PHASE IV. DOCUMENTATION

Bharath Rudra
bxr180008@utdallas.edu

Karttik Reddy Yellu
kxy170003@utdallas.edu

Sai Ram Chappidi
sxc170016@utdallas.edu

Vamshider Reddy Voncha
vxv170013@utdallas.edu

0. Pre-Illumination

In this project report we will follow the requirement of Phase IV directly. 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. Project 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 patient. 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 governs 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?

- Truly, I trust that the capacity to display super class subclass connections is imperative for such a domain. Superclass/Subclass connections enables us to demonstrate complex ideas like property legacy in such a domain. There are probably going to be numerous kinds of Persons, Employees, Doctors and numerous sorts of Payment types which are best spoken to as a subclass of their individual superclass.

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

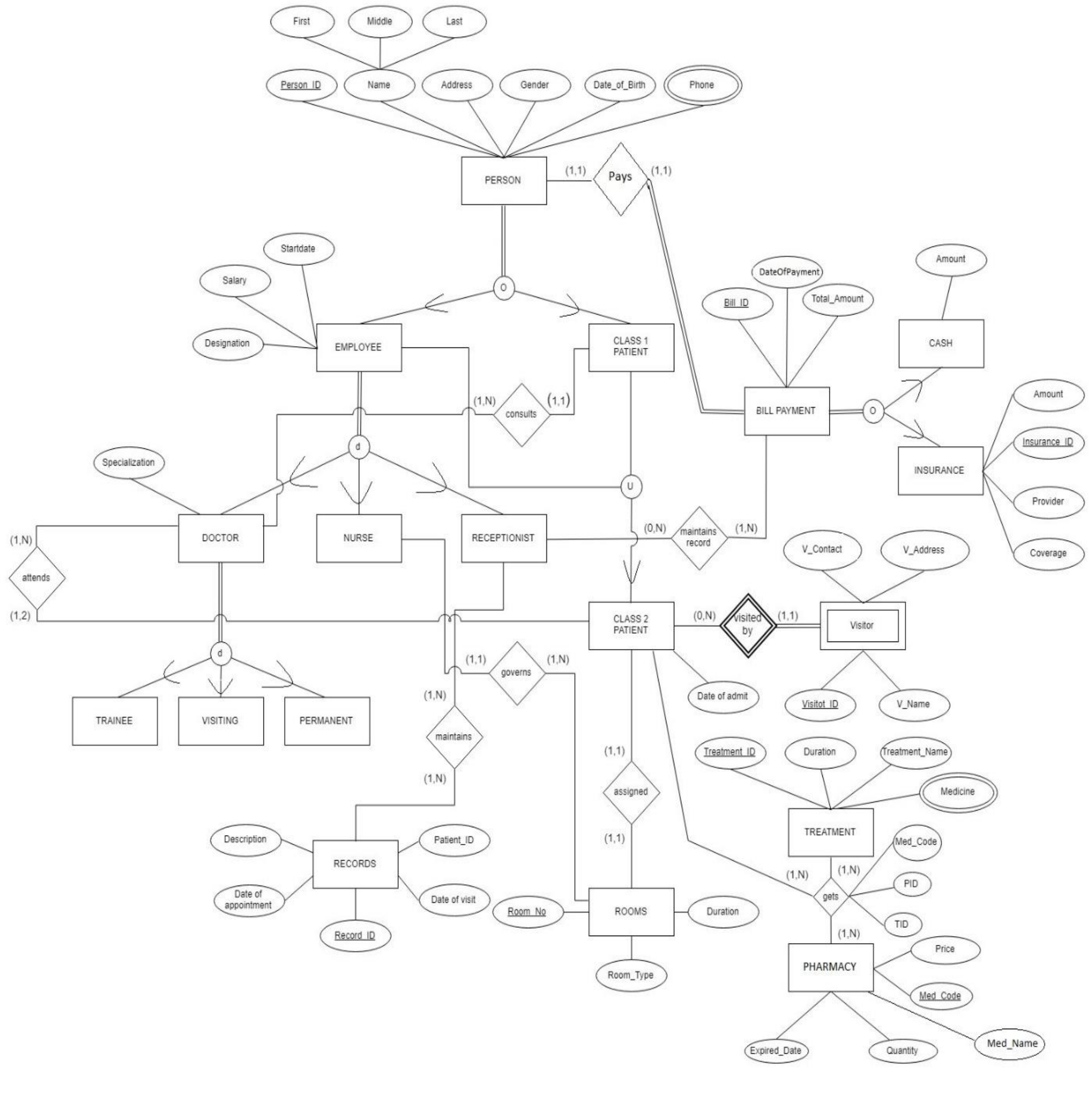
- There can be a manager for each of the doctors who can supervise them.
- There can be different kinds of employees other than doctor, nurse and receptionist like Specialists, Heads of different departments.
- Bill payment can be done even by card and not only cash or insurance. We can add that rule to the existing model.
- Doctors should also be able to access the records of patients, admitted in the hospital, directly and not through receptionist.
- In this model, we assumed that only single person can make a bill payment of an individual, but we can add another facility to split the bill payments between multiple persons.

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

- Oracle is unquestionably a standout amongst other usage of RDBMS. It oversees memory effectively and can deal with complex JOIN tasks which easily cripple MySQL and MSSQL. Oracle has an incredible Architecture, it makes it shake strong in light of the fact that it is anything but difficult to arrange diverse applications information extremely well. I think the best in addition to for me is the SQL Engine that Oracle has, it is extremely best in class, making it the main database to work an information stockroom! Running complex questions on Oracle is simply easy. Other highlights like Materialized Views, PL/SQL etc. influence Oracle to have an incentive for cash.

3. EER diagram with all assumptions

Dallas Care Hospital and Medical Care Center



CONSIDERATIONS

Serial No.	Super Class	Sub Class	Specialization Constraint
1.	Person	• Patient • Employee	Overlapping
2.	Employee	• Doctor • Nurse • Receptionist	Disjoint
3.	Doctor	• Permanent • Trainee • Visiting	Disjoint
4.	Bill_Payment	• Insurance • Cash	Overlapping
5.	• Employee • Class_1 (Patient)	Class_2 (Patient)	Union

Assumptions

<u>Entities & Attributes:</u>

1. The entity **Persons** will be of two types **Patient** and **Employee** where an Employee also can be a Patient and a Patient can be an Employee.
2. Every **Person** has a UNIQUE Person_id (Format: PXXX where 100<= XXX <= 999), Name with (First_Name, Middle_Name, Last_Name) sub-attributes, Address, Gender, Date_Of_Birth with (Date, Month, Year) sub-attributes and an OPTIONAL Phone_no.
3. Every **Patient** IS-A **Class_1** patient who can be a **Class_2** patient who is admitted or who is an **Employee**.
4. Every **Employee** has the attributes Start_Date, a unique SSN and a Job_Type.
5. Based on the Job_Type, every Employee is further categorized as a **Doctor**, a **Nurse** and a **Receptionist**.
6. Every **Doctor** has a Specialization, Type as his/her attributes.
7. Based on the Doctor Type, he/she is classified among a **Permanent** or a **Trainee** or a **Visiting** Doctor.
8. Every **Class_2** patient can have **Visitor**s.
9. Every **Visitor** has a unique Visitor_Id, Name, Contact_Info, Address as attributes.
10. Every **Class_2** patient undergoes **Treatment**.
11. Every **Treatment** has a unique Treatment_Id, Name, Duration as attributes.
12. Every **Treatment** may or may not be assigned with a **Pharmacy**.
13. For a **Pharmacy**, it has medicines with a Medicine_Name, Price, a unique Medicine_Code, Quantity and Date_expires as attributes.
14. Every **Class_2** patient is also admitted to a **Room.**
15. Every **Room** has a Room_Id, Type, Duration as its attributes.
16. There are **Records** with a unique Record_Id, Patient_Id, Description, Appointment, Date_Visited attributes.

