Project for Database Design

Phase IV. Documentation

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.

0. 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

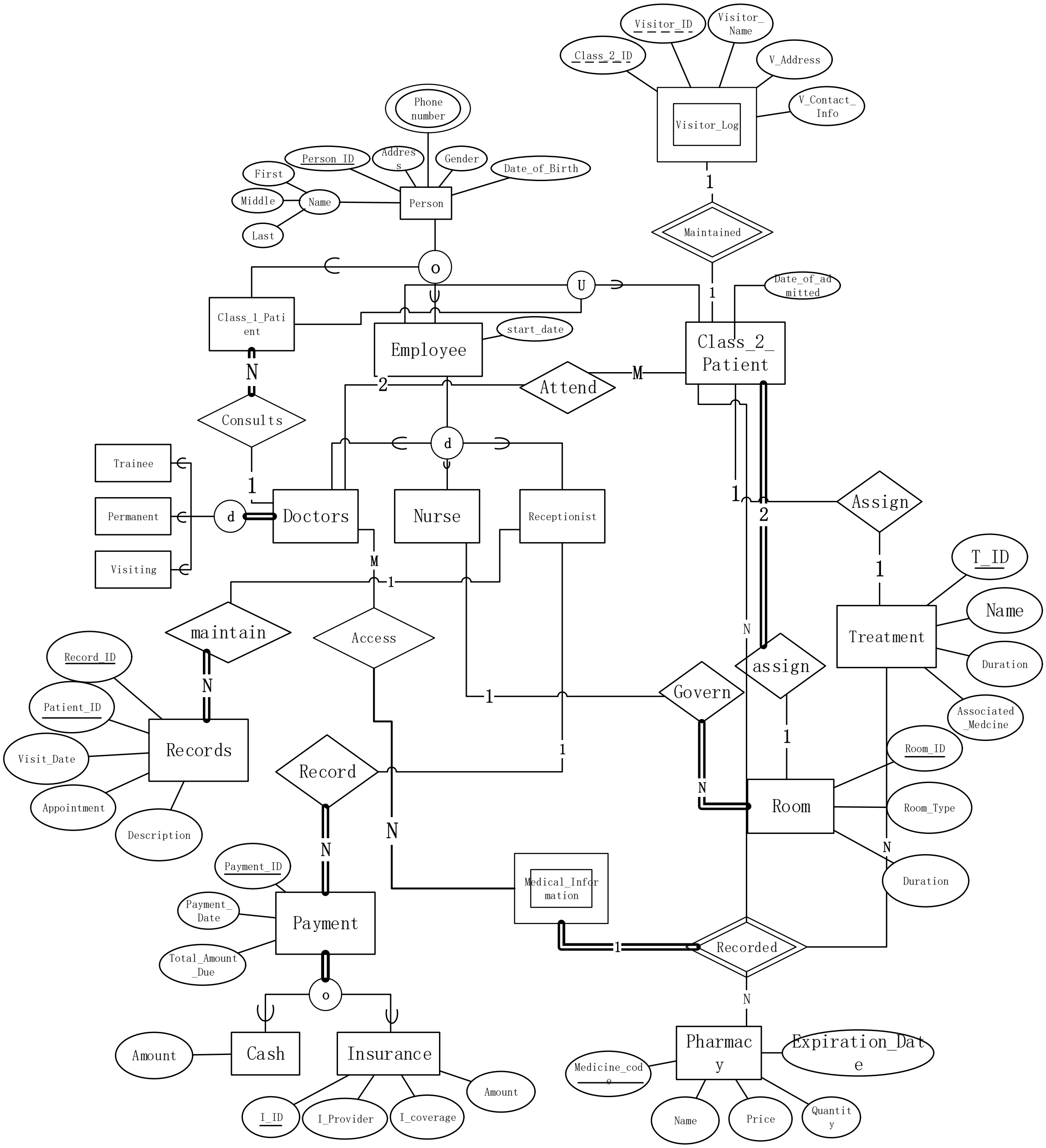
When data is better managed and access is improved, quality information is generated and the user is enabled to make faster decisions. A good database managing system helps in providing a framework to facilitate data quality initiatives and in turn, higher quality information helps in making better,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

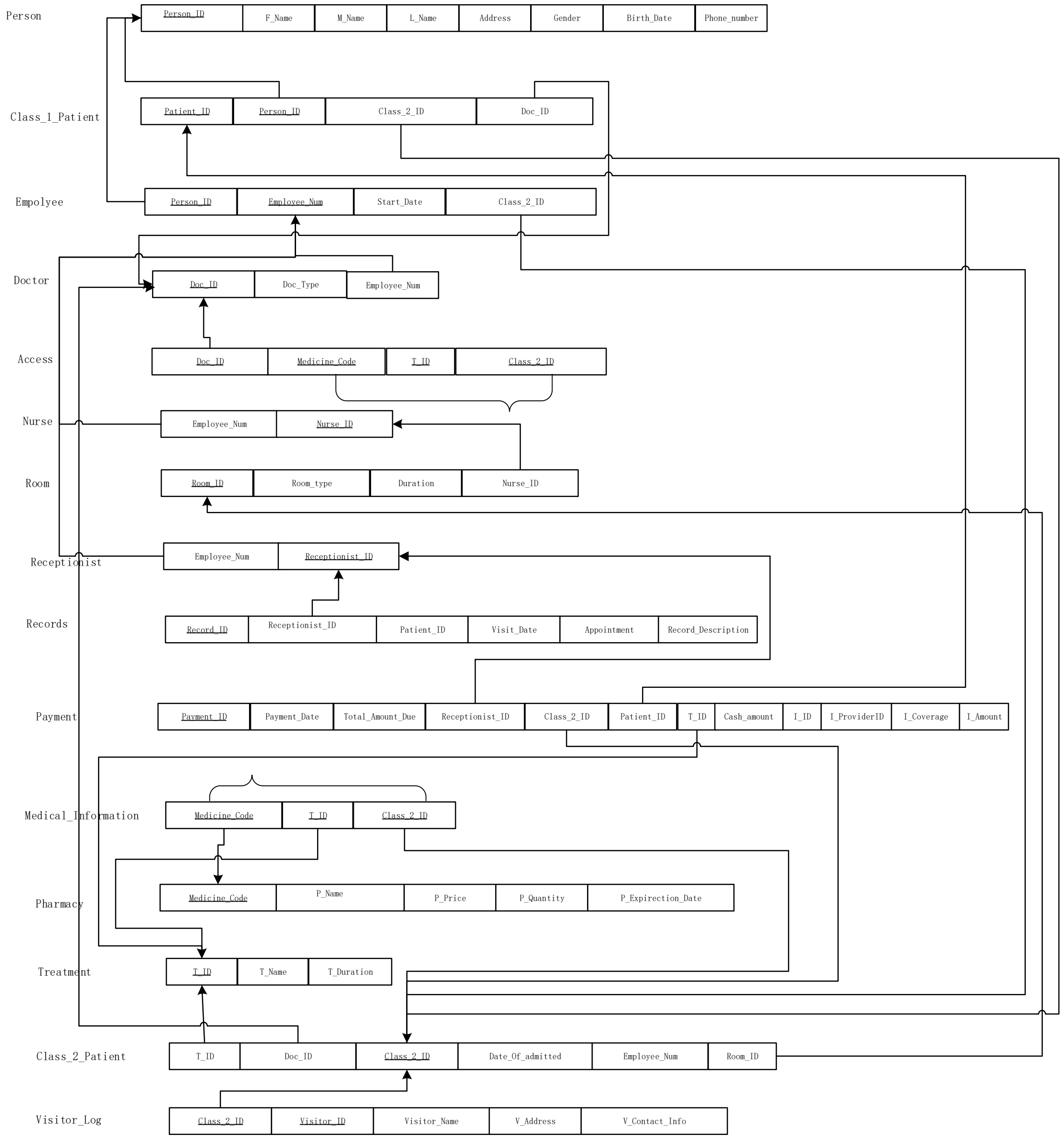
For the hospital, it meets all the requirements and the cost is not hinder the way compared with the business benefits. On the other hand, the hospital indeed requires the different access for different users and the whole system is not sample enough with just basic records.

2. EER diagram with all assumptions



3. Relational Schema in Third Normal Form

3.1 Relational Schema



3.2 Format for Every Relation

Person

Person_ID	F_Name	M_Name	L_Name	Address	Gender	Birth_Date	Phone

Class_1_Patient

Patient_ID	Person_ID	Class_2_ID	Employee_Num

Employee	Employee_Num	Person_ID	Start_Date	Class_2_ID							
Doctor	DOC_ID	Employee_Num	DOC_TYPE								
Access	Medicine_Code	T_ID	Employee_Num	Class_2_ID							
Nurse	Nurse_ID	Employee_Num									
Room	Room_ID	Room_type	Duration	Employee_Num							
Receptionist	Receptionist_ID	Employee_Num									
Records	Record_ID	Employee_Num	Patient_ID	Visit_Date	Appointment	Description					
Payment	Payment_ID	Patient_ID	Payment_Date	Total_Amount_Due	Employee_Num	T_ID	Cash_Amount	I_ID	I_ProviderID	I_Coverage	I_Amount
Medical_information	T_ID	Class_2_ID	Medicine_Code								
Pharmacy	Medicine_Code	Name	Price	Quantity	Expirection_Date						
Treatment	T_ID	Name	Duration								
Class_2_Patient	Class_2_ID	Employee_Num	Date_Of_admitted	T_ID	Room_ID						
Visitor_Log	Visitor_ID	Class_2_ID	Visitor_Name	V_Address	V_Contact						

4. 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

```sql
CREATE TABLE ACCESS1
(
  DOC_ID VARCHAR(200) NOT NULL,
  MEDICINE_CODE VARCHAR(200) NOT NULL,
  T_ID VARCHAR(200) NOT NULL,
  CLASS_2_ID VARCHAR(200) NOT NULL,
  primary key (DOC_ID, MEDICINE_CODE, T_ID, CLASS_2_PATIENT_ID),
  foreign key (MEDICINE_CODE, T_ID, CLASS_2_ID) references MEDICAL_INFORMATION(MEDICINE_CODE, T_ID, CLASS_2_ID),
  foreign key (DOC_ID) references DOCTOR(DOC_ID)
);
```

- CLASS_1_PATIENT

```sql
CREATE TABLE CLASS_1_PATIENT
(
  PATIENT_ID VARCHAR(255) NOT NULL,
  PERSON_ID VARCHAR(255) NOT NULL,
  CLASS_2_ID VARCHAR(255),
  EMPLOYEE_NUM VARCHAR(255),
  primary key (PATIENT_ID),
  foreign key (PERSON_ID) references PERSON(PERSON_ID)
);
alter table CLASS_1_PATIENT
add DOC_ID varchar(255) null;
alter table CLASS_1_PATIENT
add foreign key (DOC_ID) references DOCTOR(DOC_ID);
```

- CLASS_2_PATIENT

```sql
CREATE TABLE CLASS_2_PATIENT
(
  T_ID VARCHAR(200),
  EMPLOYEE_NUM VARCHAR(200),
  CLASS_2_ID VARCHAR(200) NOT NULL,
  DATE_OF_ADMITTED DATE,
  ROOM_ID VARCHAR(255),
  primary key(CLASS_2_ID),
  foreign key(T_ID) references TREATMENT(T_ID),
  foreign key(ROOM_ID) references ROOM(ROOM_ID)
);
alter table CLASS_2_PATIENT
add DOC_ID varchar(255) null;
alter table CLASS_2_PATIENT
add foreign key (DOC_ID) references DOCTOR(DOC_ID);
```

- DOCTOR

```sql
CREATE TABLE DOCTOR
(
  EMPLOYEE_NUM VARCHAR(255) NOT NULL,
  DOC_ID VARCHAR(255),
  DOC_TYPE VARCHAR(255),
  primary key(DOC_ID),
  foreign key (EMPLOYEE_NUM) references EMPLOYEE(EMPLOYEE_NUM)
);
```

- EMPLOYEE

```
CREATE TABLE EMPLOYEE
(
  PERSON_ID VARCHAR(255) NOT NULL,
  EMPLOYEE_NUM VARCHAR(255) NOT NULL,
  START_DATE DATE,
  CLASS_2_ID VARCHAR(255),
  primary key (EMPLOYEE_NUM),
  foreign key (PERSON_ID) references PERSON(PERSON_ID)
);
```

- MEDICAL_INFORMATION

```
CREATE TABLE MEDICAL_INFORMATION
(
  MEDICINE_CODE VARCHAR(200) NOT NULL,
  T_ID VARCHAR(200) NOT NULL,
  CLASS_2_ID VARCHAR2(200) NOT NULL,
  primary key (MEDICINE_CODE,T_ID,CLASS_2_ID),
  foreign key(T_ID) references TREATMENT(T_ID),
  foreign key(CLASS_2_ID) references CLASS_2_PATIENT(CLASS_2_ID)
);
```

