Project for Database Design

Phase IV. Documentation

Ayan Paul
axp170036@utdallas.edu

Jiten Girdhar
jxg170021@utdallas.edu

Rajarshi Chattopadhyay
rxc170010@utdallas.edu

Poonam Gillurkar
pxg180009@utdallas.edu

0. Pre-Illumination

In Section 1 we gave problem description copied from Web site; in Section 2 we answered 3 questions listed in the project and justified our solution; in Section 3 we exhibited EER diagram with all assumptions; in Section 4 we showed our relational schema after normalization; in Section 5 we gave all requested SQL statements for both views and queries; and in Section 6 we gave dependency diagram induced from relational schemas. Finally, a short summary is given at the end of this report.

1. Problem Description

Dallas Care is a hospital and medical care center. Dallas Care would like one relational database to be able to smoothly carry out their work in an organized way. The hospital has following modules: Person, Employee, Patient, Visitors, Pharmacy, Treatment, Rooms, Records and Medical Bill Payment.

A Person can be an Employee or a Class 1 Patient. Details of a person such as Person ID, Name (First, Middle, Last), Address, Gender, Date of Birth, and Phone number (one person can have more than one phone number) are recorded. A person ID should be in the format, 'PXXX', where XXX can be a value between 100 and 999. A Class 1 patient is a person who visits the hospital just for a doctor consultation. A person can be both an employee and a Class 1 patient.

Employee is further classified as Doctors, Nurses or Receptionists. The start date of the employee is recorded. The specialization of the doctor is stored and doctors are further classified into Trainee, Permanent or Visiting. Every Class 1 patient consults a doctor. A Class 1 patient can consult at most one doctor but one doctor can be consulted by more than one Class 1 patient.

A Class 2 patient is someone who is admitted into the hospital. A Class 2 patient can be an Employee or a Class 1 Patient or both. A doctor attends Class 2 patients. One doctor can attend many Class 2 patients but a Class 2 patient can be attended to by at most 2 doctors. The date of patient being admitted into the hospital is recorded.

A Visitor log is maintained for the Class2 Patients, which stores information such as patient ID, visitor ID, visitor name, visitor's address, and visitor's contact information.

Pharmacy details such as Medicine code, Name, Price, Quantity and Date of expiration is recorded. The database also stores the information of the various kinds of treatments that are offered in the hospital. The treatment details such as ID, name, duration and associated medicines are recorded. When a treatment is assigned to a Class 2 patient, the treatment details, medicine details and patient details are recorded so that the doctor can easily access this information.

Nurses govern rooms. Each nurse can govern more than one room, but each room has only one nurse assigned to it. The room details such as room ID, room type and duration is recorded. Each Class 2 patient is assigned a room on being admitted to the hospital.

A records database is maintained by the receptionist who keeps record of information such as record ID, patient ID, date of visit, appointment and description. The receptionist also records the payment information with the patient's ID, date of payment and the total amount due. Payment is further classified into Cash or Insurance. A person can pay by cash, or by insurance or pay via a combination of both. The cash amount is recorded if a person pays by cash. For Insurance, the insurance details such as Insurance ID, Insurance Provider, Insurance coverage and the amount is recorded.

2. Three Questions

2.1 Is the ability to model superclass/subclass relationships likely to be important in a hospital environment such as Dallas Care? Why or why not?

In the process of designing our entity relationship diagram for a database, we may find that attributes of two or more entities overlap, meaning that these entities seem very similar but still have a few differences. In this case, we may create a subtype of the parent entity that contains distinct attributes. A parent entity becomes a super type that has a relationship with one or more subtypes. These are similar to "Disjoint" and "Union" Operation of an EER diagram model also known as Specialization and Generalization.

The following problem condition directs us to use subclass structure in the environment:
1> A Person can be an Employee or a Class 1 Patient
Here Employee and Class 1 Patient are subclass of Person

2>A person can be both an employee and a Class 1 patient
Person is superclass of employee and a Class 1 patient

3>Employee is further classified as Doctors, Nurses or Receptionists
Doctors, Nurses or Receptionists are subclasses of the Employee class.

2.2 Can you think of 5 more rules (other than the one explicitly described above) that are likely to be used in a school environment? Add your rules to the above requirement to be implemented.

Rules likely to be applied in a school environment are:

1. A person can be a student, teacher or school staff.
2. Each teacher must teach at least one course E
3. Every student must take at least three courses.
4. A staff member can also be a student.
5. Every student must enroll for at most two extracurricular activities.

2.3 Justify using a Relational DBMS like Oracle for this project.

We could have used OODBMS to represent the problem statement but b because of the following reasons:
Lack of standards: There is a general lack of standards of OODBMSs. We have already mentioned that there is not universally agreed data model. Similarly, there is no standard object-oriented query language.

Competition: Perhaps one of the most significant issues that face OODBMS vendors is the competition posed by the RDBMS and the emerging ORDBMS products. These products have an established user base with significant experience available. SQL is an approved standard and the relational data model has a solid theoretical formation and relational products have many supporting tools to help .both end-users and developers.

Query optimization compromises encapsulations: Query optimization requires. An understanding of the underlying implementation to access the database efficiently. However, this compromises the concept of incrassation.

Locking at object level may impact performance Many OODBMSs use locking as the basis for concurrency control protocol. However, if locking is applied at the object level, locking of an inheritance hierarchy may be problematic, as well as impacting performance.

Complexity: The increased functionality provided by the OODBMS (such as the illusion of a single-level storage model, pointer sizzling, long-duration transactions, version management, and schema evolution--makes the system more complex than that of traditional DBMSs. In complexity leads to products that are more expensive and more difficult to use.

Lack of support for views: Currently, most OODBMSs do not provide a view mechanism, which, as we have seen previously, provides many advantages such as data independence, security, reduced complexity, and customization.

