Hospital Management System

Abstract:

The Hospital Management Software simplifies the work of healthcare professionals and their interactions with their patients. As we move toward digitizing health care, this takes the tedious task of managing paperwork online.

The potential users of this system are:
1. Receptionist (Appointments, Room Info, etc.)
2. Nurse
3. Doctor
4. Accountant
5. System Admin

Objectives:

- Describe core concepts of database and model a database management system through ER modeling.
- Apply knowledge of relational algebra and structured query language to retrieve and manage data from relational databases.
- To study the functioning of the Hospital management System.
- To make a software fast in processing, with a good user interface, efficient to use. It should be used for a long time without error and maintenance.
- Provide a comprehensive introduction to the fundamental concepts for design and development of database systems, with an emphasis on how to organize, maintain and retrieve - efficiently, and effectively in a hospital management system
- Reducing paperwork.

Current limitations:

- In the hospital management system the records of patients / doctors have to be maintained manually which is cumbersome and takes a lot of time.
- The Storage requires large amount of files which takes up large amount of space
- A lot of paperwork is required as they are maintained in files and registers

Proposed System:

There is no consensus regarding the hospital database management system. In order to have a successful hospital database management system, we need to make many decisions related to the patient, drug price, the doctor's specialty, and the doctor's year of experience. Each record should be added in a way to increase the scalability. Hospital management is more complicated than other types of database management because it affects a large number of people who need the right information.

Software requirements:

Frontend -

Python, along with packages such as -

- CX_Oracle - for enabling the connection of the Oracle Database to the Python IDE.
- SQLAlchemy - for executing SQL commands in a proper syntax and efficiently in Python

Backend -

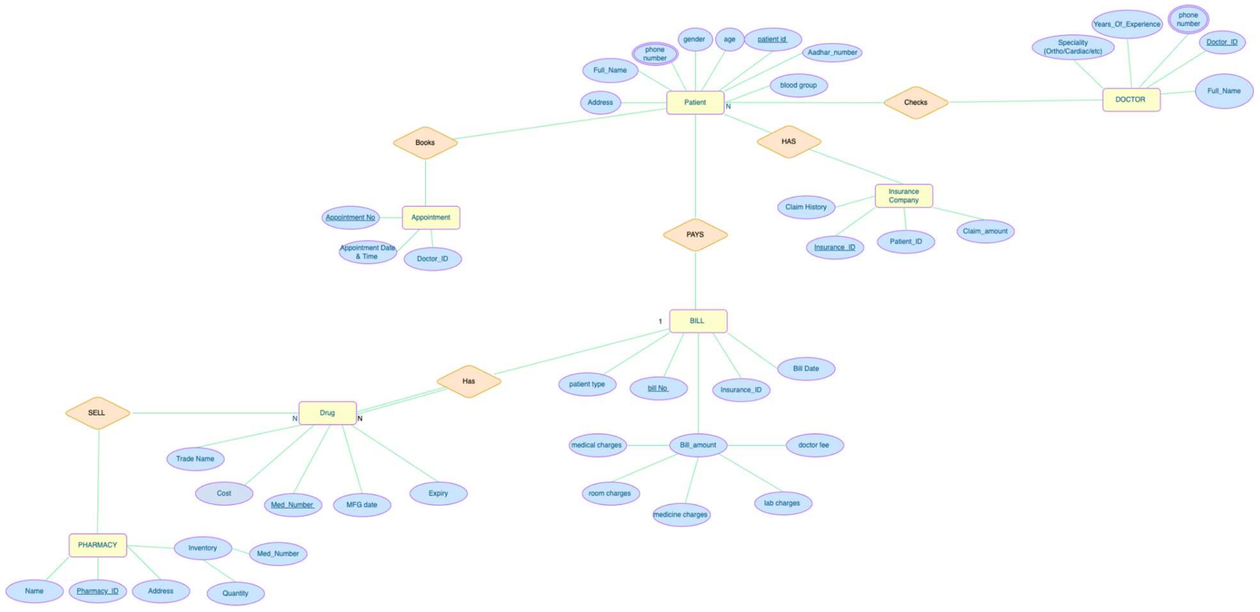
- SQL Developer - for creating tables, updating values
- Oracle XE - for the database to be stored along with the libraries
- Oracle instaclient - for enabling the connection between Python and Oracle Database

