Project for Database Design

Phase III. Implementation

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

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

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

Poonam Gillurkar   
[pxg180009@utdallas.edu](mailto:pxg180009@utdallas.edu)

(Week 11-15: Oct.31-Nov.26)

1. **Pre-Illumination**

For clearly describing the implementation of our database, we separate this report into four sections.

In Section 1 we normalize the original relational schema into third normal form and changed part of our relational schema because of some requirement from Phase II. We then explained what are changed.

In Section 2 we drew a dependency diagram for each relation table one by one.

In Section 3 we began our process of building a database in Oracle using SQL statements, which contains three parts.

Part one is the creation of database, including tables, all other structures as well as data type and format, Part two is the creation of views corresponding to five distinct requirements from Question d, and Part three is the creation of Queries to satisfy the 14 requirements from Question e.

Finally, in Section 4, a short summary is given at the end of this report.

1. **3NF Normalized Relational Schema**

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.

|  |
| --- |
| **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 |
| Description |

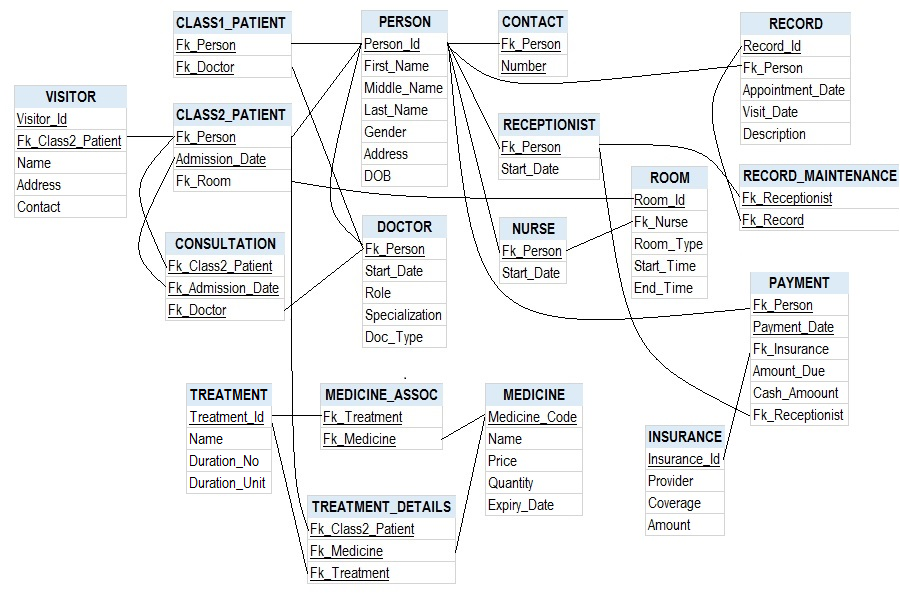
* The INSURANCE table has primary key Insurance\_Id. The other non-prime attributes Provider, Coverage, and Amount are directly functionally dependent on the primary key.

|  |
| --- |
| **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 |
|  |

The Normalized Relational Schema can be found in Figure 1



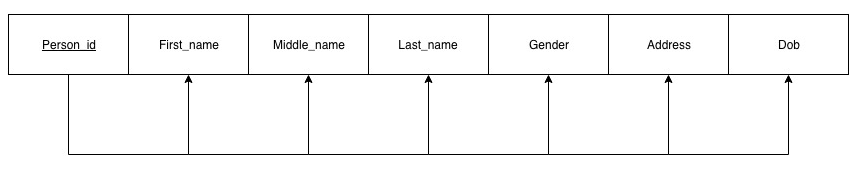
*Figure 1. Normalized relational Schema for Dallas Care database*

1. **Dependency Diagram**

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

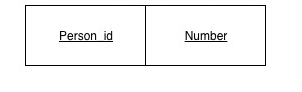
**PERSON** : {Person\_id} 🡪 {First\_name, Middle\_name, Last\_name, Gender, Address, Dob}

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

****

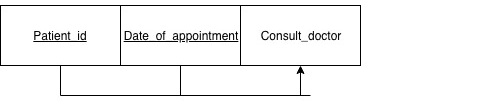
**CONTACT:** {Person\_id, Number} 🡪 {Person\_id, Number}

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.