3. EER Diagram with all assumptions

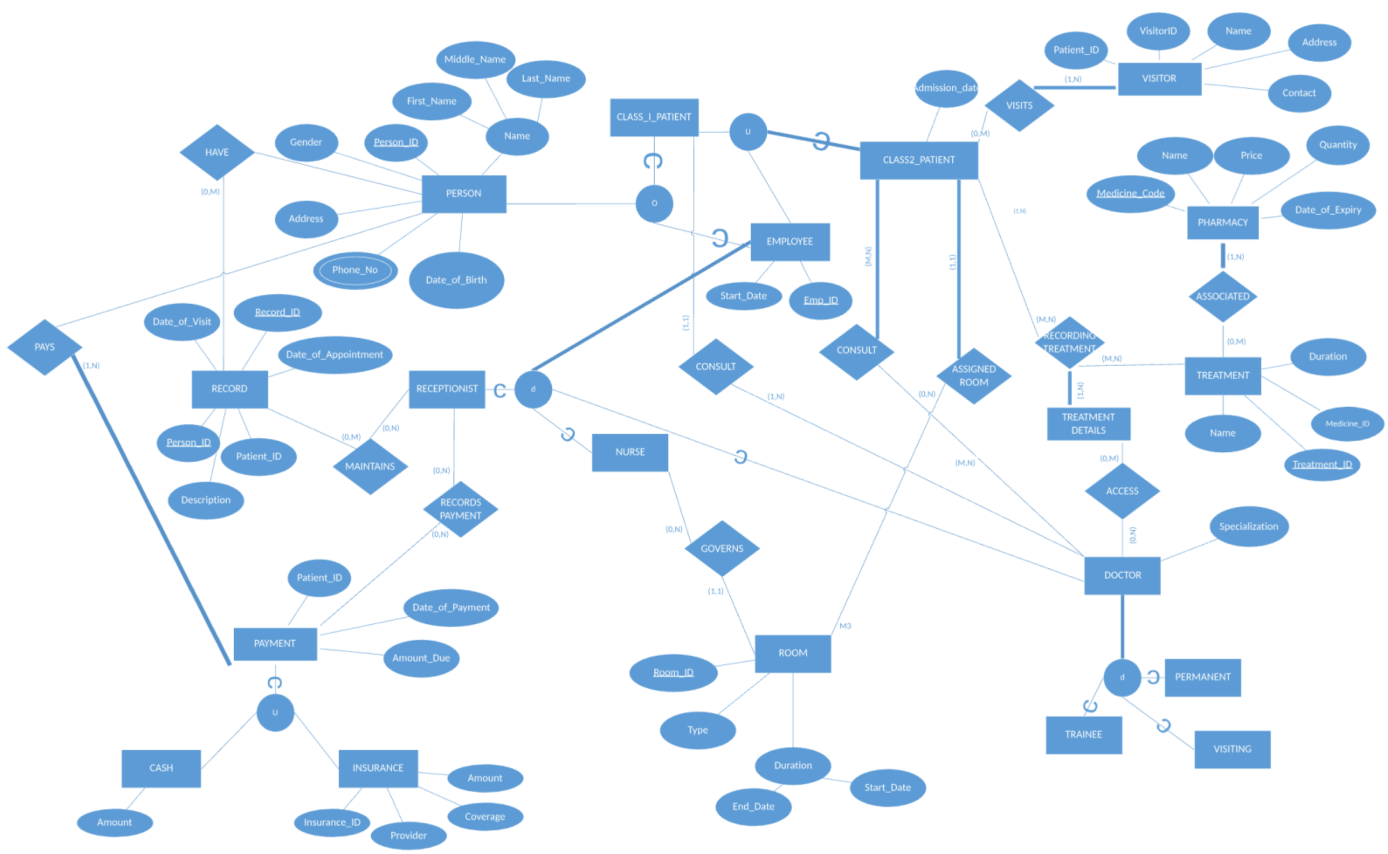


Fig 1. EER Diagram

4. Relational Schema in Third Normal Form

4.1 Relational Schema

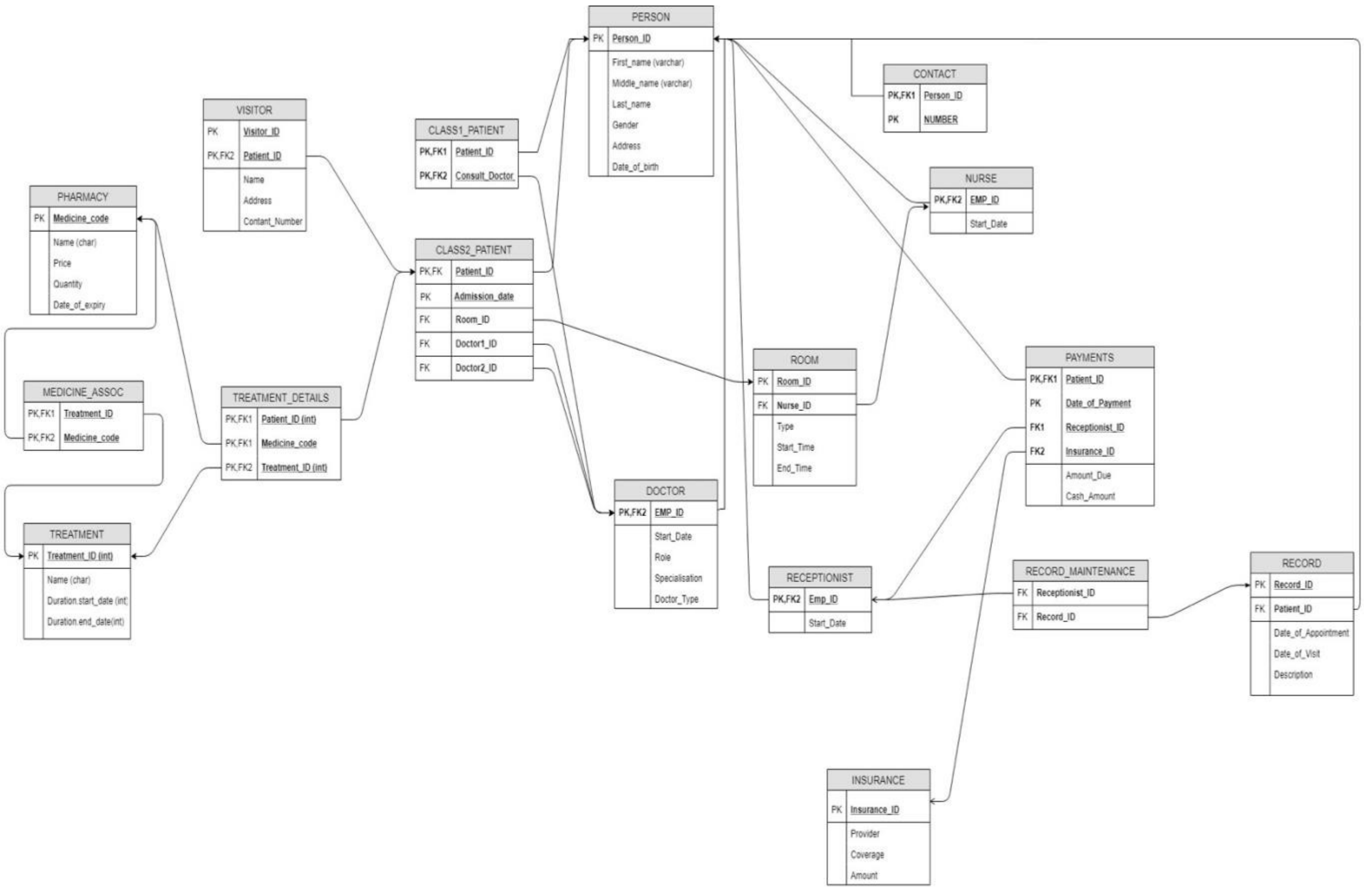


Fig 2. Normalized Relational Schema

4.2 Format for Every Relation

Firstly, according to the requirement of phase III and with purpose to simplify the relational model for this database, we have set the relations/tables conforming to 3NF Normalization. There is no transitive dependency of the non-prime attributes of a relation to the key attribute:

