Project

On

Hospital Management System

Prepaired By :

Name : Himanshu Dodrajka

Roll No : U18CO056

HOSPITAL

A hospital is a health care institution providing patient treatment with specialized staff and equipment. Hospitals are usually funded by public sector by health organisations (for profit or non-profit), by health insurance companies, or by charities, including direct charitable donations. Hospitals have a range of departments (e.g., surgery, and urgent care etc).

HOSPITAL MANAGEMENT SYSTEM

A hospital management system is an information system that manages the aspects of a hospital. This may include the administrative, financial, and medical processing. It is an integrated end-to-end Hospital Management System that provides relevant information across the hospital to support effective decision making for patient care, hospital administration and critical financial accounting, in a seamless flow. This program can look after Inpatients, OPD patients, records, database treatments, status illness, billings etc. it also maintains their hospital info such as ward id, Doctor in Charge, Department administering etc. Now with a laboratory module to handle all lab operations...!!! Not only has this it also looked after doctor and staff records and payments. Now with advanced features like LAN connectivity, ICD10 disease database, Webcam support.

NEED OF HMS

1. Minimized documentation and no duplication of records.
2. Reduced paper work.
3. Improved Patient Care
4. Better Administration Control
5. Faster information flow between various departments

6. Smart Revenue Management
7. Effective billing of various services
8. Exact stock information

Product Function

The data represented in hospital management application will perform the following major function:-

- Patient Details: - It includes inpatient and outpatient details.
- Tasks:- It includes the various tasks which are used in hospital
- Billing Details:-This software will help to calculate the bill much quicker and simpler way. This enables the organization to keep the information in efficient and systematic way.
- Doctor Details : - It includes detail of doctor and detail of doctor attending a patient.
- Deleting and Updating Details : - It delete the patient Details which are discharged from hospital or Doctor who left the job.
  It updates the condition of patient, their age , Doctor details etc. Which changes due time.