****

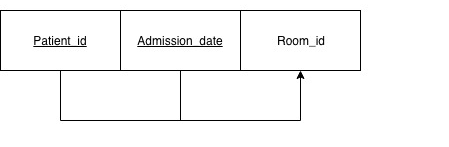
**CLASS1\_PATIENT :**  {Patient\_id, Date\_of\_appointment} 🡪 {Consult\_doctor}

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}

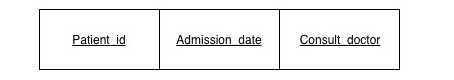
In this relation Patient\_id, Admission\_date for the primary key. Hence the remaining attributes are functionally dependent on {Patient\_id, Admission\_date}.

****

**CONSULTATION : { Patient\_id**,Admission\_date,Consult\_doctor} 🡪 {Patient\_id, Admission\_date, Consult\_doctor}

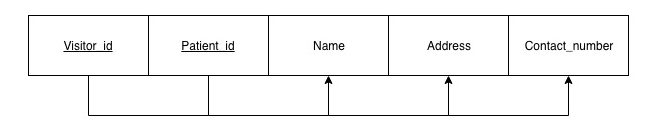
In this relation we have all the attributes as the key, using the property that X 🡪 X .

We can define the functionaly dependence of this relation.



**VISITOR : {**Visitor\_id,Patient\_id} 🡪 {Name,Address,Contact\_number}

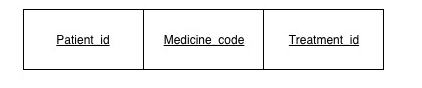
As the key here consists of single attribute hence we have functional dependency with all other attributes.



**TREATMENT\_DETAILS:**

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

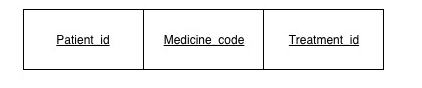
The key for this table is the complete set of attributes, hence we can say that attribute closure of the key of this set includes all the attributes of the relation and hence defines functional dependency in this way.



**MEDICINE\_ASSOC**:

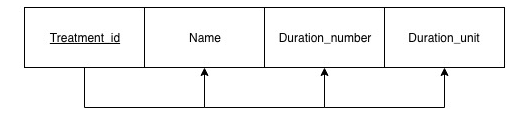
{Patient\_id, Medicine\_code, Treatment\_id} 🡪 {Patient\_id, Medicine\_code, Treatment\_id}

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



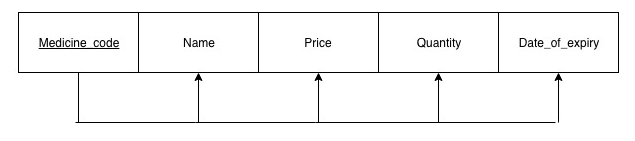
**TREATMENT**: {Treatment\_id} 🡪 {Name, Duration\_number, Duration\_unit}

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



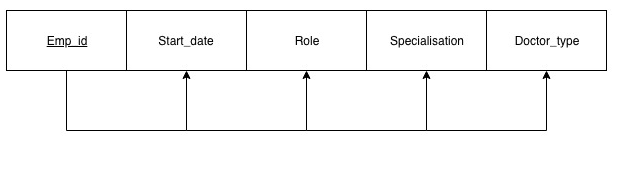
**PHARMACY:** {Medicine\_code} 🡪 {Name, Price, Quantity, Date\_of\_expiry}

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

****

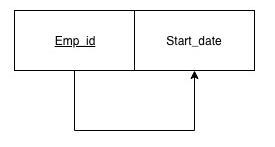
**DOCTOR: {**Emp\_id} 🡪 {Start\_date, Role, Specialisation, Doctor\_type}

