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 |
| Date\_Of\_Birth |

* 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 Fk\_Person 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.

|  |
| --- |
| **CONTACTS** |
| Person\_ID |
| Phone\_number |

* Class1 Patient is a Person who can consult only one doctor. So CLASS1\_PATIENT is comprised of composite primary key Date\_of\_appointment and Patient\_ID, which is the foreign key referencing to Person\_ID which is the primary key of PERSON, and Consult\_doctor, which references the primary key of DOCTOR table. The non-prime attribute is directly 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 |
| Room\_ID |

* 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.

|  |
| --- |
| **CLASS2\_PATIENT\_CONSULTATION** |
| Patient\_ID |
| Admission\_date |
| Consult\_doctor |

* The VISITOR table has Visitor\_ID, and Patient\_ID and Fk\_Admission\_Date, 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 |
| Visitor\_name |
| Visitor\_address |
| Contact\_info |

* The TREATMENT\_DETAILS table has the foreign keys Fk\_Class2\_Patient referencing to primary key of CLASS2\_PATIENT, Fk\_Medicine referencing to primary key of MEDICINE, and Fk\_Treatment referencing to primary key of TREATMENT as the super key.

|  |
| --- |
| **TREATMENT\_DETAILS** |
| Fk\_Class2\_Patient |
| Fk\_Admission\_Date |
| Fk\_Medicine |
| Fk\_Treatment |

* The MEDICINE\_ASSOC table has the foreign keys Fk\_Treatment referencing to primary key of TREATMENT and Fk\_Medicine 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.

|  |
| --- |
| **MEDICINE\_ASSOC** |
| Fk\_Treatment |
| Fk\_Medicine |

* 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 |
| Treatment\_Name |
| Duration\_No |
| Duration\_Unit |

* The MEDICINE 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.

|  |
| --- |
| **MEDICINE** |
| Medicine\_Code |
| Medicine\_Name |
| Price |
| Quantity |
| Expiry\_Date |

* The DOCTOR table has primary key Fk\_Person, which is the foreign key referencing to Person\_Id of PERSON table. The other non-prime attributes Start\_Date, Role, Specialization, and Doc\_Type are directly functionally dependent on the primary key.

|  |
| --- |
| **DOCTOR** |
| Fk\_Person |
| Start\_Date |
| Doc\_Role |
| Specialization |
| Doc\_Type |

* The RECEPTIONIST table has primary key Fk\_Person, 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** |
| Fk\_Person |
| Start\_Date |

* The NURSE table has primary key Fk\_Person, 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** |
| Fk\_Person |
| Start\_Date |

* The ROOM table has the primary key Room\_Id. The Fk\_Nurse 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 directly functionally dependent on the primary key.

|  |
| --- |
| **ROOM** |
| Room\_Id |
| Fk\_Nurse |
| Room\_Type |
| Start\_Time |
| End\_Time |

* The RECORD table has the primary key Record\_Id. The Fk\_Person is the foreign key referencing to primary key of PERSON table. This, along with the other non-prime attributes Appointment\_Date, Visit\_Date, and Description, are directly functionally dependent on the primary key.

|  |
| --- |
| **RECORD\_APPOINTMENT** |
| Record\_Id |
| Fk\_Person |
| Appointment\_Date |
| Visit\_Date |
| App\_Description |

* The RECORD\_MAINTENANCE table has Fk\_Receptionist, which is the foreign key referencing to primary key of RECEPTIONIST table, and Fk\_Record, which is the foreign key referencing to primary key of RECORD table as the super key.