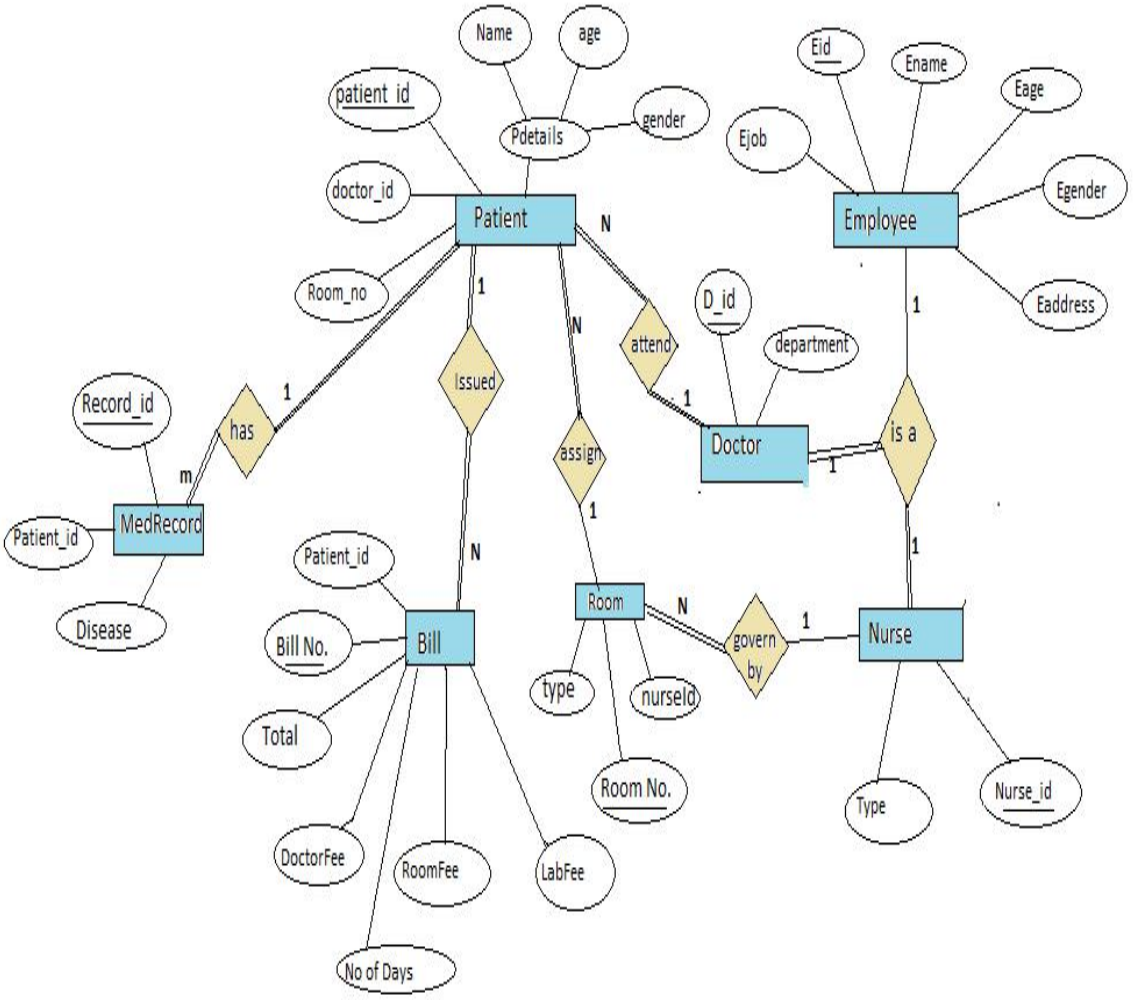
SPECIFIC REQUIREMENTS

It describes all the details that the software developer need to know for designing and developing the system. This section Describe the software or language used by developer.

--NO Front-End.

- Oracle live online .
- Pl SQL for Back-end programming.

ER Diagram of Hospital Management System :



For normalied form

Note :- Address here is taken as only as city name . so it is not a composite attribute.

Relational Model:

Conversion of Entity into Relational Model:

Tables:

Denormalized :

Doctor

Attribute	Description	Data type
Doctor_id	Id of Doctor	varchar
Doctor_name	Name	varchar
Doctor_age	Age	number
Doctor_gender	Gender	varchar
Doctor_address	Address	varchar
department	Department	varchar

Nurse

Attribute	Description	Data type
Nurse _id	Id of Nurse	varchar
Nurse _name	Name	varchar
Nurse _age	Age	number
Nurse _gender	Gender	varchar
Nurse _address	Address	varchar
Nurse_type	Type of work	varchar

Patient

Attribute	Description	Data type
Patient_Id	Id of patient	varchar

name	Name	varchar
age	Age	number
gender	Gender	varchar
Room No.	Room no. Of patient	number
Disease	Disease to patient	varchar
Doctor_id	Id of Doctor	varchar

Room

Attribute	Description	Data type
Room No.	Number of room	number
R_type	Type of Room	varchar
Nurse_id	Id of nurse	varchar

Bill

Attribute	Description	Data type
Patient_Id	Id of patient	varchar
Bill_no	No. On bill recipt	number
No_of_days	Number of days Patient admitted	Number
Doctor_charge	Doctor's Fee	number
Lab_charge	Lab's Fee	number
Room_charge	Room Fee	number
Total	Total amount	number

As seen from above table some data can be redundant

So Normalization is require.

Himanshu

Normalization of Relational Model

All the FD's of table is given below:

For Doctor table:

(doctor_id,department) --> (doctor_name,doctor_age,doctor_gender,doctor_address)

For Nurse table:

(nurse_id,nurse_type) --> (nurse_name,nurse_age,nurse_gender,nurse_address)

For Patient table:

(patient_id) --> (name,age,gender,room_no,disease,doctor_id)

For Bill table:

(Bill_no) --> (no_of_days,patient_id,doctor_charge,room_charge,lab_charge,total)

For Room table :

(room_no) --> (nurse_id,r_type)

1NF:

Patient table is not in 1NF state as **disease** attribute may have more than one value as a person may have more than one disease at same time.

