Project for Database Design

Phase IV. Documentation

Ayan Paul [axp170036@utdallas.edu](mailto:axp170036@utdallas.edu)

Jiten Girdhar

[jxg170021@utdallas.edu](mailto:jxg170021@utdallas.edu)

Rajarshi Chattopadhyay [rxc170010@utdallas.edu](mailto:rxc170010@utdallas.edu)

Poonam Gillurkar

pxg180009@utdallas.edu

# 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.

# 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.

# 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:

# A person can be a student, teacher or school staff.

# Each teacher must teach at least one course E

# Every student must take at least three courses.

# A staff member can also be a student.

# 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.

# EER Diagram with all assumptions

# C:\Users\aakanksha\Downloads\EER_Diagram.png

# Fig 1. EER Diagram

# 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** |
| Person\_Id |
| 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** |
| Person\_id |
| Number |

* + 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** |
| Patient\_id Date\_of\_appointment |
| 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** |
| Patient\_id |
| Admission\_Date |
| 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** |
| Patient\_id |
| Admission\_date |
| Doctor\_id |

* 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** |
| Insurance\_Id |
| 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** |
| Patient\_id |
| Payment\_Date Receptionist\_id |
| Insurance\_id |
| Amount\_Due |
| Cash\_Amoount |

1. **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](http://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;

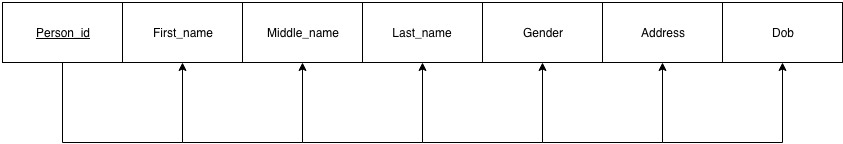
q14.select PP.name, year(current\_date())-year(P.Date\_of\_Birth) as AGE from Person, PotentialPatient PP where P.Person\_ID=PP.Person\_ID;

1. **Dependency Diagram**

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

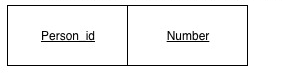
**PERSON** : {Person\_id} 🡪 {First\_name, Middle\_name, Last\_name, Gender, Address, Dob}

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



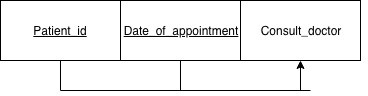
**CONTACT:** {Person\_id, Number} 🡪 {Person\_id, Number}

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.



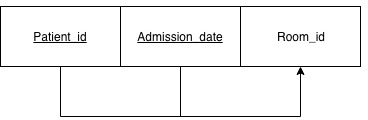
**CLASS1\_PATIENT :** {Patient\_id, Date\_of\_appointment} 🡪 {Consult\_doctor}

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.



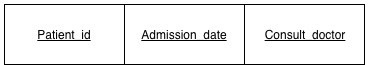
**CLASS2\_PATIENT : {**Patient\_id, Admission\_date} 🡪 {Room\_id, Doctor\_id}

In this relation Patient\_id, Admission\_date for the primary key. Hence the remaining attributes are functionally dependent on {Patient\_id, Admission\_date}.



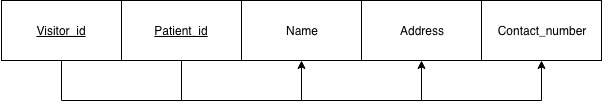
**CLASS2\_PATIENT\_CONSULTATION: { Patient\_id**,Admission\_date,Consult\_doctor} 🡪 {Patient\_id, Admission\_date, Consult\_doctor}

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}

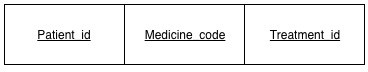
As the key here consists of single attribute hence we have functional dependency with all other attributes.



## TREATMENT\_DETAILS:

{Patient\_id, Medicine\_code, Treatment\_id} 🡪 {Patient\_id,Medicine\_id,Treatment\_id}

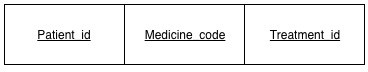
The key for this table is the complete set of attributes, hence we can say that attribute closure of the key of this set includes all the attributes of the relation and hence defines functional dependency in this way.



## MEDICINE\_ASSOC:

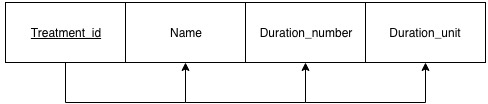
{Patient\_id, Medicine\_code, Treatment\_id} 🡪 {Patient\_id, Medicine\_code, Treatment\_id}

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



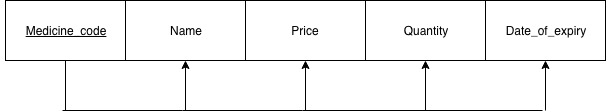
**TREATMENT**: {Treatment\_id} 🡪 {Name, Duration\_number, Duration\_unit}

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



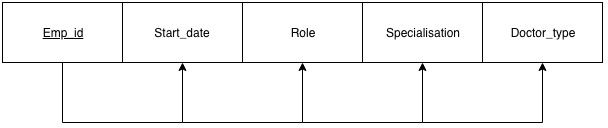
**PHARMACY:** {Medicine\_code} 🡪 {Name, Price, Quantity, Date\_of\_expiry}

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



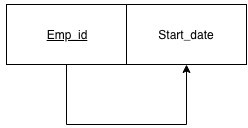
**DOCTOR: {**Emp\_id} 🡪 {Start\_date, Role, Specialisation, Doctor\_type}

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



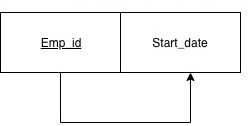
**RECEPTIONIST: {**Emp\_id} 🡪 {Start\_date}

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



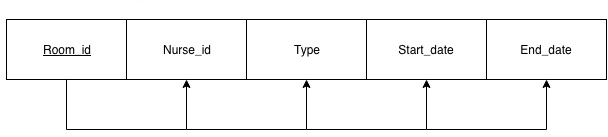
**NURSE:** {Emp\_id} 🡪 {Start\_date}

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



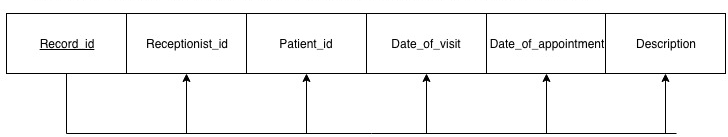
**ROOM: {**Room\_id, Nurse\_id} 🡪 {Type, Start\_date, End\_date}

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



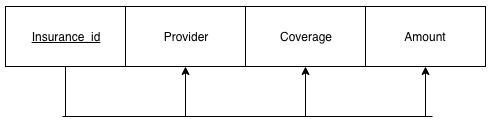
**RECORD: {**Record\_id} 🡪 {Receptionist\_id,Patient\_id,Date\_of\_visit, Date\_of\_appointment, Description}

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



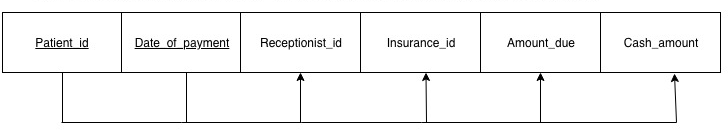
**INSURANCE: {**Insurance\_id} 🡪 {Provider, Coverage, Amount}

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.



# 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.