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



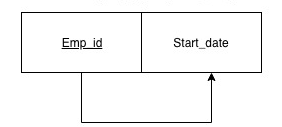
**RECEPTIONIST: {**Emp\_id} 🡪 {Start\_date}

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

****

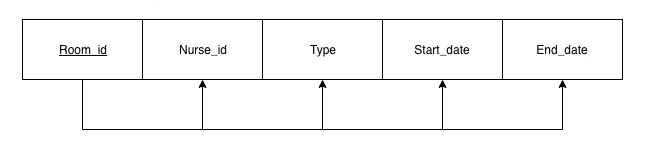
**NURSE:** {Emp\_id} 🡪 {Start\_date}

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

****

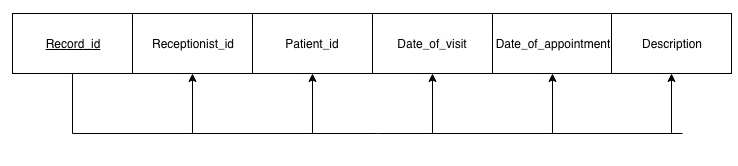
**ROOM: {**Room\_id, Nurse\_id} 🡪 {Type, Start\_date, End\_date}

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

****

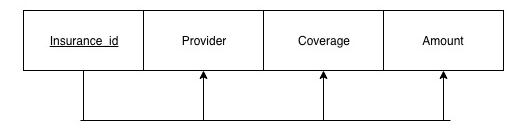
**RECORD: {**Record\_id} 🡪 {Receptionist\_id,Patient\_id,Date\_of\_visit, Date\_of\_appointment, Description}

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

****

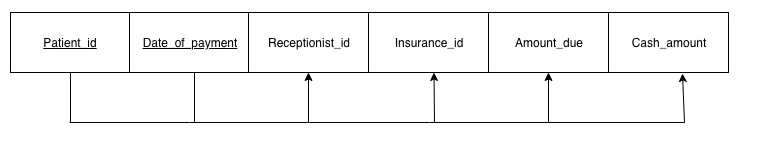
**INSURANCE: {**Insurance\_id} 🡪 {Provider, Coverage, Amount}

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.

****

1. **Implementation of Database** 
   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.

* + 1. **Table Creation**

Using SQL statement, we created the tables as follows:

* create table PERSON(

Person\_Id char(4) check(length(Person\_Id)=4 and Person\_Id like 'P\_ \_ \_' and cast(substr(Person\_Id,2,3) as decimal)>=100 and cast(substr(Person\_Id,2,3) as decimal)<=999),

First\_name varchar(50) not null,

Middle\_name varchar(50),

Last\_name varchar(50) not null,

Gender char(1) not null check(Gender in ('M','F','U')),

Address varchar(100) not null,

DOB date check(DOB<=curdate()),

primary key(Person\_Id));

* create table CONTACT(

Fk\_Person char(4) not null,

Number decimal(10) check(length(Number)=10),

primary key(Fk\_Person, Number),

foreign key(Fk\_Person) references PERSON(Person\_Id));

* create table NURSE(

Fk\_Person char(4),

Start\_Date date not null check(Start\_Date<=curdate()),

primary key(Fk\_Person),

foreign key(Fk\_Person) references PERSON(Person\_Id));

* create table DOCTOR(

Fk\_Person char(4),

Start\_Date date not null check(Start\_Date<=curdate()),

Role varchar(20) check(Role in ('Trainee','Permanent','Visiting')),

Specilization varchar(20),

Doc\_Type varchar(20) not null,

primary key(Fk\_Person),

foreign key(Fk\_Person) references PERSON(Person\_Id));

* create table RECEPTIONIST(

Fk\_Person char(4),

Start\_Date date not null check(Start\_Date<=curdate()),

primary key(Fk\_Person),

foreign key(Fk\_Person) references PERSON(Person\_Id));