So we decompose patient table as follows:

Patient --> Patient,MedRecord.

Patient table

(patient_id) --> (name,age,gender,room_no,disease,doctor_id)

MedRecord table

(record_id) --> (patient_id,disease)

Patient table divide in two table Patient and MedRecord

As follows:

Patient

Attribute	Description	Data type

Patient_Id	Id of patient	varchar
name	Name	varchar
age	Age	number
gender	Gender	varchar
Room No.	Room no. Of patient	number
Doctor_id	Id of Doctor	varchar

MedRecord

Attribute	Description	Data type
Patient_Id	Id of patient	varchar
Disease	Disease	varchar
Record_id	Id of record	varchar

All Other Tables **Bill , Doctor , Nurse , Room** will remain same.
So 1NF contains table :
Bill , Doctor , Nurse , Room , Patient , MedRecord.

2NF:

for Doctor table candidate key is {doctor_id,department} and for Nurse table candidate key {nurse_id,nurse_type}
And thier is also dependency from doctor_id and nurse_id too.
As Doctor and Nurse table contain partial dependecies hence it these two table are not in 2NF state .

Converting Doctor and Nurse into 2NF:
Doctor table can be decomposed into two table Doctor_info , Doctor
Similarly Nurse table can be decompose into Nurse_info , Nurse.

But both Nurse_info and Doctor_info contain similar data and attribute so Doctor_info and Nurse_info table becomes a single table Employee.

Doctor , Nurse ---> Doctor,Nurse,Employee.

Employee

Attribute	Description	Data type
Eid	Id of em[ployee	varchar
Ename	Name	varchar
Eage	Age	number
Egender	Gender	varchar
Eaddress	Address	varchar
E_type	Type Employee	varchar

Doctor

Attribute	Description	Data type
doctor _id	Id of doctor	varchar
Department	department	varchar

Nurse

Attribute	Description	Data type
Nurse _id	Id of Nurse	varchar
Nurse_type	Type of work	varchar

Other Tables **Bill , Room , Patient , MedRecord** will remain same.
So 2NF contain Tables :
Bill , Room , Patient , MedRecord , Employee , Doctor , Nurse.

3NF:

None of the above table contains transitive dependency so it is already in 3NF state.

So 3NF contain Tables :

Bill , Room , Patient , MedRecord , Employee , Doctor , Nurse.

BCNF:

In All of the above table the dependencies have only candidate key in left side of dependency so it is already in BCNF.

So BCNF contain Tables :

Bill , Room , Patient , MedRecord , Employee , Doctor , Nurse.

After Normalization final tables are :

Employee

Attribute	Description	Data type	Condition
Eid	Id of employee	varchar	Primary key
Ename	Name	varchar	Not NULL
Eage	Age	number	Not Null
Egender	Gender	varchar	Not Null
Eaddress	Address	varchar	
Ejob	Type of Employee	varchar	Not NULL

Doctor

Attribute	Description	Data type	Condition
doctor _id	Id of doctor	varchar	Primary key
Department	department	varchar	Not Null

Nurse

Attribute	Description	Data type	Condition
Nurse _id	Id of Nurse	varchar	Primary key
Nurse_type	Type of work	varchar	Not Null

Patient

Attribute	Description	Data type	Condition
Patient_Id	Id of patient	varchar	Primary key
name	Name	varchar	Not NULL
age	Age	number	Not NULL
gender	Gender	varchar	Not NULL
Room No.	Room no. Of patient	number	Foreign key (Room)
Doctor_id	Id of Doctor	varchar	Foreign key

MedRecord

Attribute	Description	Data type	Condition
Patient_Id	Id of patient	varchar	foreign key
Disease	Disease to patient	varchar	Not Null
Record_id	Id of record	varchar	Primary key

Room

Attribute	Description	Data type	Condition
Room No.	Number of room	number	Primary key
R_type	Type of Room	varchar	Not NULL
Nurse_id	Id of nurse	varchar	Foreign key

Bill

Attribute	Description	Data type	Condition
Patient_Id	Id of patient	varchar	Foreign key
Bill_no	No. On bill recipt	number	Primary key
No_of_days	Number of days Patient admitted	Number	Not NULL
Doctor_charge	Doctor's Fee	number	Not NULL
Lab_charge	Lab's Fee	number	Not NULL
Room_charge	Room Fee	number	Not NULL
Total	Total amount	number	Not NULL

Key Constraints :

primary Keys , foreign Key , null , Unique constraints are already included in the above tables.

