DBMS Project

Hospital Management System

Submitted to: DR.Sanjeev Rao

Hargun Singh	102297016
Santosh Rathi	102117186
Mrityunjay Pandey	102117182
Kushagra Chandra Agarwal	102117176

INDEX

1. Problem statement
2. Overview
3. Requirement Analysis
4. ER Diagram
5. ER to Table
6. Normalisation
7. Functional Dependencies
8. Code and Output

Problem Statement

Developing a hospital management system in order to effectively manage most aspects of hospitals such as booking appointments, managing patient records and keeping medical report.

Overview

The hospital is a multi-specialty facility that provides a wide range of healthcare services, including inpatient care, outpatient care, surgeries, diagnostics, and pharmacy services. The hospital has multiple departments, such as cardiology, oncology, orthopedics, pediatrics, etc., and each department has its own team of doctors, nurses, and support staff. Hence it is very important for a hospital to have a DBMS that easily allows patients to book appointments and allows doctors or administrators to manage patient data.

Requirements Analysis

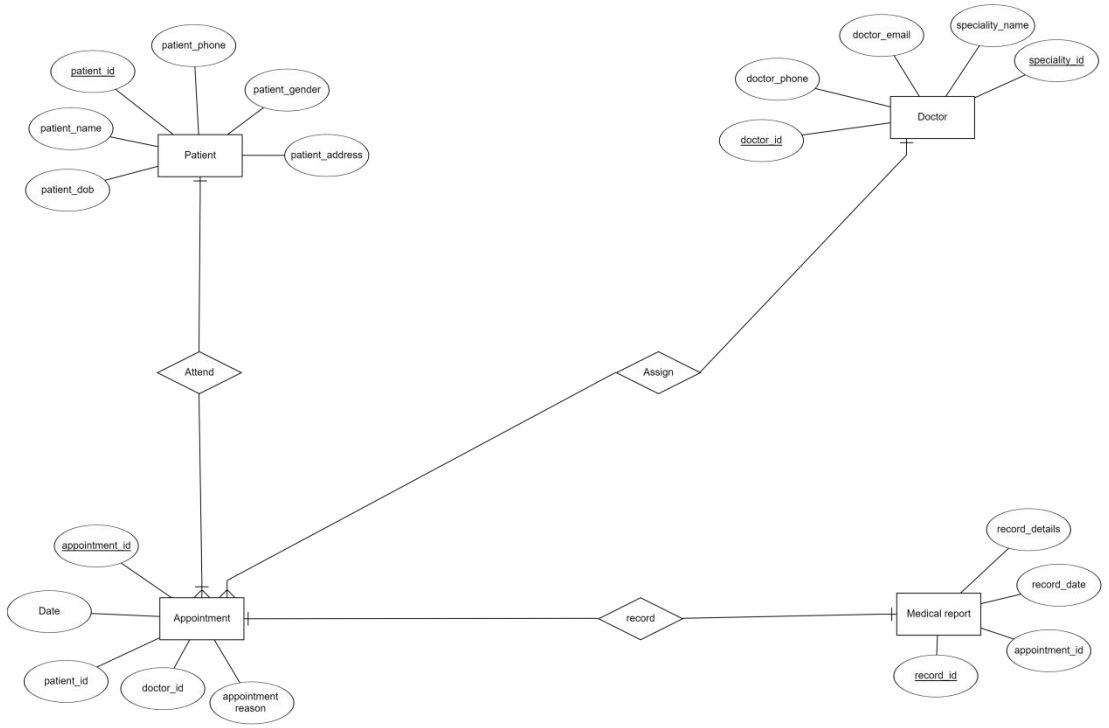
1. Patient Management:
    - Ability to add new patients with their details such as patient_id, patient_name, patient_dob, patient_gender, patient_phone, and patient_address.
    - Ability to retrieve patient information based on patient_id or patient_name.
    - Ability to update patient information such as patient_name, patient_dob, patient_gender, patient_phone, and patient_address.
2. Doctor Management:
    - Ability to add new doctors with their details such as doctor_id, doctor_name, doctor_email, and doctor_phone.
    - Ability to retrieve doctor information based on doctor_id or doctor_name.

- Ability to update doctor information such as doctor_name, doctor_email, and doctor_phone.

