Project for Database Design

Phase IV. ocumentation

GROUP 14 AUTHORS

Haoran Lou

Yao Li

Yongliang Yang

# 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.

# Problem Description

Dallas Care is a hospital and medical care center. Dallas Care would like one relational database to be able to smoothly carry out their work in an organized way. The hospital has following modules: Person, Employee, Patient, Visitors, Pharmacy, Treatment, Rooms, Records and Medical Bill Payment.

A Person can be an Employee or a Class 1 Patient. Details of a person such as Person ID, Name (First, Middle, Last), Address, Gender, Date of Birth, and Phone number (one person can have more than one phone number) are recorded. A person ID should be in the format, ‘PXXX’, where XXX can be a value between 100 and 999. A Class 1 patient is a person who visits the hospital just for a doctor consultation. A person can be both an employee and a Class 1 patient.

Employee is further classified as Doctors, Nurses or Receptionists. The start date of the employee is recorded. The specialization of the doctor is stored and doctors are further classified into Trainee, Permanent or Visiting. Every Class 1 patient consults a doctor. A Class 1 patient can consult at most one doctor but one doctor can be consulted by more than one Class 1 patient.

A Class 2 patient is a someone who is admitted into the hospital. A Class 2 patient can be an Employee or a Class 1 Patient or both. A doctor attends Class 2 patients. One doctor can attend many Class 2 patients but a Class 2 patient can be attended to by at most 2 doctors. The date of patient being admitted into the hospital is recorded.

A Visitor log is maintained for the Class2 Patients, which stores information such as patient ID, visitor ID, visitor name, visitor’s address, and visitor’s contact information.

Pharmacy details such as Medicine code, Name, Price, Quantity and Date of expiration is recorded. The database also stores the information of the various kinds of treatments that are offered in the hospital. The treatment details such as ID, name, duration and associated medicines are recorded. When a treatment is assigned to a Class 2 patient, the treatment details, medicine details and patient details are recorded so that the doctor can easily access this information.

Nurses 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.

1. **Three Questions**

## 1.1 Is the ability to model super-class / subclass relationships likely to be important in such environment? Why or why not?

Solution:

Yes .it’s important.

Since all subclass entity inherit all attribute from super-class. in this way , all the duplicate attribute could avoid。

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

Add new attribute class\_2\_ID as surrogate key to class 1 patient and employ, can easily find the information of class\_2\_patient and inherit attribute from either class 1 patient or employ or both.

For each doctor, nurse, receptionist, Add new attribute \_id ,so easily to locate which event they involved.

Assume not all people has patient id, and just class 1 patients have patient id ,to assure employee and class 1 patient could be distinguished.

Add new relation medical information, to collect information of class 2patient and their treatment and which medicine involved.

Add new relation access,to help doctor handle all class 2 patient information.

Assume every class 1 patient come hospital for consult doctors.so we can get how many times the class 1 patient consult a doctor via relation Records’ visit date

## 1.3 Justify using a Relational DBMS like Oracle for this project.

Database management systems are systems that manage the full data structure and exercise full control over the data stored in an organization’s database. As compared to the traditional approaches of maintaining data in an organization, the modern system has a number of advantages. Organizational data is always susceptible to losses and therefore a proper system is highly recommended when a lot of data exists. The following are benefits of using database management systems in an organization.

Data Sharing Is Improved In The Organization

Proper database management systems help in gaining better access to data as well as better management of the data. In turn, better access helps the end users share the data fast and effectively across the organization.

Improvement In Data Security

A better framework is provided for enforcement of data privacy and security policies. The risks of data security breaches are minimized and corporate data is used properly.

Effective Data Integration

When data management is improved, it promotes an integrated picture of an organization’s operations. It becomes easy to see how operations in one segment of the organization affects other segments of the organization. Thus, effective integration of data is accomplished through the use of data management solutions.

Database Management Systems Minimize Data Inconsistency

Data inconsistency occurs when different versions of data exist in different places in an organization. By using a proper management system and data quality management tools, the problem of data inconsistency is minimized.

Better Access To Data

A management system helps in getting quick solutions to database queries, and therefore, data access is faster and more accurate. End users like sales people will have enhanced access to the data, enabling a faster sales cycle and a more sound decision making process.

Increase In Productivity Of The End User

By deploying the best data quality tools and database management systems, the productivity of the end user is increased. With the data management tools, the end users are empowered to make quick and informed decisions that can decide the success and failure of a company in a long run.

Quick Decision Making

When data is better managed and access is improved, quality information is generated and the user is enabled to make faster decisions. A good database managing system helps in providing a framework to facilitate data quality initiatives and in turn, higher quality information helps in making better, faster decisions in an organization.

Looking to implement a data management system into your organization? Look no further because RingLead Data Management Solutions (DMS) has got you covered. RingLead’s cloud-based DMS platform can capture, clean, protect and enrich all of the data inside your CRM or Marketing Automation System in real time. RingLead DMS Cleanse can remove duplicates currently clogging up your database and prevent more from entering from web submissions, list imports and manual entry using DMS Duplicate Prevention. Additionally, RingLead DMS Enrichment can enrich all of your data in batch or list using crowdsourced data for the highest per field match rates in the industry.

For the hospital, it meets all the requirements and the cost is not hinder the way compared with the business benefits. On the other hand, the hospital indeed requires the different access for different users and the whole system is not sample enough with just basic records.