Topics Covered

1. **ER Diagram** of the hospital database system
2. **Translating the ER diagram** into relational database tables
3. **Back end** of the hospital database management system

4. **Front end** of the hospital database management system
5. **Modules** necessary for an efficient hospital management system

ER Diagram



Front End Code

```
import cx_Oracle

cx_Oracle.init_oracle_client(lib_dir=r"C:\Users\happy\Downloads\Programs\oracle_instant_client\in
   stantclient_21_7") #for assigning path

connect= cx_Oracle.connect("SYSTEM/1234@localhost:1521/xe")

curr=connect.cursor()

curr.execute("ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YY'") #for setting uniform date format

#module runner codes-----



#Appointment mgmt module

def New_Appointment():

   appointment_number = int(input("Enter Appointment Number: "))
   date_of_appt = input("Enter Date of Appointment (DD-MM-YY): ")

   time_of_appt = input("Enter Time of Appointment: ")

   doctor_id_for_appt = input("Enter the Doctor for the appointment: ")

   values_for_appt = (appointment_number, date_of_appt, time_of_appt, doctor_id_for_appt)

   print(values_for_appt)

   query_for_appt = "INSERT INTO Appointment VALUES(:appt_no, to_date(:date_appt), :time_appt,
    :doctor_id_for_appt)"

   curr.execute(query_for_appt, values_for_appt)

   connect.commit()

   print("Appointment Created! \n")



def Check_existing_appointment():

   Query_for_appts="SELECT * FROM Appointment"

   print("The existing appointments are: \n")

   print("Appointment Number, Date, Time, Doc_ID")

   curr.execute(Query_for_appts)

   rows=curr.fetchall()

   for a in rows:

       print(a)

   curr.close()



#Billing management

def print_bills():

   patient_id_for_bill_details = input("Enter Patient_ID of the patient whose bill details you
    want: ")
   bill_details_for_patient_query = "SELECT * FROM Bill NATURAL JOIN Bill_Amount WHERE Patient_ID
   = :pat_id"

   print("The bill details for patient are: \n")

   print("Bill No, Date, Patient_ID, Costs")



   for row in curr.execute(bill_details_for_patient_query, pat_id = patient_id_for_bill_details):

       print(row)

   curr.close()



def make_a_bill():

   Bill_Number = input("Enter Bill No: ")

   Bill_Date = input("Enter date of the bill (DD-MM-YY): ")

   Bill_Patient_ID = input("Enter Patient_ID: ")

   Medical_Charges = input("Enter Medical Charges: ")

   Room_Charges = input("Enter Room Charges: ")

   Medicine_Charges = input("Enter Medicine Charges: ")

   Lab_Charges = input("Enter Lab Charges: ")

   Doctor_fee = input("Enter Doctor Fee: ")

   Values_Main_Bill = (Bill_Number, Bill_Date, Bill_Patient_ID)

   Values_Bill_Amt = (Bill_Number, Medical_Charges, Room_Charges, Medicine_Charges, Lab_Charges,
   Doctor_fee)

   query_main = "INSERT INTO Bill Values (:Bill_no, to_date(:bill_date), :bill_pat_id)"

   query_amt = "INSERT INTO Bill_Amount VALUES (:Bill_no, :med_char, :room_char, :medi_charg,
   :lab_char, :doc_fee)"

   curr.execute(query_main, Values_Main_Bill)
```

```
    curr.execute(query_amt, Values_Bill_Amt)

    connect.commit()

    print("Bill created!\n")



#Doctor module

def fetch_doc_details():

    query_doc_details = "SELECT * FROM Doctor"

    print("Existing doctors are: \n")

    curr.execute(query_doc_details)

    print("Doc_ID, Full_Name, Phone Number, Years of Experience, Speciality")

    rows=curr.fetchall()

    for a in rows:

        print(a)

    curr.close()



def add_new_doc():

    query_new_doc="INSERT INTO Doctor VALUES(:docid,:fn,:prino,:yoe,:spec)"

    doctor_id = input("Enter Doctor ID: ")

    full_name_doc = input("Enter Full Name of the Doctor: ")

    Primary_Phone_No = int(input("Enter the Primary Phone Number of the Doctor: "))

    Yoe = int(input("Enter the Years of Experience of the Doctor: "))

    Speciality = input("Enter the speciality of the Doctor: ")

    Values_for_new_doc = (doctor_id, full_name_doc, Primary_Phone_No, Yoe, Speciality)

    curr.execute(query_new_doc, Values_for_new_doc)
```

```python
    connect.commit()
    print("Doc added. \n")
    connect.commit()


#Drug Mgmt.
def add_new_drug():