- The PERSON table has primary key Person_Id. The other attributes are non-prime and are directly functionally dependent on the primary key.

PERSON

<u>Person_Id</u>
First_Name
Middle_Name
Last_Name
Gender
Address
Dob

- Since a person can have multiple contact numbers, and more than one person can have the same contact number (for example a minor has same contact as of their parent), so a separate table CONTACT with super key Person_id and Number is created. This also follows 3NF as there is no non-prime key. Fk_Person is the foreign key referencing Person_Id of PERSON table.

CONTACT

<u>Person_id</u>
<u>Number</u>

- Class1 Patient is a Person who can consult only one doctor. So CLASS1_PATIENT is comprised of Patient_id, which with Date_of_appointment acts as the primary key. Patient_id is the foreign key referencing Person_Id which is the primary key of PERSON. The non-prime attribute Consult_doctor is the foreign key to the DOCTOR table and is functionally dependent on the primary key.

CLASS1_PATIENT

<u>Patient_id</u>
<u>Date_of_appointment</u>
Consult_doctor

- Class2 Patient is identified by a person and the admission date. So CLASS2_PATIENT comprises of Patient_id, which is the foreign key referencing to Person_Id of PERSON table, and the Admission_Date, which together act as the composite primary key. The other attribute Room_id is the foreign key which references to Room_Id of the ROOM table, and it is directly functionally dependent on the primary key attribute.

CLASS2_PATIENT

<u>Patient_id</u>
<u>Admission_Date</u>
Fk_Room

- Since a Class2 Patient can consult multiple Doctors and a Doctor can be consulted by multiple Class2 Patients, so a separate relation CONSULTATION is created which contains foreign keys to primary key of CLASS2_PATIENT and primary key of DOCTOR as the super key. The primary key of this table is the set of Patient_id, Admission_date and Doctor_id. Doctor_id is the foreign key to Emp_id of DOCTOR.

CLASS2_PATIENT_CONSULTATION

<u>Patient_id</u>
<u>Admission_date</u>
<u>Doctor_id</u>

- The VISITOR table has Visitor_Id and Patient_id, which is the foreign key referencing to the primary key of CLASS2_PATIENT, as the composite primary key. This is because a visitor can have multiple class 2 patients. The other non-prime attributes are directly functionally dependent on the primary key.

VISITOR

Visitor_Id
Patient_id
Name
Address
Contact

- The TREATMENT_DETAILS table has the foreign keys Patient_id referencing to primary key of CLASS2_PATIENT, Medicine_id referencing to primary key of MEDICINE, and Treatment_id referencing to primary key of TREATMENT as the key. As we know that X -> X is True, in this way we define the functional dependency for this table.

TREATMENT_DETAILS

Patient_id
Medicine_id
Treatment_id

- The MEDICINE_ASSOC table has the foreign keys Treatment_id referencing to primary key of TREATMENT and Medicine_id referencing to primary key of MEDICINE as the super key. This table is created to signify that multiple medicines can be used for a treatment, and multiple treatments can require the same medicine. We define the functional dependency using the above given property of the functional dependency.

MEDICINE_ASSOC

Treatment_id
Medicine_id

- The TREATMENT table has primary key Treatment_Id, and the other non-prime attributes Name, Duration_No and Duration_Unit are directly functionally dependent on the primary key.

TREATMENT

Treatment_Id
Name
Duration_No
Duration_Unit

- The PHARMACY table has primary key Medicine_Code, and the other non-prime attributes Name, Price, Quantity, and Expiry_Date are directly functionally dependent on the primary key.

PHARMACY

Medicine_Code
Name
Price
Quantity
Expiry_Date

- The DOCTOR table has primary key Emp_id, which is the foreign key referencing to Person_Id of PERSON table. The non-prime attributes Start_Date, Role, Specialization, and Doc_Type are directly functionally dependent on the primary key.

DOCTOR

Emp_id
Start_Date
Role
Specialization
Doc_Type

- The RECEPTIONIST table has primary key Emp_id, which is the foreign key referencing to Person_Id of PERSON table. The other non-prime attribute Start_Date, is directly functionally dependent on the primary key.

RECEPTIONIST

Emp_id
Start_Date

- The NURSE table has primary key Nurse_id, which is the foreign key referencing to Person_Id of PERSON table. The other non-prime attribute Start_Date, is directly functionally dependent on the primary key.

NURSE

Nurse_id
Start_Date

- The ROOM table has the primary key Room_Id. The Nurse_id is the foreign key referencing to primary key of NURSE table. This, along with the other non-prime attributes Room_Type, Start_Time, End_Time are functionally dependent on the primary key.

ROOM

Room_Id
Nurse_id
Room_Type
Start_Time
End_Time

- The RECORD table has the primary key Record_Id. The Patient_id is the foreign key referencing to primary key of PERSON table. This, along with the other non-prime attributes Receptionist_id, Appointment_Date, Visit_Date, and Description, are directly functionally dependent on the primary key.

RECORD

Record_Id Receptionist_id

Patient_id
Appointment_Date
Visit_Date
Record_description

- The INSURANCE table has primary key Insurance_Id. The other non-prime attributes Provider, Coverage, and Amount are directly functionally dependent on the primary key.

INSURANCE

<u>Insurance_Id</u>
Provider
Coverage
Amount

- The PAYMENT table has the composite primary key Patient_id, which is the foreign key referencing to primary key of PERSON table, and Payment_Date. This, along with the other non-prime attributes Insurance_Id which is the foreign key referencing to primary key of INSURANCE table, Amount_Due, Cash_Amount, and Receptionist_id:which is the foreign key referencing to primary key of RECEPTIONIST table, are functionally dependent on the primary key.