# EER diagram with all assumptions



# Relational Schema in Third Normal Form

## Relational Schema



## Format for Every Relation































# All Requested SQL Statements

## 4.1Creation of Database with SQL Statements

### **4.1.1 Table Creation**

Using SQL statement, we created 15 tables as follows:

* ACCSEE1

1. **CREATE** **TABLE** ACCESS1
2. (
3. DOC\_ID **VARCHAR**(200) NOT NULL,
4. MEDICINE\_CODE **VARCHAR**(200) NOT NULL,
5. T\_ID **VARCHAR**(200) NOT NULL,
6. CLASS\_2\_ID **VARCHAR**(200) NOT NULL,
7. **primary** **key** (DOC\_ID, MEDICINE\_CODE, T\_ID, CLASS\_2\_PATIENT\_ID),
8. **foreign** **key** (MEDICINE\_CODE, T\_ID, CLASS\_2\_ID) **references** MEDICAL\_INFORMATION(MEDICINE\_CODE, T\_ID, CLASS\_2\_ID),
9. **foreign** **key** (DOC\_ID) **references** DOCTOR(DOC\_ID)
10. );

* CLASS\_1\_PATIENT

1. **CREATE** **TABLE** CLASS\_1\_PATIENT
2. (
3. PATIENT\_ID **VARCHAR**(255) NOT NULL,
4. PERSON\_ID **VARCHAR**(255) NOT NULL,
5. CLASS\_2\_ID **VARCHAR**(255),
6. EMPLOYEE\_NUM **VARCHAR**(255),
7. **primary** **key** (PATIENT\_ID),
8. **foreign** **key** (PERSON\_ID) **references** PERSON(PERSON\_ID)
9. );
10. **alter** **table** CLASS\_1\_PATIENT
11. **add** DOC\_ID **varchar**(255) null;
12. **alter** **table** CLASS\_1\_PATIENT
13. **add** **foreign** **key** (DOC\_ID) **references** DOCTOR(DOC\_ID);

* CLASS\_2\_PATIENT

1. **CREATE** **TABLE** CLASS\_2\_PATIENT
2. (
3. T\_ID **VARCHAR**(200),
4. EMPLOYEE\_NUM **VARCHAR**(200),
5. CLASS\_2\_ID **VARCHAR**(200) NOT NULL,
6. DATE\_OF\_ADMITTED **DATE**,
7. ROOM\_ID **VARCHAR**(255),
8. **primary** **key**(CLASS\_2\_ID),
9. **foreign** **key**(T\_ID) **references** TREATMENT(T\_ID),
10. **foreign** **key**(ROOM\_ID) **references** ROOM(ROOM\_ID)
11. );
12. **alter** **table** CLASS\_2\_PATIENT
13. **add** DOC\_ID **varchar**(255) null;
14. **alter** **table** CLASS\_2\_PATIENT
15. **add** **foreign** **key** (DOC\_ID) **references** DOCTOR(DOC\_ID);

* DOCTOR

1. **CREATE** **TABLE** DOCTOR
2. (
3. EMPLOYEE\_NUM **VARCHAR**(255) NOT NULL,
4. DOC\_ID **VARCHAR**(255),
5. DOC\_TYPE **VARCHAR**(255),
6. **primary** **key**(DOC\_ID),
7. **foreign** **key** (EMPLOYEE\_NUM) **references** EMPLOYEE(EMPLOYEE\_NUM)
8. );

* EMPLOYEE

1. **CREATE** **TABLE** EMPLOYEE
2. (
3. PERSON\_ID **VARCHAR**(255) NOT NULL,
4. EMPLOYEE\_NUM **VARCHAR**(255) NOT NULL,
5. START\_DATE **DATE**,
6. CLASS\_2\_ID **VARCHAR**(255),
7. **primary** **key** (EMPLOYEE\_NUM),
8. **foreign** **key** (PERSON\_ID) **references** PERSON(PERSON\_ID)
9. );

* MEDICAL\_INFORMATION

1. **CREATE** **TABLE** MEDICAL\_INFORMATION
2. (
3. MEDICINE\_CODE **VARCHAR**(200) NOT NULL,
4. T\_ID **VARCHAR**(200) NOT NULL,
5. CLASS\_2\_ID VARCHAR2(200) NOT NULL,
6. **primary** **key** (MEDICINE\_CODE,T\_ID,CLASS\_2\_ID),
7. **foreign** **key**(T\_ID) **references** TREATMENT(T\_ID),
8. **foreign** **key**(CLASS\_2\_ID) **references** CLASS\_2\_PATIENT(CLASS\_2\_ID)
9. );

* NURSE

1. **CREATE** **TABLE** NURSE
2. (
3. EMPLOYEE\_NUM **VARCHAR**(255) NOT NULL,
4. NURSE\_ID **VARCHAR**(255) NOT NULL,
5. **primary** **key**(NURSE\_ID),
6. **foreign** **key** (EMPLOYEE\_NUM) **references** EMPLOYEE(EMPLOYEE\_NUM)
7. );