    med_number = input("Enter the Medicine Number")
    trade_name = input("Enter the Trade Name of the Drug: ")
    cost = input("Enter Cost of the Drug: ")
    Mfg_Date = input("Enter the MFG Date(YYYY-MM-DD): ")
    Expiry_Date = input("Enter the Expiry Date(YYYY-MM-DD): ")
    drug_info=(med_number, trade_name, cost, Mfg_Date, Expiry_Date)
    new_drug_query = "INSERT INTO Drugs (%s,%s,%s,%s)"
    curr.execute(new_drug_query, drug_info)
    print("New drug added. \n")


def display_drugs():
    query_for_all_drugs = "SELECT * FROM Drugs"
    print("All existing drugs in the system: \n")
    print("DRUG_NO, Name, Cost, Manf. Date, Exp Date")
    curr.execute(query_for_all_drugs)
    rows=curr.fetchall()
    for a in rows:
        print(a)
    curr.close()

#Insurance module

def check_insuarance():

    patient_id_for_insurance_check = input("Please enter the Patient_ID for which you want to
    check the Insuarance Status: ")

    query_for_ic = "SELECT Max_Claim_Amt FROM INSUARANCE WHERE PATIENT_ID= :pat_id_insuarance"

    print("Max claim amount of patient is: \n")

    try:

        for row in curr.execute(query_for_ic, pat_id_insuarance = patient_id_for_insurance_check):

            print(row)

    except:

        print("Patient doesn't have an Insuarnace.")

    curr.close()



#patient mgmt.

def add_patient():

    register_query="INSERT INTO Patient VALUES (:pat_id_reg, :aadhar, :full_Name, :address,
    :gender, :age, :Bg, :phnno)"

    Patient_ID = input("Enter Patient_ID: ")

    Aadhar_no= input("Enter Aadhar No: ")

    Full_Name= input("Enter Full Name: ")

    Address = input("Enter address: ")

    Gender = input("Enter Gender (M/F/NB/OT): ")

    Age = int(input("Enter Age: "))

    Blood_Group = input("Enter Blood Group: ")
    Phone_Number = int(input("Enter Phone Number: "))

    values1= (Patient_ID, Aadhar_no, Full_Name, Address, Gender, Age, Blood_Group, Phone_Number)

    print(values1,  register_query)

    curr.execute(register_query, values1)

    connect.commit()

    print("Patient registered! \n")



def check_patients():

    patient_show_query = "SELECT * FROM Patient"

    print("Existing Patients are: \n")

    print("Patient_ID, Aadhar, Full Name, City, Gender, Age, Blood_Group, Phone_Number")

    curr.execute(patient_show_query)

    rows=curr.fetchall()

    for a in rows:

        print(a)

    curr.close()



#driver code

print("Options: ")

print("\n 1 - Appointment Management \n 2 - Billing Management \n 3 - Doctor Management \n 4 -
    Drug Management \n 5 - Insuarance Module \n 6 - Patient Management")

choice = int(input("Enter your choice: "))

if choice == 1:

    print("1 - Create new Appointment \t 2 - Check existing appointments \n")
    choice1=int(input("Enter your choice: "))

    if choice1==1:

        New_Appointment()

    elif choice1 == 2:

        Check_existing_appointment()

    else:

        print("Wrong Input, please try again. ")

elif choice == 2:

    print("\n 1 - Print existing Bills  \t 2 - Make a new bill \n")

    choice2=int(input("Enter your choice: "))

    if choice2==1:

        print_bills()

    elif choice2 == 2:

        make_a_bill()

    else:

        print("Wrong Input, please try again. ")

elif choice == 3:

    print("\n 1 - Fetch all doc details \t 2 - Add new doc \n")

    choice3=int(input("Enter your choice: "))

    if choice3==1:

        fetch_doc_details()

    elif choice3 == 2:

        add_new_doc()

    else:
        print("Wrong Input, please try again. ")

elif choice == 4:

    print("\n1 - Add a new drug \t 2 - Fetch all drugs")

    choice4=int(input("Enter your choice: "))

    if choice4==1:

        add_new_drug()

    elif choice4 == 2:

        display_drugs()

    else:

        print("Wrong Input, please try again. ")

elif choice == 5:

    check_insuarance()

elif choice == 6:

    print("\n1 - Add a new Patient \t 2 - Fetch all Patients \n")

    choice4=int(input("Enter your choice: "))

    if choice4==1:

        add_patient()

    elif choice4 == 2:

        check_patients()

    else:

        print("Wrong Input, please try again. ")

else:

    print("Wrong Input, please try again. ")
```