PAYMENT

<u>Patient_id</u>
<u>Payment_Date</u>
Receptionist_id
Insurance_id
Amount_Due
Cash_Amoount

5. All requested SQL Statements

5.1 Creation of Database with SQL Statements

After normalizing every relational schema into third normal form and modifying some details, it is the time to implement our database using SQL languages into Oracle.

5.1.1 Table Creation

Using SQL statement, we created the tables as follows:

```
create database FINALPROJECT;

USE FINALPROJECT;

create table PERSON(
Person_ID varchar(4) check(length(Person_ID)=4 and Person_ID like 'P%' and
cast(substr(PersonID,2,3) as decimal)>=100 and cast(substr(PersonID,2,3) as decimal)<=999),
First_name varchar(50) not null,
Middle_name varchar(50), Last_name varchar(50) not null,
Gender char(1) check(Gender in ('M','F','O')), Address varchar(100) not null,
Date_Of_Birth date check(Date_of_Birth<=curdate()), primary key(Person_ID));
```

create table CONTACTS(Person_ID varchar(4) not null,
Phone_number decimal(10) unique check(length(Phone_number)=10) , primary key(Person_ID,Phone_number),
foreign key(Person_ID) references PERSON(Person_ID) on delete cascade on update cascade);

create table NURSE(Emp_ID varchar(4),
Start_date date not null check(Start_date<=curdate()), primary key(Emp_ID),
foreign key(Emp_ID) references PERSON(Person_ID) on update cascade);

create table DOCTOR(Emp_ID varchar(4),
Start_date date not null check(Start_date<=curdate()), Specilization varchar(20),
Doctor_type varchar(15) check(Doctor_role in ('Trainee','Permanent','Visiting')), primary key(Emp_ID),
foreign key(Emp_ID) references PERSON(Person_ID) on update cascade);

create table RECEPTIONIST(Emp_ID varchar(4),
Start_date date not null check(Start_date<=curdate()), primary key(Emp_ID),
foreign key(Emp_ID) references PERSON(Person_ID) on update cascade);

create table CLASS1_PATIENT(Patient_ID varchar(4) not null,
Date_of_appointment date not null check(Date_of_appointment<=curdate()), Consult_doctor varchar(4) not null,
primary key(Patient_ID,Date_of_appointment),
foreign key(Patient_ID) references PERSON(Person_ID) on update cascade, foreign key(Consult_doctor) references DOCTOR(Emp_ID) on update cascade);

create table ROOM(Room_ID varchar(5), Nurse_ID varchar(4) not null,
Room_type varchar(10) not null,
start_date time check(start_date<=curdate()), end_date time check(enddate>curdate()), primary key(Room_ID),
foreign key(Nurse_ID) references NURSE(Emp_ID) on update cascade,
check(end_date>start_date));

create table CLASS2_PATIENT(Patient_ID varchar(4),
Room_ID varchar(5) not null,
Admission_date date check(Admission_date<=curdate()), primary key(Patient_ID,Admission_date),
foreign key(Patient_ID) references PERSON(Person_ID) on update cascade, foreign key(Room_ID) references ROOM(Room_ID) on update cascade);

create table CLASS2_PATIENT_CONSULTATION(Patient_ID varchar(4) not null, Admission_date date check(Admission_date<=curdate()), Consult_doctor varchar(4) not null, primary key(Patient_ID,Admission_date,Consult_doctor), foreign key(Patient_ID,Admission_date) references CLASS2_PATIENT(Patient_ID,Admission_date),
foreign key(Consult_doctor) references DOCTOR(Emp_ID) on update cascade);

create table RECORD(Record_ID varchar(7),
Recetionist_ID varchar(4) not null, Patient_ID varchar(4) not null,
Date_of_appointment date not null check(Date_of_appointment>=curdate()), Date_of_visit date not null check(Date_of_visit>=curdate()), Record_description varchar(200),
primary key(Record_ID),
foreign key(Patient_ID) references PERSON(Person_ID) on update cascade,
foreign key(Recetionist_ID) references RECEPTIONIST(Emp_ID) on update cascade);

create table INSURANCE(Insurance_ID varchar(10), Provider varchar(30) not null, Coverage decimal(10) not null, Amount decimal(10) not null, primary key(Insurance_ID));

create table PAYMENTS(Patient_ID varchar(4),
Date_of_payment date not null check(Date_of_payment>=curdate()), Recetionist_ID varchar(4) not null,
Insurance_ID varchar(10),
Amount_due decimal(10) not null check(Amount_due>=0), Cash_amount decimal(10) not null default 0 check(Cash_amount>=0), primary key(Patient_ID,Date_of_payment),
foreign key(Patient_ID) references PERSON(Person_ID),
foreign key(Recetionist_ID) references RECEPTIONIST(Emp_ID), foreign key(Insurance_ID) references INSURANCE(Insurance_ID));

create table VISITOR(Visitor_ID varchar(10), Patient_ID varchar(4) not null,
Visitor_name varchar(30) not null, Visitor_address varchar(50) not null, Contact_info decimal(10),
primary key(Visitor_ID,Patient_ID),
foreign key(Patient_ID) references CLASS2_PATIENT(Patient_ID));

create table PHARMACY(Medicene_code varchar(6), Medicene_name varchar(20) not null,
Price decimal(10,2) not null check(Price>0), Quantity decimal(4) not null check(Quantity>=0),
Date_of_expiry date not null check(Date_of_expiry>=curdate()), primary key(Medicene_code));

create table TREATMENT(Treatment_ID varchar(6), Treatment_name varchar(20) not null,
Duration decimal(3,1) not null check(Duration_number>0),
Duration_unit varchar(10) not null check(Duration_unit in ('Months','Days','Years')), primary key(Treatment_ID));

```
create table MEDICENE_ASSOC( Treatment_ID varchar(6), Medicene_code varchar(6),
primary key(Treatment_ID, Medicene_code),
foreign key(Treatment_ID) references TREATMENT(Treatment_ID), foreign key(Medicene_code)
references PHARMACY(Medicene_code));


create table TREATMENT_DETAILS( Patient_ID varchar(4),
Admission_date date check(Admission_date<=curdate()), Treatment_ID varchar(6),
Medicene_code varchar(6),
primary key(Patient_ID, Admission_date, Treatment_ID, Medicene_code), foreign
key(Patient_ID,Admission_date) references CLASS2_PATIENT(Patient_ID,Admission_date),
foreign key(Treatment_ID) references TREATMENT(Treatment_ID), foreign key(Medicene_code)
references PHARMACY(Medicene_code));
```

5.1.2 Database State

We insert some values into the database in order to test our SQL create view and query statement.