* PYMENT

1. **CREATE** **TABLE** PAYMENT
2. (
3. PAYMENT\_ID VARCHAR2(20) NOT NULL,
4. PAYMENT\_DATE **DATE**,
5. TOTAL\_AMOUNT\_DUE **VARCHAR**(20),
6. RECEPTIONIST\_ID **VARCHAR**(20),
7. PATIENT\_ID **VARCHAR**(40),
8. T\_ID **VARCHAR**(20),
9. CASH\_AMOUNT **VARCHAR**(100),
10. I\_ID **VARCHAR**(60),
11. I\_PROVIDERID **VARCHAR**(100),
12. I\_COVERAGE **VARCHAR**(250),
13. I\_AMOUNT **VARCHAR**(100),
14. **primary** **key** (PAYMENT\_ID),
15. **foreign** **key** (RECEPTIONIST\_ID) **references** RECEPTIONIST(RECEPTIONIST\_ID),
16. **foreign** **key** (PATIENT\_ID) **references** CLASS\_1\_PATIENT(PATIENT\_ID)
17. );
18. **alter** **table** PAYMENT
19. **add** CLASS\_2\_ID **varchar**(255) null;
20. **alter** **table** PAYMENT
21. **add** **foreign** **key** (CLASS\_2\_ID) **references** CLASS\_2\_PATIENT(CLASS\_2\_ID);
22. **alter** **table** PAYMENT
23. **add** **foreign** **key** (T\_ID) **references** TREATMENT(T\_ID);

* PERSON

1. **CREATE** **TABLE** PERSON
2. (
3. PERSON\_ID **VARCHAR**(255) NOT NULL,
4. F\_NAME **VARCHAR**(255) NOT NULL,
5. M\_NAME **VARCHAR**(255),
6. L\_NAME **VARCHAR**(255) NOT NULL,
7. ADDRESS **VARCHAR**(255),
8. GENDER **VARCHAR**(255),
9. BITH\_DATE **DATE**,
10. PHONE\_NUMBER **VARCHAR**(255),
11. **primary** **key** (PERSON\_ID)
12. );

* PHARMACY

1. **CREATE** **TABLE** PHARMACY
2. (
3. MEDICINE\_CODE **VARCHAR**(200) NOT NULL,
4. P\_PRICE **VARCHAR**(200),
5. P\_NAME **VARCHAR**(200),
6. P\_QUANTITY **VARCHAR**(200),
7. P\_EXPIRECTION\_DATE **DATE**,
8. **primary** **key**(MEDICINE\_CODE)
9. );

* RECEPTIONIST

1. **CREATE** **TABLE** RECEPTIONIST
2. (
3. RECEPTIONIST\_ID **VARCHAR**(255) NOT NULL,
4. EMPLOYEE\_NUM **VARCHAR**(255) NOT NULL,
5. **primary** **key** (RECEPTIONIST\_ID),
6. **foreign** **key** (EMPLOYEE\_NUM) **references** EMPLOYEE(EMPLOYEE\_NUM)
7. );

* RECORDS

1. **CREATE** **TABLE** RECORDS
2. (
3. RECORD\_ID **VARCHAR**(255) NOT NULL,
4. RECEPTIONIST\_ID **VARCHAR**(255),
5. PATIENT\_ID **VARCHAR**(255),
6. VISIT\_DATE **DATE**,
7. APPOINTMENT **DATE**,
8. RECORD\_DESCRIPTION **VARCHAR**(255),
9. **primary** **key**(RECORD\_ID),
10. **foreign** **key** (RECEPTIONIST\_ID) **references** RECEPTIONIST(RECEPTIONIST\_ID),
11. **foreign** **key** (PATIENT\_ID) **references** CLASS\_1\_PATIENT(PATIENT\_ID)
12. );

* ROOM

1. **CREATE** **TABLE** ROOM
2. (
3. ROOM\_ID **VARCHAR**(255) NOT NULL,
4. ROOM\_TYPE **VARCHAR**(255),
5. ROOM\_DURATION **VARCHAR**(255),
6. NURSE\_ID **VARCHAR**(255) NOT NULL,
7. **primary** **key**(ROOM\_ID),
8. **foreign** **key** (NURSE\_ID) **references** NURSE(NURSE\_ID)
9. );

* TREATMENT

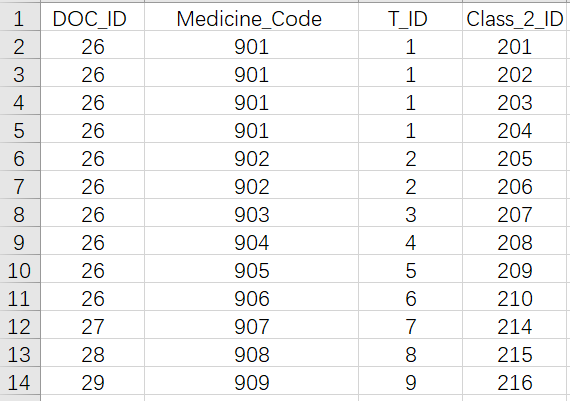
1. **CREATE** **TABLE** TREATMENT
2. (
3. T\_ID **VARCHAR**(200) NOT NULL,
4. T\_NAME **VARCHAR**(250),
5. T\_DURATION **VARCHAR**(200),
6. **primary** **key**(T\_ID)
7. );

