Assignment 10

Topic: Final Documentation

Group Members :

Sukhjot Saggu: 500883082

Mohammad Zaman: 500890883

Ransika Perera: 500903101

Section: 08

T.A.: Mr. Haythan Qushtom

Table of Contents

Assignment 1

Question

Assignment 1: Application Description: Week of Sep. 14. Finalize the application in consultation with the lab TA. Submit a report on description of the application, its functions and the information that you expect from it, in high level to the TA (6 marks)

Data Layout

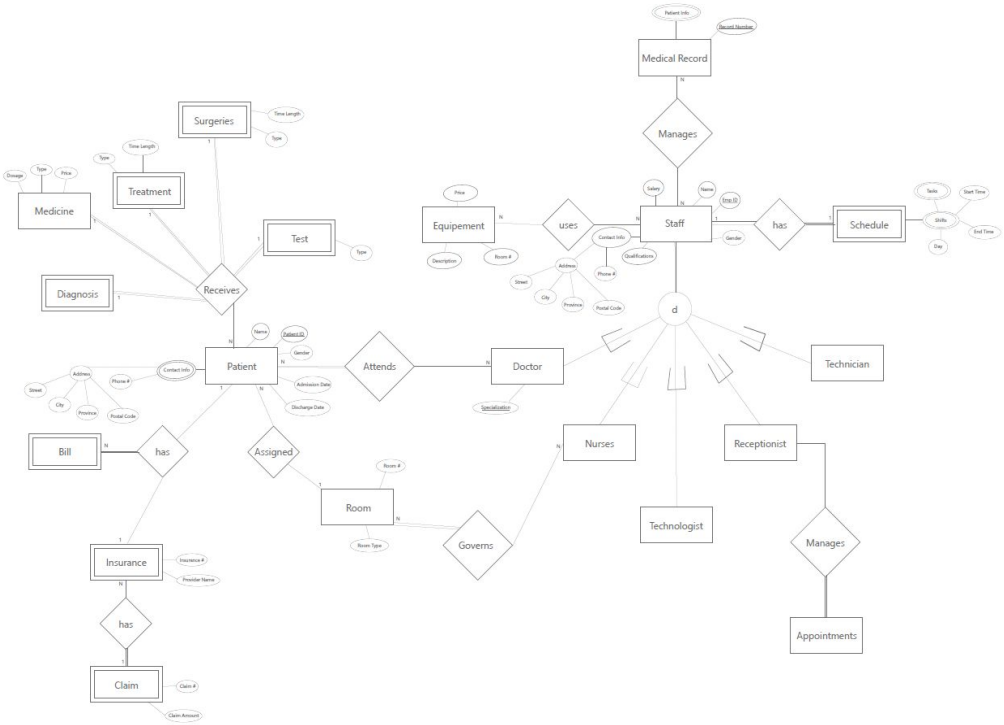
The application that our group will be creating is a patient information and record management system. This system can be used by medical institutions in order to keep track of the patients they have and related information of each patient. The information would consist of their personal info which is their name, number, age, blood group, address, any special characteristics (allergy, disabilities, etc), the reason for their appointment, possible prescriptions were given, any diagnosis of an illness, and if provided, the date to check back with the doctor.