* create table CLASS1\_PATIENT(

Fk\_Person char(4) not null,

Fk\_Doctor char(4) not null,

foreign key(Fk\_Person) references PERSON(Person\_Id),

foreign key(Fk\_Doctor) references DOCTOR(Fk\_Person));

* create table ROOM(

Room\_Id varchar(5),

Fk\_Nurse char(4) not null,

Room\_Type varchar(10) not null,

Start\_Time time,

End\_Time time,

primary key(Room\_Id),

foreign key(Fk\_Nurse) references NURSE(Fk\_Person));

* create table CLASS2\_PATIENT(

Fk\_Person char(4) not null,

Admission\_Date date not null,

Fk\_Room varchar(5) not null,

primary key(Fk\_Person, Admission\_Date),

foreign key(Fk\_Person) references PERSON(Person\_Id),

foreign key(Fk\_Room) references ROOM(Room\_Id));

* create table CONSULTATION(

Fk\_Class2\_Patient char(4) not null,

Fk\_Admission\_Date date check(Fk\_Admission\_Date>=curdate()),

Fk\_Doctor char(4) not null,

primary key(Fk\_Class2\_Patient, Fk\_Admission\_Date, Fk\_Doctor),

foreign key(Fk\_Class2\_Patient, Fk\_Admission\_Date) references CLASS2\_PATIENT(Fk\_Person, Admission\_Date),

foreign key(Fk\_Doctor) references DOCTOR(Fk\_Person));

* create table RECORD(

Record\_Id varchar(7),

Fk\_Person char(4),

Appointment\_Date date check(Appointment\_Date>=curdate()),

Visit\_Date date,

Description varchar(200),

primary key(Record\_Id),

foreign key(Fk\_Person) references PERSON(Person\_Id));

* create table RECORD\_MAINTENANCE(

Fk\_Receptionist char(4),

Fk\_Record varchar(7),

Primary\_Key(Fk\_Receptionist, Fk\_Record),

foreign key(Fk\_Record) references RECORD(Record\_Id),

foreign key(Fk\_Receptionist) references RECEPTIONIST(Fk\_Person));

* 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 PAYMENT(

Fk\_Person char(4),

Payment\_Date date not null check(Payment\_Date>=curdate()),

Fk\_Insurance varchar(10),

Amount\_Due decimal(10) not null,

Cash\_Amount decimal(10),

Fk\_Recetionist char(4) not null,

primary key(Fk\_Person, Payment\_Date),

foreign key(Fk\_Person) references PERSON(Person\_Id),

foreign key(Fk\_Recetionist) references RECEPTIONIST(Fk\_Person),

foreign key(Fk\_Insurance) references INSURANCE(Insurance\_Id));

* create table VISITOR(

Visitor\_Id varchar(10),

Fk\_Class2\_Patient char(4) not null,

Name varchar(30) not null,

Address varchar(50) not null,

Contact decimal(10),

primary key(Visitor\_Id, Fk\_Class2\_Patient),

foreign key(Fk\_Class2\_Patient) references CLASS2\_PATIENT(Fk\_Person));

* create table MEDICINE(

Medicine\_Code varchar(6),

Name varchar(20) not null,

Price decimal(10,2) not null check(Price>0),

Quantity decimal(4) not null check(Quantity>0),

Expiry\_Date date not null check(Expiry\_Date>=curdate()),

primary key(Medicine\_code));

* create table TREATMENT(

Treatment\_Id varchar(6),

Name varchar(20) not null,

Duration\_Number decimal(2) not null check(Duration\_Number>0),

Duration\_Unit date not null check(Duration\_Unit in ('Months','Days','Years')),

primary key(Treatment\_Id));

* create table MEDICINE\_ASSOC(

Fk\_Treatment varchar(6),

Fk\_Medicine varchar(6),

primary key(Fk\_Treatment, Fk\_Medicine),

foreign key(Fk\_Treatment) references TREATMENT(Treatment\_Id),

foreign key(Fk\_Medicine) references MEDICINE(Medicine\_Code));