17. A **Payment** has a unique Payment_Id, Patient_Id, Date_Of_Payment, Payment_Mode, Total_Amount_Due as its attributes.
18. A **Payment** can be through **Insurance** or **Cash** or by both.
19. An **Insurance** have a unique Insurance_Id, its Provider, Amount, Insurance_Coverage as its attributes.
20. A payment made in **Cash** has a Bill_Id and Amount as its attributes.

Relationships among the Entities:

1. A Person can be a Patient or an Employee in the hospital.
2. Every Class_1 Patient Consults at most a Doctor and a Doctor can be consulted by more than one Class_1 patient.
3. A Doctor can Attend multiple Class_2 patients but a Class_2 patient is attended by either one or at most two Doctors.
4. A Visitor can visit at least one Class_2 patient and a Class_2 patient is visited by any number of Visitors.
5. Every Class_2 patient is Assigned with a Treatment or multiple Treatments and every Treatment may or may not have a Class_2 patient.
6. A Treatment Prescribes a minimum of one medicine from the Pharmacy and a medicine can be prescribed to different treatments.
7. Every Class_2 patient is Admitted to only one Room and the Rooms can admit any number of Class_2 patients.
8. A Nurse Governs none or multiple Rooms but a Room is governed by only one Nurse.
9. A Receptionist Maintains a single or multiple Patient Records but a Record is maintained by at most one Receptionist.
10. A Receptionist also updates Payment_Info into the Bill_Payment entity.

4. Relational Schema in Third Normal Form

4.1 Relational Schema

Firstly, according to the requirement of phase III and with purpose to simplify the relational model for this database, we changed the below mentioned things respect to original relational models. We will list them as follows.

NOTE: To maintain the appropriate naming conventions, some of the relations and attributes names have been changed.

- The attribute PhoneNumber is a multi-valued attribute. So, to maintain 1 NF, we decompose the relation into PERSON and PERSON_PHONE resulting the both relations in their 3NF form.
- The attribute Class2_PersonId is removed from the relation EMPLOYEE.
- The attribute Class2_PersonId is removed from the relation CLASS1_PATIENT, we included PersonId which references to PERSON.PersonId and introduced a new primary key Class1PatientId.
- In the relation CLASS2_PATIENT, we included PersonId which references to PERSON.PersonId and introduced a new primary key Class2PatientId.
- As a correction mentioned for Phase II of the project, we changed the attribute PatientId to PersonId which references to PERSON.PersonId in the relation PATIENT_RECORDS.
- We excluded the attribute PersonId in the relation BILL_CASH as we found it to be redundant.
- We excluded the attribute PersonId in the relation BILL_INSURANCE as we found it to be redundant.
- As InsuranceId creates a partial dependency with the Provider and Coverage of the insurance, we decomposed the relation INSURANCE into BILL_INSURANCE and INSURANCE_DETAILS.

The modified relational schema is shown in Table below.

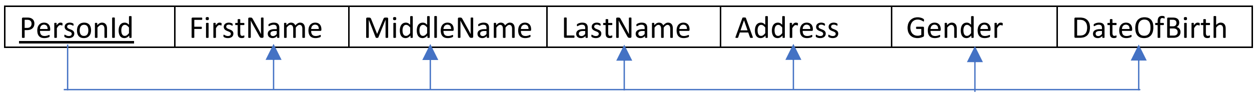
Relations	Attributes						
PERSON	PersonId (VARCHAR2(4))	FirstName (VARCHAR2(20[CHAR]))	MiddleName (VARCHAR2(20[CHAR]))	LastName (VARCHAR2(20[CHAR]))	Address (VARCHAR 2(200))	Gender (VARCH AR26 [CHAR]))	DateOf Birth (DATE)
PERSON_PHONE	PersonId (VARCHAR2(4))	PhoneNumber (NUMBER(11)					
EMPLOYEE	EmployeePersonId (VARCHAR2(4))	Salary (NUMBER(10,2)	StartDate (DATE)	Designation (VARCHAR2(20 0 [CHAR]))			
CLASS1_PATIENT	Class1PatientId (VARCHAR2(4))	PersonId (VARCHAR2(4) [CHAR])	DoctorPersonId (VARCHAR2(4))				

DOCTOR	DoctorPersonId (VARCHAR2(4))	Specialization (VARCHAR2(20 [CHAR]))					
NURSE	NursePersonId (VARCHAR2(4))						
RECEPTIONIST	ReceptionistPersonId (VARCHAR2(4))						
TRAINEE	TraineePersonId (VARCHAR2(4))						
VISITING	VisitingPersonId (VARCHAR2(4))						
PERMANENT_DOCTOR	PermanentPersonId (VARCHAR2(4))						
CLASS2_PATIENT	Class2PatientId(VARCHAR2(4))	PersonId (VARCHAR2(4))	DoctorPersonId (VARCHAR2(4))	RoomNo (Number (10)	DateOfAdmit (DATE)		
ROOMS	RoomNumber (Number 10)	RoomType (VARCHAR2(20[CHAR]))	RoomDuration NUMBER(5)	NursePersonId (VARCHAR2(4))			
PATIENT_RECORDS	RecordID (VARCHAR2(20))	DateOfAppointment (DATE)	DateOfVisit (DATE)	Description (VARCHAR2(100))	PersonId		
PHARMACY	MedicineCode (VARCHAR2(20))	MedicineName (VARCHAR2(20))	MedicinePrice NUMBER(10)	Quantity NUMBER(5)	DateOfExpiry (DATE)		
TREATMENT	TreatmentId (VARCHAR2(20))	TreatmentName (VARCHAR2(20[CHAR))	TreatmentDuration NUMBER(5)	TreatmentDescription (VARCHAR2(100))			
TREATMENT_MEDICINE	TreatmentId (VARCHAR2(20))	MedicineCode (VARCHAR2(20))					
VISITOR	Visitorid (VARCHAR2(20))	Class2PatientId (VARCHAR2(4))	VisitorName (VARCHAR2(20[CHAR]))	VisitorAddress (VARCHAR2(200))	VisitorContact (NUMBER(11))		
BILL_PAYMENT	BillId (VARCHAR2(20))	PersonId (VARCHAR2(4))	DateOfPayment (DATE)	TotalAmountDue NUMBER(10,2)			
BILL_CASH	Billid (VARCHAR2(20))	BillAmount NUMBER(10,2)					
BILL_INSURANCE	Insuranceid (VARCHAR2(20))	BillId (VARCHAR2(20))	Amount NUMBER(10,2)				
INSURANCE_DETAILS	Insuranceid (VARCHAR2(20))	Provider (VARCHAR2(20))	Coverage (VARCHAR2(20))				
MAINTAINS_RECORDS	RecordId (VARCHAR2(20))	ReceptionistPersonId (VARCHAR2(4))					
MAINTAINS_PAYMENTS	ReceptionistPersonId (VARCHAR2(4))	BillId (VARCHAR2(20))					
GETS_TREATMENT_PHARMACY	MedicineCode (VARCHAR2(20))	TreatmentId (VARCHAR2(20))	Class2PatientId (VARCHAR2(4))				

4.2 Format for every relation

4.2.1 PERSON & PERSON_PHONE

PERSON



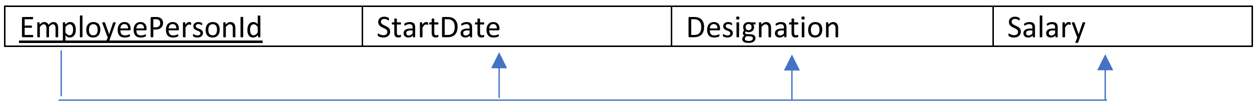
PersonId	FirstName	MiddleName	LastName	Address	Gender	DateOfBirth