* VISITOR\_LOG

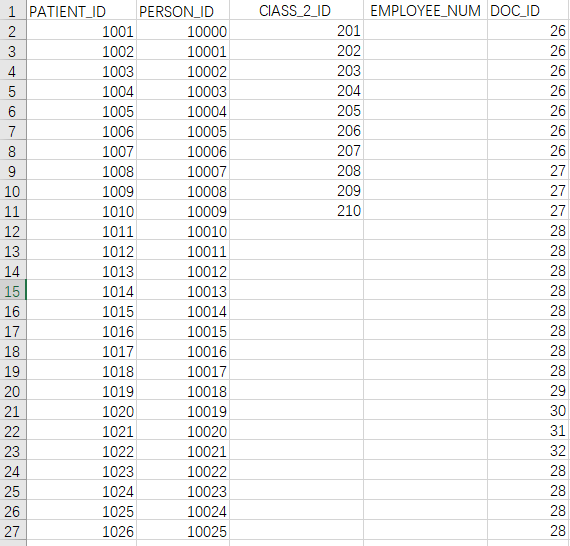
1. **CREATE** **TABLE** VISITOR\_LOG
2. (
3. CLASS\_2\_ID **VARCHAR**(200),
4. VISITOR\_ID **VARCHAR**(200) NOT NULL,
5. VISITOR\_NAME **VARCHAR**(200),
6. V\_ADDRESS **VARCHAR**(200),
7. V\_CONTACT\_INFO **VARCHAR**(200),
8. **primary** **key**(VISITOR\_ID),
9. **foreign** **key** (CLASS\_2\_ID) **references** CLASS\_2\_PATIENT(CLASS\_2\_ID)
10. );

### **4.1.2 A Database State**

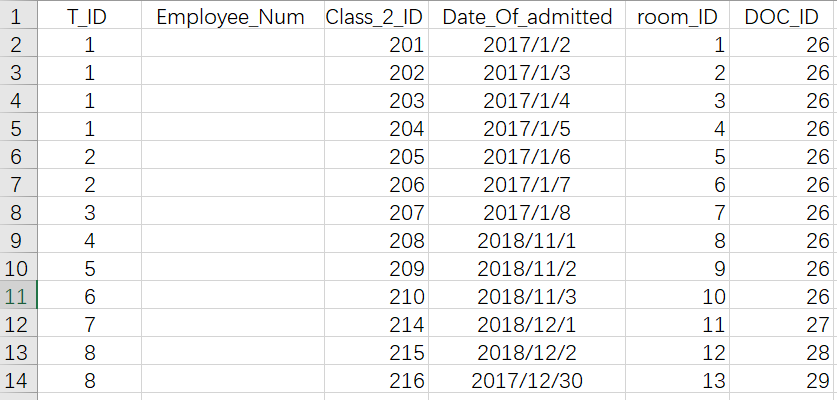
* ACCSEE1



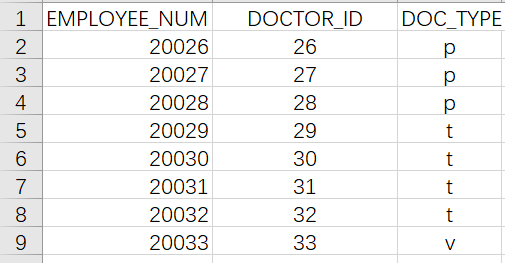
* CLASS\_1\_PATIENT;