- NURSE

```
CREATE TABLE NURSE
(
  EMPLOYEE_NUM VARCHAR(255) NOT NULL,
  NURSE_ID VARCHAR(255) NOT NULL,
  primary key(NURSE_ID),
  foreign key (EMPLOYEE_NUM) references EMPLOYEE(EMPLOYEE_NUM)
);
```

- PYMENT

```
CREATE TABLE PAYMENT
(
  PAYMENT_ID VARCHAR2(20) NOT NULL,
  PAYMENT_DATE DATE,
  TOTAL_AMOUNT_DUE VARCHAR(20),
  RECEPTIONIST_ID VARCHAR(20),
  PATIENT_ID VARCHAR(40),
  T_ID VARCHAR(20),

  CASH_AMOUNT VARCHAR(100),
  I_ID VARCHAR(60),
  I_PROVIDERID VARCHAR(100),
  I_COVERAGE VARCHAR(250),
  I_AMOUNT VARCHAR(100),
  primary key (PAYMENT_ID),
  foreign key (RECEPTIONIST_ID) references RECEPTIONIST(RECEPTIONIST_ID),
  foreign key (PATIENT_ID) references CLASS_1_PATIENT(PATIENT_ID)
);
alter table PAYMENT
add CLASS_2_ID varchar(255) null;
alter table PAYMENT
add foreign key (CLASS_2_ID) references CLASS_2_PATIENT(CLASS_2_ID);
alter table PAYMENT
add foreign key (T_ID) references TREATMENT(T_ID);
```

- PERSON

```
CREATE TABLE PERSON
(
  PERSON_ID VARCHAR(255) NOT NULL,
  F_NAME VARCHAR(255) NOT NULL,
  M_NAME VARCHAR(255),
  L_NAME VARCHAR(255) NOT NULL,
  ADDRESS VARCHAR(255),
  GENDER VARCHAR(255),
  BITH_DATE DATE,
  PHONE_NUMBER VARCHAR(255),
  primary key (PERSON_ID)
);
```

- PHARMACY

```
CREATE TABLE PHARMACY
(
  MEDICINE_CODE VARCHAR(200) NOT NULL,
  P_PRICE VARCHAR(200),
  P_NAME VARCHAR(200),
  P_QUANTITY VARCHAR(200),
  P_EXPIRECTION_DATE DATE,
  primary key(MEDICINE_CODE)
);
```

- RECEPTIONIST

```
CREATE TABLE RECEPTIONIST
(
  RECEPTIONIST_ID VARCHAR(255) NOT NULL,
  EMPLOYEE_NUM VARCHAR(255) NOT NULL,
  primary key (RECEPTIONIST_ID),
  foreign key (EMPLOYEE_NUM) references EMPLOYEE(EMPLOYEE_NUM)
);
```

- RECORDS

```
CREATE TABLE RECORDS
(
  RECORD_ID VARCHAR(255) NOT NULL,
  RECEPTIONIST_ID VARCHAR(255),
  PATIENT_ID VARCHAR(255),
  VISIT_DATE DATE,
  APPOINTMENT DATE,
  RECORD_DESCRIPTION VARCHAR(255),
  primary key(RECORD_ID),
  foreign key (RECEPTIONIST_ID) references RECEPTIONIST(RECEPTIONIST_ID),
  foreign key (PATIENT_ID) references CLASS_1_PATIENT(PATIENT_ID)
);
```

- ROOM

```
CREATE TABLE ROOM
(
    ROOM_ID VARCHAR(255) NOT NULL,
    ROOM_TYPE VARCHAR(255),
    ROOM_DURATION VARCHAR(255),
    NURSE_ID VARCHAR(255) NOT NULL,
    primary key(ROOM_ID),
    foreign key (NURSE_ID) references NURSE(NURSE_ID)
);
```

- TREATMENT

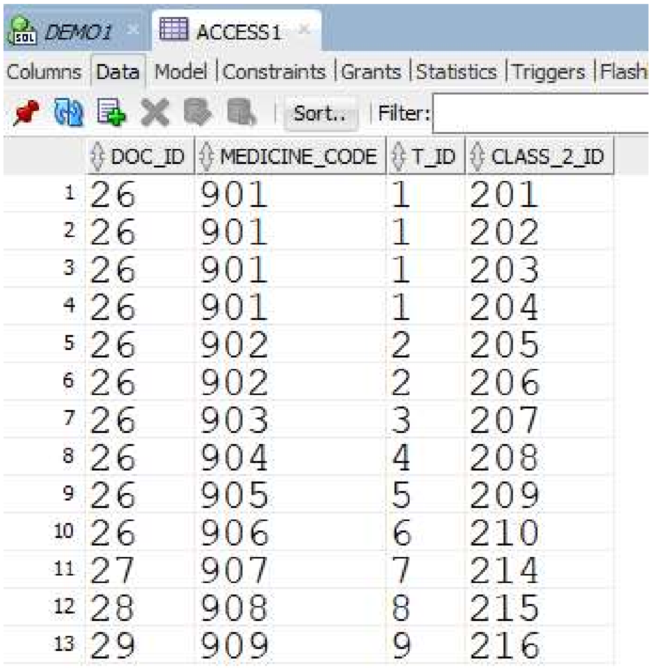
```
CREATE TABLE TREATMENT
(
    T_ID VARCHAR(200) NOT NULL,
    T_NAME VARCHAR(250),
    T_DURATION VARCHAR(200),
    primary key(T_ID)
);
```

- VISITOR_LOG

```
CREATE TABLE VISITOR_LOG
(
    CLASS_2_ID VARCHAR(200),
    VISITOR_ID VARCHAR(200) NOT NULL,
    VISITOR_NAME VARCHAR(200),
    V_ADDRESS VARCHAR(200),
    V_CONTACT_INFO VARCHAR(200),
    primary key(VISITOR_ID),
    foreign key (CLASS_2_ID) references CLASS_2_PATIENT(CLASS_2_ID)
);
```

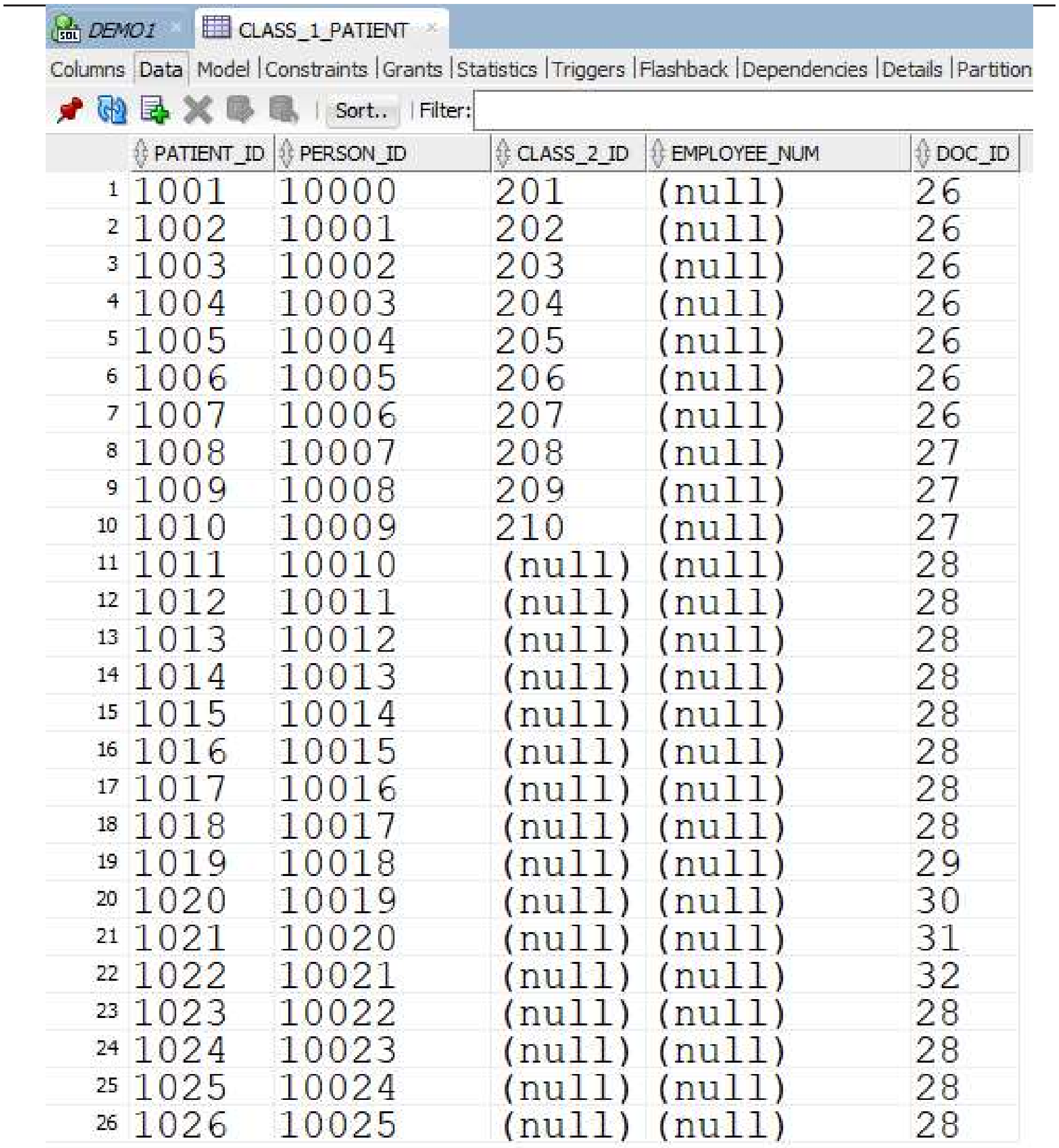
4.1.2 A Database State

- ACCSEE1


DEMO1
ACCESS1
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flash
Sort..
Filter:

	DOC_ID	MEDICINE_CODE	T_ID	CLASS_2_ID
1	26	901	1	201
2	26	901	1	202
3	26	901	1	203
4	26	901	1	204
5	26	902	2	205
6	26	902	2	206
7	26	903	3	207
8	26	904	4	208
9	26	905	5	209
10	26	906	6	210
11	27	907	7	214
12	28	908	8	215
13	29	909	9	216

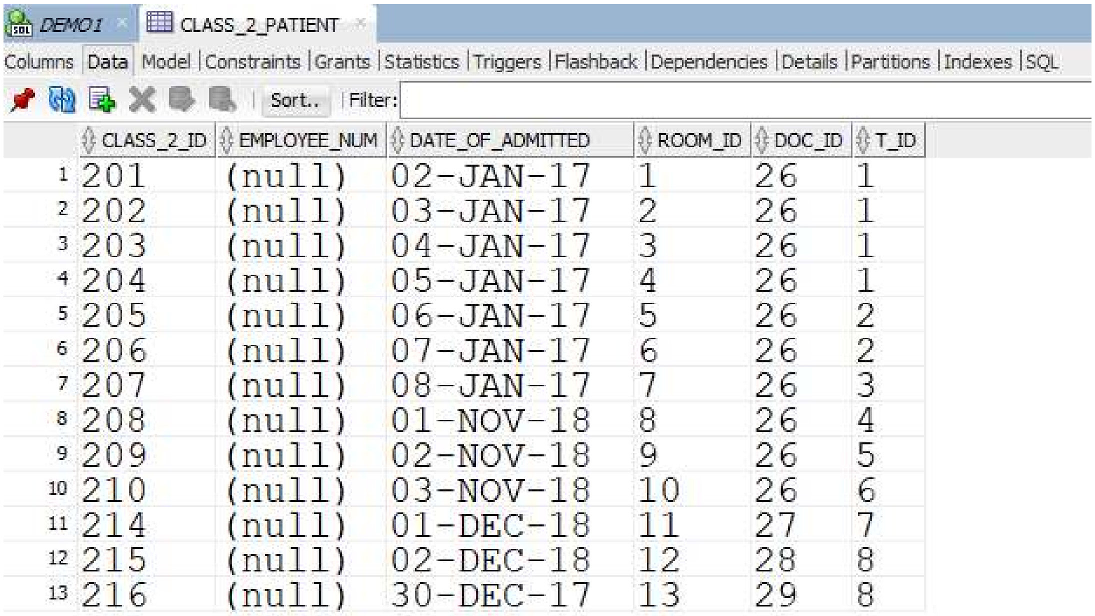
- CLASS_1_PATIENT；



DEMO1
CLASS_1_PATIENT
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partition
Sort..
Filter:

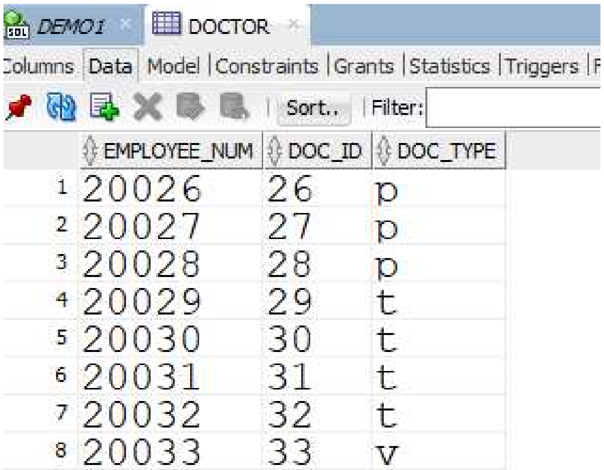
	PATIENT_ID	PERSON_ID	CLASS_2_ID	EMPLOYEE_NUM	DOC_ID
1	1001	10000	201	(null)	26
2	1002	10001	202	(null)	26
3	1003	10002	203	(null)	26
4	1004	10003	204	(null)	26
5	1005	10004	205	(null)	26
6	1006	10005	206	(null)	26
7	1007	10006	207	(null)	26
8	1008	10007	208	(null)	27
9	1009	10008	209	(null)	27
10	1010	10009	210	(null)	27
11	1011	10010	(null)	(null)	28
12	1012	10011	(null)	(null)	28
13	1013	10012	(null)	(null)	28
14	1014	10013	(null)	(null)	28
15	1015	10014	(null)	(null)	28
16	1016	10015	(null)	(null)	28
17	1017	10016	(null)	(null)	28
18	1018	10017	(null)	(null)	28
19	1019	10018	(null)	(null)	29
20	1020	10019	(null)	(null)	30
21	1021	10020	(null)	(null)	31
22	1022	10021	(null)	(null)	32
23	1023	10022	(null)	(null)	28
24	1024	10023	(null)	(null)	28
25	1025	10024	(null)	(null)	28
26	1026	10025	(null)	(null)	28

- CLASS_2_PATIENT


DEMO1
CLASS_2_PATIENT
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

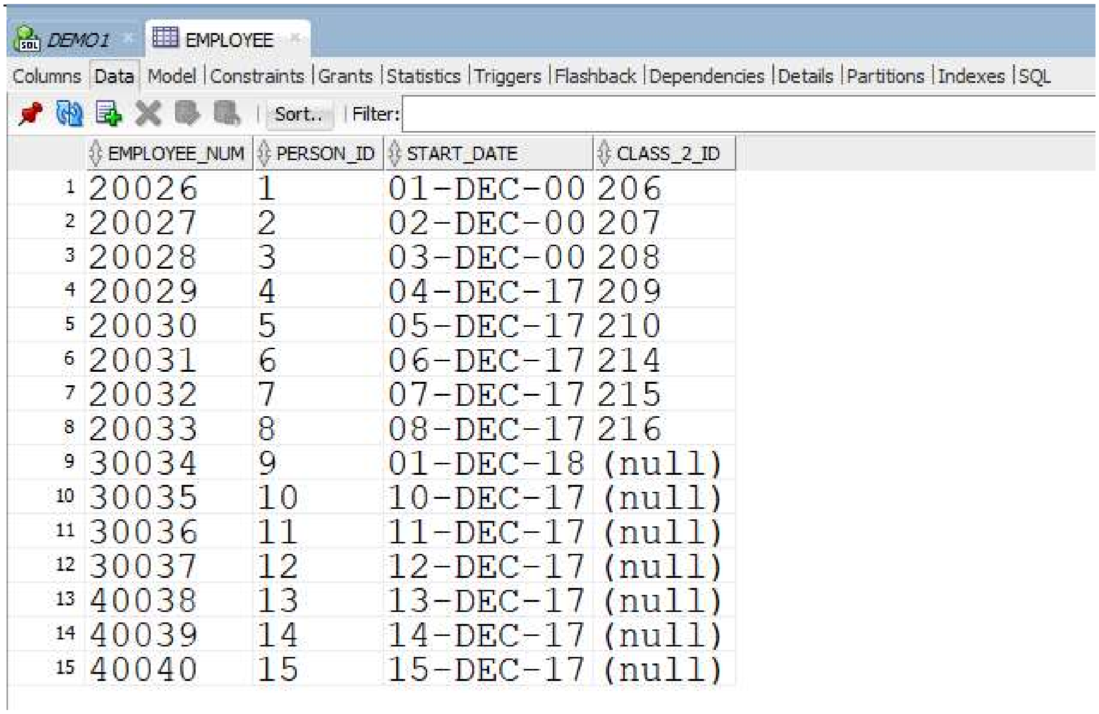
	CLASS_2_ID	EMPLOYEE_NUM	DATE_OF_ADMITTED	ROOM_ID	DOC_ID	T_ID
1	201	(null)	02-JAN-17	1	26	1
2	202	(null)	03-JAN-17	2	26	1
3	203	(null)	04-JAN-17	3	26	1
4	204	(null)	05-JAN-17	4	26	1
5	205	(null)	06-JAN-17	5	26	2
6	206	(null)	07-JAN-17	6	26	2
7	207	(null)	08-JAN-17	7	26	3
8	208	(null)	01-NOV-18	8	26	4
9	209	(null)	02-NOV-18	9	26	5
10	210	(null)	03-NOV-18	10	26	6
11	214	(null)	01-DEC-18	11	27	7
12	215	(null)	02-DEC-18	12	28	8
13	216	(null)	30-DEC-17	13	29	8

- DOCTOR


DEMO1
DOCTOR
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | F
Sort..
Filter:

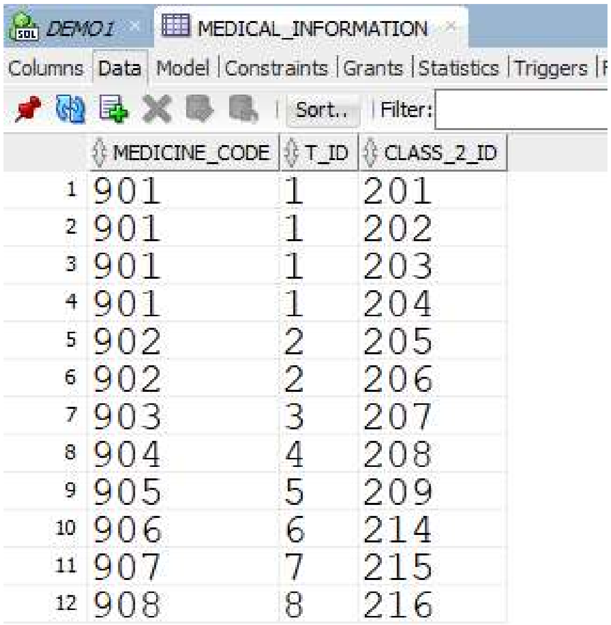
	EMPLOYEE_NUM	DOC_ID	DOC_TYPE
1	20026	26	p
2	20027	27	p
3	20028	28	p
4	20029	29	t
5	20030	30	t
6	20031	31	t
7	20032	32	t
8	20033	33	v

- EMPLOYEE



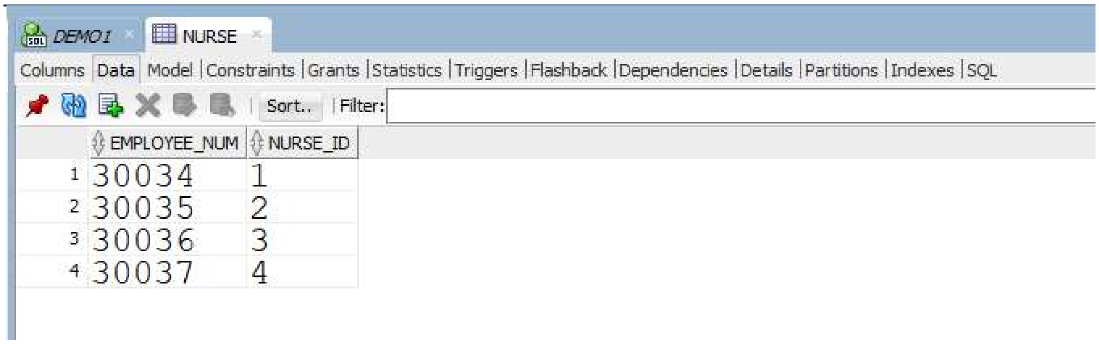
	EMPLOYEE_NUM	PERSON_ID	START_DATE	CLASS_2_ID
1	20026	1	01-DEC-00	206
2	20027	2	02-DEC-00	207
3	20028	3	03-DEC-00	208
4	20029	4	04-DEC-17	209
5	20030	5	05-DEC-17	210
6	20031	6	06-DEC-17	214
7	20032	7	07-DEC-17	215
8	20033	8	08-DEC-17	216
9	30034	9	01-DEC-18	(null)
10	30035	10	10-DEC-17	(null)
11	30036	11	11-DEC-17	(null)
12	30037	12	12-DEC-17	(null)
13	40038	13	13-DEC-17	(null)
14	40039	14	14-DEC-17	(null)
15	40040	15	15-DEC-17	(null)

- MEDICAL_INFORMATION



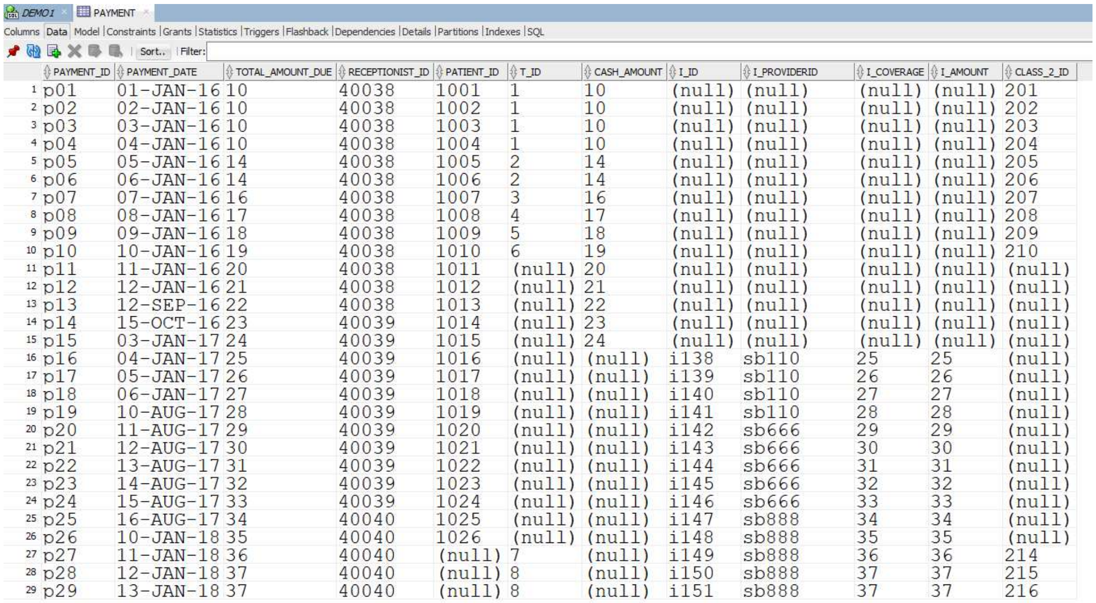
	MEDICINE_CODE	T_ID	CLASS_2_ID
1	901	1	201
2	901	1	202
3	901	1	203
4	901	1	204
5	902	2	205
6	902	2	206
7	903	3	207
8	904	4	208
9	905	5	209
10	906	6	214
11	907	7	215
12	908	8	216

- NURSE


DEMO1
NURSE
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

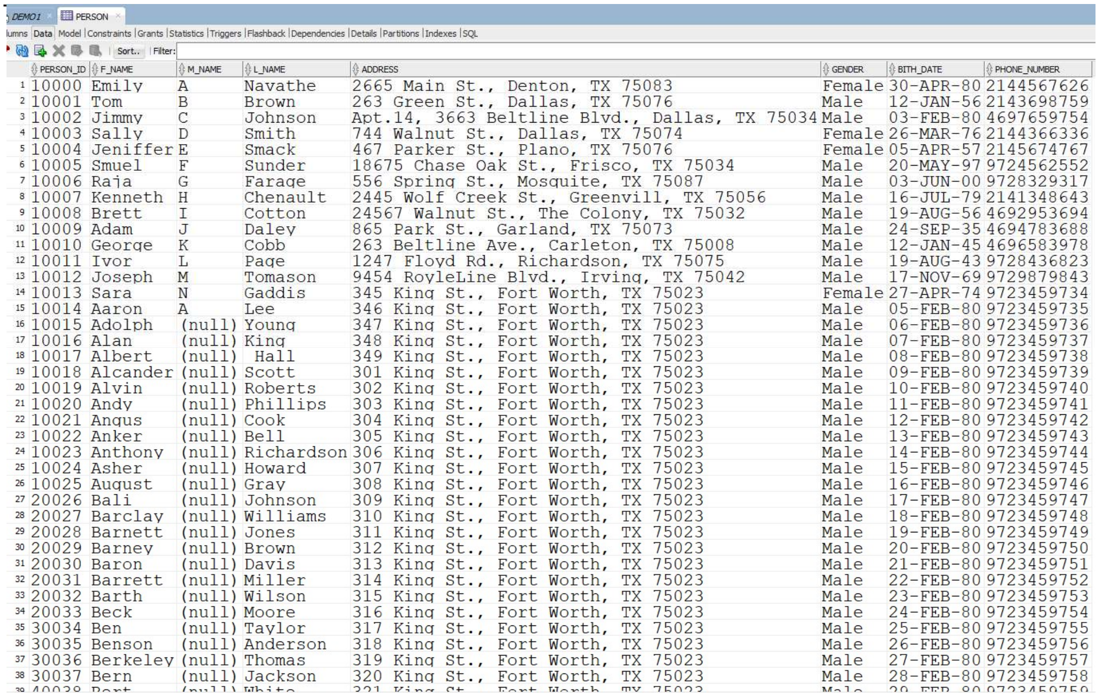
	EMPLOYEE_NUM	NURSE_ID
1	30034	1
2	30035	2
3	30036	3
4	30037	4