3. Appointment Management:
    - Ability to add new appointments with their details such as appointment_id, appointment_date, patient_id, doctor_id, and appointment_reason.
    - Ability to retrieve appointment information based on appointment_id, patient_id, or doctor_id.
    - Ability to update appointment information such as appointment_date and appointment_reason.

4. Medical Record Management:
    - Ability to add new medical records with their details such as record_id, appointment_id, record_date, and record_details.
    - Ability to retrieve medical record information based on record_id or appointment_id.
    - Ability to update medical record information such as record_date and record_details.
5. Error Handling:
    - Ability to handle errors that may occur during data insertion, retrieval, update, or deletion.
    - Display appropriate error messages when errors occur, indicating the nature of the error and possible solutions.

ER Diagram



ER Diagram To Table

1 Patient

Pid	name	dob	gender	address	phone

2 Doctor

Did	doctor_name	doctor_email	doctor_phone
speciality_id	speciality_name		

3 medical_records

Rid	Aid	record_date	record_details

4 Appointment

Aid	appointment_date	Pid	Did	appointment_reason

Normalisation

Relation: Attend

UNF: patient_id, patient_dob, patient_name, patient_gender, patient_phone,patient_address,appointment_id, appointment_date, patient_id, doctor_id, appointment_reason

2NF:

1 **Patient** :

R = (**patient_id**, patient_dob, patient_name, patient_gender, patient_phone,patient_address)

2 Appointment:

R = (**appointment_id**, appointment_date, patient_id, doctor_id, appointment_reason)

Already in 3nf

Relation: Assign

UNF:doctor_id, doctor_name, doctor_email, doctor_phone,speciality_id, speciality_name, appointment_id, appointment_date, patient_id, doctor_id, appointment_reason

2NF:

1 Doctor :

R = (**doctor_id**, doctor_name, doctor_email, doctor_phone)

2 Appointment:

R = (**appointment_id**, appointment_date, patient_id, doctor_id, appointment_reason)

3 specialties:

R = (**speciality_id,** speciality_name)

3NF:

1 Doctor :

R = (**doctor_id**, doctor_name, doctor_email, doctor_phone)

2 Appointment:

R = (**appointment_id**, appointment_date, patient_id, doctor_id, appointment_reason)

3 specialties:

R = (**speciality_id,** speciality_name)

4 doctors_specialties:

R = (doctor_id(FK), speciality_id(FK))

Relation: Record

UNF: appointment_id, appointment_date, patient_id, doctor_id, appointment_reason, record_id, appointment_id, record_date, record_details

2 NF:

1 Appointment:
R = (**appointment_id**, appointment_date, patient_id, doctor_id, appointment_reason)

2 Medical_records :
R = (**record_id**, appointment_id, record_date, record_details)

Already in 3nf

Functional Dependencies:

1. **Patient** :
   R = (**patient_id**, patient_dob, patient_name, patient_gender, patient_phone,patient_address)
   FDs:
   a. patient_id-> patient_name
   b. patient_id-> patient_dob
   c. patient_id-> patient_gender
   d. patient_id-> patient_phone
   e. patient_id->patient_address

2. **medical_records** :
   R = (**record_id**, appointment_id, record_date, record_details)
   FDs:
   a. record_id-> record_date
   b. record_id-> record_details
   c. record_id-> appointment_id

3. **Doctor** :
   R = (**doctor_id**, doctor_name, doctor_email, doctor_phone)
   FDs:
   a. doctor_id-> doctor_name
   b. doctor_id-> doctor_email

c. doctor_id-> doctor_phone

4. **Appointment:**
   R = (**<u>appointment_id</u>**, appointment_date, patient_id, doctor_id, appointment_reason)
   FDs:
   a. appointment_id-> appointment_date
   b. appointment_id-> patient_id
   c. appointment_id->doctor_id
   d. appointment_id-> appointment_reason

5. **doctors_specialities:**
   R = (**<u>doctor_id, speciality_id</u>**)
   Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes.

