SQL

ORACLE

DBMS PROJECT

HOSPITAL MANAGEMENT SYSTEM

A PROJECT REPORT

ON

HOSPITAL MANAGEMENT SYSTEM

COURSE: DATABASE MANAGEMENT SYSTEM (CSE1401)

UNDER GUIDANCE OF : PROF. ANJALI G JIVANI

SUBMITTED BY :-

NAME	SEAT NO.	PRN
HIMILSINH SINDHA	453070	8021075138
PREET MISTRY	453029	8021075004
KETUL PRAJAPATI	453057	8021031843
MANAV PARMAR	453036	8021019677

INDEX

PROJECT DESCRIPTION

SCOPE

The hospital management system is designed to manage and streamline the operations of a hospital or medical centre. The system can help healthcare providers manage patient records, appointments, prescriptions, tests, and billing, as well as other important aspects of hospital operations such as staff and resource management.

MODULES

The hospital management system project typically consists of several interconnected modules or components that work together to provide a complete solution. Some of the key components of the system include:

1) Patient management module : This module allows healthcare providers to manage patient records, including medical history, and treatment plans. It also allows for scheduling of appointments and tracking of patient progress.

2) Prescription and test management module : This module allows doctors and other healthcare providers to prescribe medication and tests for patients, and track the results of those tests.

3) Billing and payment module : This module helps manage the financial aspects of hospital operations, including billing patients for services rendered, tracking payments, and generating financial reports.

4) Staffing and resource management module : This module helps manage staffing levels and resource allocation, including the scheduling of doctors, nurses, and other staff members, and managing the availability of hospital rooms and equipment.

BENEFITS

The benefits of a hospital management system project are numerous. It can help improve the efficiency of hospital operations, and improve patient safety, and improve the quality of care delivered to patients. It can also help streamline billing and payment processes, reducing administrative overhead and improving financial performance.

EPILOGUE

Overall, a hospital management system project is an important investment for healthcare providers looking to improve the quality and efficiency of their operations. With the right system in place, hospitals can better manage patient care and financial operations, leading to more sustainable business model for the hospital.

ASSUMPTIONS

- Here we have only considered for inpatient.
- For appointment , only perfect hours can be selected.
- Each patient can only be assigned to one room at a time.
- Each patient can only be treated by one doctor at a time.
- Any nurse can treat any patient, according to their availability.
- Records have history regarding patient's entry, exit and treatment only.
- We have considered only following room types :

1. Special Room (Capacity : 1 patient)
2. Semi - Special Room (Capacity : 3 patients)
3. General Ward (Capacity : 10 patients)

NORMALIZED TABLES

STAFF	
CONSTRAINT	**COLUMN**
PK	Staff_Id
	Staff_Name
	SGender
	SPhno
	SAddress

PATIENT	
CONSTRAINT	**COLUMN**
PK	P_Id
	Patient_Name
	PGender
	PPh_No
	PAddress

DOCTOR	
CONSTRAINT	**COLUMN**
PK , FK	Doc_Id
	Speciality

NURSE	
CONSTRAINT	**COLUMN**
PK , FK	Nurse_Id
	Shift_Type

APPOINTMENT	
CONSTRAINT	**COLUMN**
PK	Apt_Id
FK	Doc_Id
FK	P_Id
	Apt_Date_Slot

DIAGNOSE	
CONSTRAINT	**COLUMN**
PK , FK	Doc_Id
PK , FK	P_Id
PK , FK	Test_Id
	Diag_Disease

RECORDS	
CONSTRAINT	COLUMN
PK	Record_Id
PK , FK	P_Id
	Dt_Admitted
	Dt_Discharged
	Treatment

PRES_TEST	
CONSTRAINT	COLUMN
PK , FK	Test_Id
FK	Pres_Id
	Test_Result
	Test_Cost
	Test_Date

TEST_DETAILS	
CONSTRAINT	COLUMN
PK	Test_Id
	Test_Type
	P_Range
	Bio_Range_M/F

PRES_MEDICINE	
CONSTRAINT	COLUMN
PK , FK	Med_Id
PK , FK	Pres_Id
	M_Qty
	M_Dosage

ROOM_INFO	
CONSTRAINT	COLUMN
PK , FK	Room_Id
PK , FK	P_Id
	R_Entry_Date
	R_Exit_Date
	Room_Status