Summary

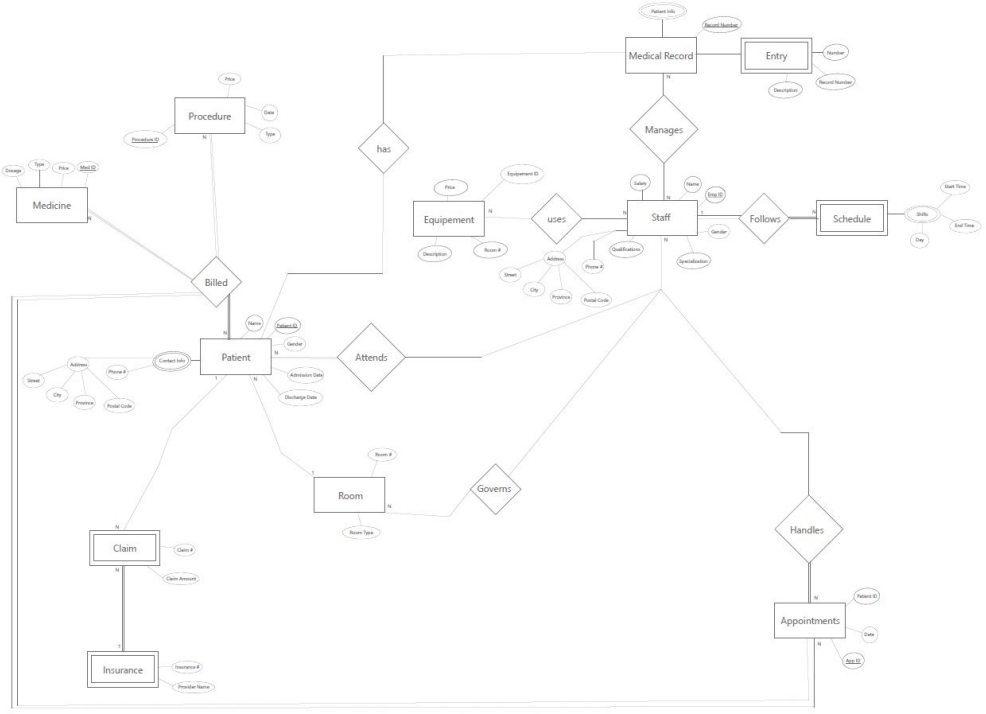
This type of system is very crucial for medical institutions such as hospitals and family doctors as it provides them with a way to efficiently manage the large volumes of patients that visit as well as keep track of what has happened in each appointment to help the doctor keep track of the patient's health. This system would mainly be used by the doctor during an appointment with the patient to view information on that patient but can also be used by the office secretary in order to get the personal information of a patient such as their phone number in order to book a visit for the patient. It will also include information if the patient has experienced any health emergency to their record which the doctor will be able to refer to if needed. When a new patient arrives, their personal information listed above will be filled into the database with a new ID created for that patient. That ID will store all of the information related to the specific patient. When accessing information from the database a variety of methods will be available such as, by patient ID, patient name, or patient phone number. The user will also have a way to sort the list of patients through alphabetical order, or ID number. Finally, if the medical institution no longer requires the information for a specific patient then they will also be able to remove that specific patient from the database.

Assignment 2

Original:



Final (changed over the semester):