INSERTION DATA FOR POPULATING TABLE:

```
insert into PERSON values ('P500','Adam','M','Morgan','M','Texas','1965-03-24');
insert into PERSON values ('P501','Lily',null,'Pulsic','F','New Jersey','1970-05-14');
insert into PERSON values ('P502','Bryan','','Shaw','M','Texas','1987-08-20');
insert into PERSON values ('P503','Jil','A','Heather','F','California','2000-11-20');
insert into PERSON values ('P504','Tyler','R','Fox','U','Washington','1985-01-07');


insert into DOCTOR values ('P500','2005-06-15','Opthelamy','Trainee');
insert into DOCTOR values ('P501','2005-06-15','Gynacology','Permanent');
insert into DOCTOR values ('P502','2005-06-15','Dentist','Visiting');
insert into DOCTOR values ('P503','2005-06-15','Eye','Trainee');
insert into DOCTOR values ('P504','2005-06-15','Neurology','Permanent');


insert into PERSON values ('P505','Ane','F','Humpry','F','Arizona','2000-03-24');
insert into PERSON values ('P506','Lacey',null,'John','F','New Jersey','1995-05-14');
insert into PERSON values ('P507','Ram','Chandra','Dev','M','Chicago','1997-08-20');


insert into NURSE values ('P505','2017-12-31');
insert into NURSE values ('P506','2018-01-01');


insert into RECEPTIONIST values ('P507','2017-04-04');


insert into ROOM values ('R1304','P506','Cabin','22:30:45','08:30:00');
```

```
insert into ROOM values ('R2310','P505','Ward','11:30:00','21:30:30');


insert into CLASS2_PATIENT values ('P502','R1304','2005-06-27');
insert into CLASS2_PATIENT values ('P507','R2310','2018-02-28');


insert into PERSON values ('P508','Anish','','Hegde','M','Mangalore','2000-03-24');
insert into PERSON values ('P509','Angad','Murthy','Vittal','M','Bangalore','1995-05-14');
insert into PERSON values ('P510','Meghna',null,'Kurrup','F','Kolkata','1997-02-02');
insert into PERSON values ('P511','Ram','Chandra','Dev','M','Chicago','1997-08-20');


insert into class1_patient values('P508','2018-11-20','P501');
insert into class1_patient values('P508','2018-10-20','P504');
insert into class1_patient values('P509','2018-08-02','P502');
insert into class1_patient values('P510','2018-06-15','P503');


insert class2_patient values('P510','R2310','2018-06-15');


insert into PHARMACY values('M1', 'Calpol', 1.25, 50, '2019-02-26');
insert into PHARMACY values('M2', 'VR654', 5.00, 50, '2020-11-26');
insert into PHARMACY values('M3', 'Nycil4', 8.75, 50, '2019-09-26');
insert into PHARMACY values('M4', 'CandidB', 5.25, 50, '2019-05-26');
insert into PHARMACY values('M5', 'Mega6', 8.25, 50, '2019-08-26');
insert into PHARMACY values('M6', 'Norflox 40', 3.50, 50, '2020-02-26');
insert into PHARMACY values('M7', 'Metrogyl', 10.25, 50, '2019-02-01');


insert into TREATMENT values('1', 'Antibiotic', 7.0, 'Days');
insert into TREATMENT values('2', 'Diarrhoea Meds', 5.0, 'Days');
insert into TREATMENT values('3', 'TetVac', 1.0, 'Days');
insert into TREATMENT values('4', 'Cough Meds', 5.0, 'Days');
insert into TREATMENT values('5', 'Indigestion meds', 3.0, 'Days');


insert into MEDICENE_ASSOC values('1', 'M1');
insert into MEDICENE_ASSOC values('1', 'M3');


insert into MEDICENE_ASSOC values('5', 'M3');
insert into MEDICENE_ASSOC values('2', 'M2');
insert into MEDICENE_ASSOC values('1', 'M2');
```

```
insert into TREATMENT_DETAILS values('P502', '2005-06-27', '1', 'M1');
insert into TREATMENT_DETAILS values('P507', '2018-02-28', '1', 'M3');
insert into TREATMENT_DETAILS values('P510', '2018-06-15', '5', 'M3');


insert into INSURANCE values('1', 'SHIP', 500, 1000);


insert into PAYMENTS values('P509', '2018-10-26', 'P507', null , 200, 200);
insert into PAYMENTS values('P510', '2018-02-26', 'P507', '1', 100, 100);
insert into PAYMENTS values('P511', '2018-10-26', 'P507', '1', 120, 120);
insert into PAYMENTS values('P507', '2018-02-26', 'P507', '1', 300, 300);
insert into PAYMENTS values('P509', '2018-10-26', 'P507', '1', 300, 0);


insert into RECORD values('R001', 'P507','P510','2018-10-05', '2018-10-05', 'fever');
insert into RECORD values('R002', 'P507','P511', '2018-08-30', '2018-10-31', 'indigestion');
insert into RECORD values('R003', 'P507','P507', '2017-10-01', '2017-10-05', 'cough');
insert into RECORD values('R004', 'P507','P510', '2017-12-31', '2018-01-05', 'loose motion');
```