ROOM_TYPE	
CONSTRAINT	COLUMN
PK	Type_Id
	Type_Name
	R_Cost

BILL	
CONSTRAINT	**COLUMN**
PK	Bill_Id
FK	P_Id
	Bill_Dt

MEDICINE_DETAILS	
CONSTRAINT	**COLUMN**
PK	Med_Id
	Name
	M_Unit_Price

BILL_DETAILS	
CONSTRAINT	**COLUMN**
PK , FK	Bill_Id
	Pay_Id
	Doc_Charges
	Room_Charges
	Med_Charges
	Test_Charges
	Other_Charges
	Mode_Of_Pay
	Status

ROOM	
CONSTRAINT	**COLUMN**
PK	Room_Id
FK	Type_id

PRESCRIPTION	
CONSTRAINT	**COLUMN**
PK	Pres_Id
FK	Doc_Id
FK	P_Id

NOTATION : PK - PRIMARY KEY

FK - FOREIGN KEY

CARDINALITY

• DOCTOR - APPOINTMENT	1 : M	
• APPOINTMENT - PATIENT	1 : 1	
• PATIENT - RECORDS	1 : M	WEAK
• PATIENT - DIAGNOSE	1 : M	WEAK
• PATIENT - ROOM	1 : 1	
• ROOM - ROOM_INFO	1 : M	
• ROOM_TYPE - ROOM	1 : M	
• PATIENT - BILL	1 : M	
• BILL - BILL_DETAILS	1 : 1	WEAK
• PATIENT - PRESCRIPTION	1 : M	
• DOCTOR - PRESCRIPTION	1 : M	
• DOCTOR - PATIENT	1 : M	
• PRESCRIPTION - PRES_TEST	1 : M	
• PRESCTIPTION - PRES_MEDICINE	1 : M	
• PRES_TEST - TEST_DETAILS	1 : 1	WEAK
• PRES_MEDICINE - MED_DETAILS	1 : 1	WEAK

FUNCTIONS

CALCULATE TOTAL BILL

```
CREATE OR replace FUNCTION calculate_total_bill(p IN patient.p_id%TYPE)
RETURN NUMBER
IS
 total_bill NUMBER := 0
BEGIN
   SELECT ( doc_charges + room_charges + med_charges
            + test_charges + other_charges )
   INTO   total_bill
   FROM   bill_details
   WHERE  bill_id = (SELECT bill_id
                     FROM   bill
                     WHERE  p_id = p);
   RETURN total_bill;
END;
```

SEARCH ROOM ID WITH PATIENT NAME

```
CREATE OR replace FUNCTION get_patient_room(p_name IN
 patient.patient_name%TYPE)
RETURN room.room_id%TYPE
AS
 r_id room.room_id%TYPE;
BEGIN
   SELECT r.room_id
   INTO   r_id
   FROM   room r
          inner join room_info ri ON r.room_id = ri.room_id
          inner join patient p ON ri.p_id = p.p_id
          WHERE  upper(p.patient_name) = upper(p_name);
   RETURN r_id;
END;
```

FIND NUMBER OF PATIENTS WITH A GIVEN DISEASE

```
CREATE OR replace FUNCTION patients_count_of_disease(d_name IN
diagnose.diag_disease%TYPE)
RETURN NUMBER
AS
  p_count NUMBER(5);
BEGIN
    SELECT count(*)
    INTO   p_count
    FROM   diagnose
    WHERE  upper(diag_disease) = upper(d_name);

    RETURN p_count;

EXCEPTION
     WHEN no_data_found THEN
          dbms_output.put_line('There is no patient with this disease.');
END;
```

PROCEDURES

ADD NEW PATIENT DETAILS

```
CREATE OR replace PROCEDURE add_patient (
                p_pph_no IN patient.pph_no%TYPE,
                p_pgender IN patient.pgender%TYPE,
                p_patient_name IN patient.patient_name%TYPE,
                p_paddress IN patient.paddress%TYPE,
                p_dt_admitted IN records.dt_admitted%TYPE)
IS
   patient_id  patient.p_id%TYPE;
   p_count  NUMBER(10);
   bill_id1  bill_details.bill_id%TYPE;
   b_count  NUMBER(10);
   r_count  NUMBER(10);
   r_id  records.record_id%TYPE;

BEGIN
   SELECT max(p_id)
   INTO p_count
   FROM PATIENT;

   patient_id := p_count + 1;

   INSERT INTO PATIENT (p_id,pph_no,pgender,patient_name,paddress)
   VALUES (patient_id,p_pph_no,p_pgender,p_patient_name,p_paddress);

   dbms_output.put_line
            ('Patient is successfully registered with Patient ID :' || patient_id);

   SELECT max(record_id)
   INTO   r_count
   FROM   RECORDS;

   r_id := r_count + 1;
```

```
    INSERT INTO RECORDS
    (record_id,p_id,dt_admitted,dt_discharged,treatment)
    VALUES (r_id,patient_id,p_dt_admitted,null,null);

    assign_room_to_patient(patient_id);

    UPDATE ROOM_INFO
    SET  r_entry_date = p_dt_admitted
    WHERE  p_id = patient_id;

    SELECT max(bill_id)
    INTO  b_count
    FROM  BILL_DETAILS;

    bill_id1 := b_count + 1;

    INSERT INTO BILL (bill_id,bill_dt,p_id)
    VALUES (bill_id1,sysdate,patient_id);

    INSERT INTO BILL_DETAILS
     (bill_id,doc_charges,room_charges,med_charges,other_charges,test_charges,
      mode_of_pay,status,pay_id)
    VALUES  (bill_id1,0,0,0,0,0,null,'unpaid',null);

END add_patient;
```