6. **specialities:**
   R = (**<u>speciality_id,</u>** speciality_name)
   a. **<u>speciality_id</u>**-> speciality_name

Final Database After Normalization

1 Patient

<u>Pid</u>	name	dob	gender	address	phone

2 Doctor

<u>Did</u>	doctor_name	doctor_email	doctor_phone

3 medical_records

<u>Rid</u>	Aid	record_date	record_details

4 Appointment

<u>Aid</u>	appointment_date	Pid	Did	appointment_reason

5 doctors_specialties

Did	Pid

6 specialties

speciality_id	speciality_name

Code For Creation of Tables

```
CREATE TABLE patients (
   patient_id INT PRIMARY KEY,
   patient_name VARCHAR2(50),
   patient_dob DATE,
   patient_gender CHAR(1),
   patient_phone VARCHAR2(20),
   patient_address VARCHAR2(100)
);



CREATE TABLE doctors (
   doctor_id INT PRIMARY KEY,
   doctor_name VARCHAR2(50),
   doctor_email VARCHAR2(50),
   doctor_phone VARCHAR2(20)
);



CREATE TABLE specialities (
   speciality_id INT PRIMARY KEY,
   speciality_name VARCHAR2(50)
);

CREATE TABLE doctors_specialities (
   doctor_id INT,
   speciality_id INT,
   FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id),
   FOREIGN KEY (speciality_id) REFERENCES specialities(speciality_id)
);



CREATE TABLE appointments (
   appointment_id INT PRIMARY KEY,
   appointment_date DATE,
   patient_id INT,
   doctor_id INT,
   appointment_reason VARCHAR2(100),
   CONSTRAINT fk_patient FOREIGN KEY (patient_id) REFERENCES
patients(patient_id),
```

```
    CONSTRAINT fk_doctor FOREIGN KEY (doctor_id) REFERENCES
doctors(doctor_id)
);
```

Procedure To insert values into tables

```
-- To add a patient
CREATE OR REPLACE PROCEDURE add_patient (
    p_patient_id IN patients.patient_id%TYPE,
    p_patient_name IN patients.patient_name%TYPE,
    p_patient_dob IN patients.patient_dob%TYPE,
    p_patient_gender IN patients.patient_gender%TYPE,
    p_patient_phone IN patients.patient_phone%TYPE,
    p_patient_address IN patients.patient_address%TYPE
)
IS
BEGIN
    INSERT INTO patients (
        patient_id,
        patient_name,
        patient_dob,
        patient_gender,
        patient_phone,
        patient_address
    ) VALUES (
```

```
        p_patient_id,
        p_patient_name,
        p_patient_dob,
        p_patient_gender,
        p_patient_phone,
        p_patient_address
    );
    COMMIT;
END;
/


-- TO add doctor
CREATE OR REPLACE PROCEDURE add_doctor(
    p_doctor_id IN doctors.doctor_id%TYPE,
    p_doctor_name IN doctors.doctor_name%TYPE,
    p_doctor_email IN doctors.doctor_email%TYPE,
    p_doctor_phone IN doctors.doctor_phone%TYPE
)
IS
BEGIN
    INSERT INTO doctors(doctor_id, doctor_name, doctor_email, doctor_phone)
    VALUES(p_doctor_id, p_doctor_name, p_doctor_email, p_doctor_phone);
    COMMIT;
    dbms_output.put_line('Doctor added successfully');
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error occurred while adding doctor: ' || SQLERRM);
END;
```

```
/


-- To add Specialities

CREATE OR REPLACE PROCEDURE add_speciality(
   p_speciality_id IN specialities.speciality_id%TYPE,
   p_speciality_name IN specialities.speciality_name%TYPE

)
IS
BEGIN

   -- insert the new speciality into the table
   INSERT INTO specialities(speciality_id, speciality_name)
   VALUES(p_speciality_id, p_speciality_name);

   DBMS_OUTPUT.PUT_LINE('Speciality added successfully');
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error adding speciality: ' || SQLERRM);
END;
/

--to add doctor_specilisation
CREATE OR REPLACE PROCEDURE add_doctor_speciality(p_doctor_id IN INT,
p_speciality_id IN INT)
IS
BEGIN
   INSERT INTO doctors_specialities(doctor_id, speciality_id)
```

```
    VALUES (p_doctor_id, p_speciality_id);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Doctor speciality added successfully.');
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error adding doctor speciality: ' || SQLERRM);
END;
/




-- Add an appointment
CREATE OR REPLACE PROCEDURE add_appointment (
    p_appointment_id IN appointments.appointment_id%TYPE,
    p_appointment_date IN appointments.appointment_date%TYPE,
    p_patient_id IN appointments.patient_id%TYPE,
    p_doctor_id IN appointments.doctor_id%TYPE,
    p_appointment_reason IN appointments.appointment_reason%TYPE
)
IS
BEGIN
    -- Insert appointment
    INSERT INTO appointments (appointment_id, appointment_date, patient_id,
doctor_id, appointment_reason)
    VALUES (p_appointment_id, p_appointment_date, p_patient_id, p_doctor_id,
p_appointment_reason);


    -- Commit the changes
    COMMIT;
```

```
    -- Display a message indicating successful insertion
    DBMS_OUTPUT.PUT_LINE('Appointment added successfully!');
EXCEPTION
    WHEN OTHERS THEN
        -- Display an error message if an exception occurs
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
        ROLLBACK;
END;
/




-- To add Medical_record
CREATE OR REPLACE PROCEDURE add_medical_report (
    p_record_id IN medical_report.record_id%TYPE,
    p_appointment_id IN medical_report.appointment_id%TYPE,
    p_record_date IN medical_report.record_date%TYPE,
    p_record_details IN medical_report.record_details%TYPE
)
IS
BEGIN
    -- Insert medical record
    INSERT INTO medical_report (record_id, appointment_id, record_date,
record_details)
    VALUES (p_record_id, p_appointment_id, p_record_date, p_record_details);


    -- Commit the changes
    COMMIT;
```

```
    -- Display a message indicating successful insertion
    DBMS_OUTPUT.PUT_LINE('Medical report added successfully!');
EXCEPTION
    WHEN OTHERS THEN
        -- Display an error message if an exception occurs
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
        ROLLBACK;
END;
/



BEGIN
    add_patient(1,'Mahesh','11-FEB-2000','M','7589461236','Mohali');
    add_patient(2,'Suresh','10-JAN-1998','M','5896412368','Patiala');
      add_patient(3,'Sheela','01-NOV-2002','F','6258974131','Lucknow') ;

    add_doctor(1, 'Dr. John Doe', 'john.doe@example.com', '555-555-1234');
    add_doctor(2,'Himesh','himesh1@gmail.com','8974561256');
    add_doctor(3,'Sonia','soni5@gmail.com','9658741288') ;

            add_speciality(1,'Cardiology');
            add_speciality(2,'ENT');
            add_speciality(3,'Brain');

    add_doctor_speciality(1, 1);
    -- Adding doctor with doctor_id = 1 to speciality with speciality_id = 1
    add_doctor_speciality(2, 3);
    add_doctor_speciality(3, 2);
```