PERSON_PHONE

PersonId	PhoneNumber

4.2.2 EMPLOYEE

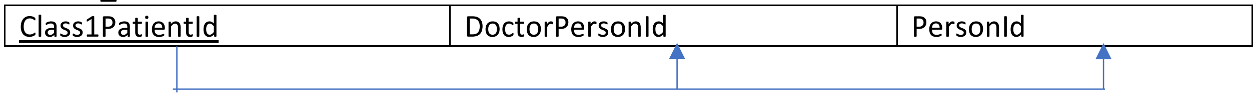
EMPLOYEE



EmployeePersonId	StartDate	Designation	Salary

4.2.3 CLASS1_PATIENT

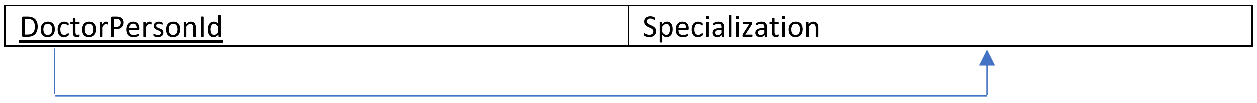
CLASS1_PATIENT



Class1PatientId	DoctorPersonId	PersonId

4.2.4 DOCTOR

DOCTOR



DoctorPersonId	Specialization

4.2.5 NURSE, RECEPTIONIST, TRAINEE, VISITING, PERMANENT_DOCTOR

NURSE

NursePersonId

RECEPTIONIST

ReceptionistPersonId

TRAINEE

TraineePersonId

VISITING

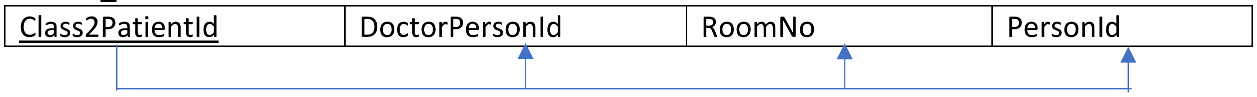
VistingPersonId

PERMANENT_DOCTOR

PermanentPersonId

4.2.6 CLASS2_PATIENT

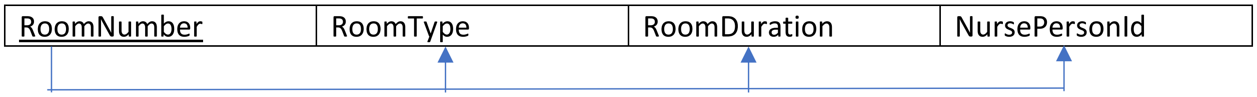
CLASS2_PATIENT



Class2PatientId	DoctorPersonId	RoomNo	PersonId

4.2.7 ROOMS

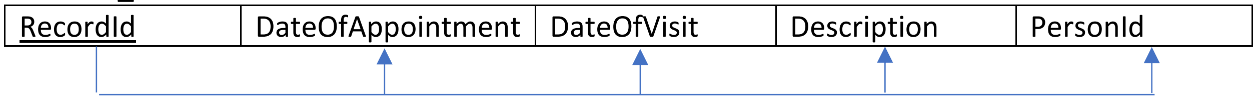
ROOMS



RoomNumber	RoomType	RoomDuration	NursePersonId

4.2.8 PATIENT_RECORDS

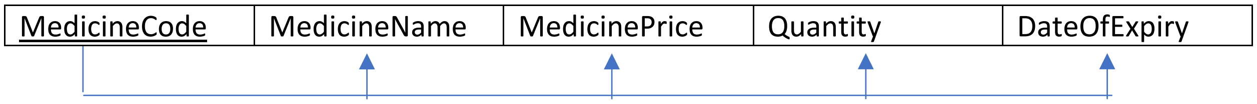
PATIENT_RECORDS



RecordId	DateOfAppointment	DateOfVisit	Description	PersonId

4.2.9 PHARMACY

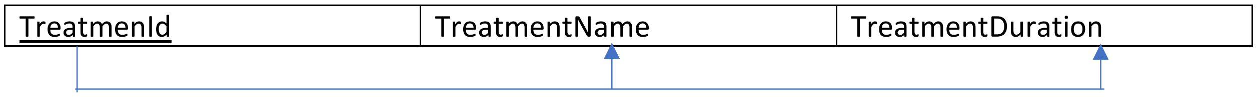
PHARMACY



MedicineCode	MedicineName	MedicinePrice	Quantity	DateOfExpiry

4.2.10 TREATMENT

TREATMENT



TreatmenId	TreatmentName	TreatmentDuration

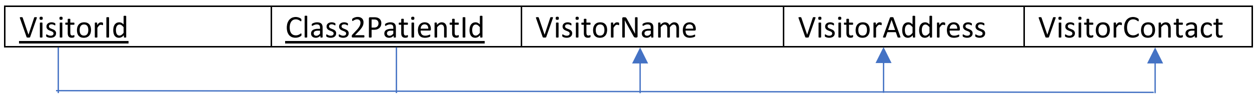
4.2.11 TREATMENT_MEDICINE

TREATMENT_MEDICINE

TreatmentId	MedicineCode

4.2.12 VISITOR

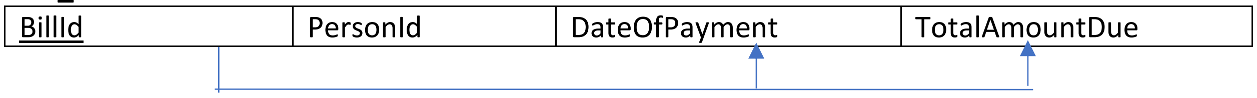
VISITOR



VisitorId	Class2PatientId	VisitorName	VisitorAddress	VisitorContact

4.2.13 BILL_PAYMENT

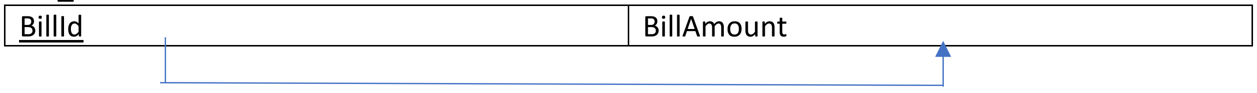
BILL_PAYMENT



BillId	PersonId	DateOfPayment	TotalAmountDue

4.2.14 BILL_CASH

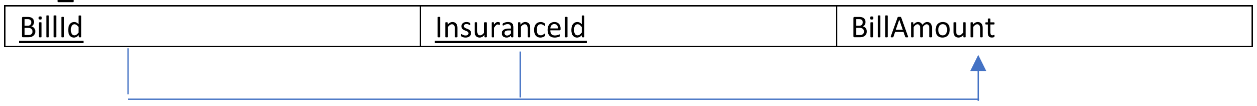
BILL_CASH



BillId	BillAmount

4.2.15 BILL_INSURANCE

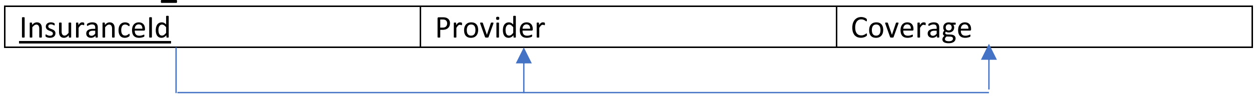
BILL_INSURANCE



BillId	InsuranceId	BillAmount

4.2.16 INSURANCE_DETAILS

INSURANCE_DETAILS



InsuranceId	Provider	Coverage

4.2.17 MAINTAINS_RECORDS

MAINTAINS_RECORDS

<u>RecordId</u>	<u>ReceptionistPersonId</u>

4.2.18 MAINTAINS_PAYMENTS

MAINTAINS_PAYMENTS

<u>ReceptionistId</u>	<u>BillId</u>

4.2.19 GETS_TREATMENT_MEDICINE

GETS_TREATMENT_PHARMACY

<u>MedicineCode</u>	<u>TreatmentId</u>	<u>Class2PersonId</u>

5. SQL statements

5.1 Creation of Database with SQL Statements

5.1.1 Table Creation

- **PERSON**

```
CREATE TABLE PERSON(

   PersonId varchar2(4) not null,
   FirstName varchar2(20) not null,
   MiddleName varchar2(20),
   LastName varchar2(20) not null,
   Address varchar(200) not null,
   Gender char(2) not null,
   DateOfBirth date not null,

   PRIMARY KEY (PersonId)
);
```