- PYMENT


DEMO1
PAYMENT
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

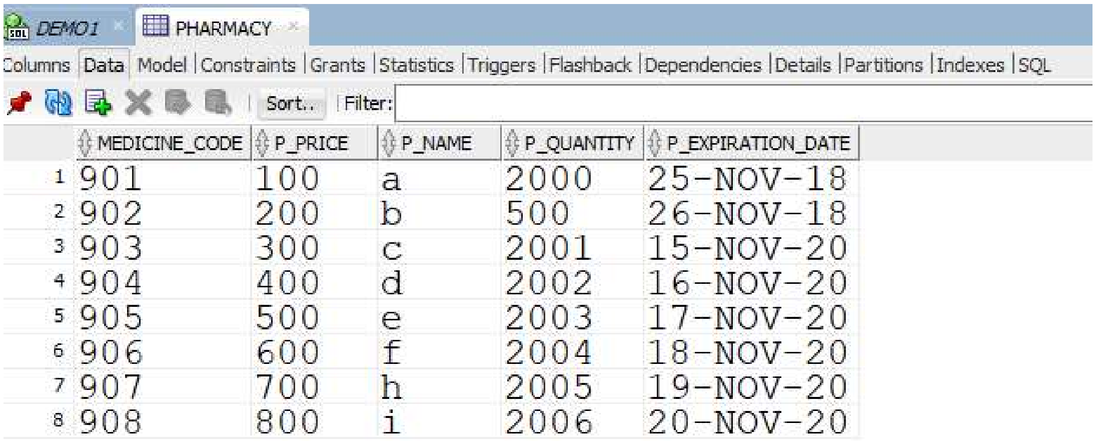
	PAYMENT_ID	PAYMENT_DATE	TOTAL_AMOUNT_DUE	RECEPTIONIST_ID	PATIENT_ID	T_ID	CASH_AMOUNT	I_ID	I_PROVIDERID	I_COVERAGE	I_AMOUNT	CLASS_2_ID
1	p01	01-JAN-16	10	40038	1001	1	10	(null)	(null)	(null)	(null)	201
2	p02	02-JAN-16	10	40038	1002	1	10	(null)	(null)	(null)	(null)	202
3	p03	03-JAN-16	10	40038	1003	1	10	(null)	(null)	(null)	(null)	203
4	p04	04-JAN-16	10	40038	1004	1	10	(null)	(null)	(null)	(null)	204
5	p05	05-JAN-16	14	40038	1005	2	14	(null)	(null)	(null)	(null)	205
6	p06	06-JAN-16	14	40038	1006	2	14	(null)	(null)	(null)	(null)	206
7	p07	07-JAN-16	16	40038	1007	3	16	(null)	(null)	(null)	(null)	207
8	p08	08-JAN-16	17	40038	1008	4	17	(null)	(null)	(null)	(null)	208
9	p09	09-JAN-16	18	40038	1009	5	18	(null)	(null)	(null)	(null)	209
10	p10	10-JAN-16	19	40038	1010	6	19	(null)	(null)	(null)	(null)	210
11	p11	11-JAN-16	20	40038	1011	(null)	20	(null)	(null)	(null)	(null)	(null)
12	p12	12-JAN-16	21	40038	1012	(null)	21	(null)	(null)	(null)	(null)	(null)
13	p13	12-SEP-16	22	40038	1013	(null)	22	(null)	(null)	(null)	(null)	(null)
14	p14	15-OCT-16	23	40039	1014	(null)	23	(null)	(null)	(null)	(null)	(null)
15	p15	03-JAN-17	24	40039	1015	(null)	24	(null)	(null)	(null)	(null)	(null)
16	p16	04-JAN-17	25	40039	1016	(null)	(null)	i138	sb110	25	25	(null)
17	p17	05-JAN-17	26	40039	1017	(null)	(null)	i139	sb110	26	26	(null)
18	p18	06-JAN-17	27	40039	1018	(null)	(null)	i140	sb110	27	27	(null)
19	p19	10-AUG-17	28	40039	1019	(null)	(null)	i141	sb110	28	28	(null)
20	p20	11-AUG-17	29	40039	1020	(null)	(null)	i142	sb666	29	29	(null)
21	p21	12-AUG-17	30	40039	1021	(null)	(null)	i143	sb666	30	30	(null)
22	p22	13-AUG-17	31	40039	1022	(null)	(null)	i144	sb666	31	31	(null)
23	p23	14-AUG-17	32	40039	1023	(null)	(null)	i145	sb666	32	32	(null)
24	p24	15-AUG-17	33	40039	1024	(null)	(null)	i146	sb666	33	33	(null)
25	p25	16-AUG-17	34	40040	1025	(null)	(null)	i147	sb888	34	34	(null)
26	p26	10-JAN-18	35	40040	1026	(null)	(null)	i148	sb888	35	35	(null)
27	p27	11-JAN-18	36	40040	(null)	7	(null)	i149	sb888	36	36	214
28	p28	12-JAN-18	37	40040	(null)	8	(null)	i150	sb888	37	37	215
29	p29	13-JAN-18	37	40040	(null)	8	(null)	i151	sb888	37	37	216

- PERSON


DEMO1
PERSON
lumns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

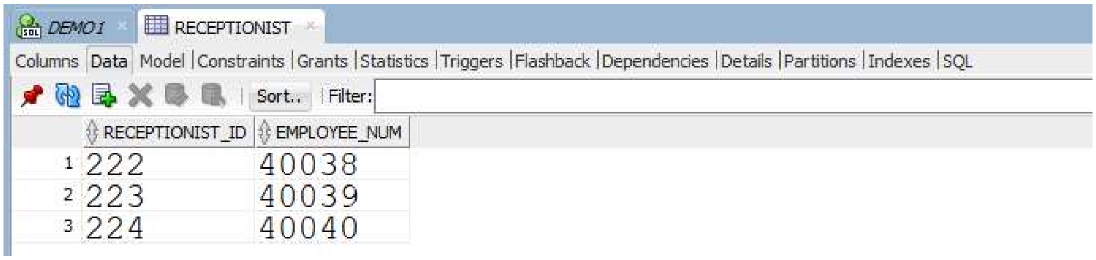
	PERSON_ID	F_NAME	M_NAME	L_NAME	ADDRESS	GENDER	BITH_DATE	PHONE_NUMBER
1	10000	Emily	A	Navathe	2665 Main St., Denton, TX 75083	Female	30-APR-80	2144567626
2	10001	Tom	B	Brown	263 Green St., Dallas, TX 75076	Male	12-JAN-56	2143698759
3	10002	Jimmy	C	Johnson	Apt.14, 3663 Beltline Blvd., Dallas, TX 75034	Male	03-FEB-80	4697659754
4	10003	Sally	D	Smith	744 Walnut St., Dallas, TX 75074	Female	26-MAR-76	2144366336
5	10004	Jeniffer	E	Smack	467 Parker St., Plano, TX 75076	Female	05-APR-57	2145674767
6	10005	Smuel	F	Sunder	18675 Chase Oak St., Frisco, TX 75034	Male	20-MAY-97	9724562552
7	10006	Raja	G	Farage	556 Spring St., Mosquite, TX 75087	Male	03-JUN-00	9728329317
8	10007	Kenneth	H	Chenault	2445 Wolf Creek St., Greenvill, TX 75056	Male	16-JUL-79	2141348643
9	10008	Brett	I	Cotton	24567 Walnut St., The Colony, TX 75032	Male	19-AUG-56	4692953694
10	10009	Adam	J	Daley	865 Park St., Garland, TX 75073	Male	24-SEP-35	4694783688
11	10010	George	K	Cobb	263 Beltline Ave., Carleton, TX 75008	Male	12-JAN-45	4696583978
12	10011	Ivor	L	Page	1247 Floyd Rd., Richardson, TX 75075	Male	19-AUG-43	9728436823
13	10012	Joseph	M	Tomason	9454 RoyleLine Blvd., Irving, TX 75042	Male	17-NOV-69	9729879843
14	10013	Sara	N	Gaddis	345 King St., Fort Worth, TX 75023	Female	27-APR-74	9723459734
15	10014	Aaron	A	Lee	346 King St., Fort Worth, TX 75023	Male	05-FEB-80	9723459735
16	10015	Adolph	(null)	Young	347 King St., Fort Worth, TX 75023	Male	06-FEB-80	9723459736
17	10016	Alan	(null)	King	348 King St., Fort Worth, TX 75023	Male	07-FEB-80	9723459737
18	10017	Albert	(null)	Hall	349 King St., Fort Worth, TX 75023	Male	08-FEB-80	9723459738
19	10018	Alcander	(null)	Scott	301 King St., Fort Worth, TX 75023	Male	09-FEB-80	9723459739
20	10019	Alvin	(null)	Roberts	302 King St., Fort Worth, TX 75023	Male	10-FEB-80	9723459740
21	10020	Andy	(null)	Phillips	303 King St., Fort Worth, TX 75023	Male	11-FEB-80	9723459741
22	10021	Angus	(null)	Cook	304 King St., Fort Worth, TX 75023	Male	12-FEB-80	9723459742
23	10022	Anker	(null)	Bell	305 King St., Fort Worth, TX 75023	Male	13-FEB-80	9723459743
24	10023	Anthony	(null)	Richardson	306 King St., Fort Worth, TX 75023	Male	14-FEB-80	9723459744
25	10024	Asher	(null)	Howard	307 King St., Fort Worth, TX 75023	Male	15-FEB-80	9723459745
26	10025	August	(null)	Gray	308 King St., Fort Worth, TX 75023	Male	16-FEB-80	9723459746
27	20026	Bali	(null)	Johnson	309 King St., Fort Worth, TX 75023	Male	17-FEB-80	9723459747
28	20027	Barclay	(null)	Williams	310 King St., Fort Worth, TX 75023	Male	18-FEB-80	9723459748
29	20028	Barnett	(null)	Jones	311 King St., Fort Worth, TX 75023	Male	19-FEB-80	9723459749
30	20029	Barney	(null)	Brown	312 King St., Fort Worth, TX 75023	Male	20-FEB-80	9723459750
31	20030	Baron	(null)	Davis	313 King St., Fort Worth, TX 75023	Male	21-FEB-80	9723459751
32	20031	Barrett	(null)	Miller	314 King St., Fort Worth, TX 75023	Male	22-FEB-80	9723459752
33	20032	Barth	(null)	Wilson	315 King St., Fort Worth, TX 75023	Male	23-FEB-80	9723459753
34	20033	Beck	(null)	Moore	316 King St., Fort Worth, TX 75023	Male	24-FEB-80	9723459754
35	30034	Ben	(null)	Taylor	317 King St., Fort Worth, TX 75023	Male	25-FEB-80	9723459755
36	30035	Benson	(null)	Anderson	318 King St., Fort Worth, TX 75023	Male	26-FEB-80	9723459756
37	30036	Berkeley	(null)	Thomas	319 King St., Fort Worth, TX 75023	Male	27-FEB-80	9723459757
38	30037	Bern	(null)	Jackson	320 King St., Fort Worth, TX 75023	Male	28-FEB-80	9723459758
39	40038	Bert	(null)	White	321 King St., Fort Worth, TX 75023	Male	29-FEB-80	9723459759