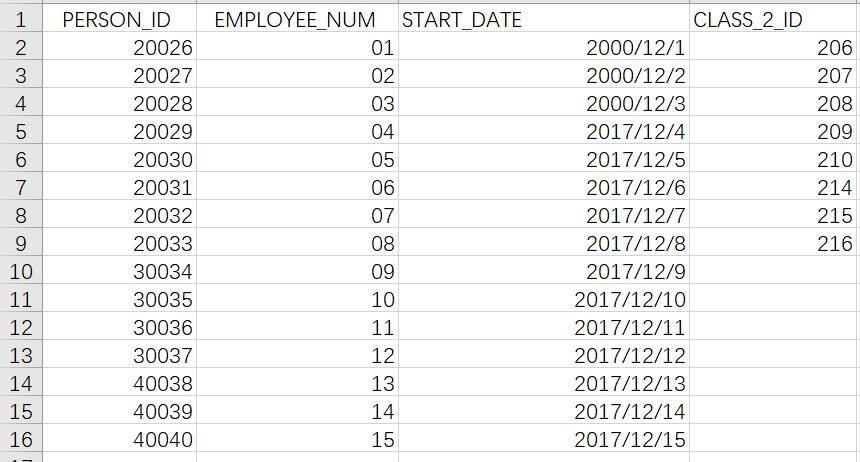
* CLASS\_2\_PATIENT



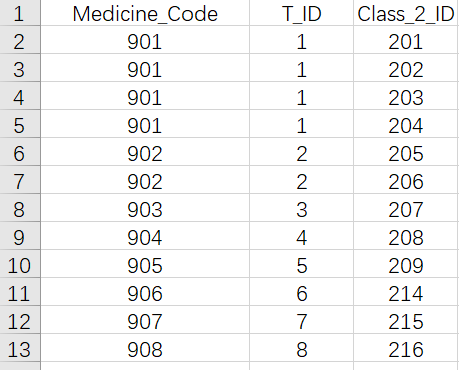
* DOCTOR



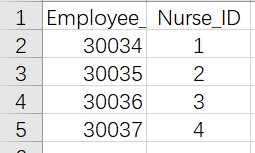
* EMPLOYEE



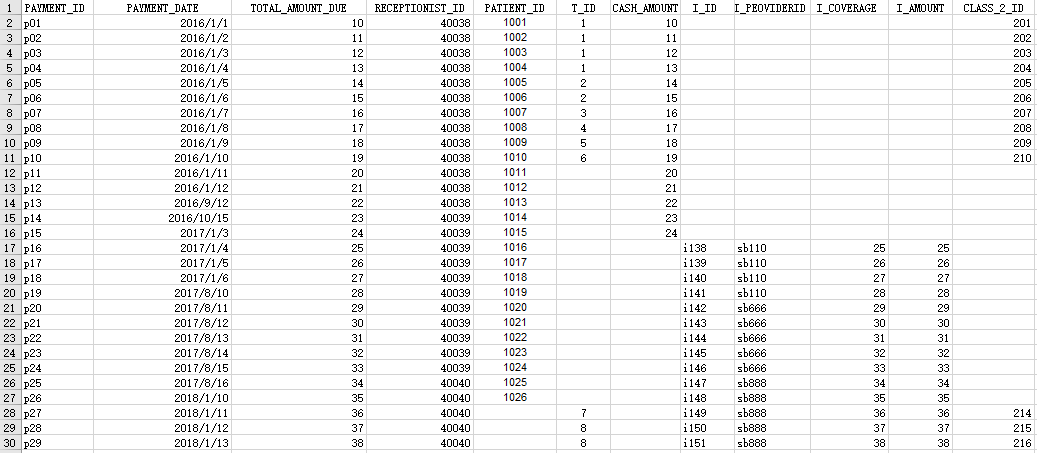
* MEDICAL\_INFORMATION



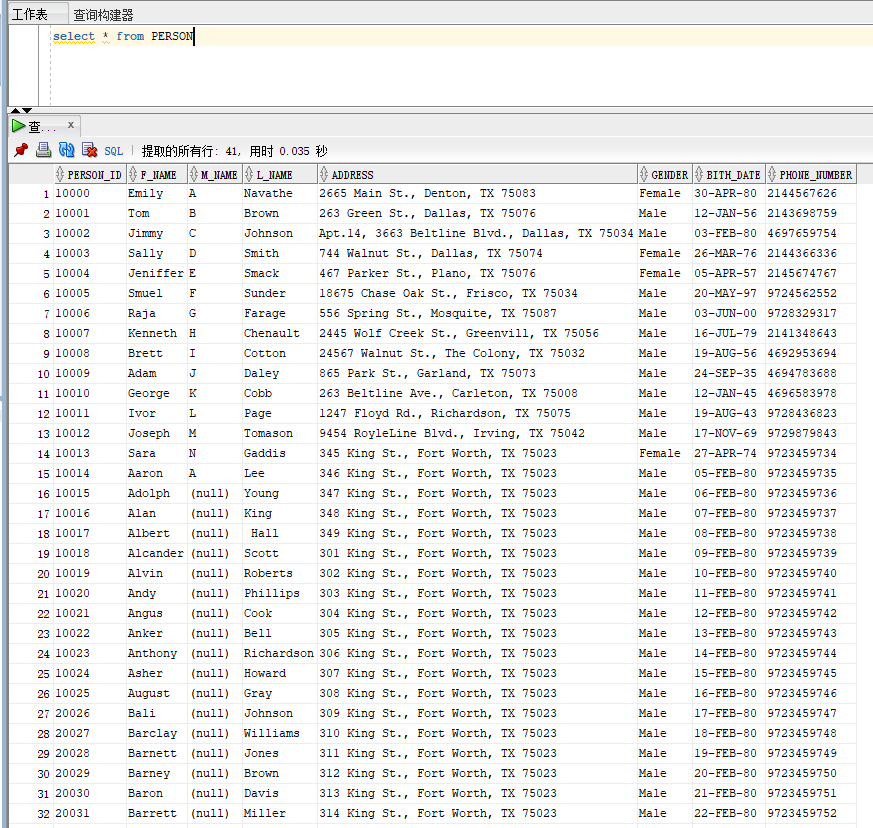
* NURSE



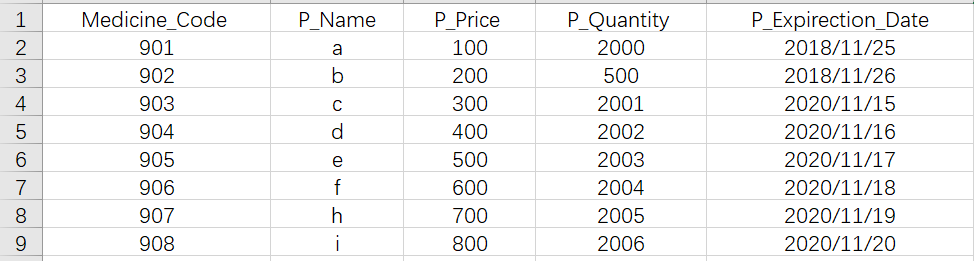
* PYMENT



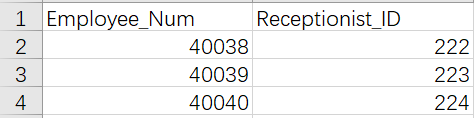
* PERSON



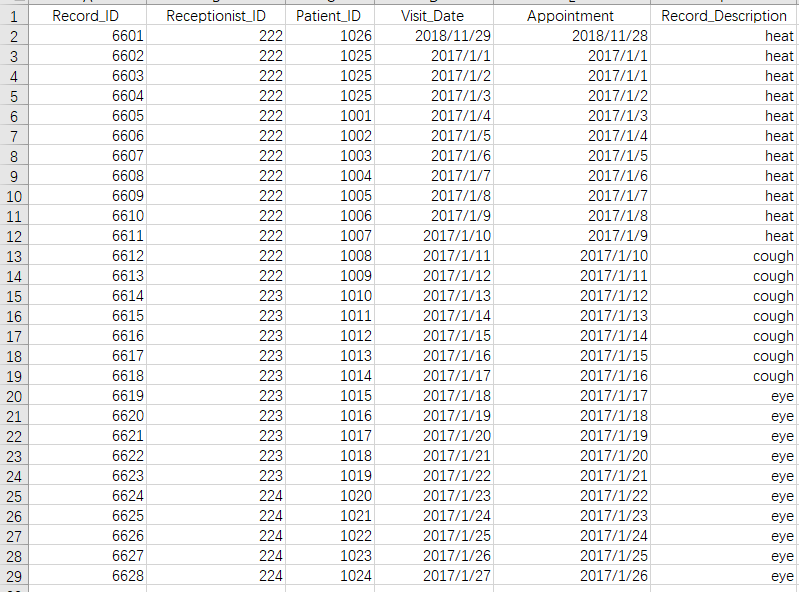
* PHARMACY



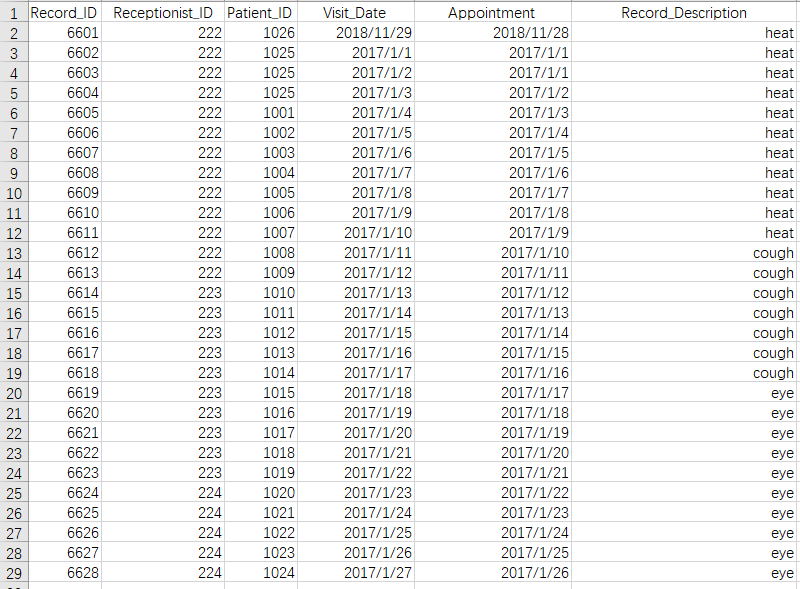
* RECEPTIONIST



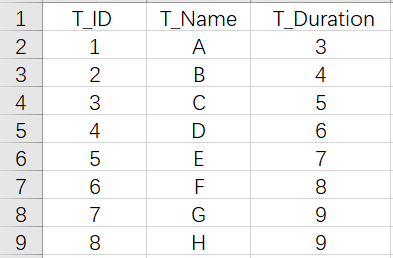
* RECORDS



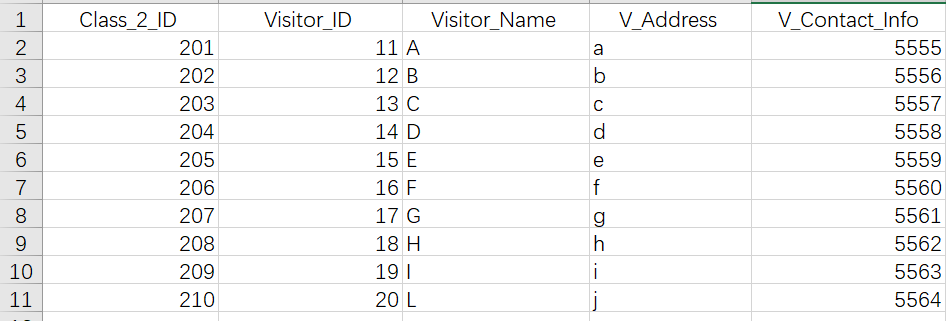
* ROOM



* TREATMENT



* VISITOR\_LOG



## 4.2 Creation of Views (Answer for Question d/Phase III)

Use the Create View statement to create the following views:

1. TopDoctor- This view returns the First Name, Last Name and Date of Joining of those doctors who have made more than 5 Class 1 patients and over 10 Class 2 patients.

1. CREATE VIEW TopDoctor as
2. SELECT P.FName,P.LName,E.Start\_Date ,Doc\_ID
3. FROM  Person as P,Employee E, Doctor D
4. WHERE E.Person\_ID =P.Person\_ID and D.Employee\_Num = E.Employee\_Num and E.Employee\_Num in
5. (
6. (SELECT count(\*)
7. FROM class \_1\_patient as c1p,
8. Group by c1p.Employee\_Num
9. HAVING count(\*) > 5)
10. and
11. (SELECT count(\*)
12. FROM Class\_2\_Patient as c2p
13. Group by c2p.Employee\_Num
14. HAVING count(\*)>10)
15. );

2 TopTreatment- This view returns the treatment name of the most common treatment in Dallas Care along with the bill payment amount when a person receives that treatment.