- **PERSON_PHONE**

```
CREATE TABLE PERSON_PHONE(

   PersonId varchar2(4) not null,
   PhoneNumber NUMBER(10) not null,

   PRIMARY KEY (PersonId, PhoneNumber),
   FOREIGN KEY (PersonId) REFERENCES PERSON(PersonId)
);
```

- **EMPLOYEE**

```
CREATE TABLE EMPLOYEE(

   EmployeePersonId varchar2(4) not null,
   StartDate date not null,
   Salary NUMBER(10,2) not null,
   Designation varchar2(200) not null,

   PRIMARY KEY (EmployeePersonId),
   FOREIGN KEY (EmployeePersonId) REFERENCES PERSON(PersonId) );
```

- **DOCTOR**

```
CREATE TABLE DOCTOR(
    DoctorPersonId varchar2(4) not null,
    Specialization varchar2(20) not null,

    PRIMARY KEY (DoctorPersonId),
    FOREIGN KEY (DoctorPersonId) REFERENCES PERSON(PersonId)
);
```

- **CLASS1_PATIENT**

```
CREATE TABLE CLASS1_PATIENT(

    Class1PatientId varchar2(4) not null,
    PersonId varchar(4) not null,
    DoctorPersonId varchar(4) not null,

    PRIMARY KEY (Class1PatientId),
    FOREIGN KEY (PersonId) REFERENCES PERSON(PersonId),
    FOREIGN KEY (DoctorPersonId) REFERENCES DOCTOR(DoctorPersonId)
);
```

- **NURSE**

```
CREATE TABLE NURSE(

    NursePersonId varchar2(4) not null,

    PRIMARY KEY (NursePersonId),
    FOREIGN KEY (NursePersonId) REFERENCES PERSON(PersonId)
);
```

- **RECEPTIONIST**

```
CREATE TABLE RECEPTIONIST(

    ReceptionistPersonId varchar2(4) not null,

    PRIMARY KEY (ReceptionistPersonId),
    FOREIGN KEY (ReceptionistPersonId) REFERENCES PERSON(PersonId)
);
```

- **TRAINEE**

```
CREATE TABLE TRAINEE(

    TraineePersonId varchar2(4) not null,
    PRIMARY KEY (TraineePersonId),
    FOREIGN KEY (TraineePersonId) REFERENCES PERSON(PersonId)
);
```

- **VISITING**

```
CREATE TABLE VISITING(

    VisitingPersonId varchar2(4) not null,

    PRIMARY KEY (VisitingPersonId),
    FOREIGN KEY (VisitingPersonId) REFERENCES PERSON(PersonId)
);
```

- **PERMANENT_DOCTOR**

```
CREATE TABLE PERMANENT_DOCTOR(

    PermanentPersonId varchar2(4) not null,

    PRIMARY KEY (PermanentPersonId),
    FOREIGN KEY (PermanentPersonId) REFERENCES PERSON(PersonId)
);
```

- **CLASS2_PATIENT**

```
CREATE TABLE CLASS2_PATIENT(

    Class2PatientId varchar2(4) not null,
    PersonId varchar(4) not null,
    DoctorPersonId varchar(4) not null,
    RoomNo number(10) not null,
    DateOfAdmit date not null,

    PRIMARY KEY (Class2PatientId),
    FOREIGN KEY (PersonId) REFERENCES PERSON(PersonId),
    FOREIGN KEY (DoctorPersonId) REFERENCES DOCTOR(DoctorPersonId)
);
```

- **ROOMS**

```
CREATE TABLE ROOMS(

    RoomNumber number(10) not null,
    RoomType varchar2(20) not null,
    RoomDuration number(5) not null,
    NursePersonId varchar2(4) not null,
    PRIMARY KEY (RoomNumber),
    FOREIGN KEY (NursePersonId) REFERENCES NURSE(NursePersonId)
);
```

- **PATIENT_RECORDS**

```
CREATE TABLE PATIENT_RECORDS(

   RecordId varchar2(20) not null,
   DateOfAppointment date not null,
   DateOfVisit date not null,
   Description varchar2(100) not null,
   PersonId varchar2(4) not null,

   PRIMARY KEY (RecordId),
   FOREIGN KEY (PersonId) REFERENCES PERSON(PersonId)
);
```

- **PHARMACY**

```
CREATE TABLE PHARMACY(

   MedicineCode varchar2(20) not null,
   MedicineName varchar2(20) unique not null,
   MedicinePrice number(10) not null,
   Quantity number(5) not null,
   DateOfExpiry date not null,

   PRIMARY KEY (MedicineCode)
);
```

- **TREATMENT**

```
CREATE TABLE TREATMENT(

   TreatmentId varchar2(20) not null,
   TreatmentName varchar2(20) not null,
   TreatmentDuration number(5) not null,
   TreatmentDescription VARCHAR(100) NOT NULL,

   PRIMARY KEY (TreatmentId)
);
```

- **MEDICINE**

```
CREATE TABLE TREATMENT_MEDICINE(
    TreatmentId varchar2(20) not null,
    MedicineCode varchar2(20) not null,

    PRIMARY KEY (TreatmentId, MedicineCode),
    FOREIGN KEY (TreatmentId) REFERENCES TREATMENT(TreatmentId)
);
```

- **VISITOR**

```
CREATE TABLE VISITOR(

    VisitorId varchar2(20) not null,
    Class2PatientId varchar2(4) not null,
    VisitorName varchar2(20) not null,
    VisitorAddress varchar2(100) not null,
    VisitorContact number(10) not null,

    PRIMARY KEY (VisitorId, Class2PatientId),
    FOREIGN KEY (Class2patientId) REFERENCES CLASS2_PATIENT(Class2PatientId)
);
```

- **PAYMENT**

```
CREATE TABLE BILL_PAYMENT(

    BillId varchar2(20) not null,
    PersonId varchar2(4) not null,
    DateOfPayment date not null,
    TotalAmountDue number(10,2) not null,

    PRIMARY KEY (BillId),
    FOREIGN KEY (PersonId) REFERENCES PERSON(PersonId)
);
```

- **BILL_CASH**

```
CREATE TABLE BILL_CASH(

    BillId varchar2(20) not null,
    BillAmount number(10,2) not null,

    PRIMARY KEY (BillId),
    FOREIGN KEY (BillId) REFERENCES BILL_PAYMENT(BillId)
);
```

- **BILL_INSURANCE**

```
CREATE TABLE BILL_INSURANCE(

   InsuranceId varchar2(20) not null,
   BillId varchar2(20) not null,
   BillAmount number(10,2) not null,

   PRIMARY KEY (InsuranceId),
   FOREIGN KEY (BillId) REFERENCES BILL_PAYMENT(BillId)
);
```

- **INSURANCE_DETAILS**

```
CREATE TABLE INSURANCE_DETAILS(

   InsuranceId varchar2(20) not null,
   InsuranceProvider varchar2(20) not null,
   InsuranceCoverage number(10,2) not null,

   PRIMARY KEY (InsuranceId),
   FOREIGN KEY (InsuranceId) REFERENCES BILL_Insurance(InsuranceId)
);
```

- **RECORDS**

```
CREATE TABLE MAINTAINS_RECORDS(

   RecordId varchar2(20) not null,
   ReceptionistPersonId varchar2(4) not null,

   PRIMARY KEY (RecordId, ReceptionistPersonId),
   FOREIGN KEY (RecordId) REFERENCES PATIENT_RECORDS(RecordId),
   FOREIGN KEY (ReceptionistPersonId) REFERENCES RECEPTIONIST(ReceptionistPersonId)
);
```