Assignment 3

```
CREATE TABLE staff(
    staff_name VARCHAR(25) NOT NULL,
    emp_id NUMBER PRIMARY KEY,
    gender VARCHAR(10),
    salary NUMBER,
    qulifications VARCHAR(25),
    phone_number NUMBER,
    address VARCHAR(50)
);

CREATE TABLE doctor (
    emp_id NUMBER REFERENCES staff(emp_id),
    specialization VARCHAR(100),
    PRIMARY KEY(emp_id)
);

CREATE TABLE nurse (
    emp_id NUMBER,
    FOREIGN KEY (emp_id) REFERENCES doctor(emp_id),
    PRIMARY KEY(emp_id)
);

CREATE TABLE technologist (
    emp_id NUMBER,
    FOREIGN KEY (emp_id) REFERENCES doctor(emp_id),
    PRIMARY KEY(emp_id)
);

CREATE TABLE receptionist (
    emp_id NUMBER,
    FOREIGN KEY (emp_id) REFERENCES doctor(emp_id),
    PRIMARY KEY(emp_id)
);

CREATE TABLE billed (
    bill_id NUMBER PRIMARY KEY,
    patient_id NUMBER REFERENCES patient(patient_id),
    procedure_id NUMBER NOT NULL,
    procedure_date DATE,
    procedure_amount NUMBER
);
```

```
CREATE TABLE appointment (
    app_date TIMESTAMP NOT NULL,
    patient_id NUMBER,
    procedure_id NUMBER,
    FOREIGN KEY (patient_id, procedure_id) REFERENCES billed(patient_id, procedure_id),
    PRIMARY KEY(app_date, patient_id)
);

CREATE TABLE technician (
    emp_id NUMBER,
    FOREIGN KEY (emp_id) REFERENCES doctor(emp_id),
    PRIMARY KEY(emp_id)
);

CREATE TABLE patient(
    p_name VARCHAR(25),
    patient_id NUMBER PRIMARY KEY,
    gender VARCHAR(10),
    admission_date DATE,
    discharge_date DATE,
    address VARCHAR(50),
    phone_number NUMBER
);

CREATE TABLE schedule(
    emp_id NUMBER,
    tasks VARCHAR(100),
    start_time TIMESTAMP,
    end_time TIMESTAMP
);

CREATE TABLE assigned (
    patient_id NUMBER REFERENCES patient(patient_id),
    room_number NUMBER REFERENCES room(room_number),
    PRIMARY KEY(patient_id, room_number)
);

CREATE TABLE equipment (
    equipment_id NUMBER PRIMARY KEY,
    room_number NUMBER REFERENCES room(room_number),
    price NUMBER,
    description VARCHAR(100)
);

CREATE TABLE medical_record (
    record_number NUMBER PRIMARY KEY,
```

```
    patient_info VARCHAR(1000)
);

CREATE TABLE surgeries (
    surgeries_type VARCHAR(25),
    len NUMBER
);

CREATE TABLE treatment(
    treatment_type VARCHAR(25),
    len NUMBER
);

CREATE TABLE testing(
    test_type VARCHAR(25)
);

CREATE TABLE medicine(
    medicine_type VARCHAR(25),
    price NUMBER,
    dosage NUMBER
);

CREATE TABLE insurace(
    provider_name VARCHAR(25),
    insurance_number NUMBER,
    phone_number NUMBER,
    address VARCHAR(50)
);

CREATE TABLE claim(
    claim_number NUMBER PRIMARY KEY,
    bill_id NUMBER REFERENCES billed(bill_id)
);

CREATE TABLE room(
    room_number NUMBER PRIMARY KEY,
    room_type VARCHAR(25)
);

CREATE TABLE files (
    patient_ID NUMBER,
    FOREIGN KEY (patient_id) REFERENCES billed(patient_id),
    claim_number NUMBER,
    PRIMARY KEY (patient_id, claim_number)
);
```

```
CREATE TABLE processes (
   claim_number REFERENCES claim(claim_number),
   insurance_number REFERENCES insurance(insurance_number),
   PRIMARY KEY(claim_number, insurance_number)
);

CREATE TABLE uses(
   equipment_id NUMBER REFERENCES equipment(equipment_id),
   emp_id NUMBER REFERENCES staff(emp_id),
   PRIMARY KEY (equipment_id, emp_id)
);

CREATE TABLE manages(
   record_number NUMBER REFERENCES medical_record(record_number),
   emp_id NUMBER REFERENCES staff(emp_id)
);

CREATE TABLE follow (
   emp_id NUMBER REFERENCES staff(emp_id)
);

CREATE TABLE attends(
   emp_id NUMBER,
   specialization VARCHAR(100) REFERENCES doctor(specialization),
   patient_id NUMBER,
   FOREIGN KEY (patient_id) REFERENCES billed(patient_id),
   FOREIGN KEY (emp_id) REFERENCES doctor(emp_id),
   PRIMARY KEY (emp_id, patient_id)
);

CREATE TABLE governs (
   emp_id NUMBER REFERENCES staff(emp_id),
   room_number NUMBER REFERENCES room(room_number),
   PRIMARY KEY (emp_id, room_number)
);

CREATE TABLE manages (
   emp_id NUMBER,
   patient_id NUMBER,
   FOREIGN KEY (patient_id) REFERENCES billed(patient_id),
   FOREIGN KEY (app_date) REFERENCES appointment(app_date),
   FOREIGN KEY (emp_id) REFERENCES doctor(emp_id),
   PRIMARY KEY(emp_id, patient_id, app_date)
);
```