1. CREATE VIEW TopTreatment as
2. SELECT T.name, T.T\_ID
3. FROM Treatment T,Payment P,
4. WHERE P.T\_ID=T.T\_ID and T.T\_ID exist
5. (SELECT P.T\_ID ,max(count(\*))
6. FROM Payment P
7. Group by P.T\_ID
8. )

3 ReorderMeds- This view returns the medicines that need to be reordered. A medicine needs to be reordered if the expiration date is 1 month FROM current date or quantity is less than 1000.

1. CREATE VIEW ReorderMeds as
2. SELECT  Medicine\_Code
3. FROM  Pharmacy
4. WHERE ((to\_char(sysdate,'mm') - (to\_char(P\_Expirection\_Date,'mm')))<=1 or (P\_Quantity<1000)

4. PotentialPatient- This view returns the name, phone number and ID of patients who visited the hospital more than 3 times as a Class 1 patient but has not been admitted yet.

1. **CREATE** **VIEW** PotentialPatient **as**
2. **SELECT** F\_name, L\_name, Phone\_Number,Patient\_ID,Person\_ID
3. **FROM** Record r, Person p,Class\_1\_Patient c1p
4. **WHERE** r.Patient\_ID =c1p.Patient\_ID and c1p.person\_ID =p.Person\_ID and c1p.class\_2\_ID = Null and p.Patient\_ID in
5. (
6. **SELECT**  r.patient\_ID,count(\*)
7. Form    Record
8. **Group** **by** r.patient\_ID
9. **HAVING** count(\*) >3
10. )

5. MostFrequentIssues - This view returns the maximum frequency of the reason that patients visit the hospital for and the associated treatment for the same. For example, if patients visit the hospital mostly complaining about heart issues then what are the treatment associated with heart issues.

1. **CREATE** **VIEW** PotentialPatient **as**
2. View5:
3. **CREATE** **VIEW** FrequentIssues **AS**
5. **SELECT** T.T\_Name ,count(\*)
6. **FROM** Treatment T
7. **Group** **by** **distinct** T.T\_name
8. **Order** **by** count(\*) **desc**
9. limit 1

## 4.3 Creation of SQL Queries (Answer for Question e/Phase III)

Now we give out the SQL Queries for all questions listed in **Question e** as follows:

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

1. **SELECT** E.start\_date,D.type
2. **FROM** Doctor D,Employee E
3. **WHERE** D.Employee\_Num = E.Employee\_Num

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

1. **SELECT** P.F\_name ,P.L\_name
2. **FROM** Person p,Employee e
3. **WHERE** p.person\_ID = e.person\_ID and e.start\_date =(sysdate -e.start\_date)<90)

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

1. **SELECT** F\_name,L\_name,((to\_char(sysdate,'yyyy'))to\_char(E.birth\_date,'yyyy'))) **as** age,Doc\_type
2. **FROM** TopDoctor **as** T, Person P, Employee E, Doctor D
3. **WHERE** T.Doc\_ID=D.Doc\_ID

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

1. **SELECT** ph.**name**
2. **FROM** Medical\_infomation me,Pharmacy **as** ph,TopTreatment **as** **top**
3. **WHERE** me.medicine\_code = ph.medicine\_code and **top**.T\_ID = me.T\_ID

5. Find all the doctors who have not had a patient in the last 5 months. (Hint: Consider the date of payment as the day the doctor has attended a patient/been consulted by a patient.)

1. **SELECT** F\_name ,L\_name
2. **FROM** Person p ,Employee e ,Doctor d
3. **WHERE** p.Person\_ID =e.Person\_ID and d.Employee\_Num=e.Employee.Num and (D.doc\_ID ,D.Employee\_Num) in
4. (
5. **SELECT** d.Doc\_ID ,d.Employee\_Num,
6. **FROM** class\_2\_patient c2p, class\_1patient c1p ,payment p
7. **WHERE** c1p.class\_2\_ID=c2p.class\_2\_ID and c2p.doc\_ID=D.doc\_ID and     ((to\_char(sysdate,'mm') - (to\_char(Date\_of\_admitted,'mm')))>5 and    ((to\_char(sysdate,'mm')     - (to\_char(payment\_date,'mm')))>5
8. ）

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

1. **SELECT** I\_Provider\_name ,count(\*)
2. **FROM** Class\_1\_Patient c1p, Class\_2\_Patient c2p,Payment p
3. **WHERE** c1p.Class\_2\_ID= c2p.Class\_2\_ID and p.Patient\_ID = c1p.Patient\_ID and p.I\_ID <> null
4. **Group** **by** p.I\_Provider\_name

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

1. **SELECT** r.Room\_ID ,r.duration
2. **FROM** Room r
3. **WHERE** duration = **max**(r.duration)

8. Find the year with the maximum number of patient visiting the hospital and the reason for their visit.

1. **SELECT** r.Record\_Description, to\_char(Visit\_Date,'yyyy') **as** year
2. **FROM** Records r
3. **WHERE** year in
4. (
5. **SELECT** year ,r.Patient\_ID, **max**(count(\*))
6. **FROM** Records r
7. **Group** **by** r.Patient\_ID
8. )

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

1. **SELECT** **Name**
2. **FROM** TREATEMENT
3. **WHERE** duration=
4. (
5. **SELECT** **Min**(Duration)
6. **FROM** TREATEMENT
7. )