Views:

- The view Room_status shows Room number along with the the number of patiends assigned to given room.

create or replace view Room_status as

```
select room_no,count(*) as Patients
    from Patient
     group by room_no
     order by room_no;

select * from Room_status;
```

- The view doctor_status shows Doctor id , name of doctor , the number of patient which are currently under treatment by doctor and the department or speciallity of doctor.

```
create or replace view doctor_status as
        with ttable(ID,count) as
        (select doctor_id ,count(*)
         from Patient group by doctor_id)
      select distinct ttable.ID as ID , Ename as Name ,ttable.count as
Patients,department
         from ttable inner join Patient on Patient.doctor_id = ttable.ID
inner join Doctor on Patient.doctor_id = Doctor.doctor_id
         inner join Employee on Eid = Patient.doctor_id;

select * from doctor_status;
```

```
View created.
```

ROOM_NO	PATIENTS
101	2
102	1

```
Download CSV
2 rows selected.

View created.
```

ID	NAME	PATIENTS	DEPARTMENT
d1	Ashish	1	Surgeon
d2	Kavita	2	Surgeon

```
Download CSV
```

Joins:

Inner join is used in the above table for joining four different tables **Employee , Patient , Doctor , ttable (a temperory table created by WITH query).**

Implementation

Completely populated table with corrected data elements reflecting the ERD design

Employee

Eid	Ename	Eage	Egender	Eaddress	Ejob
d1	Ashish	35	male	Delhi	Doctor
d2	Kavita	30	female	Jodhpur	Doctor
n1	Arjun	45	male	Ajmer	Nurse
n2	Keshav	35	male	Surat	Nurse

Doctor

doctor_id	department
d1	Surgeon
d2	Surgeon

Nurse

nurse_id	nurse_type
n1	Medical-surgical
n2	Emergency room

Bill
Himanshu

Bill_no	Patient_id	Doctor_fee	Room_fee	days	Lab_fee	total
1	p1	20000	10000	10	20000	50000
2	p2	20000	20000	10	20000	60000
3	p3	20000	15000	10	20000	55000

Patient

Patient_id	Name	age	gender	Doctor_id	Room_no
P1	Arvind	34	male	D1	101
P2	Arun	36	male	D2	101

MedRecord

Record_id	disease	Patient_id
R1	Asthama	P1
R2	Corona	P2

Implementation of PLSQL functiionalities and Snapshots of PL/SQL extension and querys :

Creating all tables:

```
create table Employee(
  Eid varchar(10) primary key,
  Ename varchar(30),
  Eage number,
  Egender varchar(10),
  Eaddress varchar(50),
  Ejob varchar(20)
```

```
 );

create table Doctor(
 doctor_id varchar(10),
 department varchar(30)
);

create table Nurse(
 nurse_id varchar(10),
 nurse_type varchar(20)
);

 create table MedRecord(
    record_id varchar(10),
    disease varchar(50),
    patient_id varchar(10)
 );
create table room (
   Rno number,
   Rtype varchar(10),
   nurse_id varchar(10)
);
create table Bill (
    bill_no number,
    patient_id varchar(10),
    doctor_fee number,
    room_fee number ,
    no_of_days number,
    lab_fee number,
    total number
);
```

Creating and Implementing Triggers :