Assignment 4

```
CREATE TABLE staff(
   staff_name VARCHAR(25) NOT NULL,
   emp_id NUMBER PRIMARY KEY,
   gender VARCHAR(10),
   salary NUMBER,
   qulifications VARCHAR(25),
   phone_number NUMBER,
   address VARCHAR(50),
   special VARCHAR(50) NOT NULL
);
insert into staff values('John Doe', 1, 'male', 200000, 'degree', 416, 'XX Place Drive',
'neuroseurgeon');
insert into staff values('Jane Albert', 2, 'female', 210000, 'degree', 4161234566, '45 Study Drive',
'technician');
insert into staff values('Michael Toon', 3, 'male', 80000, 'diploma', 8974561234, '15 Library
Avenue', 'receptionist');
insert into staff values('Emily Blake', 4, 'female', 95000, 'diploma', 6458921678, '125 Desert Road',
'nurse');
insert into staff values('Lydia Chen', 5, 'female', 65000, 'doctoral degree', 5684359635, '7 Best
Drive', 'technologist');
select * from staff;

CREATE TABLE billed (
   bill_id NUMBER PRIMARY KEY,
   patient_id NUMBER REFERENCES patient(patient_id),
   procedure_id NUMBER NOT NULL,
   procedure_date DATE,
   procedure_amount NUMBER
);
insert into billed values(1, 3, 1, '2020-09-05',5000);
insert into billed values(2, 1, 2, '2020-08-07',8000);
select * from billed;

CREATE TABLE appointment (
   app_date TIMESTAMP NOT NULL,
   patient_id NUMBER REFERENCES patient(patient_id),
   PRIMARY KEY (app_date, patient_id)
);
insert into appointment values('2020-08-04 13:00:00', 3);
insert into appointment values('2020-04-06 10:00:00', 1);
select * from appointment;

CREATE TABLE patient(
   p_name VARCHAR(25),
```

```
    patient_id NUMBER PRIMARY KEY,
    gender VARCHAR(10),
    admission_date DATE,
    discharge_date DATE,
    address VARCHAR(50),
    phone_number NUMBER
);
insert into patient values('Sara', 1, 'female', '2020-09-12', '2020-09-17', '123 Anywhere Drive',
'4169876543');
insert into patient values('David James', 2, 'male', '2020-06-11', '2020-07-12', '456 Lily Drive',
'4164536785');
insert into patient values('Joe Lad', 3, 'male', '2020-02-14', '2020-02-14', '789 Fall Drive',
'4164890785');
select * from patient;

drop table schedule;
CREATE TABLE schedule(
    s_id NUMBER PRIMARY KEY,
    emp_id NUMBER,
    tasks VARCHAR(100),
    start_time TIMESTAMP,
    end_time TIMESTAMP
);
insert into schedule values(1, 1, 'Surgery', '2020-08-04 12:00:00', '2020-08-04 15:00:00');
insert into schedule values(2, 2, 'Check Up', '2020-09-05 08:00:00', '2020-09-05 12:00:00');
select * from schedule;

CREATE TABLE assigned (
    patient_id NUMBER REFERENCES patient(patient_id),
    room_number NUMBER REFERENCES room(room_number),
    PRIMARY KEY(patient_id, room_number)
);
insert into assigned values(1, 1);
insert into assigned values(2,2);
select * from assigned;

CREATE TABLE equipment (
    equipment_id NUMBER PRIMARY KEY,
    room_number NUMBER REFERENCES room(room_number),
    price NUMBER,
    description VARCHAR(100)
);
insert into equipment values(1, 1, 99, 'Used for MRI Sacn');
select * from equipment;

CREATE TABLE medical_record (
```

```
    record_number NUMBER PRIMARY KEY,
    patient_id NUMBER REFERENCES patient(patient_id),
    patient_info VARCHAR(1000)
);
insert into medical_record values(1,1, 'Asthma Problem');
insert into medical_record values(3,2, 'Living Problem');
insert into medical_record values(2,3, 'Lung Problem');
select * from medical_record;

CREATE TABLE surgeries (
    surgeries_type VARCHAR(25),
    len NUMBER
);
insert into surgeries values('amputation', 4);
insert into surgeries values('laser eye', 1);
select * from surgeries;

CREATE TABLE treatment(
    treatment_type VARCHAR(25),
    len NUMBER
);
insert into treatment values('casting', 1);
insert into treatment values('injection', 3);
select * from treatment;

CREATE TABLE testing(
    test_type VARCHAR(25)
);
insert into testing values('blood test');
insert into testing values('physical');
select * from testing;

CREATE TABLE medicine(
    medicine_type VARCHAR(25),
    price NUMBER,
    dosage NUMBER
);
insert into medicine values('pill',45,90);
insert into medicine values('drug',23,15);
select * from medicine;

CREATE TABLE insurance(
    provider_name VARCHAR(25),
    insurance_number NUMBER PRIMARY KEY,
    phone_number NUMBER,
    address VARCHAR(50)
```

```
);
insert into insurance values('Sunlife',1,4157891234,'67 Bank Drive');
insert into insurance values('Moonlife',2,456789123,'98 Wall Street');
select * from insurance;

CREATE TABLE claim(
   claim_number NUMBER PRIMARY KEY,
   bill_id NUMBER REFERENCES billed(bill_id)
);
insert into claim values(1, 2);
insert into claim values(2, 1);
select * from claim;

CREATE TABLE room(
   room_number NUMBER PRIMARY KEY,
   room_type VARCHAR(25)
);
insert into room values(1, 'recovery');
insert into room values(2, 'surgery');
select * from room;

CREATE TABLE files (
   patient_ID NUMBER,
   FOREIGN KEY (patient_id) REFERENCES billed(patient_id),
   claim_number NUMBER,
   PRIMARY KEY (patient_id, claim_number)
);

CREATE TABLE processes (
   claim_number NUMBER REFERENCES claim(claim_number),
   insurance_number NUMBER REFERENCES insurance(insurance_number),
   PRIMARY KEY(claim_number, insurance_number)
);
insert into processes values(1, 1);
insert into processes values(2, 2);
select * from processes;

CREATE TABLE uses(
   equipment_id NUMBER REFERENCES equipment(equipment_id),
   emp_id NUMBER REFERENCES staff(emp_id),
   PRIMARY KEY (equipment_id, emp_id)
);
insert into uses values(1,3);
select * from uses;

CREATE TABLE manages(
```

```
    record_number NUMBER REFERENCES medical_record(record_number),
    emp_id NUMBER REFERENCES staff(emp_id)
);
insert into manages values(3,2);
insert into manages values(1,3);
select * from manages;

CREATE TABLE follow (
    emp_id NUMBER REFERENCES staff(emp_id),
    s_id NUMBER REFERENCES schedule(s_id),
    PRIMARY KEY(emp_id, s_id)
);
insert into follow values(1,1);
insert into follow values(2,2);
select * from follow;

CREATE TABLE attends(
    emp_id NUMBER REFERENCES staff(emp_id),
    patient_id NUMBER REFERENCES patient(patient_id),
    PRIMARY KEY (emp_id, patient_id)
);
insert into attends values(1,1);
insert into attends values(2,2);
select * from attends;

CREATE TABLE governs (
    emp_id NUMBER REFERENCES staff(emp_id),
    room_number NUMBER REFERENCES room(room_number),
    PRIMARY KEY (emp_id, room_number)
);
insert into governs values(1, 1);
insert into governs values(2, 2);
select * from governs;

CREATE TABLE handles (
    emp_id NUMBER REFERENCES staff(emp_id),
    patient_id NUMBER,
    app_date TIMESTAMP,
    FOREIGN KEY (app_date, patient_id) REFERENCES appointment(app_date, patient_id),
    PRIMARY KEY(emp_id, patient_id, app_date)
);
insert into handles values(2,3,'2020-08-04 9:00:00');
insert into handles values(4,2,'2020-02-03 17:00:00');
select * from handles;
```