ASSIGN ROOM TO PATIENT

```
CREATE OR replace PROCEDURE assign_room_to_patient(patient_id IN
          patient.p_id%TYPE)
IS
   v_room_status  room_info.room_status%TYPE;
   assign  BOOLEAN := false;
   p_count  NUMBER(2);
   t_count  NUMBER(2);
   present_id  patient.p_id%TYPE := null;
   r_id  room.room_id%TYPE;
   CURSOR c1 IS SELECT * FROM ROOM_INFO ORDER  BY room_id;
    r1 c1%ROWTYPE;

BEGIN
   OPEN c1;

   SELECT p_id
   INTO present_id
   FROM ROOM_INFO
   WHERE p_id = patient_id;

   IF present_id IS NOT NULL THEN
         dbms_output.put_line('Room is already assigned to Patient');
   END IF;

   CLOSE c1;
EXCEPTION
   WHEN no_data_found THEN
      LOOP
        FETCH c1 INTO r1;
         r_id := r1.room_id;
         SELECT count(*)
         INTO t_count
         FROM ROOM_INFO;
```

```
IF r1.room_status = 'vacant' THEN
   IF r1.room_id IN ( 1, 2, 3 ) THEN

     UPDATE ROOM_INFO SET p_id = patient_id,
     room_status = 'occupied'
     WHERE  room_id = r_id AND p_id = 0;
      dbms_output.put_line('Patient ID :' || patient_id ||
       ' Is Assigned To Special Room With Room ID :' || r_id);
       assign := true;

   ELSIF r1.room_id IN ( 4, 5, 6, 7, 8 ) THEN

     SELECT count(*) INTO p_count FROM ROOM_INFO
     WHERE  room_id = r_id;
      IF p_count <= 3 THEN

        IF p_count < 3 THEN

          UPDATE ROOM_INFO SET  p_id = patient_id,
          room_status = 'occupied'
          WHERE  room_id = r_id AND p_id = 0;
          dbms_output.put_line('Patient ID :' || patient_id ||
          ' Is Assigned To Semi-Special Room With Room ID :' || r_id);
           assign := true;
           INSERT INTO ROOM_INFO
           VALUES ( r1.room_id,0,null,null,'vacant');

         ELSIF p_count = 3 THEN

           UPDATE ROOM_INFO SET p_id = patient_id,
            room_status = 'occupied'
            WHERE room_id = r_id AND p_id = 0;
             assign := true;

          END IF;
        END IF;
```

```
        ELSIF r1.room_id IN ( 9, 10 ) THEN

          SELECT count(*) INTO p_count FROM ROOM_INFO
          WHERE room_id = r_id;
           IF p_count <= 10 THEN

              IF p_count < 10 THEN

                 UPDATE ROOM_INFO SET p_id = patient_id,
                  room_status = 'occupied'
                  WHERE room_id = r_id AND p_id = 0;
                  dbms_output.put_line('Patient ID :' || patient_id ||
                  ' Is Assigned To General Ward With Room ID :' || r_id);
                   assign := true;
                   INSERT INTO ROOM_INFO
                   VALUES (r1.room_id,0,null,null,'vacant');

               ELSIF p_count = 10 THEN

                   UPDATE ROOM_INFO SET p_id = patient_id,
                   room_status = 'occupied'
                   WHERE room_id = r_id AND p_id = 0;
                   assign := true;

                 END IF;
               END IF;
            END IF;
      ELSIF t_count = 38 THEN
             dbms_output.put_line('No Room Is Available');
              exit;
       END IF;
    END LOOP;
  WHEN OTHERS THEN
      dbms_output.put_line('Error assigning room to patient');
END assign_room_to_patient;
```