|  |
| --- |
| **RECORD\_MAINTENANCE** |
| Fk\_Receptionist |
| Fk\_Record |

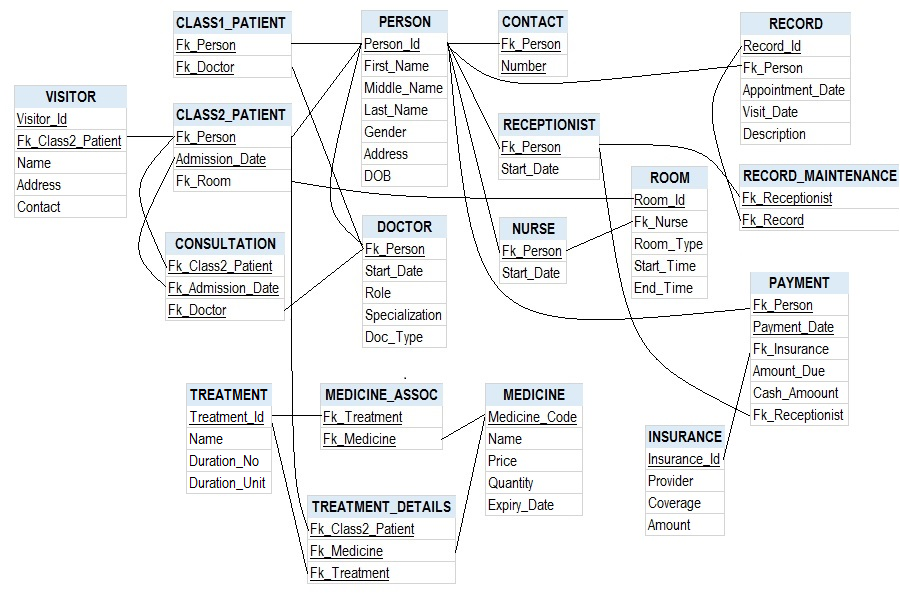
* The INSURANCE table has primary key Insurance\_Id. The other non-prime attributes Provider, Coverage, and Amount are directly functionally dependent on the primary key.

|  |
| --- |
| **INSURANCE** |
| Insurance\_Id |
| Provider |
| Coverage |
| Amount |

* The PAYMENT table has the composite primary key Fk\_Person, which is the foreign key referencing to primary key of PERSON table, and Payment\_Date. This, along with the other non-prime attributes Fk\_Insurance\_Id which is the foreign key referencing to primary key of INSURANCE table, Amount\_Due, Cash\_Amount, and Fk\_Receptionist, which is the foreign key referencing to primary key of RECEPTIONIST table, are directly functionally dependent on the primary key.

|  |
| --- |
| **PAYMENT** |
| Fk\_Person |
| Payment\_Date |
| Fk\_Insurance |
| Amount\_Due |
| Cash\_Amoount |
| Fk\_Receptionist |

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 from Figure 1 as follows:

* 1. Person

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relation Person, Person\_Id. Therefore, every other attribute of this relational schema is functionally dependent on Person\_Id.

The dependency diagram is shown as Figure 2.

*Figure 3. Dependency Diagram of Person*

2.2 XXXXXX Please list dependency diagram for every table using above forms.

2.XX Final Results

After drawing the dependency diagrams one after another, Figure XX shows the final results for the whole database including the ones who do not have any functional dependencies.

*Figure XX. Whole Dependency Diagram for Dallas Care Database*

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.

create database DALLAS\_CARE;

use DALLAS\_CARE;

* + 1. **Table Creation**

Using SQL statement, we created the tables as follows:

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)

);

commit;

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

);

commit;

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

);

commit;

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

);

commit;

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

);

commit;

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

);

commit;

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)

);

commit;

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

);

commit;

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

);

commit;

create table RECORD(

Record\_ID varchar(7),

Receptionist\_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

);

commit;

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)

);

commit;

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)

);

commit;

create table VISITOR(

Visitor\_ID varchar(10),

Patient\_ID varchar(4) not null,

Admission\_date date,

Visitor\_name varchar(30) not null,

Visitor\_address varchar(50) not null,

Contact\_info decimal(10),

primary key(Visitor\_ID,Patient\_ID, Admission\_date),

foreign key(Patient\_ID, Admission\_date) references CLASS2\_PATIENT(Patient\_ID, Admission\_date)

);

commit;

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)

);

commit;

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)

);

commit;

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)

);

commit;

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)

);

commit;

* + 1. Database State

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

(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.