- **MAINTAINS_PAYMENTS**

```
CREATE TABLE MAINTAINS_PAYMENTS(

   BillId varchar2(20) not null,
   ReceptionistPersonId varchar2(4) not null,

   PRIMARY KEY (BillId, ReceptionistPersonId),
   FOREIGN KEY (BillId) REFERENCES BILL_PAYMENT(BillId),
   FOREIGN KEY (ReceptionistPersonId) REFERENCES RECEPTIONIST(ReceptionistPersonId)
);
```

- **GETS_TREATMENT_PHARMACY**

```
CREATE TABLE GETS_TREATMENT_PHARMACY(

   MedicineCode varchar2(20) not null,
   TreatmentId varchar2(20) not null,
   Class2PatientId varchar2(4) not null,

   PRIMARY KEY (MedicineCode, TreatmentId, Class2PatientId),
   FOREIGN KEY (MedicineCode) REFERENCES PHARMACY(MedicineCode),
   FOREIGN KEY (TreatmentId) REFERENCES TREATMENT(TreatmentId),
   FOREIGN KEY (Class2PatientId) REFERENCES CLASS2_PATIENT(Class2PatientId)
);
```

5.1.2 Database State

We insert some values into the database in order to test our SQL create view and query statement. Here we just give one example of insertions as follows:

```
/*Inserting to PERSON*/
INSERT INTO PERSON
VALUES('P101', 'Emily', 'A', 'Navathe','2665 Main St., Denton, TX 75083' ,'F',DATE'1980-04-30');
```

PersonId	**FirstName**	**MiddleName**	**LastName**	**Address**	**Gender**	**DateOfBirth**
P101	Emily	A	Navathe	2665 Main St., Denton, TX 75083	F	30-04-1980
P102	Tom	B	Brown	263 Gree St., Dallas, TX 75076	M	12-01-1956
P103	Jimmy	C	Johnson	Apt 14, 3663 Beltline Blvd., Dallas, TX 75074	M	03-02-1980
P104	Sally	D	Smith	744 Walnut St., Dallas, TX 75074	F	26-03-1976
P105	Jeniffer	E	Smack	467 Parker St., Plano, TX 75076	F	05-04-1957

```
/*INSERTING INTO PERSON_PHONE*/
Insert INTO PERSON_PHONE
VALUES ('P123',9728245628);
```

PersonId	**PhoneNumber**
P101	2222908717
P101	3333908717
P102	1234509871
P103	1234509762
P103	3330926762
P103	5409871984

```
/*INSERTING INTO EMPLOYEE*/
INSERT INTO EMPLOYEE
VALUES('P101', DATE'2001-01-01', 150000, 'DOCTOR');
```

EmployeePersonId	StartDate	Salary	Designation
P101	01-01-2001	150000	DOCTOR
P102	20-05-2002	150000	DOCTOR
P103	21-04-2001	150000	DOCTOR
P106	11-10-2001	75000	NURSE
P107	21-09-2005	75000	NURSE
P109	10-01-2001	60000	RECEPTIONIST
P110	10-10-2010	60000	RECEPTIONIST

```
/*INSERTING INTO DOCTOR*/
INSERT INTO DOCTOR
VALUES ('P101', 'Cardiologist', 'PERMANENT_DOCTOR');
```

DoctorPersonId	Specialization	DoctorType
P101	Cardiologist	PERMANENT_DOCTOR
P102	Cardiologist	PERMANENT_DOCTOR
P103	Neurologist	VISITING
P104	Physician	PERMANENT_DOCTOR
P105	ENT Specialist	TRAINEE

```
/*INSERTING INTO NURSE*/
INSERT INTO NURSE
VALUES('P106');
```

NursePersonId
P106
P107
P108

```
/*INSERTING INTO RECEPTIONIST*/
INSERT INTO RECEPTIONIST
VALUES('P109');
```

ReceptionistPersonId
P109
P110

```
/*INSERTING INTO CLASS1_PATIENT*/
INSERT INTO CLASS1_PATIENT
VALUES ('C101','P111','P104');
```

Class1PatientId	PersonId	DoctorPersonId
C105	P111	P105
C104	P114	P105
C106	P115	P104
C107	P116	P104
C108	P113	P101
C109	P117	P104
C110	P118	P104
C111	P119	P101
C112	P112	P101

```
/*INSERTING INTO TRAINEE*/
INSERT INTO TRAINEE
VALUES('P105');
```

TraineePersonId
P105

```
/*INSERTING INTO VISITING*/
INSERT INTO VISITING
VALUES('P103');
```

VisitingPersonId
P103

```
/*INSERTING INTO PERMANENT_DOCTOR*/
INSERT INTO PERMANENT_DOCTOR
VALUES('P101');
```

PermanentPersonId
P101
P102

```
/*INSERTING INTO ROOMS*/
INSERT INTO ROOMS
VALUES ('11','PRIVATE',10,'P106');
```

RoomNumber	RoomType	RoomDuration	NursePersonId
11	PRIVATE	10	P106

12	PRIVATE	5	P106
24	SEMI-PRIVATE		P107
25	SEMI-PRIVATE	7	P107
34	STANDARD	1	P108
35	STANDARD		P108

```
/*INSERTING INTO CLASS2_PATIENT*/
INSERT INTO CLASS2_PATIENT
VALUES('C201', 'P111', 'P101', '11', DATE'2012-08-18');
```

Class2PatientId	**PersonId**	**DoctorPersonId**	**RoomNo**	**DateOfAdmit**
C201	P111	P101	11	18-08-2012
C202	P112	P101	12	18-08-2011
C203	P113	P102	21	28-04-2011
C204	P114	P103	22	08-04-2013
C205	P115	P103	23	08-01-2012
C206	P116	P104	31	18-01-2014
C207	P117	P104	32	08-09-2016

```
/*INSERTING INTO PATIENT_RECORDS*/
INSERT INTO PATIENT_RECORDS
VALUES('R101',DATE'2012-08-16', DATE'2012-08-18','HEART-ISSUE', 'P111');
```

RecordId	**DateOfAppointment**	**DateOfVisit**	**Description**	**PersonId**
R101	16-08-2012	18-08-2012	HEART-ISSUE	P111
R102	16-08-2011	18-08-2011	HEART-ISSUE	P112
R104	05-04-2013	07-04-2013	NEUROLOGICAL-DISORDER	P114
R105	06-01-2012	08-01-2012	NEUROLOGICAL-DISORDER	P115
R107	08-09-2016	08-09-2016	GENERAL-ISSUE	P117
R108	26-02-2014	28-02-2014	HEARING-ISSUE	P118

```
/*INSERTING INTO PHARMACY*/
INSERT INTO PHARMACY
VALUES('M101', 'Rivaroxaban', '25', '1000', DATE'2020-01-01');
```

MedicineCode	**MedicineName**	**MedicinePrice**	**Quantity**	**DateOfExpiry**
M101	Rivaroxaban	25	1000	01-01-2020
M102	Dabigatran	50	1500	01-08-2024
M103	Apixaban	10	700	01-07-2020

```
/*INSERTING INTO TREATMENT*/
INSERT INTO TREATMENT
```

VALUES('T101', 'Bypass Grafting', 30, 'HEART-ISSUE');

TreatmentId	TreatmentName	TreatmentDuration	TreatmentDescription
T101	Bypass Grafting	30	HEART-ISSUE
T102	Revascularization	20	HEART-ISSUE
T107	Brain Mapping	12	NEUROLOGICAL-DISORDER
T108	Cyberknife	15	NEUROLOGICAL-DISORDER
T114	Treatment for Flu	10	GENERAL-ISSUE
T115	Treatment for Common Cold	3	GENERAL-ISSUE
T116	Treatment for Ear	5	GENERAL-ISSUE
T117	Treatment for Nose	4	GENERAL-ISSUE