5.2 Creation of Views

Required Views creation:

```
create or replace view toptreatment as with top_treatm as(
select td.treatment_id ti, count(treatment_id) as c from treatment_details td
group by ti order by c desc limit 1
),
patient as(
select td.patient_id pi
from treatment_details td, top_treatm tt where td.treatment_id = tt.ti
),
amt as(
select sum(p.amount_due) s from payments p, patient pa where p.patient_id = pa.pi
)
select t.treatment_name, amt.s from treatment t, amt, top_treatm tt where t.treatment_id = tt.ti;



create or replace view topdoctor as with checked_class1 as(
select consult_doctor d1, count(consult_doctor) as c1 from class1_patient
where consult_doctor in (select emp_id from doctor) group by consult_doctor
),
checked_class2 as(
select consult_doctor d2, count(consult_doctor) as c2 from class2_patient_consultation
where consult_doctor in (select emp_id from doctor) group by consult_doctor
)
```

```
select p.person_id, p.first_name, p.last_name, start_date, sum(c1+c2) from person p, doctor d,
checked_class1 cn1, checked_class2 cn2 where p.person_id = d.emp_id
and d.emp_id = cn1.d1
and cn1.c1 > 5
and d.emp_id = cn2.d2 and cn2.c2 > 10
group by person_id, fitst_name, last_name, start_date ;
```

```
CREATE OR REPLACE VIEW ReorderMeds AS( SELECT medicene_code, medicene_name,
quantity, date_of_expiry FROM PHARMACY
WHERE Date_of_expiry <= DATE_ADD(CURDATE(), INTERVAL 1 month) OR Quantity <
1000 );
```

```
create or replace view potentialpatient as with freq as(
select patient_id pi, count(patient_id) c from class1_patient
group by pi having c>3
),
notadmit as( select pi from freq
where exists (select patient_id from class2_patient)
)
SELECT p.Person_ID, p.First_name, p.Last_name, c.phone_number FROM person p, contacts c,
notadmit na
WHERE c.person_id = p.person_id and p.person_id = na.pi;
```

```
create or replace view mostfreqissues as with reason as(
select record_description rd, count(record_description) c from record
group by record_description order by c desc
limit 1
),
patient as( select patient_id
from record, reason
where record.record_description = reason.rd
)
select r.rd as reason, r.c as freq, t.treatment_name
from treatment t, treatment_details td, patient p, reason r where td.patient_id = p.patient_id;
```

5.3 Creation of SQL Queries:

```
q1. select Doctor_type, group_concat(Start_date) from DOCTOR group by(Doctor_type);
```

```
q2. SELECT P.First_name,P.Middle_name,P.Last_name from PERSON P,CLASS2_PATIENT A,
(SELECT Emp_ID,Start_date FROM DOCTOR union SELECT Emp_ID,Start_date FROM
NURSE union SELECT Emp_ID,Start_date FROM RECEPTIONIST) B
where P.Person_ID=A.patient_ID and A.patient_ID=B.Emp_ID and
datediff(A.Admission_date,B.Start_date)<=90 ;
```

```
q3. with top5doctors as(
select person_id, total_patients from topdoctor
order by total_patients desc limit 5
)
select avg(year(curdate())-year(p.date_of_birth)) avg_age, d.doctor_type from person p, doctor d,
top5doctors t5
where p.person_id = d.emp_id and d.emp_id = t5.person_id group by d.doctor_type;
```

```
q4.select P.Medicene_name from MEDICENE_ASSOC M, PHARMACY P,
(select A.Treatment_id,max(A.new_count) from (select Treatment_id,count(patient_ID) as
new_count from TREATMENT_DETAILS group by(Treatment_ID)) A) B
where M.Treatment_id= B.Treatment_id and M.Medicene_code=P.Medicene_code;
```

```
q5.select emp_id from doctor where emp_id not in
(select consult_doctor from class1_patient where datediff(curdate(),date_of_appointment)<=150
union
select consult_doctor from class2_patient_consultation where
datediff(curdate(),Admission_date)<=150);
```

```
q6.select P.Person_ID,
P.First_name,P.Middle_name,P.Last_name,P.Gender,P.Address,P.Date_of_Birth,I.Provider from
person P, Payments B, Insurance I
where P.Person_ID=B.Patient_ID and B.Insurance_ID=I.Insurance_ID and B.Cash_Amount=0;
```

```
q7.select A.room_id,SEC_TO_TIME( TIME_TO_SEC(A.room_time) * max(A.book_count) ) from
(select R.room_id, timediff(R.end_date,R.start_date) as room_time,count(*) as book_count from
class2_patient C, room R where R.room_id=C.Room_id group by(R.room_id)) A;
```

```
q8. select max(A.count), A.year, A.info from
(select year(date_of_visit) as year,count(*) as count, group_concat(Record_Description) as info
from record group by(year(date_of_visit))) A;
```

```
q9.select treatment_id, Duration, Duration_unit from treatment where treatment_id in ( select
Min(Treatment_id) from treatment_details group by patient_id);
```

```
q10. with emp as(
select emp_id, start_date from doctor union
select emp_id, start_date from nurse union
select emp_id, start_date from receptionist
),
last_join as(
select max(start_date) d from emp
)
select count(*) from class2_patient p, last_join j where p.admission_date > j.d;
```

```
q11. select P.Person_ID,
P.First_name,P.Middle_name,P.Last_name,P.Gender,P.Address,P.Date_of_Birth from person P,
class1_patient c1, class2_patient c2
where P.Person_ID=C1.Patient_ID and C1.Patient_ID=C2.Patient_ID and
datediff(C2.Admission_date,C1.Date_of_appointment)<=7;
```

```
q12. select sum(Amount_due),month(Date_of_payment) from payments where
year(Date_of_payment)='2017' group by(month(Date_of_payment));
```

```
q13. select distinctrow First_name,P.Middle_name,P.Last_name from PERSON P, DOCTOR D,
CLASS1_PATIENT C ,(
select Patient_id,count(*) from CLASS1_PATIENT where Patient_id not in (select Patient_id from
CLASS2_PATIENT) group by(patient_id) having count(*)=1) R where P.Person_ID=D.Emp_ID
and D.Emp_ID=C.Consult_doctor and C.Patient_ID=R.Patient_ID;
```

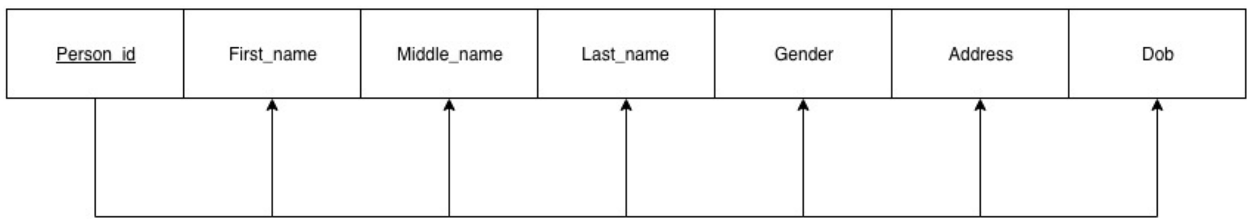
```
q14.select PP.name, year(current_date())-year(P.Date_of_Birth) as AGE from Person,
PotentialPatient PP where P.Person_ID=PP.Person_ID;
```

6. Dependency Diagram

We now draw a dependency diagram for each table in our Relational Schema as follows:

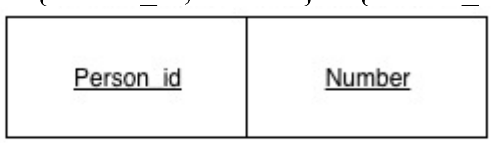
PERSON : {Person_id} → {First_name, Middle_name, Last_name, Gender, Address, Dob}

In this Relation there is only one attribute as the primary key, hence all the other attributes are functionally dependent on it.