* create table TREATMENT\_DETAILS(

Fk\_Class2\_Patient char(4),

Fk\_Medicine varchar(6),

Fk\_Treatment varchar(6),

primary key(Fk\_Class2\_Patient, Fk\_Medicine, Fk\_Treatment),

foreign key(Fk\_Class2\_Patient) references CLASS2\_PATIENT(Fk\_Person),

foreign key(Fk\_Treatment) references TREATMENT(Treatment\_Id),

foreign key(Fk\_Medicine) references MEDICINE(Medicine\_Code));

* + 1. 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:

INSERTION OF TABLE HOSPITAL\_PERSONNEL -------------------------------------------------------------------------------------------------------- insert into Hospital\_Personnel values ('000-00-0000', 'Emily', 'A', 'Navathe', date'1980-04-30', --------------------------------------------------------------------------------------------------------------------------- Table 2 shows the states for Hospital Personnel database schemas. (Example) Hospital\_Personnel Person\_ID FName M LName Birth\_Date Phone Address 000-00-0000 Emily A Navathe 1980-04-30 214-456-7626 2665 Main St., Denton, TX 75083 111-11-1111 Tom B Brown 1956-01-12 214-369-8759 263 Green St., Dallas, TX 75076 222-22-2222 Jimmy C Johnson 1980-02-03 469-765-9754 Apt.14, 3663 Beltline Blvd., Dallas, TX 75034’ 333-33-3333 Sally D Smith 1976-03-26 214-436-6336 744 Walnut St., Dallas, TX 75074 444-44-4444 Jeniffer E Smack 1957-04-05 214-567-4767 467 Parker St., Plano, TX 75076 555-55-5555 Smuel F Sunder 1997-05-20 972-456-2552 18675 Chase Oak St., Frisco, TX 75034 666-66-6666 Raja G Farage 2000-06-03 972-832-9317 556 Spring St., Mosquite, TX 75087 777-77-7777 Kenneth H Chenault 1979-07-16 214-134-8643 2445 Wolf Creek St., Greenvill, TX 75056 888-88-8888 Brett I Cotton 1956-08-19 469-295-3694 24567 Walnut St., The Colony, TX 75032 999-99-9999 Adam J Daley 1935-09-24 469-478-3688 865 Park St., Garland, TX 75073’ 101-01-0101 George K Cobb 1945-01-12 469-658-3978 263 Beltline Ave., Carleton, TX 75008 121-21-2121 Ivor L Page 1943-08-19 972-843-6823 1247 Floyd Rd., Richardson, TX 75075 131-31-3131 Joseph M Tomason 1969-11-17 972-987-9843 9454 RoyleLine Blvd., Irving, TX 75042 141-41-4141 Sara N Gaddis 1974-04-27 972-345-9734 345 King St., Fort Worth, TX 75023

(Please list all your table instance/data here.)

Till now we finished the process of creating tables and database states.

* 1. Creation of Views (Answer for Question d)
     1. Employees-Hired (This is an Example)

This view returns the First Name, Last Name, and Date Hired of all Hospital Employees CREAT VIEW Employees-Hired AS SELECT First\_Name, Last\_Name, Date\_Hired FROM Hospital\_Personnel, Employees WHERE Person\_ID=Emp\_ID

* 1. Creation of SQL Queries (Answer for Question f)

Now we give out the SQL Queries for each of 14 questions listed in Question e as follows:

* + 1. For each Job Class list all the staff members belonging to this class. (This is an example)

SELECT Job\_Class, Emp\_Type, First\_Name, Last\_Name FROM Employees, Hospital\_Personnel WHERE Person\_ID=Emp\_ID AND Emp\_type = 'S' ORDER BY Job\_Class;

1. **Conclusion**

In this report we modified the EER diagram and relational schemas for XXX Database according to the requirement of Phase III. We also give dependency diagram for each relational schema in database. Then we created tables for each relational schema and write the SQL statements for the views and queries listed in Question d and Question e.