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

1. **SELECT** COUNT(C.Class\_2\_ID) **as** COUNTNUMBER
2. **FROM** Class\_2\_Patient C
3. **WHERE** C.Date\_Of\_Admitted>(
4. **SELECT** **MAX**(Start\_date)
5. **FROM** DOCTOR,EMPLOYEE
6. **WHERE** DOCTOR.Employee\_Num=EMPLOYEE.Employee\_Num
7. )

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

1. **SELECT** **DISTINCT** t3.\*
2. **FROM** Class\_1\_Patient t1 LEFT JOIN Class\_2\_Patient t2 **on** t2.Class\_2\_ID=t1.Class\_2\_ID
3. LEFT JOIN RECORDS t3 **on** t1.Patient\_ID=t3.Patient\_ID
4. **WHERE** (to\_char(t2.Date\_Of\_admitted,'yyyymmdd') - (to\_char(t3.Visit\_Date,'yyyymmdd'))<7

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

1. **SELECT** sum(Total\_Amount\_Due)
2. **FROM** Payment
3. **WHERE** Payment\_Date = to\_char(Payment\_Date,'2017')
4. **Group** **by** (Payment\_ID)

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.

1. **SELECT** F\_name,L\_name
2. **FROM** Person p,Doctor d, Class\_1\_Patient **as** c1p, Employee e
3. **WHERE** p.person\_id =e.person\_id and e.Employee\_Num = d.Employee\_Num and c1p.Doc\_ID =d.Doc\_ID and c1p.Class\_2\_ID = null and (c1p.Patient\_ID) exist
4. (
5. **SELECT** r.Patient\_ID,count(Visit\_Date)
6. **FROM** Records r
7. **WHERE** count(Visit\_Date)=1
8. **Group** **by** r.patient\_id
9. ）

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

1. **SELECT**   F\_name, L\_name,,((to\_char(sysdate,'yyyy'))-to\_char(E.birth\_date,'yyyy'))) **as** age
2. **FROM**   PotentialPatient po,Person p
3. **WHERE**  po.Person\_ID= p.Person\_ID

# Dependency Diagram

We now draw a dependency diagram for each table from diagram above as follows:

## Hospital Personnel

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema Hospital Personnel, Person\_ID. Therefore, every other attribute of this relational schema is functionally dependent on Person\_ID. The dependency diagram is shown as Figure 1.



Figure 1. Dependency Diagram of Hospital Personnel

## Class\_1\_Patient

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema Class\_1\_Patient, Patient\_ID. Therefore, every other attribute of this relational schema is functionally dependent on Patient\_ID. The dependency diagram is shown as Figure 2.



Figure 2. Dependency Diagram of Class\_1\_Patient

## Hospital Employee

There is only one attribute in the left-hand side of the functional dependencies, which is the key of relational schema Hospital Employee,Employee\_num. Therefore, every other attribute of this relational schema is functionally dependent on Employee\_num. The dependency diagram is shown as Figure 3.



Figure 3. Dependency Diagram of Employee

## Hospital Doctor

There is only one attribute in the left-hand side of the functional dependencies, which is DOC\_ID. The dependency diagram is shown as Figure 4.



Figure 2. Dependency Diagram of Doctor

## Relation Access

There are two attribute in the left-hand side of the functional dependencies, which are Medicine code and Treatment ID. The dependency diagram is shown as Figure 5.



Figure 5. Dependency Diagram of Access

## Hospital Nurse

The dependency diagram is shown as Figure 6.



Figure 6. Dependency Diagram of Nurse

## Hospital Room

The dependency diagram is shown as Figure 7.



Figure 7. Dependency Diagram of Room

## Hospital Receptionist

The dependency diagram is shown as Figure 8.



Figure 8. Dependency Diagram Receptionist

## Patient Records

The dependency diagram is shown as Figure 9.



Figure 9. Dependency Diagram of Records

## Payment Information

The dependency diagram is shown as Figure 10.



Figure 10. Dependency Diagram of Payment

## Medical Information

The medicine code is depend on both T\_ID and Class\_2\_ID, thus the primary of this relation is T\_ID together with Class\_2\_ID. The dependency diagram is shown as Figure 11.



Figure 11. Dependency Diagram

## Hospital Pharmacy

The dependency diagram is shown as Figure 12.



Figure 12. Dependency Diagram

## Hospital Treatment

The dependency diagram is shown as Figure 13.



Figure 13. Dependency Diagram

## Class 2 Patient

The dependency diagram is shown as Figure 14.



Figure 14. Dependency Diagram

## Visitor Log

The dependency diagram is shown as Figure 15.



Figure 15. Dependency Diagram of Hospital Personnel

## Final Results

After drawing the dependency diagrams one after another, Figure 16 shows the final results for the whole database including the ones who do not have any functional dependencies.



Figure 16. Whole Dependency Diagram for Dallas Care Database

# Conclusion

In this final report we summarized all the necessary descriptions and solutions for Dallas Care database, including process and result of EER diagrams, relational schemas in third normal form, SQL statements to create database, create view and solve corresponding queries, as well as dependency diagram. We also implement the whole database in Oracle and using a database state to test every query. In section 2 we also explained why we use superclass/subclass relationship to build relational schema, why we choose a Relational DBMS to implement our database, and the additional five business rules shown from implementation.