/*INSERTING INTO TREATMENT_MEDICINE*/
INSERT INTO TREATMENT_MEDICINE VALUES('T101', 'M101');

TreatmentId	MedicineCode
T101	M101
T101	M102
T101	M103
T101	M104
T101	M105
T102	M101
T102	M103

/*INSERTING INTO BILL_PAYMENT*/
INSERT INTO BILL_PAYMENT
VALUES('B101', 'P111', DATE'2012-08-25', 250.00);

BillId	PersonId	DateOfPayment	TotalAmountDue
B101	P111	25-08-2012	250
B102	P112	23-08-2011	250
B103	P113	10-05-2011	200.75
B104	P114	18-04-2013	100
B105	P115	18-01-2012	120.22

/*INSERTING INTO BILL_CASH*/
INSERT INTO BILL_CASH VALUES('B101', 250.00);

BillId	BillAmount
B101	250
B102	250
B103	200.75

/*INSERTING INTO VISITOR*/

INSERT INTO VISITOR
VALUES('V101', 'C201', 'Alex', '223 gold Rd' ,9876785549);

VisitorId	**Class2PatientId**	**VisitorName**	**VisitorAddress**	**VisitorContact**
V106	C203	George	90 Elv Street	1234569898
V107	C204	Almond	97 Elven Street	5555098198
V108	C205	Ghost	9888 Richmen Street	2222444678
V109	C205	Rider	9888 Richmen Street	9876540090
V110	C206	Ravi	7760 McCallum Blvd	4444569898
V111	C207	Sahith	7760 McCallum Blvd	8907654132

/*INSERTING INTO BILL_INSURANCE*/
INSERT INTO BILL_INSURANCE
VALUES('I101', 'B104', 50.00);

InsuranceId	**BillId**	**BillAmount**
I101	B104	50
I102	B105	60.22
I103	B106	60.35
I104	B107	48.25

/*INSERTING INTO INSURANCE_DETAILS*/
INSERT INTO INSURANCE_DETAILS
VALUES('I101', ' Aetna Health Insurance', '2500');

InsuranceId	**InsuranceProvider**	**InsuranceCoverage**
I101	Aetna Health Insurance	2500
I102	Assurant Health	3500
I103	Aetna Health Insurance	1500
I104	BCBS	7500
I105	Celtic	1000

/*INSERTING INTO MAINTAINS_PAYMENTS*/
INSERT INTO MAINTAINS_PAYMENTS
VALUES('B101', 'P109');

BillId	**ReceptionistPersonId**
B107	P109
B108	P109
B109	P110
B110	P110
B111	P110

```
/*INSERTING INTO MAINTAINS_RECORDS*/
INSERT INTO MAINTAINS_RECORDS
VALUES('R101', 'P109');
```

RecordId	**ReceptionistPersonId**
R109	P109
R110	P109
R111	P110
R112	P110

```
/*INSERTING INTO GETS_TREATMENT_PHARMACY*/
INSERT INTO GETS_TREATMENT_PHARMACY
VALUES('T101', 'M101', 'C201');
```

TreatmentId	**MedicineCode**	**Class2PatientId**
T101	M103	C214
T105	M103	C210
T101	M104	C201
T104	M104	C215
T106	M104	C209

5.2 Creation of views

5.2.1 TopDoctor

```
CREATE VIEW TopDoctor AS
SELECT p.firstname, p.lastname, e.startdate
FROM PERSON P, EMPLOYEE E
WHERE e.employeepersonid = p.personid AND p.personid IN
(
SELECT p.personid
FROM CLASS2_PATIENT C2, PERSON P
WHERE c2.doctorpersonID = p.personid AND p.personid IN
(
SELECT  p.personid
FROM CLASS1_PATIENT C1, PERSON P
WHERE c1.doctorpersonid = p.personid
GROUP BY p.personid
HAVING COUNT(c1.doctorpersonid) > 5
)
GROUP BY p.personid
HAVING COUNT(c2.doctorpersonid) > 10
);
```

5.2.2 TOPTREATMENT

```
CREATE view TopTreatment as
SELECT distinct T.TreatmentName , B.TOTALAMOUNTDUE
from Treatment T, GETS_TREATMENT_PHARMACY G , Class2_Patient C2, BILL_payment B
where T.TreatmentID = G.TreatmentID and G.CLASS2PATIENTID = C2.CLASS2PATIENTID
AND C2.PERSONID = B.PERSONID AND T.TreatmentID = (
   SELECT   G.TreatmentID
   FROM     GETS_TREATMENT_PHARMACY G
   GROUP BY G.TreatmentID
   ORDER BY COUNT(*) DESC
   OFFSET 0 ROWS FETCH NEXT 1 ROWS ONLY
);
```

5.2.3 ReorderMeds

```
CREATE VIEW ReorderMeds AS
SELECT *
FROM PHARMACY
WHERE (dateofexpiry - sysdate) < 30 OR quantity < 1000;
```

5.2.4 - PotentialPatient

```
CREATE VIEW PotentialPatient AS
SELECT p.firstname, p.lastname, p.personid, ph.phonenumber
FROM PERSON P INNER JOIN PERSON_PHONE PH ON P.PERSONID = PH.PERSONID
GROUP BY p.firstname, p.lastname, p.personid, ph.phonenumber
HAVING p.personid = (
   SELECT PERSONID
   FROM class1_patient
   WHERE CLASS1_PATIENT.personid NOT IN (SELECT PERSONID FROM CLASS2_PATIENT)
   GROUP BY personid
   HAVING COUNT(class1patientid) > 3 );
```

5.2.5 - MostFrequentIssues

```
CREATE VIEW MostFrequentIssues AS
SELECT TREATMENTNAME, TREATMENTDESCRIPTION
FROM TREATMENT
WHERE TREATMENTDESCRIPTION = (

      SELECT DESCRIPTION
      FROM PATIENT_RECORDS
      GROUP BY DESCRIPTION
      HAVING COUNT(DESCRIPTION) = (

      SELECT MAX(c) as maxcount
      FROM ((
         SELECT DESCRIPTION, COUNT(DESCRIPTION) as c
         FROM PATIENT_RECORDS
         GROUP BY DESCRIPTION))));
```

5.3 Creation of SQL queries

5.3.1 For each Doctor class, list the start date and specialization of the doctor.

```
SELECT D.Specialization, E.StartDate
FROM EMPLOYEE E, DOCTOR D
WHERE e.employeepersonid = d.doctorpersonid;
```

5.3.2 Find the names of employees who have been admitted to the hospital within 3 months of joining.

```
SELECT p.firstname, p.middlename, p.lastname
FROM CLASS2_PATIENT C2, EMPLOYEE E, PERSON P
WHERE c2.personid = e.employeepersonid AND E.employeepersonid = p.personid AND
((c2.dateofadmit-e.startdate) < 90);
```

5.3.3 Find the average age and class (trainee, visiting or permanent) of top 5 doctors in the hospital.

```
SELECT d.doctortype, ROUND(avg((sysdate - p.dateofbirth)/365)) as AverageAge
FROM TOPDOCTOR T, DOCTOR D, PERSON P
WHERE T.PERSONID = D.DOCTORPERSONID and P.PERSONID = D.DOCTORPERSONID
GROUP BY D.DOCTORTYPE;
```

5.3.4 Find the name of medicines associated with the most common treatment in the hospital.

```
SELECT P.MedicineName
FROM TopTreatment TT, Treatment T, Treatment_Medicine TM, Pharmacy P
WHERE TT.TreatmentName = T.TreatmentName and T.TreatmentId = TM.TreatmentId and
TM.MedicineCode = P.MedicineCode
```