- PHARMACY


DEMO1
PHARMACY
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

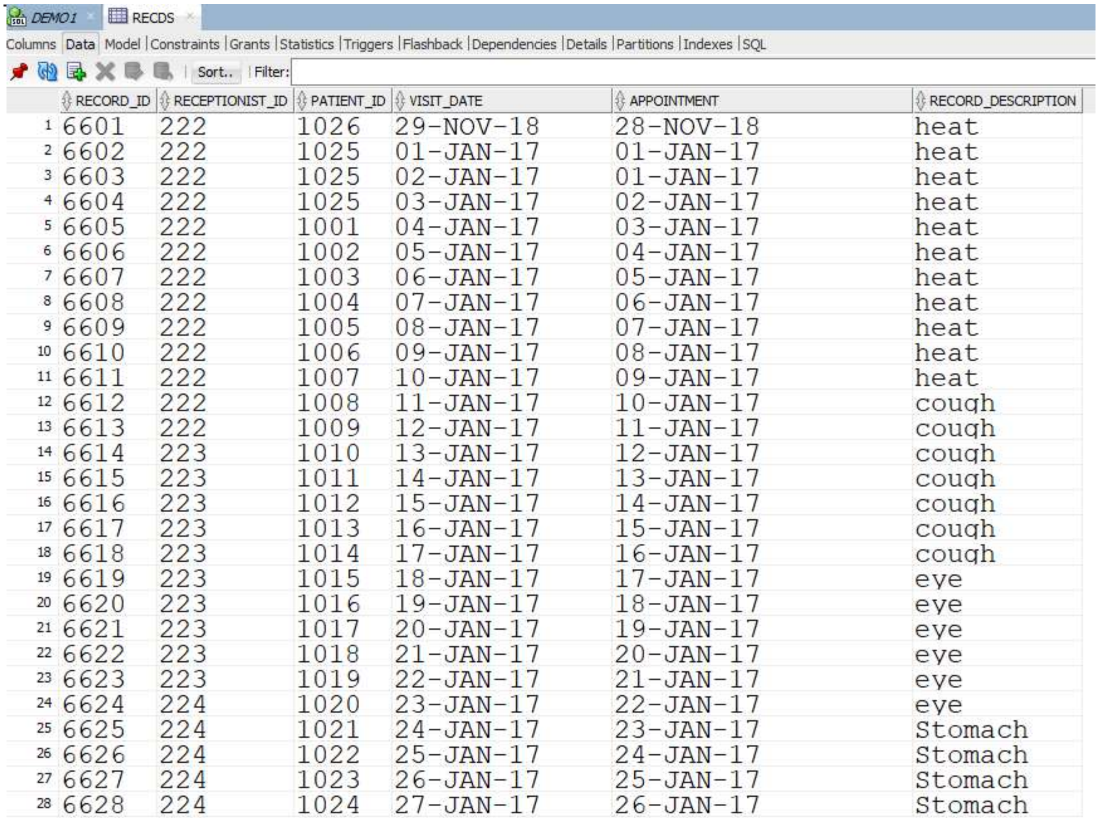
	MEDICINE_CODE	P_PRICE	P_NAME	P_QUANTITY	P_EXPIRATION_DATE
1	901	100	a	2000	25-NOV-18
2	902	200	b	500	26-NOV-18
3	903	300	c	2001	15-NOV-20
4	904	400	d	2002	16-NOV-20
5	905	500	e	2003	17-NOV-20
6	906	600	f	2004	18-NOV-20
7	907	700	h	2005	19-NOV-20
8	908	800	i	2006	20-NOV-20

- RECEPTIONIST


DEMO1
RECEPTIONIST
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

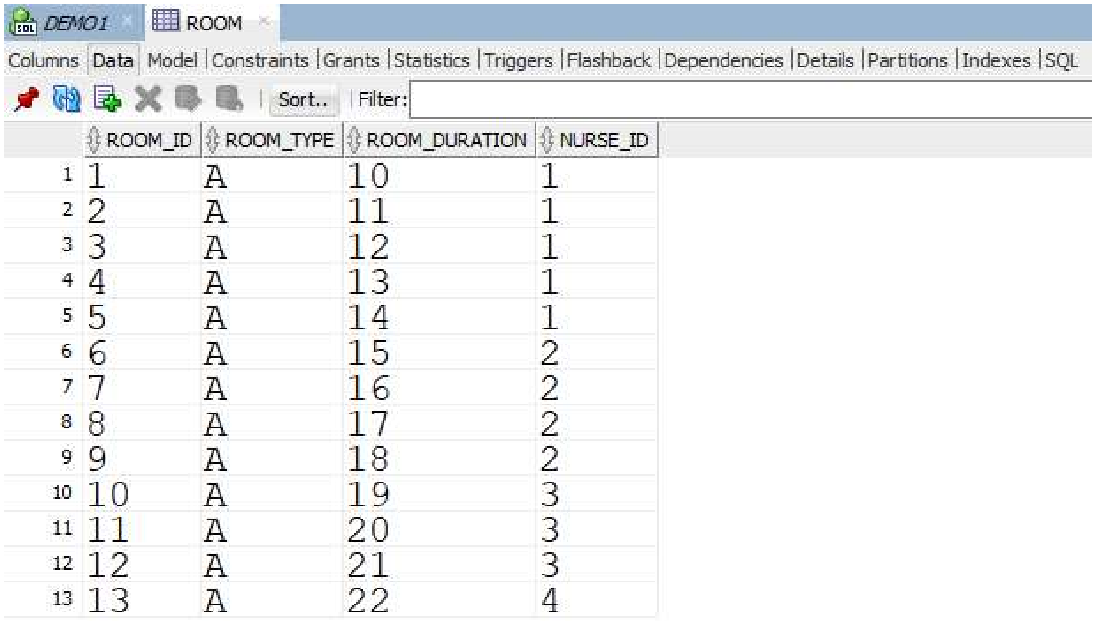
	RECEPTIONIST_ID	EMPLOYEE_NUM
1	222	40038
2	223	40039
3	224	40040

- RECORDS


DEMO1
RECDS
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

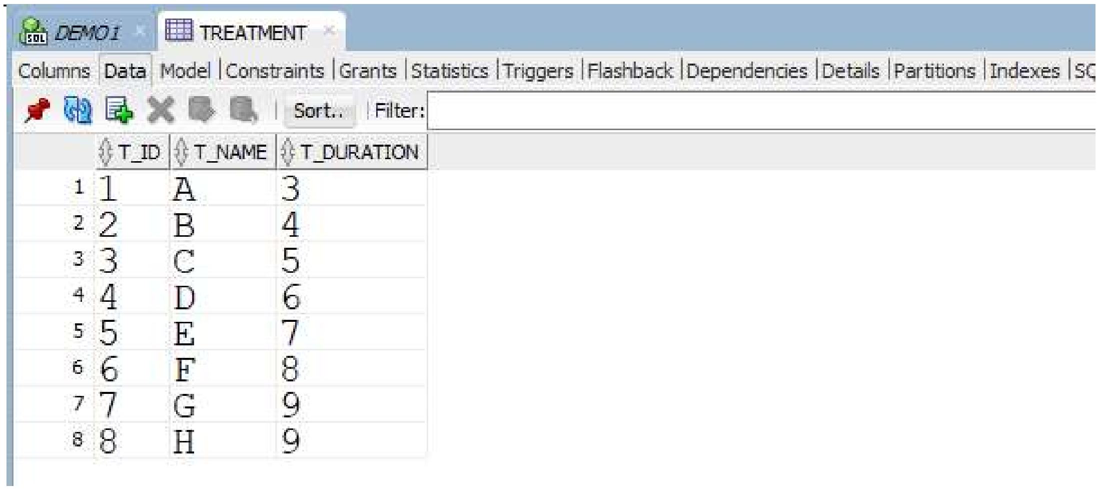
	RECORD_ID	RECEPTIONIST_ID	PATIENT_ID	VISIT_DATE	APPOINTMENT	RECORD_DESCRIPTION
1	6601	222	1026	29-NOV-18	28-NOV-18	heat
2	6602	222	1025	01-JAN-17	01-JAN-17	heat
3	6603	222	1025	02-JAN-17	01-JAN-17	heat
4	6604	222	1025	03-JAN-17	02-JAN-17	heat
5	6605	222	1001	04-JAN-17	03-JAN-17	heat
6	6606	222	1002	05-JAN-17	04-JAN-17	heat
7	6607	222	1003	06-JAN-17	05-JAN-17	heat
8	6608	222	1004	07-JAN-17	06-JAN-17	heat
9	6609	222	1005	08-JAN-17	07-JAN-17	heat
10	6610	222	1006	09-JAN-17	08-JAN-17	heat
11	6611	222	1007	10-JAN-17	09-JAN-17	heat
12	6612	222	1008	11-JAN-17	10-JAN-17	cough
13	6613	222	1009	12-JAN-17	11-JAN-17	cough
14	6614	223	1010	13-JAN-17	12-JAN-17	cough
15	6615	223	1011	14-JAN-17	13-JAN-17	cough
16	6616	223	1012	15-JAN-17	14-JAN-17	cough
17	6617	223	1013	16-JAN-17	15-JAN-17	cough
18	6618	223	1014	17-JAN-17	16-JAN-17	cough
19	6619	223	1015	18-JAN-17	17-JAN-17	eye
20	6620	223	1016	19-JAN-17	18-JAN-17	eye
21	6621	223	1017	20-JAN-17	19-JAN-17	eye
22	6622	223	1018	21-JAN-17	20-JAN-17	eye
23	6623	223	1019	22-JAN-17	21-JAN-17	eye
24	6624	224	1020	23-JAN-17	22-JAN-17	eye
25	6625	224	1021	24-JAN-17	23-JAN-17	Stomach
26	6626	224	1022	25-JAN-17	24-JAN-17	Stomach
27	6627	224	1023	26-JAN-17	25-JAN-17	Stomach
28	6628	224	1024	27-JAN-17	26-JAN-17	Stomach

- ROOM


DEMO1
ROOM
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

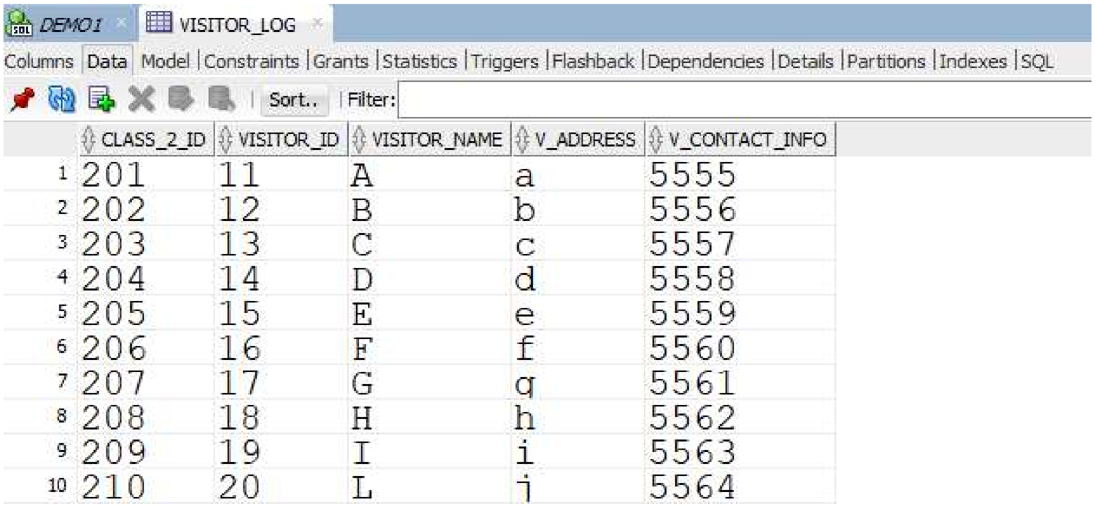
	ROOM_ID	ROOM_TYPE	ROOM_DURATION	NURSE_ID
1	1	A	10	1
2	2	A	11	1
3	3	A	12	1
4	4	A	13	1
5	5	A	14	1
6	6	A	15	2
7	7	A	16	2
8	8	A	17	2
9	9	A	18	2
10	10	A	19	3
11	11	A	20	3
12	12	A	21	3
13	13	A	22	4

- TREATMENT


DEMO1
TREATMENT
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQ
Sort..
Filter:

	T_ID	T_NAME	T_DURATION
1	1	A	3
2	2	B	4
3	3	C	5
4	4	D	6
5	5	E	7
6	6	F	8
7	7	G	9
8	8	H	9

- VISITOR_LOG


DEMO1
VISITOR_LOG
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

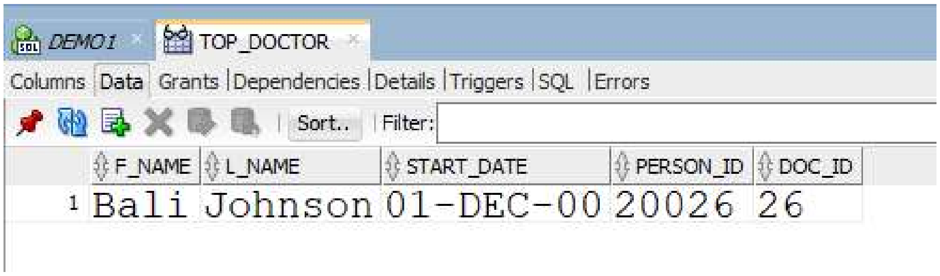
	CLASS_2_ID	VISITOR_ID	VISITOR_NAME	V_ADDRESS	V_CONTACT_INFO
1	201	11	A	a	5555
2	202	12	B	b	5556
3	203	13	C	c	5557
4	204	14	D	d	5558
5	205	15	E	e	5559
6	206	16	F	f	5560
7	207	17	G	g	5561
8	208	18	H	h	5562
9	209	19	I	i	5563
10	210	20	L	j	5564

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.