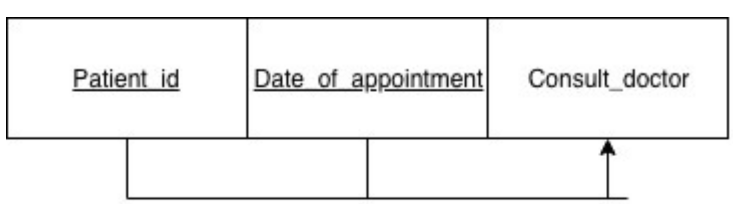
CONTACT: {Person_id, Number} → {Person_id, Number}

In this table we have both the attributes as the primary key of the relation. We have attribute closure of set {Person_id,Number} as {Person_id,Number} using the the property that X → X is true.



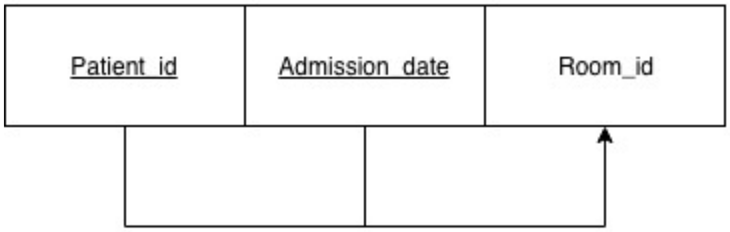
CLASS1_PATIENT : {Patient_id, Date_of_appointment} → {Consult_doctor}

In this table the attribute Consult_doctor is functionally dependent on patient_id and Date_of_appointment which is also the key of the given relation.



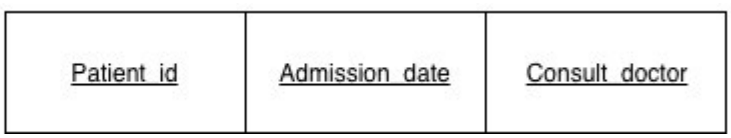
CLASS2_PATIENT : {Patient_id, Admission_date} → {Room_id, Doctor_id}

In this relation Patient_id, Admission_date for the primary key. Hence the remaining attributes are functionally dependent on {Patient_id, Admission_date}.



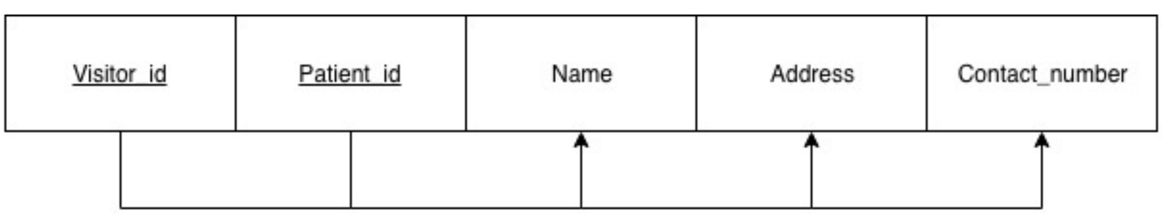
CLASS2_PATIENT_CONSULTATION: { Patient_id,Admission_date,Consult_doctor} → {Patient_id, Admission_date, Consult_doctor}

In this relation we have all the attributes as the key, using the property that X → XWe can define the functionaly dependence of this relation.



VISITOR : {Visitor_id,Patient_id} → {Name,Address,Contact_number}

As the key here consists of single attribute hence we have functional dependency with all other attributes.



TREATMENT_DETAILS:

{Patient_id, Medicine_code, Treatment_id} → {Patient_id,Medicine_id,Treatment_id}

The key for this table is the complete set of attributes, hence we can say that attribute closure of the key of this set includes all the attributes of the relation and hence defines functional dependency in this way.

Patient_id	Medicine_code	Treatment_id

MEDICINE_ASSOC:

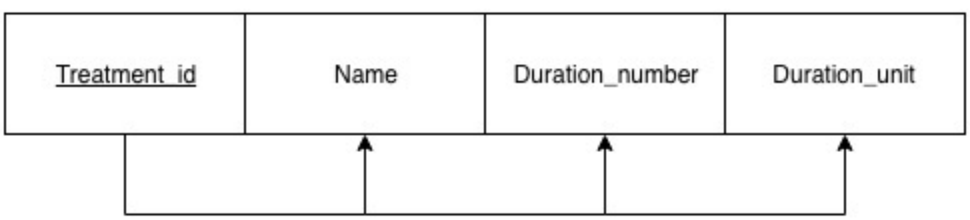
{Patient_id, Medicine_code, Treatment_id} → {Patient_id, Medicine_code, Treatment_id}

The key for this table is the complete set of attributes, hence we can say that attribute closure of the key of this set includes all the attributes of the relation and hence defines functional dependency in this way

Patient_id	Medicine_code	Treatment_id

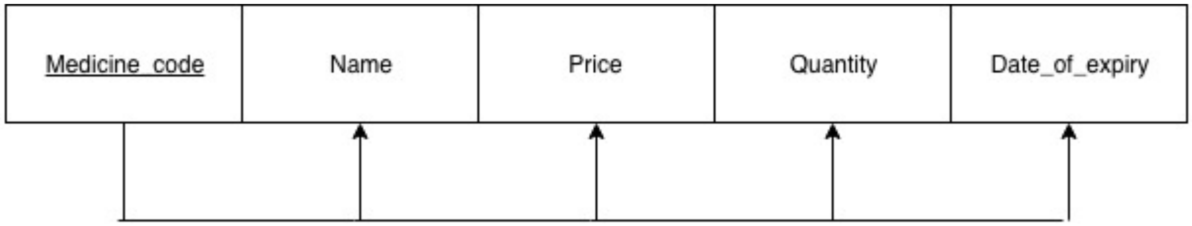
TREATMENT: {Treatment_id} → {Name, Duration_number, Duration_unit}

As the key here consists of single attribute hence we have functional dependency with all other attributes.



