PHASE III. IMPLEMENTATION

|  |  |
| --- | --- |
| Bharath Rudra  bxr180008@utdallas.edu | Karttik Reddy Yellu  kxy170003@utdallas.edu |
| Sai Ram Chappidi  sxc170016@utdallas.edu | Vamshider Reddy Voncha  vxv170013@utdallas.edu |

1. **Pre-Illumination**

For clearly describing the implementation of our database, we separate this report into four sections. In Section 1 we normalized the original relational schema into third normal form and changed part of our relational schema because of some requirement from Phase III. 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 14 requirements from Question e. Finally, a short summary is given at the end of this report.

1. **Modified 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 (VARCHAR2(200 )) | Gender (VARCHAR26 [CHAR])) | DateOfBirth (DATE) |
| PERSON\_PHONE | PersonId (VARCHAR2(4)) | PhoneNumber (NUMBER(11) |  |  |  |  |  |
| EMPLOYEE | EmployeePersonId (VARCHAR2(4)) | Salary (NUMBER(10,2) | StartDate (DATE) | Designation (VARCHAR2(200 [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)) |  |  |  |  |

1. **Dependency Diagram**

**2.1 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**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PersonId | FirstName | MiddleName | LastName | Address | Gender | DateOfBirth |

**PERSON\_PHONE**

|  |  |
| --- | --- |
| PersonId | PhoneNumber |

**2.2 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**

|  |  |  |  |
| --- | --- | --- | --- |
| EmployeePersonId | StartDate | Designation | Salary |

**2.3 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**

|  |  |  |
| --- | --- | --- |
| Class1PatientId | DoctorPersonId | PersonId |

**2.4 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**

|  |  |
| --- | --- |
| DoctorPersonId | Specialization |

**2.5 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 |

**2.6 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**

|  |  |  |  |
| --- | --- | --- | --- |
| Class2PatientId | DoctorPersonId | RoomNo | PersonId |

**2.7 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**

|  |  |  |  |
| --- | --- | --- | --- |
| RoomNumber | RoomType | RoomDuration | NursePersonId |

**2.8 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**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RecordId | DateOfAppointment | DateOfVisit | Description | PersonId |

**2.9 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**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MedicineCode | MedicineName | MedicinePrice | Quantity | DateOfExpiry |

**2.10 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**

|  |  |  |
| --- | --- | --- |
| TreatmenId | TreatmentName | TreatmentDuration |

**2.11 TREATMENT\_MEDICINE**

Here both attributes combine to form a Primary Key.

**TREATMENT\_MEDICINE**

|  |  |
| --- | --- |
| TreatmentId | MedicineCode |

**2.12 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**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VisitorId | Class2PatientId | VisitorName | VisitorAddress | VisitorContact |

**2.13 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**

|  |  |  |  |
| --- | --- | --- | --- |
| BillId | PersonId | DateOfPayment | TotalAmountDue |

**2.14 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**

|  |  |
| --- | --- |
| BillId | BillAmount |

**2.15 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**

|  |  |  |
| --- | --- | --- |
| BillId | InsuranceId | BillAmount |

**2.16 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 |

**2.17 MAINTAINS\_RECORDS**

Here both attributes combine to form a Primary Key.

**MAINTAINS\_RECORDS**

|  |  |
| --- | --- |
| RecordId | ReceptionistPersonId |

**2.18 MAINTAINS\_PAYMENTS**

Here both attributes combine to form a Primary Key.

**MAINTAINS\_PAYMENTS**

|  |  |
| --- | --- |
| ReceptionistId | BillId |

**2.19 GETS\_TREATMENT\_MEDICINE**

Here all the attributes combine to form a Primary Key.

**GETS\_TREATMENT\_PHARMACY**

|  |  |  |
| --- | --- | --- |
| MedicineCode | TreatmentId | Class2PersonId |

1. **Implementation of Database**

**3.1 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.

**3.1.1 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)

);

**3.1.2 Insertion and A 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 |

------------------------------------------------------------------------------------------------------------------------------------------

**3.2 Creation of Views**

**3.2.1 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

);

-------------------------------------------------------------------------------------------------------------------------------

**3.2.2 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

);

-------------------------------------------------------------------------------------------------------------------------------

**3.2.3 ReorderMeds**

CREATE VIEW ReorderMeds AS

SELECT \*

FROM PHARMACY

WHERE (dateofexpiry - sysdate) < 30 OR quantity < 1000;

-------------------------------------------------------------------------------------------------------------------------------

**3.2.4 - 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 );

-------------------------------------------------------------------------------------------------------------------------------

**3.2.5 - 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))));

-------------------------------------------------------------------------------------------------------------------------------

**3.2 Creation of SQL Queries**

3.3.1 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;

3.3.2 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);

3.3.3 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;

3.3.4 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

3.3.5 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

3.3.6 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

3.3.7 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))

)

3.3.8 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))))

)

3.3.9 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)))

3.3.10 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

);

3.3.11 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

3.3.12 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)

3.3.13 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

)

3.3.14 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

1. **Conclusion**

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