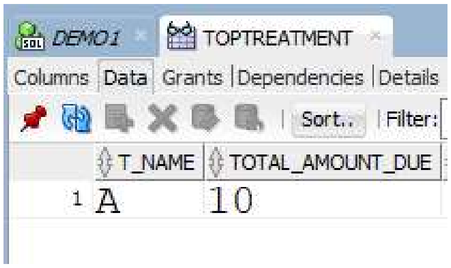
```
CREATE VIEW TOP_DOCTOR AS
SELECT P.F_Name,P.L_Name,E.Start_Date
FROM PERSON P,DOCTOR D,employee E
WHERE P.Person_ID=e.employee_num and d.employee_num=e.employee_num and d.doc_id
in
(
    (
    SELECT DOC_ID
    FROM class_1_patient
    group by Doc_id
    having count(*)>5
    )
    INTERSECT
    (
    SELECT DOC_ID
    FROM class_2_patient
    group by Doc_id
    having count(*)>=10
    )
)
```



	F_NAME	L_NAME	START_DATE	PERSON_ID	DOC_ID
1	Bali	Johnson	01-DEC-00	20026	26

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.

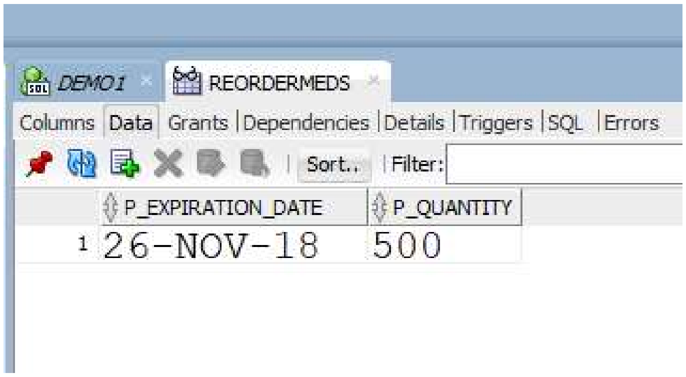
```
CREATE VIEW TopTreatment AS
SELECT DISTINCT T_NAME, TOTAL_AMOUNT_DUE
FROM PAYMENT, TREATMENT
WHERE PAYMENT.T_ID = TREATMENT.T_ID AND PAYMENT.T_ID IN
(
    SELECT T_ID FROM
        (
        SELECT P.T_ID, COUNT(*)AS COUNT
        FROM PAYMENT P
        WHERE P.T_ID IS NOT NULL
        GROUP BY P.T_ID
        ORDER BY COUNT DESC
        )
    WHERE ROWNUM<2
)
```



	T_NAME	TOTAL_AMOUNT_DUE
1	A	10

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.

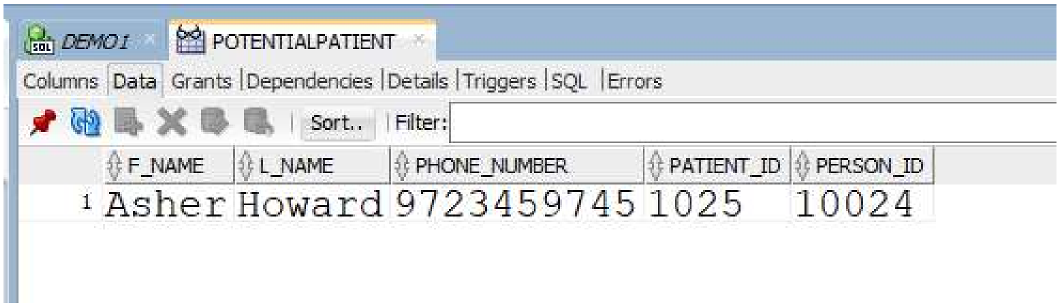
```
CREATE VIEW ReorderMeds AS
SELECT  P_EXPIRATION_DATE, P_QUANTITY
FROM PHARMACY
WHERE P_QUANTITY<1000 AND (TO_DATE(P_EXPIRATION_DATE, 'DD-MON-YY') - TO_DATE(sysdate, 'DD-MON-YY'))<30
```



	P_EXPIRATION_DATE	P_QUANTITY
1	26-NOV-18	500

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.

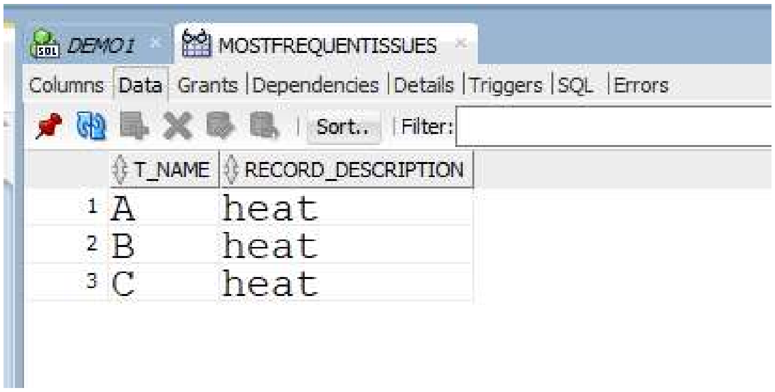
```
CREATE VIEW POTIENTIALPATIENT AS
SELECT distinct P.F_NAME, P.L_NAME,P.PHONE_NUMBER,C1P.PATIENT_ID,p.person_id
FROM RECDS R, PERSON P,CLASS_1_PATIENT C1P
WHERE R.PATIENT_ID =C1P.PATIENT_ID AND C1P.PERSON_ID =P.PERSON_ID AND C1P.CLASS_2_ID IS NULL AND C1P.PATIENT_ID IN
(
    SELECT Patient_ID
    FROM
        (
        SELECT R.PATIENT_ID,COUNT(*)
        FROM RECDS R
        GROUP BY R.PATIENT_ID
        HAVING COUNT(*) >= 3
        )
);
```



	F_NAME	L_NAME	PHONE_NUMBER	PATIENT_ID	PERSON_ID
1	Asher	Howard	9723459745	1025	10024

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.

```
CREATE VIEW MostFrequentIssues as
SELECT DISTINCT T.T_NAME, RECORD_DESCRIPTION
FROM TREATMENT T,RECDS R,PAYMENT P
WHERE T.T_ID = P.T_ID AND R.PATIENT_ID =P.PATIENT_ID AND P.PATIENT_ID IN
(
    SELECT PATIENT_ID
    FROM  RECDS
    WHERE RECORD_DESCRIPTION IN
    (
        SELECT RECORD_DESCRIPTION
        FROM
         (
            SELECT RECORD_DESCRIPTION,COUNT(*)
            FROM RECDS
            GROUP BY RECORD_DESCRIPTION
            ORDER BY RECORD_DESCRIPTION DESC
        )
        WHERE ROWNUM<2
    )
)
```



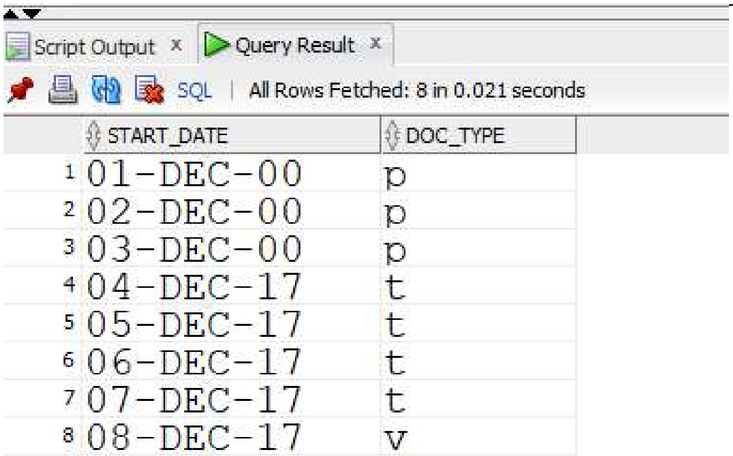
	T_NAME	RECORD_DESCRIPTION
1	A	heat
2	B	heat
3	C	heat

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.

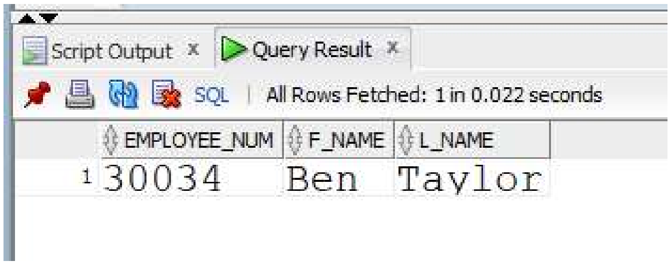
```
SELECT E.start_date,D.DOC_TYPE
FROM DOCTOR D,EMPLOYEE E
WHERE D.EMPLOYEE_NUM = E.EMPLOYEE_NUM
```



```
   START_DATE   DOC_TYPE
1  01-DEC-00    p
2  02-DEC-00    p
3  03-DEC-00    p
4  04-DEC-17    t
5  05-DEC-17    t
6  06-DEC-17    t
7  07-DEC-17    t
8  08-DEC-17    v
```

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

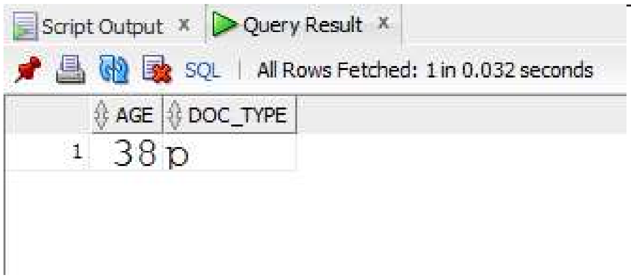
```
SELECT e.employee_num,P.F_name,P.L_name
FROM Person p,Employee e
WHERE p.person_ID = e.EMPLOYEE_NUM and(SYSDATE-E.start_DATE)<100;
```



```
   EMPLOYEE_NUM  F_NAME  L_NAME
1  30034         Ben     Taylor
```

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

```
select age, doc_type
from
    (
    SELECT distinct FLOOR((SYSDATE-P.BITH_DATE)/365)as age,Doc_type
    FROM Top_Doctor T, Person P, Employee E, Doctor D
    WHERE T.Doc_ID=D.Doc_ID and T.person_ID =P.person_ID
    order by age
    )
where rownum<6
```

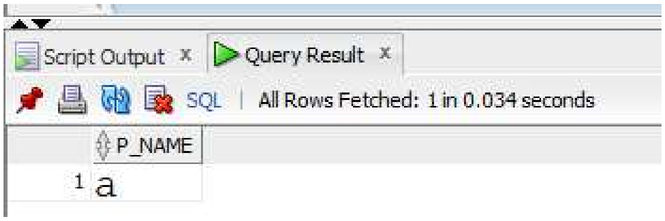


All Rows Fetched: 1 in 0.032 seconds

	AGE	DOC_TYPE
1	38	p

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

```sql
SELECT DISTINCT PH.P_NAME
FROM MEDICAL_INFORMATION ME,PHARMACY PH,TOPTREATMENT TOP
WHERE ME.MEDICINE_CODE = PH.MEDICINE_CODE AND TOP.T_ID = ME.T_ID
```

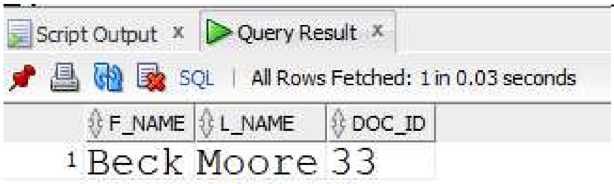


All Rows Fetched: 1 in 0.034 seconds

	P_NAME
1	a

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.)

```sql
SELECT P.F_NAME,P.L_NAME ,D.DOC_ID
FROM PERSON P ,DOCTOR D
WHERE P.PERSON_ID =D.EMPLOYEE_NUM AND D.DOC_ID NOT IN
 (
    (
    SELECT D.DOC_ID
    FROM CLASS_2_PATIENT C2P,DOCTOR D
    WHERE C2P.DOC_ID=D.DOC_ID AND FLOOR(SYSDATE-(DATE_OF_ADMITTED))>150
    )
    UNION
    (
    SELECT DOC_ID
    FROM
        (
        SELECT C1P.DOC_ID ,C1P.PATIENT_ID
        FROM PAYMENT P,CLASS_1_PATIENT C1P
        WHERE P.PATIENT_ID=C1P.PATIENT_ID AND C1P.PATIENT_ID=P.PATIENT_ID AND FLOOR(SYSDATE-(PAYMENT_DATE))>150
        )
    )
 )
```