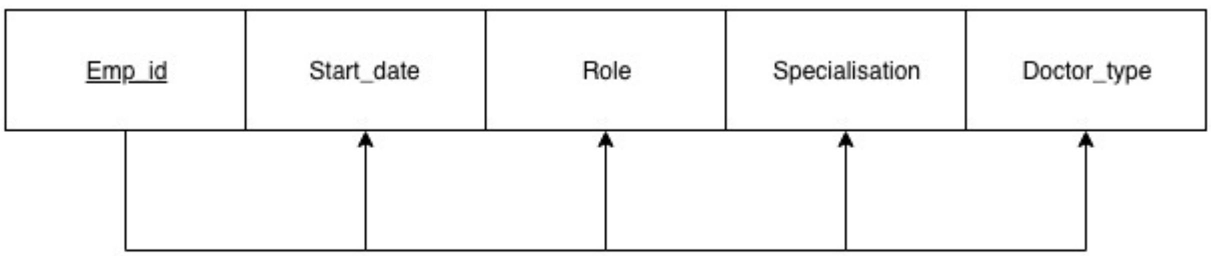
PHARMACY: {Medicine_code} → {Name, Price, Quantity, Date_of_expiry}

As the key here consists of single attribute hence we have functional dependency with all other attributes.



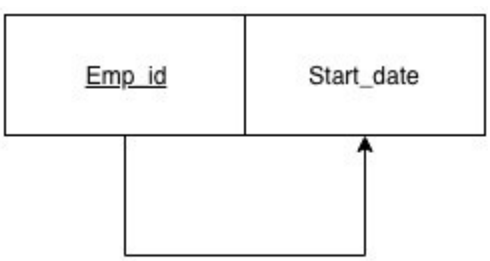
DOCTOR: {Emp_id} → {Start_date, Role, Specialisation, Doctor_type}

As the key here consists of single attribute hence we have functional dependency with all other attributes.



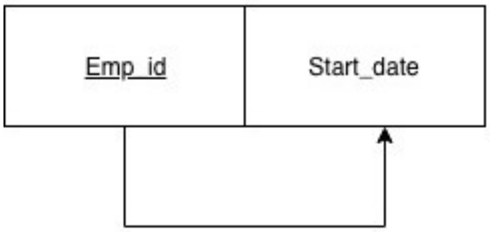
RECEPTIONIST: {Emp_id} → {Start_date}

As the key here consists of single attribute hence we have functional dependency with all other attributes.



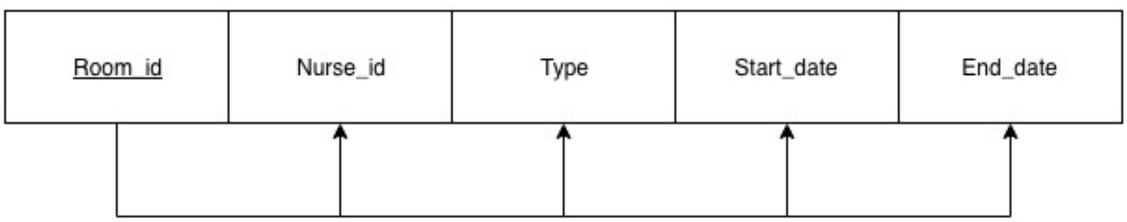
NURSE: {Emp_id} → {Start_date}

As the key here consists of single attribute hence we have functional dependency with all other attributes.



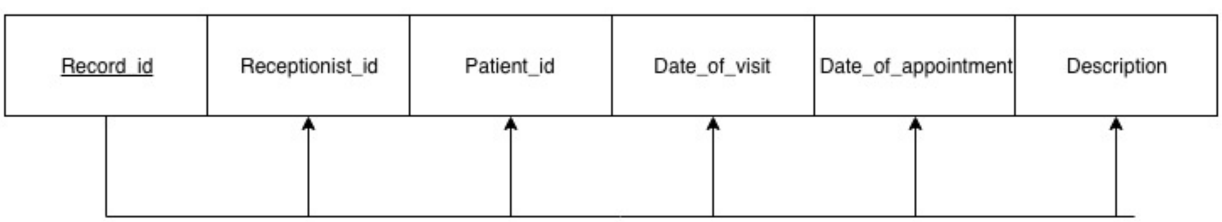
ROOM: {Room_id, Nurse_id} → {Type, Start_date, End_date}

As the key here consists of single attribute hence we have functional dependency with all other attributes.



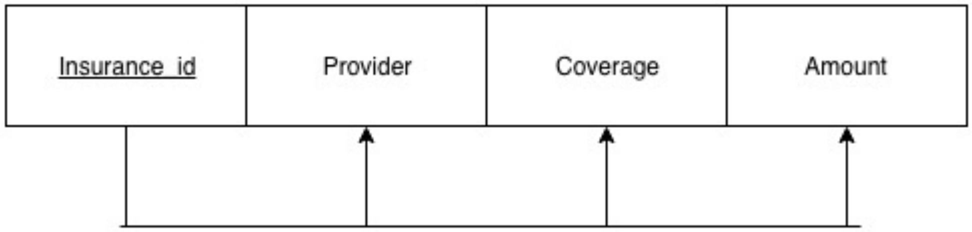
RECORD: {Record_id} → {Receptionist_id,Patient_id,Date_of_visit, Date_of_appointment, Description}

As the key here consists of single attribute hence we have functional dependency with all other attributes.



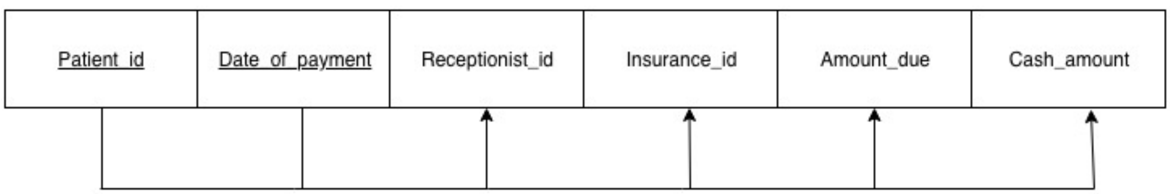
INSURANCE: {Insurance_id} → {Provider, Coverage, Amount}

As the key here consists of single attribute hence we have functional dependency with all other attributes.



PAYMENT: {Patient_id, Date_of_payment} → {Receptionist_id, Insurance_id, Amount_due, Cash_amount}

As the key here consists of single attribute hence we have functional dependency with all other attributes.



7. Conclusion

In this final report we summarized all the necessary descriptions and solutions for Dallas Care database, including process and result of EER diagrams, relational schemas in third normal form, SQL statements to create database, views and solved corresponding queries, as well as dependency diagram. We also implement the whole database in Oracle and using a database state to test every query. In section 2, we also explained why we use superclass/subclass relationship to build relational schema, why we choose a Relational DBMS to implement our database, and the additional five business rules shown from implementation.