5.3.5 Find all the doctors who have not had a patient in the last 5 months.

```
SELECT D.DOCTORPERSONID
FROM DOCTOR D
```

```
MINUS

SELECT DISTINCT D.DOCTORPERSONID
FROM DOCTOR D, CLASS1_PATIENT C1, CLASS2_PATIENT C2, BILL_PAYMENT B
WHERE (D.DoctorPersonId = C1.DoctorPersonId or D.DoctorPersonId = C2.DoctorPersonId) AND (C1.personid = B.PersonId or C2.personId = B.personid) and (sysdate - b.dateofpayment)/12 < 5
```

5.3.6 Find the total number of patients who have paid completely using insurance and the name of the insurance provider.

```
SELECT I.INSURANCEPROVIDER, COUNT(B.PersonId) as NO_OF_PATIENTS
FROM BILL_PAYMENT B, bill_insurance BI, INSURANCE_DETAILS I
WHERE B.BillId = BI.BILLID and BI.INSURANCEID = I.INSURANCEID AND B.TotalamountDue = BI.BILLAMOUNT
Group by I.INSURANCEPROVIDER
```

5.3.7 Find the most occupied room in the hospital and the duration of the stay.

```
SELECT C2.ROOMNO, R.ROOMDURATION
    FROM CLASS2_PATIENT C2, ROOMS R
    WHERE C2.ROOMNO = R.ROOMNUMBER
    GROUP BY C2.ROOMNO, R.ROOMDURATION
    HAVING COUNT(C2.ROOMNO) = (
       SELECT MAX(c) as maxcount
       FROM ((
              SELECT ROOMNO, COUNT(ROOMNO) as c
              FROM CLASS2_PATIENT
              GROUP BY ROOMNO))
       )
```

5.3.8 Find the year with the maximum number of patients visiting the hospital and the reason for their visit.

```
SELECT extract(year from DATEOFVISIT) as YEAR ,DESCRIPTION
from PATIENT_RECORDS
where extract(year from DATEOFVISIT)  IN

(SELECT extract(year from DATEOFVISIT) as year
FROM PATIENT_RECORDS
GROUP BY extract(year from DATEOFVISIT)
HAVING COUNT(RECORDID) = (
SELECT MAX(c) as maxcount
FROM ((
SELECT extract(year from DATEOFVISIT), COUNT(RECORDID) as c
FROM PATIENT_RECORDS
GROUP BY extract(year from DATEOFVISIT))))
)
```

5.3.9 Find the duration of the treatment that is provided the least to patients.

```
SELECT G.TREATMENTID, T.TREATMENTDURATION
FROM GETS_TREATMENT_PHARMACY G, TREATMENT T
WHERE G.TREATMENTID = T.TREATMENTID
GROUP BY G.TREATMENTID, T.TREATMENTDURATION
HAVING COUNT(G.TREATMENTID) = (

SELECT MIN(c) as maxcount
FROM ((
   SELECT TREATMENTID, COUNT(TREATMENTID) as c
   FROM GETS_TREATMENT_PHARMACY
   GROUP BY TREATMENTID)))
```

5.3.10 List the total number of patients that have been admitted to the hospital after the most current employee has joined.

```
SELECT count(Class2PatientID) as totalNumPatientsAdmitted
FROM class2_patient
WHERE DATEOFADMIT > (
   SELECT STARTDATE
   FROM EMPLOYEE
   ORDER BY STARTDATE desc
   OFFSET 0 ROWS FETCH NEXT 1 ROWS ONLY
);
```

5.3.11 List all the patient records of those who have been admitted to the hospital within a week of being consulted by a doctor.

```
SELECT Distinct C2.class2patientid, p.firstname, p.lastname
FROM PATIENT_RECORDS R, CLASS2_PATIENT C2, Person P
WHERE R.DATEOFVISIT <= C2.DateOfAdmit and (C2.DateOfAdmit - R.DATEOFVISIT) <= 7
and p.personid = c2.personid
```

5.3.12 Find the total amount paid by patients for each month in the year 2017.

```
select extract(month from DATEOFPAYMENT) as MONTH ,SUM(TotalamountDue) as Sum
from bill_payment
where extract(year from DATEOFPAYMENT)  = 2017
GROUP BY extract(month from DATEOFPAYMENT)
```

5.3.13 Find the name of the doctors of patients who have visited the hospital only once for consultation and have not been admitted to the hospital.

```
SELECT P.firstname, p.lastname
From Person p, Class1_patient c1
WHERE c1.doctorPersonId = p.personid and c1.personId IN
(SELECT PERSONID
        FROM class1_patient
        WHERE CLASS1_PATIENT.personid NOT IN (SELECT PERSONID FROM
        CLASS2_PATIENT)
        GROUP BY personid
        HAVING COUNT(class1patientid) = 1
)
```

5.3.14 Find the name and age of the potential patients in the hospital.

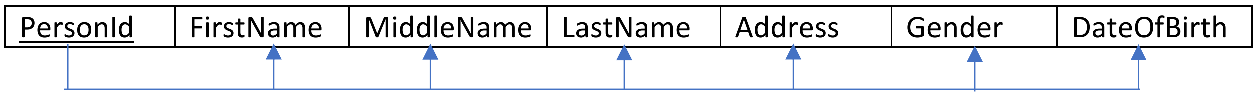
```
SELECT  DISTINCT p.firstname, p.lastname,
ROUND((sysdate - p.dateofbirth)/365) as Age
FROM POTENTIALPATIENT PP, PERSON P
WHERE pp.personid = p.personid
```

6. Dependency Diagram

6.1 PERSON & PERSON_PHONE

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema PERSON, PersonId. Therefore, every other attribute of this relational schema is functionally dependent on PersonId. Similarly, there is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema PERSON_PHONE, PersonId. Therefore, every other attribute of this relational schema is functionally dependent on PersonId. The dependency diagrams are shown as

PERSON



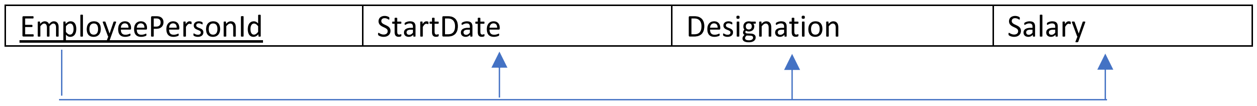
PERSON_PHONE

PersonId	PhoneNumber

6.2 EMPLOYEE

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema EMPLOYEE, EmployeePersonId. Therefore, every other attribute of this relational schema is functionally dependent on EmployeePersonId. The dependency diagram is shown as

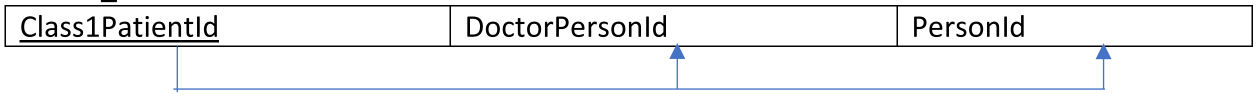
EMPLOYEE



6.3 CLASS1_PATIENT

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema CLASS1_PATIENT, Class1PatientId. Therefore, every other attribute of this relational schema is functionally dependent on Class1PatientId. The dependency diagram is shown as

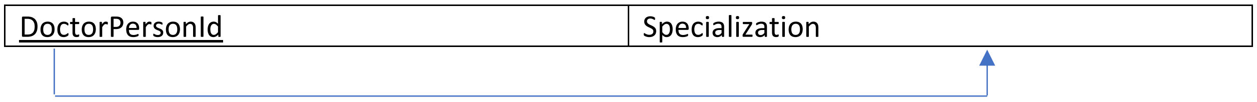
CLASS1_PATIENT



6.4 DOCTOR

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema DOCTOR, DoctorPersonId. Therefore, every other attribute of this relational schema is functionally dependent on DoctorPersonId. The dependency diagram is shown as