Back end Database (Table Creation)

```
CREATE TABLE Patient (Patient_ID Varchar(25) PRIMARY KEY, Aadhar_No Varchar(10) NOT NULL,
Full_Name Varchar(25), Address Varchar(100), Gender Varchar(2), Age number(3), Blood_Group
Varchar(2), Phone_Number Number(12));
CREATE TABLE Doctor (Doctor_ID Varchar (25) PRIMARY KEY, Full_Name_Doc Varchar(25),
Primary_Phone_Number Number(12), Years_of_Experience Number(2), Speciality Varchar(10));
CREATE TABLE Appointment(Appointment_Number Number(10) PRIMARY KEY, Date_of_appointment Date,
Time_of_appointment Varchar(10), Doctor_ID Varchar(25) FOREIGN KEY (Doctor_ID) REFERENCES
Doctor(Doctor_ID));
CREATE TABLE Pharmacy (Pharmacy_ID Varchar(25) PRIMARY KEY, Pharmacy_Name Varchar(15) NOT
NULL, Pharmacy_Address Varchar (100));
CREATE TABLE Drugs (Med_Number Number (15) PRIMARY KEY, Trade_Name Varchar(25), Cost Number
(15), MFG_Date Date, Expiry_Date DATE);
CREATE TABLE Pharmacy_Inventory(Pharmacy_ID Varchar(25), Med_Number Number(15), Quantity
Number(15), FOREIGN KEY (Pharmacy_ID) REFERENCES Pharmacy(Pharmacy_ID), FOREIGN
KEY(Med_Number)REFERENCES Drugs(Med_Number));
CREATE TABLE Insuarance (Insuarance_ID Varchar (25) PRIMARY KEY, Patient_ID Varchar(25),
Max_Claim_Amt Number(10), Claim_History Varchar(25), FOREIGN KEY (Patient_ID) REFERENCES
Patient(Patient_ID)); -- history = null/yes/2/3 etc
CREATE TABLE Bill (Bill_Number Varchar(25) PRIMARY KEY, Bill_Date DATE, Patient_ID
Varchar(25), FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID));
CREATE TABLE Bill_Amount (Bill_Number Varchar(25), Medical_Charges Number(10), Room_Charges
Number(10), Medicine_Charges Number(10), Lab_Charges Number(10), Doctor_fee
Number(15),FOREIGN KEY (Bill_Number) REFERENCES Bill(Bill_Number));
commit;
```