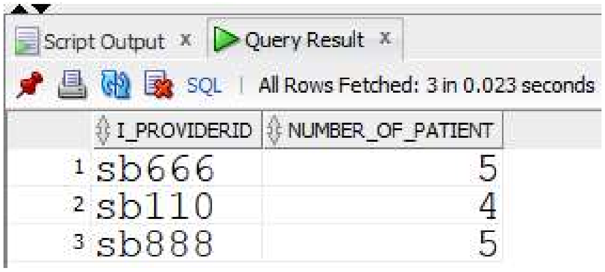


All Rows Fetched: 1 in 0.03 seconds

	F_NAME	L_NAME	DOC_ID
1	Beck	Moore	33

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

```
SELECT I_PROVIDERID,COUNT(*)AS NUMBER_OF_PATIENT
FROM PAYMENT
WHERE CASH_AMOUNT IS NULL
GROUP BY I_PROVIDERID
```

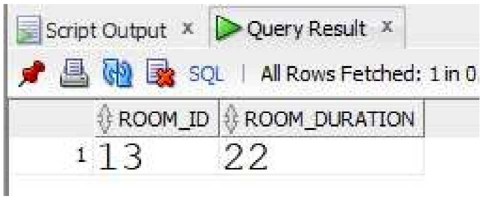


All Rows Fetched: 3 in 0.023 seconds

	I_PROVIDERID	NUMBER_OF_PATIENT
1	sb666	5
2	sb110	4
3	sb888	5

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

```
SELECT ROOM_ID, ROOM_DURATION
FROM ROOM
WHERE ROOM_DURATION IN
(
SELECT MAX(ROOM_DURATION)AS OCCUPIED
FROM ROOM
)
```

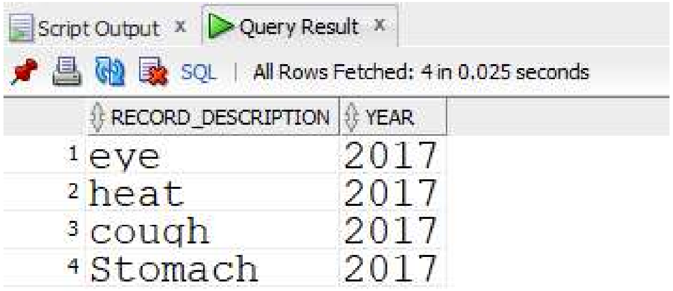


All Rows Fetched: 1 in 0

	ROOM_ID	ROOM_DURATION
1	13	22

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

```
SELECT DISTINCT RECORD_DESCRIPTION, YEAR
FROM
(
    SELECT TO_CHAR(VISIT_DATE,'yyyy') AS YEAR,RECORD_DESCRIPTION
    FROM RECDS R
    WHERE TO_CHAR(VISIT_DATE,'yyyy') IN
    (
        SELECT YEAR
        FROM
            (
            SELECT TO_CHAR(VISIT_DATE,'yyyy') AS YEAR,COUNT(*)
            FROM RECDS
            GROUP BY TO_CHAR(VISIT_DATE,'yyyy')
            ORDER BY COUNT(*) DESC
            )
        WHERE ROWNUM<2
    )
)
```



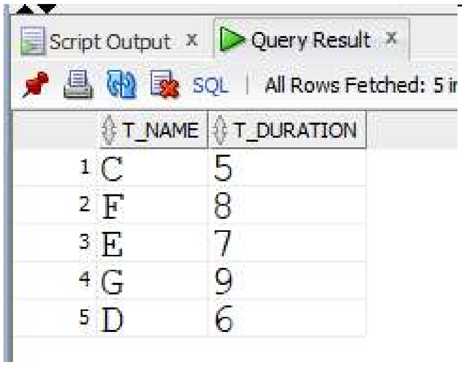
All Rows Fetched: 4 in 0.025 seconds

	RECORD_DESCRIPTION	YEAR
1	eye	2017
2	heat	2017
3	cough	2017
4	Stomach	2017

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

```
SELECT T.T_NAME,T.T_DURATION
FROM TREATMENT T,CLASS_2_PATIENT C2P
WHERE T.T_ID=C2P.T_ID AND C2P.T_ID IN
    (
    SELECT T_ID
    FROM
        (
        SELECT T_ID,COUNT(T_ID)
        FROM CLASS_2_PATIENT
        GROUP BY T_ID
        HAVING COUNT(T_ID)=1
        )
    )

    SELECT MIN (FQ)
    FROM
    (
        SELECT T_ID,COUNT(T_ID)AS FQ
        FROM CLASS_2_PATIENT
        GROUP BY T_ID
    )
```

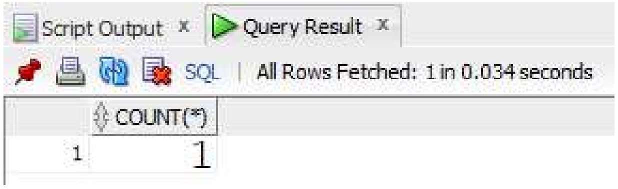


All Rows Fetched: 5 i

	T_NAME	T_DURATION
1	C	5
2	F	8
3	E	7
4	G	9
5	D	6

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

```
SELECT COUNT(*)
FROM CLASS_2_PATIENT
WHERE CLASS_2_PATIENT.DATE_OF_ADMITTED>
 (
   SELECT MAX(START_DATE)
   FROM EMPLOYEE
 )
```

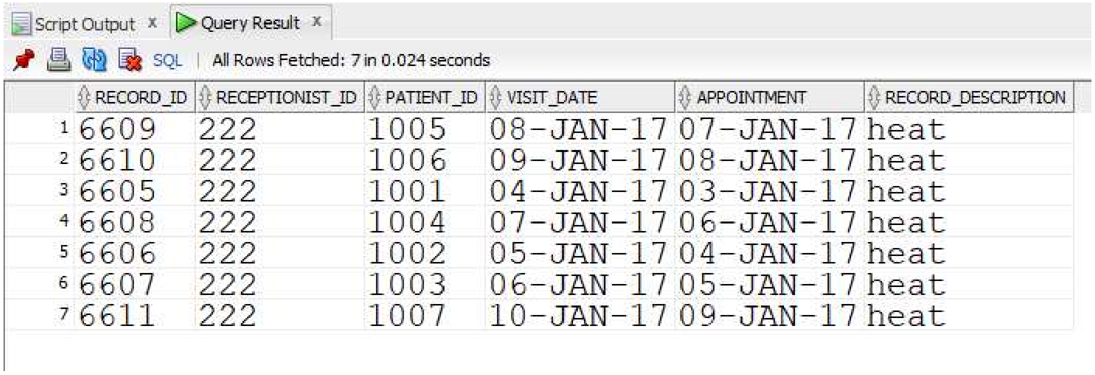


All Rows Fetched: 1 in 0.034 seconds

	COUNT(*)
1	1

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 T3.*
FROM CLASS_1_PATIENT T1 LEFT JOIN CLASS_2_PATIENT T2 ON T2.CLASS_2_ID=T1.CLASS_2_ID
LEFT JOIN RECDS T3 ON T1.PATIENT_ID=T3.PATIENT_ID
WHERE (TO_CHAR(T2.DATE_OF_ADMITTED,'yyyymmdd')-TO_CHAR(T3.VISIT_DATE,'yyyymmdd'))<7
```

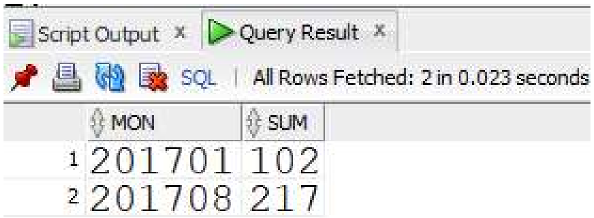


All Rows Fetched: 7 in 0.024 seconds

	RECORD_ID	RECEPTIONIST_ID	PATIENT_ID	VISIT_DATE	APPOINTMENT	RECORD_DESCRIPTION
1	6609	222	1005	08-JAN-17	07-JAN-17	heat
2	6610	222	1006	09-JAN-17	08-JAN-17	heat
3	6605	222	1001	04-JAN-17	03-JAN-17	heat
4	6608	222	1004	07-JAN-17	06-JAN-17	heat
5	6606	222	1002	05-JAN-17	04-JAN-17	heat
6	6607	222	1003	06-JAN-17	05-JAN-17	heat
7	6611	222	1007	10-JAN-17	09-JAN-17	heat

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

```
SELECT TO_CHAR(PAYMENT_DATE,'yyyymm') AS MON, SUM (TOTAL_AMOUNT_DUE) AS SUM
FROM PAYMENT
WHERE TO_CHAR(PAYMENT_DATE,'yyyy')=2017
GROUP BY TO_CHAR(PAYMENT_DATE,'yyyymm')
```

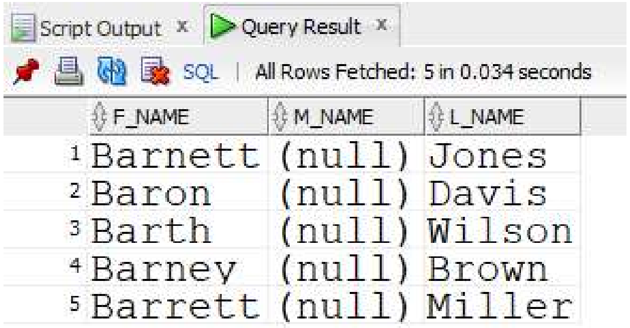


All Rows Fetched: 2 in 0.023 seconds

	MON	SUM
1	201701	102
2	201708	217

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 DISTINCT F_NAME,M_NAME,L_NAME
FROM PERSON, DOCTOR, CLASS_1_PATIENT, RECDS
WHERE RECDS.PATIENT_ID =CLASS_1_PATIENT.PATIENT_ID
    AND CLASS_1_PATIENT.DOC_ID = DOCTOR.DOC_ID
    AND DOCTOR.EMPLOYEE_NUM =PERSON.PERSON_ID
    AND CLASS_1_PATIENT.CLASS_2_ID IS NULL
    AND RECDS.PATIENT_ID  IN
    (
    SELECT PATIENT_ID
    FROM
        (
        SELECT RECDS.PATIENT_ID,COUNT(*)
        FROM RECDS
        --WHERE RECDS.VISIT_DATE
        GROUP BY RECDS.PATIENT_ID
        HAVING COUNT(*)=1
        )
    )
```

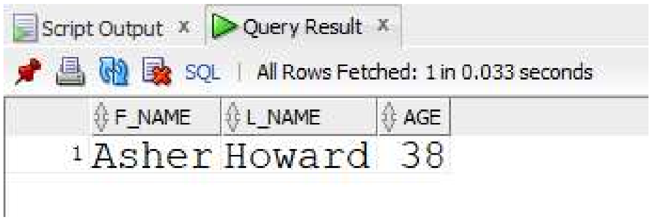


All Rows Fetched: 5 in 0.034 seconds

	F_NAME	M_NAME	L_NAME
1	Barnett	(null)	Jones
2	Baron	(null)	Davis
3	Barth	(null)	Wilson
4	Barney	(null)	Brown
5	Barrett	(null)	Miller

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

```
SELECT P.F_NAME,P.L_NAME,FLOOR((SYSDATE-P.BITH_DATE)/365) AS AGE
FROM   POTENTIALPATIENT PO,PERSON P
WHERE  PO.PERSON_ID= P.PERSON_ID
```



```
All Rows Fetched: 1 in 0.033 seconds

  F_NAME  L_NAME  AGE
1 Asher   Howard  38
```

5. Dependency Diagram

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

5.1 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.

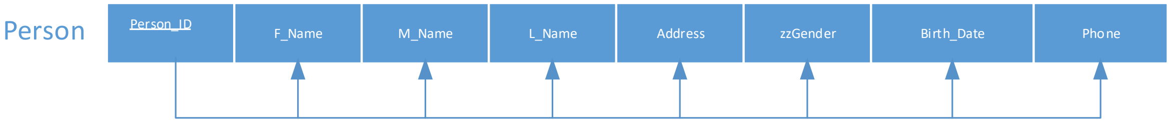


Figure 1. Dependency Diagram of Hospital Personnel

5.2 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.

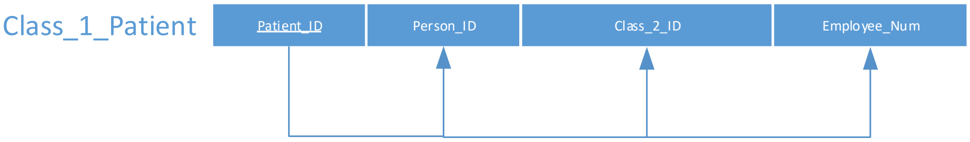


Figure 2. Dependency Diagram of Class_1_Patient

5.3 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.

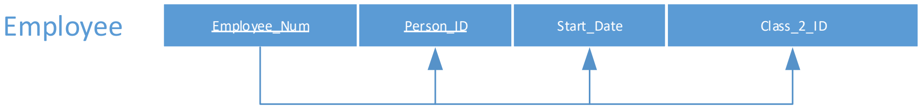


Figure 3. Dependency Diagram of Employee

5.4 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.