DOCTOR



6.5 NURSE, RECEPTIONIST, TRAINEE, VISITING, PERMANENT_DOCTOR

All the below relations have a single attribute

NURSE

NursePersonId

RECEPTIONIST

ReceptionistPersonId

TRAINEE

TraineePersonId

VISITING

VistingPersonId

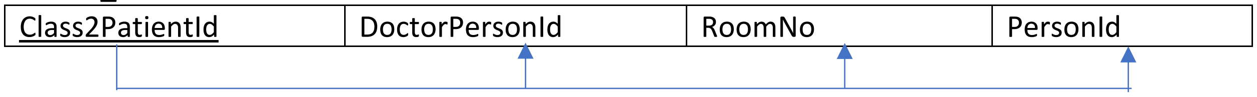
PERMANENT_DOCTOR

PermanentPersonId

6.6 CLASS2_PATIENT

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema CLASS2_PATIENT, Class2PatientId. Therefore, every other attribute of this relational schema is functionally dependent on Class2PatientId. The dependency diagram is shown as

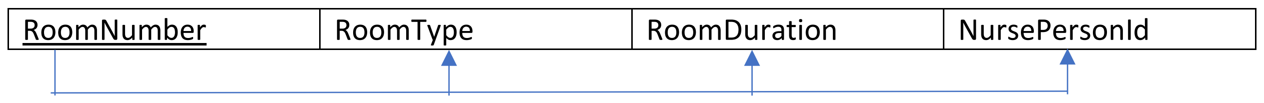
CLASS2_PATIENT



6.7 ROOMS

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema ROOMS, RoomNumber. Therefore, every other attribute of this relational schema is functionally dependent on RoomNumber. The dependency diagram is shown as

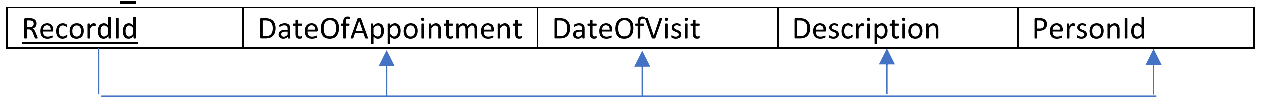
ROOMS



6.8 PATIENT_RECORDS

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema PATIENT_RECORDS, RecordId. Therefore, every other attribute of this relational schema is functionally dependent on RecordId. The dependency diagram is shown as

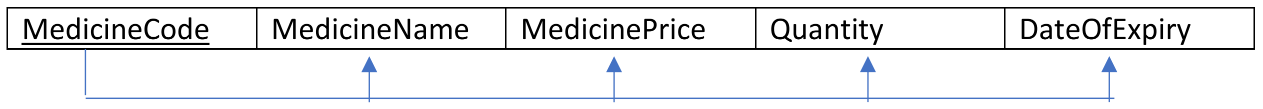
PATIENT_RECORDS



6.9 PHARMACY

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema PHARMACY, MedicineCode. Therefore, every other attribute of this relational schema is functionally dependent on MedicineCode. The dependency diagram is shown as

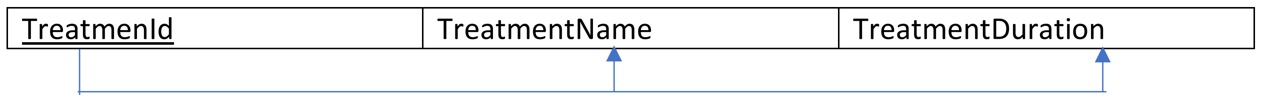
PHARMACY



6.10 TREATMENT

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema TREATMENT, TreatmentId. Therefore, every other attribute of this relational schema is functionally dependent on TreatmentId. The dependency diagram is shown as

TREATMENT



6.11 TREATMENT_MEDICINE

Here both attributes combine to form a Primary Key.

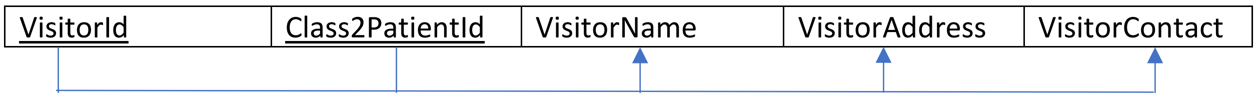
TREATMENT_MEDICINE

TreatmentId	MedicineCode

6.12 VISITOR

There are two attributes in the left-hand side of the functional dependencies, which combine to form the key of relational schema VISITOR, VisitorId & Class2PatientId. Therefore, every other attribute of this relational schema is functionally dependent on VisitorId & Class2PatientId. The dependency diagram is shown as

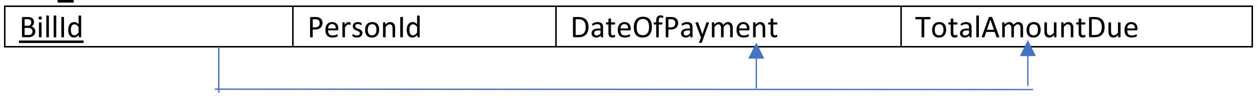
VISITOR



6.13 BILL_PAYMENT

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema BILL_PAYMENT, BillId. Therefore, every other attribute of this relational schema is functionally dependent on TreatmentId. The dependency diagram is shown as

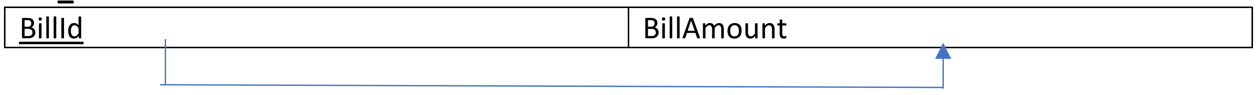
BILL_PAYMENT



6.14 BILL_CASH

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema BILL_CASH, BillId. Therefore, every other attribute of this relational schema is functionally dependent on BillId. The dependency diagram is shown as

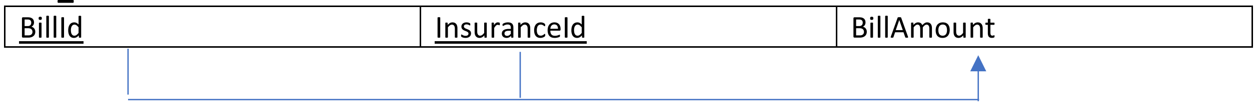
BILL_CASH



6.15 BILL_INSURANCE

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema BILL_INSURANCE, BillId. Therefore, every other attribute of this relational schema is functionally dependent on BillId. The dependency diagram is shown as

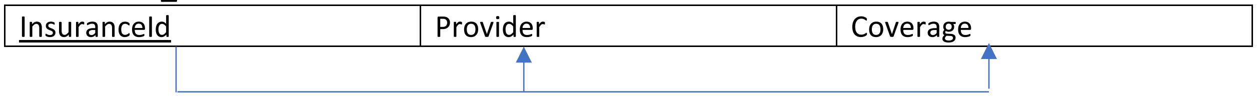
BILL_INSURANCE



6.16 INSURANCE_DETAILS

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema INSURANCE_DETAILS, InsuranceId. Therefore, every other attribute of this relational schema is functionally dependent on InsuranceId. The dependency diagram is shown as

INSURANCE_DETAILS



InsuranceId	Provider	Coverage

6.17 MAINTAINS_RECORDS

Here both attributes combine to form a Primary Key.

MAINTAINS_RECORDS

RecordId	ReceptionistPersonId

6.18 MAINTAINS_PAYMENTS

Here both attributes combine to form a Primary Key.

MAINTAINS_PAYMENTS

ReceptionistId	BillId

6.19 GETS_TREATMENT_MEDICINE

Here all the attributes combine to form a Primary Key.

GETS_TREATMENT_PHARMACY

MedicineCode	TreatmentId	Class2PersonId

7. Conclusion

In this final report we summarized all the necessary descriptions and solutions for DallasHealthCare database, including process and result of EER diagrams, relational schemas in third normal form, SQL statements to create database, create view and solve corresponding queries, as well as dependency diagram. We also implemented the whole database in Oracle and using a database state to test every query. 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.