add_appointment(1, '01-JAN-2023', 1, 1, 'Follow-up appointment');

add_medical_report(1,1,TO_DATE('2023-04-16', 'YYYY-MM-DD'),'Patient had a check-up, prescribed medication for common cold.');

```
select * from patients;
        select * from doctors;
        select * from specialities;
        select * from doctors_specialities;
        select * from appointments;
        select * from medical_report;
END;
/
```

```
Statement processed.
Doctor added successfully
Doctor added successfully
Doctor added successfully
Speciality added successfully
Speciality added successfully
Speciality added successfully
Doctor speciality added successfully.
Doctor speciality added successfully.
Doctor speciality added successfully.
Appointment added successfully!
Medical report added successfully!
```

PATIENT_ID	PATIENT_NAME	PATIENT_DOB	PATIENT_GENDER	PATIENT_PHONE	PATIENT_ADDRESS
1	Mahesh	11-FEB-00	M	7589461236	Mohali
2	Suresh	10-JAN-98	M	5896412368	Patiala
3	Sheela	01-NOV-02	F	6258974131	Lucknow

DOCTOR_ID	DOCTOR_NAME	DOCTOR_EMAIL	DOCTOR_PHONE
1	Dr. John Doe	john.doe@example.com	555-555-1234
2	Himesh	himesh1@gmail.com	8974561256
3	Sonia	soni5@gmail.com	9658741288

DOCTOR_ID	SPECIALITY_ID
1	1
2	3
3	2

SPECIALITY_ID	SPECIALITY_NAME
1	Cardiology
2	ENT
3	Brain

APPOINTMENT_ID	APPOINTMENT_DATE	PATIENT_ID	DOCTOR_ID	APPOINTMENT_REASON
1	01-JAN-23	1	1	Follow-up appointment

RECORD_ID	APPOINTMENT_ID	RECORD_DATE	RECORD_DETAILS
1	1	16-APR-23	Patient had a check-up, prescribed medication for common cold.