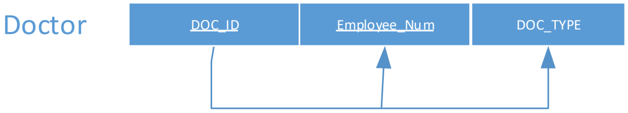


Figure 2. Dependency Diagram of Doctor

5.5 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.

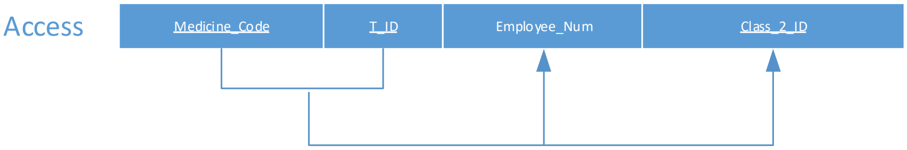


Figure 5. Dependency Diagram of Access

5.6 Hospital Nurse

The dependency diagram is shown as Figure 6.

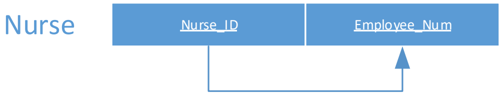


Figure 6. Dependency Diagram of Nurse

5.7 Hospital Room

The dependency diagram is shown as Figure 7.

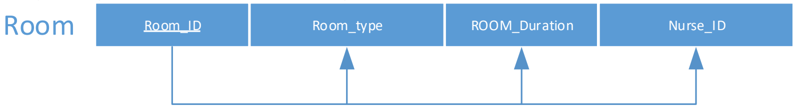


Figure 7. Dependency Diagram of Room

5.8 Hospital Receptionist

The dependency diagram is shown as Figure 8.

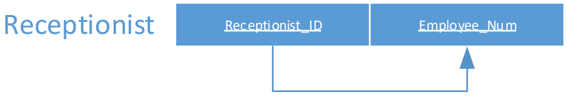


Figure 8. Dependency Diagram Receptionist

5.9 Patient Records

The dependency diagram is shown as Figure 9.

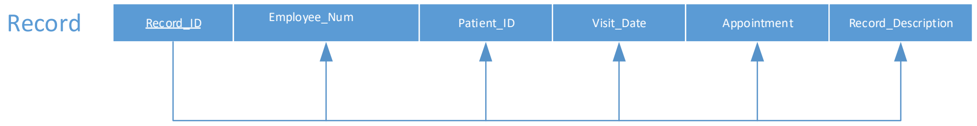


Figure 9. Dependency Diagram of Records

5.10 Payment Information

The dependency diagram is shown as Figure 10.

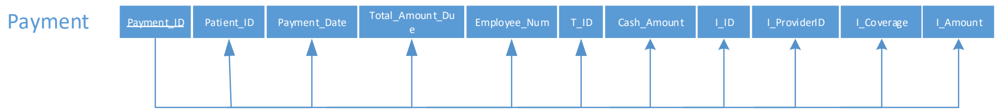


Figure 10. Dependency Diagram of Payment

5.11 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.

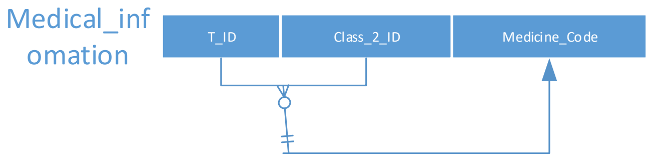


Figure 11. Dependency Diagram

5.12 Hospital Pharmacy

The dependency diagram is shown as Figure 12.

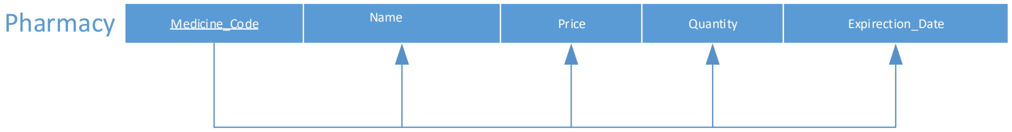


Figure 12. Dependency Diagram

5.13 Hospital Treatment

The dependency diagram is shown as Figure 13.

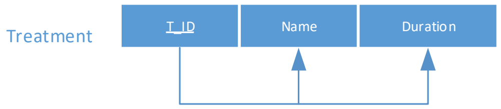


Figure 13. Dependency Diagram

5.14 Class 2 Patient

The dependency diagram is shown as Figure 14.

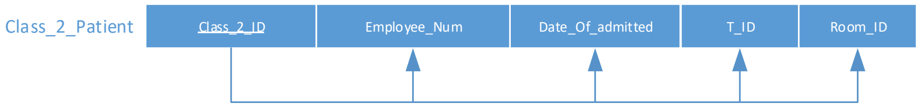


Figure 14. Dependency Diagram

5.15 Visitor Log

The dependency diagram is shown as Figure 15.

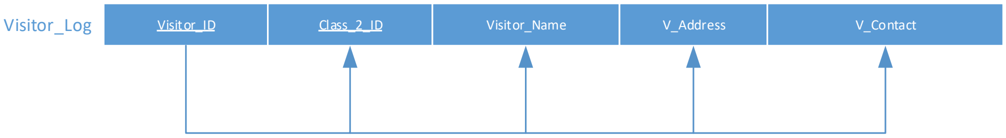
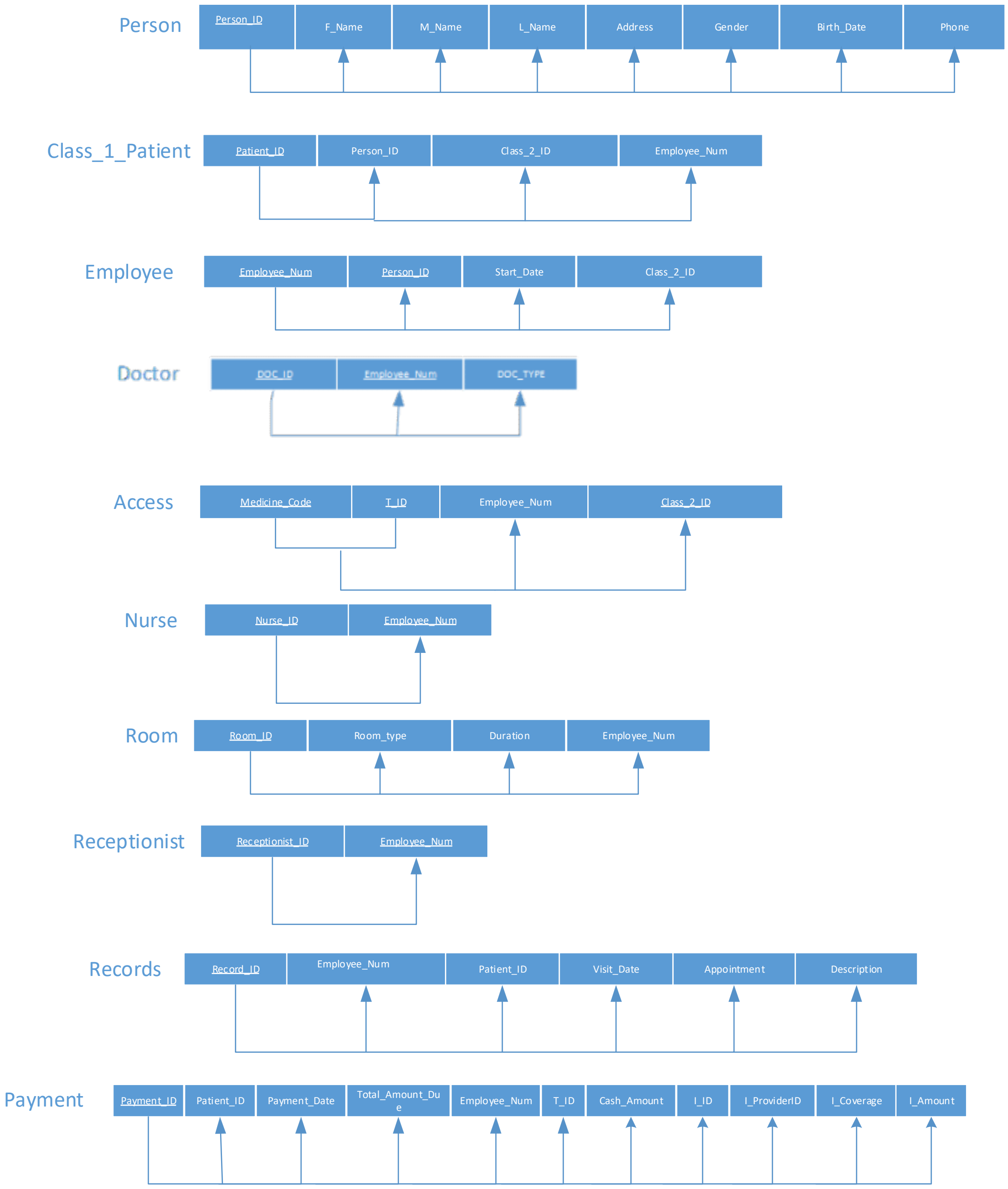


Figure 15. Dependency Diagram of Hospital Personnel

5.16 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.


Person
Person_ID F_Name M_Name L_Name Address Gender Birth_Date Phone
Class_1_Patient
Patient_ID Person_ID Class_2_ID Employee_Num
Employee
Employee_Num Person_ID Start_Date Class_2_ID
Doctor
DOC_ID Employee_Num DOC_TYPE
Access
Medicine_Code T_ID Employee_Num Class_2_ID
Nurse
Nurse_ID Employee_Num
Room
Room_ID Room_type Duration Employee_Num
Receptionist
Receptionist_ID Employee_Num
Records
Record_ID Employee_Num Patient_ID Visit_Date Appointment Description
Payment
Payment_ID Patient_ID Payment_Date Total_Amount_Du e Employee_Num T_ID Cash_Amount I_ID I_ProviderID I_Coverage I_Amount

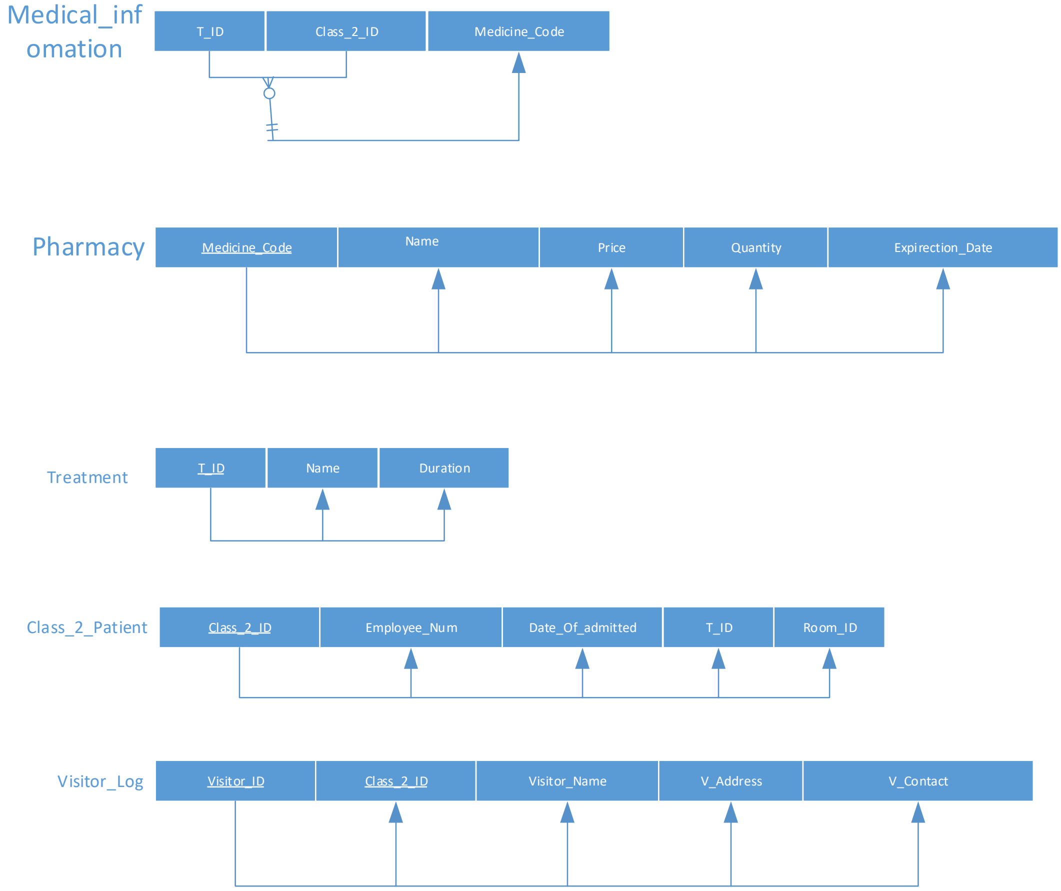


Figure 16. Whole Dependency Diagram for Dallas Care Database

6. 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.