(Hospital_id) references Hospital(Hospital_id));
```

8. **Appointment Request**

```
create table Appointment_Request(
Request_id int (10) primary key auto_increment,
Hospital_id int(10),
Doctor_id int(10),
Name varchar(45),
time datetime,
status char(1),
```

```
foreign key(Doctor_id) references Doctor(Doctor_id),
foreign key(Hospital_id) references Hospital(Hospital_id));
```

TRIGGERS

→Trigger to update the total number of items present after a transaction is done

→It activates when a record in the inventory is inserted, updated or deleted

#To delete the doctors from the database above the age of retirement

```
DELIMITER $$
CREATE TRIGGER retired_doctor
BEFORE UPDATE ON doctor
FOR EACH ROW
BEGIN
DELETE FROM doctor WHERE YEAR(curdate()) - YEAR(doctor.DOB) > 60;
DELETE FROM hospital_doctors WHERE hospital_doctors.Doctor_id = doctor.Doctor_
id;
END; $$
DELIMITER $$
```

SQL QUERIES

 # 1. Search hospital of an area

```
Select hospital.Hospital_name, hospital.Hospital_address,
hospital.Hospital_contact_no, hospital.Open_time, hospital.Close_time
from hospital
where hospital.Hospital_id in (select hospital_area.Hospital_id
from hospital_area where hospital_area.Area = "Sola");
```

 # 2. Doctor search by their Specialization

```
Select doctor.Doctor_name, doctor.Qualification,
doctor.Contact_no, doctor.Rating
from doctor where doctor.Doctor_id in (select doctor_specialization.Doctor_id
from doctor_specialization
where doctor_specialization.Specialization = "Neurosurgeon");
```

 # 3. Sort doctors by rating

```
select * from doctor order by rating asc;
```

#4. No. of doctors associated with a hospital

```
 Select count(hospital_doctors.Doctor_id) from hospital_doctors
 where hospital_doctors.Hospital_id =
      (Select hospital.Hospital_id
      from hospital
      where hospital.Hospital_name = 'Zydus');
```

 # 5. Remove the doctors having no record in hospital doctor relation

```
DELETE FROM doctor
WHERE NOT EXISTS (
    SELECT *
    FROM hospital_doctors
    WHERE doctor_id = doctor.doctor_id );
```

 # 6. Remove the doctor from govt hospitals above the age 60

```
delete from hospital_doctors
where doctor.Doctor_id = Hospital_doctors.doctor_id  and
datediff(now(),doctor.DOB)>=(21900)
and hospital_doctors.hospital_id = hospital.hospital_id
and hospital.hospital_type = "G";
```

 # 7. Sort the Doctors by Experience

```
select * from doctor order by experience asc;
```

 # 8. Sorting doctors based on expense level for particular treatment in ascending and descending order

```
select * from hospital as h inner join hospital_treatment as ht on
h.hospital_id = ht.hospital_id
where ht.treatment = s1
order by ht.expense_level desc;
```

 # 9. Hospitals open in current area

```
select hospital.Hospital_name, hospital.Hospital_address,
hospital_area.location_link, hospital.Hospital_contact_no
from hospital inner join hospital_area
where hospital_area.area = s1 and hospital.close_time > curtime()
and hospital.open_time < curtime();
```

CONCLUSION:

Hospital management system is the inevitable part of the lifecycle of the modern medical institutions. It automates numerous daily operations and enables smooth interactions of the user. This project covers the needs of the patients, staff and hospital authorities and simplifies their interaction.

The system includes database containing data about all the Hospitals in a city, in a particular area, treatments offered at the hospital, all doctors in a city, in a particular area, their specializations, experience, ratings, affiliations with various hospitals, availability of doctors at a hospital, book appointments. This database can be used by the user to find a hospital or clinic that best fits the user's requirements and affordability. The user can compare fees charged by various hospitals and doctors.

LIMITATIONS:

This project does not address meal options or prescription administration by the nurses for in patients. Our database also does not allow patient to interact with user interface , would have to call the hospital to do so.

FUTURE ENHANCEMENTS:

- Add user friendly graphic UI to get the look and feel of the management system.
- Accommodate more transactions and queries