Function to Get Doctor name From Specialization name

```
CREATE OR REPLACE FUNCTION
get_doctor_name_from_specialization(p_specialization_name IN VARCHAR2)

RETURN VARCHAR2

IS

   v_doctor_name doctors.doctor_name%TYPE;

BEGIN

   SELECT d.doctor_name

   INTO v_doctor_name
```

```
    FROM doctors d
    INNER JOIN doctors_specialities ds ON d.doctor_id = ds.doctor_id
    INNER JOIN specialities s ON ds.speciality_id = s.speciality_id
    WHERE s.speciality_name = p_specialization_name;

    RETURN v_doctor_name;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
    WHEN OTHERS THEN
        RAISE;
END;
/

DECLARE
    v_doctor_name doctors.doctor_name%TYPE;
BEGIN
    v_doctor_name := get_doctor_name_from_specialization('Cardiology');
    IF v_doctor_name IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Doctor Name: ' || v_doctor_name);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No doctor found for the given specialization
name.');
    END IF;
END;
/
```

```
Statement processed.
Doctor Name: Dr. John Doe
```

Function to retrieve medical_report from appointment_Id

```
CREATE OR REPLACE FUNCTION get_medical_record(p_appointment_id IN
appointments.appointment_id%TYPE)

RETURN medical_report%ROWTYPE

IS

   v_medical_record medical_report%ROWTYPE;

BEGIN

   -- Query the medical_report table for the given appointment ID

   SELECT *

   INTO v_medical_record

   FROM medical_report

   WHERE appointment_id = p_appointment_id;


   RETURN v_medical_record;

EXCEPTION

   WHEN NO_DATA_FOUND THEN

      -- Handle the case when no medical record is found for the given appointment ID

      RAISE_APPLICATION_ERROR(-20001, 'Medical record not found for the
given appointment ID');

END;

/


DECLARE
```

```
    v_medical_record medical_report%ROWTYPE;
BEGIN
    -- Call the function to get the medical record for appointment ID 12345
    v_medical_record := get_medical_record(1);

    -- Display the medical record details
    DBMS_OUTPUT.PUT_LINE('Record ID: ' || v_medical_record.record_id);
    DBMS_OUTPUT.PUT_LINE('Appointment ID: ' ||
v_medical_record.appointment_id);
    DBMS_OUTPUT.PUT_LINE('Record Date: ' || v_medical_record.record_date);
    DBMS_OUTPUT.PUT_LINE('Record Details: ' ||
v_medical_record.record_details);
EXCEPTION
    WHEN OTHERS THEN
        -- Handle any exceptions that may occur
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

```
Statement processed.
Record ID: 1
Appointment ID: 1
Record Date: 16-APR-23
Record Details: Patient had a check-up, prescribed medication for common cold.
```

Trigger to check slot before booking appointment

```
CREATE OR REPLACE TRIGGER check_appointment
BEFORE INSERT ON appointments
FOR EACH ROW
```

```
DECLARE
   count_appt NUMBER;
BEGIN
   SELECT COUNT(*) INTO count_appt
   FROM appointments
   WHERE patient_id = :new.patient_id
   AND doctor_id = :new.doctor_id
   AND TRUNC(appointment_date) = TRUNC(:new.appointment_date);

   IF count_appt > 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Patient has already booked an appointment with the same doctor on the same day.');
   END IF;
END;
/
insert into appointments values(1, '01-JAN-2023', 1, 1, 'Follow-up appointment');
```

```
ORA-20001: Patient has already booked an appointment with the same doctor on the same day. ORA-06512: at "SQL_YSPPWGQCIQBPCRHQPCMLKPPAH.CHECK_APPOINTMENT", line 11
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

References

www.youtube.com/parteekBhatia

Conclusion

In conclusion, the healthcare management system project requires the development of a relational database management system (DBMS) that can efficiently store and manage patient information, doctor information, appointments, and medical reports. The system should be able to enforce referential integrity, perform CRUD operations, and provide appropriate error handling, data validation, and data integrity checks.