The below Triggers will print data after inserting data into a table or updating the data into table:

```
CREATE OR REPLACE TRIGGER display_changes_p
After insert or update ON Patient
FOR EACH ROW
WHEN (NEW.age > 0)
BEGIN
    dbms_output.put_line('After Insertion Patient Data is following :');
    dbms_output.put_line('ID             : ' || :NEW.patient_id);
    dbms_output.put_line('Name           : ' || :NEW.name);
    dbms_output.put_line('Age            : ' || :NEW.age);
```

```
    dbms_output.put_line('gender      : ' || :NEW.gender);
    dbms_output.put_line('doctor_id : ' || :NEW.doctor_id);
END;
/
-----------------------------------------------------------------------------------------
CREATE OR REPLACE TRIGGER display_changes_e
After insert or update ON Employee
FOR EACH ROW
WHEN (NEW.Eage > 0)
BEGIN
    dbms_output.put_line('After Insertion Employee Data is following :');
    dbms_output.put_line('ID             : ' || :NEW.Eid);
    dbms_output.put_line('Name          : ' || :NEW.Ename);
    dbms_output.put_line('Age            : ' || :NEW.Eage);
    dbms_output.put_line('gender      : ' || :NEW.Egender);
    dbms_output.put_line('Address     : ' || :NEW.Eaddress);
END;
/
-----------------------------------------------------------------------------------------------------
CREATE OR REPLACE TRIGGER display_changes_d
After insert ON Doctor
FOR EACH ROW
WHEN (NEW.department is not NULL)
BEGIN
    dbms_output.put_line('Department: ' || :NEW.department);
END;
/
-----------------------------------------------------------------------------------------
CREATE OR REPLACE TRIGGER display_changes_m
After insert or update ON MedRecord
FOR EACH ROW
WHEN (NEW.record_id is not NULL)
BEGIN
    dbms_output.put_line('ID             : ' || :NEW.record_id);
    dbms_output.put_line('Disease     : ' || :NEW.Disease);
END;
/
-----------------------------------------------------------------------------------------
CREATE OR REPLACE TRIGGER display_changes_b
After insert or update ON Bill
FOR EACH ROW
WHEN (NEW.bill_no > 0)
BEGIN
    dbms_output.put_line('After Insertion Bill Data is following :');
    dbms_output.put_line('Patient ID        : ' || :NEW.patient_id);
    dbms_output.put_line('bill No            : ' || :NEW.bill_no);
    dbms_output.put_line('no_of_days        : ' || :NEW.no_of_days);
    dbms_output.put_line('doctor_charge   : ' || :NEW.doctor_fee);
    dbms_output.put_line('room_charge      : ' || :NEW.room_fee);
    dbms_output.put_line('lab_charge        : ' || :NEW.lab_fee);
    dbms_output.put_line('total               : ' || :NEW.total);
```

END;

/

```
53  /
54
55  --------------------------------------------------------------------------------
56
57  CREATE OR REPLACE TRIGGER display_changes_b
58  After insert or update ON Bill
59  FOR EACH ROW
60  WHEN (NEW.bill_no > 0)
61  BEGIN
62      dbms_output.put_line('After Insertion Bill Data is following :');
63      dbms_output.put_line('Patient ID      : ' || :NEW.patient_id);
64      dbms_output.put_line('bill No         : ' || :NEW.bill_no);
65      dbms_output.put_line('no_of_days      : ' || :NEW.no_of_days);
66      dbms_output.put_line('doctor_charge   : ' || :NEW.doctor_fee);
67      dbms_output.put_line('room_charge     : ' || :NEW.room_fee);
68      dbms_output.put_line('lab_charge      : ' || :NEW.lab_fee);
69      dbms_output.put_line('total           : ' || :NEW.total);
70
71  END;
72  /
```

```
Trigger created.

Trigger created.

Trigger created.
```

Below query also shows implementation of procedure , function , cursors , triggers in PLSQL .

creation:

Add New Patient :- This module is used to add a new patient.

Add New Bill :- This module is used to add a new bill to patient.

Add New Doctor :- This module is used to add a new Doctor.

Add New Nurse :- This module is used to add a new Nurse.

Snapshot of creating a new patient I.e. insertion of a new patient data into Patient and MedRecord table.