UPDATE DISCHARGE DATE

```
CREATE OR replace PROCEDURE update_discharge_date (patient_id IN
patient.p_id%TYPE)

IS
  dis_date room_info.r_exit_date%TYPE;

BEGIN

    SELECT r_exit_date
    INTO dis_date
    FROM ROOM_INFO r
    WHERE r.p_id = patient_id;

    UPDATE RECORDS re
    SET re.dt_discharged = dis_date
    WHERE re.p_id = patient_id;

    dbms_output.put_line('patient discharged on : ' || dis_date);

END update_discharge_date;
```

TRIGGERS

UPDATE ROOM AVAILABILITY AND ROOM CHARGES ON DISCHARGE

```
CREATE OR replace TRIGGER room_management
AFTER UPDATE ON RECORDS
FOR EACH ROW

DECLARE

   rc  bill_details.room_charges%TYPE := 0;
   nd  NUMBER(10);
   patient_id  records.p_id%TYPE := : old.p_id;
   discharge_date  records.dt_discharged%TYPE := : new.dt_discharged;
   previous_value  records.dt_discharged%TYPE := : old.dt_discharged;
   r_id  room.room_id%TYPE;
   p_count  NUMBER(2);

BEGIN

  IF discharge_date IS NOT NULL AND previous_value IS NULL THEN

    SELECT r_cost
    INTO rc
    FROM ROOM_TYPE
    WHERE type_id = (SELECT type_id
                     FROM ROOM
                     WHERE room_id = (SELECT room_id
                                      FROM ROOM_INFO
                                      WHERE p_id = patient_id));

    SELECT ( r_exit_date - r_entry_date )
    INTO nd
    FROM ROOM_INFO
    WHERE p_id = patient_id;
```

```
    UPDATE BILL_DETAILS
    SET room_charges = nd * rc
    WHERE bill_id = (SELECT bill_id
                     FROM BILL
                     WHERE p_id = patient_id);

    SELECT room_id
    INTO r_id
    FROM ROOM_INFO
    WHERE p_id = patient_id;

    IF r_id IN ( 1, 2, 3 ) THEN
       UPDATE ROOM_INFO
       SET room_status = 'vacant',p_id = 0,r_entry_date = null,
       r_exit_date = null
       WHERE p_id = patient_id;
    ELSIF r_id IN ( 4, 5, 6, 7,8, 9, 10 ) THEN
       SELECT count(*)
       INTO p_count
       FROM ROOM_INFO
       WHERE room_id = r_id;

       IF p_count = 1 THEN
          UPDATE ROOM_INFO
          SET room_status = 'vacant',p_id = 0,r_entry_date = null,
          r_exit_date = null
          WHERE p_id = patient_id;
       ELSE
          DELETE FROM ROOM_INFO
          WHERE room_id = r_id;
       END IF;
     END IF;

     dbms_output.put_line
               ('Patient has been discharged successfully, now room is vacant');
  END IF;
END;
```

CHECK APPOINTMENT AVAILABILITY

```
CREATE OR replace TRIGGER check_appointment_availability
BEFORE INSERT OR UPDATE ON APPOINTMENT
FOR EACH ROW
DECLARE
   apt_count NUMBER;
BEGIN
   IF TO_CHAR(:NEW.apt_date_slot,'mi') = 0 THEN
     SELECT count(*)
     INTO apt_count
     FROM APPOINTMENT
     WHERE doc_id = :NEW.doc_id AND apt_date_slot = :NEW.apt_date_slot;
     IF apt_count >= 1 THEN
          raise_application_error
                    (-20001, 'Doctor appointment slot is already booked by other.');
     END IF;
   ELSE raise_application_error
              (-20006, 'Only Perfect Hours Appointment can be selected');
   END IF;
END;
```

UPDATE TEST CHARGES ON NEW PRESCRIPTION

```
CREATE OR replace TRIGGER update_test_charges
BEFORE INSERT ON PRES_TEST
FOR EACH ROW

BEGIN
   UPDATE BILL_DETAILS
   SET test_charges = ( test_charges + :NEW.test_cost )
   WHERE bill_id = (SELECT bill_id
                    FROM BILL
                    WHERE p_id = (SELECT p_id
                                  FROM PRESCRIPTION
                                  WHERE pres_id = :NEW.pres_id));
END;
```

STAY HEALTHY

STAY SAFE