If data of patient or employee is already present then it will show a message that data is already present .

SQL Worksheet

```
163  procedure createPatient
164  is
165  begin
166  dbms_output.put_line('Enter id of patient : ');
167   --p_id := &p_id;
168   declare
169    any_rows_found number;
170  begin
171    select count(*)
172    into   any_rows_found
173    from   Patient
174    where  patient_id = p_id;
175
176    if any_rows_found = 1 then
177      dbms_output.put_line('Patient is already present');
178   else
179  dbms_output.put_line('Enter name : ');
180   --name := &name;
181  dbms_output.put_line('Enter age of patient : ');
```

```
Enter record id :
Enter desease of patient :
After Insertion Patient Data is following :
ID         : p1
Name       : himanshu
Age        : 20
gender     : male
doctor_id  : n5
ID         : r1
Disease    : Asthma
```

Deletion of data :

Delete :- This module deletes the entry of a patient or a Doctor details.

Deleting a patient record so have to delete MedRecord and Bill record of Patient too.

If data of patient or Employee is not present then will show a message that the data is not present.

SQL Worksheet

```
372
373  procedure delete_patient
374  is
375  begin
376  dbms_output.put_line('Enter id of patient : ');
377   --p_id := &p_id;
378  declare
379    any_rows_found number;
380  begin
381    select count(*)
382    into   any_rows_found
383    from   Patient
384    where  patient_id = p_id;
385
386    if any_rows_found = 1 then
387  delete from Patient
388  where patient_id=p_id;
389  delete from MedRecord
390  where patient_id=p_id;
391  delete from Bill
392  where patient_id=p_id;
393  dbms_output.put_line('Deleted');
394  else
```

```
1. Delete details of Patient
2. Delete details of Doctor
3. Delete details of Nurse
Enter your choice :
Enter id of patient :
Deleted
```

Snapshots of Updation:

Updation module:- This module used to store or produce the patient reports and Doctor Details .

Updating a Patient Record age or disease or doctor assigned to patient can be change as shown below.
If data is not present then it will show that entity is not present.

SQL Worksheet

```
30  ------------------------------------------------------------------------
31  ------------------------------------------------------------------------
32
33
34  procedure updatePatient
35  is
36  begin
37  dbms_output.put_line('Enter id of patient : ');
38   --p_id := &p_id;
39  declare
40    any_rows_found number;
41  begin
42    select count(*)
43    into   any_rows_found
44    from   Patient
45    where  patient_id = p_id;
46
47    if any_rows_found = 1 then
48      dbms_output.put_line('Enter age of patient : ');
49        --age:=&age;
50      dbms_output.put_line('Enter doctor id : ');
51        --d_id := &d_id;
52      dbms_output.put_line('Enter desease of patient : ');
```

```
After Insertion Patient Data is following :
ID          : p1
Name        : himanshu
Age         : 20
gender      : male
doctor_id   : n5
```

Viewing or Searching Data

View Details :- This module is used to view the details of a admitted patient and Doctor appointed in hospital.

Search :- This module searches the record of desired patient using his ID and Doctor using doctor id.

Below snapshot contains searching bill details and geting bill details of Patient .

If data is not present then it will show that entity is not present.

SQL Worksheet

```
401
402
403  procedure viewBill
404  is
405  begin
406  dbms_output.put_line('Enter the id of Patient');
407  --p_id := &p_id;
408  declare
409    any_rows_found number;
410  begin
411    select count(*)
412    into   any_rows_found
413    from   Patient
414    where  patient_id = p_id;
415
416    if any_rows_found = 1 then
417    declare
418    p_name Patient.name%type;
419    p_doctor_fee Bill.doctor_fee%type;
420    p_room_fee  Bill.room_fee%type;
421    p_lab_fee Bill.lab_fee%type;
```

```
5. View Room details
Enter your choice :
Enter the id of Patient
name          : himanshu
Doctor fee    : 100
Room fee      : 10000
Lab fee       : 100
Total amount  : 10200
```

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.