Assignment 4 Part II

Query 1

Code

```
SELECT medical_record.record_number, patient.p_name
FROM medical_record, patient
WHERE medical_record.patient_id = patient.patient_id
ORDER BY medical_record.patient_id ASC;
```

Output

	RECORD_NUMBER	P_NAME
1	1	Sara
2	3	David James
3	2	Joe Lad

Description

This is querying to show which patients have a medical record already created by showing their record number and patient name.

Query 2

Code

```
SELECT schedule.s_id, staff.staff_name
FROM schedule, staff
WHERE staff.emp_id = schedule.emp_id
ORDER BY schedule.s_id ASC;
```

Output

	S_ID	STAFF_NAME
1	1	John Doe
2	2	Jane Albert
3	3	John Doe

Description

This is querying to show which employees have a schedule made by showing their schedule id and the staff name.

Query 3

Code

```
SELECT billed.bill_id, claim.claim_number, patient.p_name
FROM billed, patient,claim
WHERE billed.patient_id = patient.patient_id AND claim.bill_id = billed.bill_id
ORDER BY billed.bill_id;
```

Output

	BILL_ID	CLAIM_NUMBER	P_NAME
1	1	2	Joe Lad
2	2	1	Sara

Description

This is querying for patients who have been billed and have made a claim on that bill by showing their bill id and patient name.

View 1

Code

```
Create VIEW
Male_Staff AS(
SELECT staff_name, gender
FROM staff
WHERE gender='male');
select * from male_staff;
```

Output

	STAFF_NAME	GENDER
1	John Doe	male
2	Michael Toon	male

Description

This view displays the name and gender of only the male staff members.

View 2

Code

```
Create VIEW Large_Salary AS(
SELECT staff_name, gender, salary
FROM staff
WHERE salary>80000);
select * from large_salary;
```

Output

	STAFF_NAME	GENDER	SALARY
1	John Doe	male	200000
2	Jane Albert	female	210000
3	Emily Blake	female	95000

Description

This view displays the staff members which have a salary that is greater than $80000 along with their gender.

View 3

Code

```
Create VIEW Qualify AS(
SELECT staff_name, salary, qulifications, gender
FROM staff
WHERE salary>90000 AND (qulifications='degree' OR qulifications='diploma'));
select * from qualify;
```

Output

	STAFF_NAME	SALARY	QULIFICATIONS	GENDER
1	John Doe	200000	degree	male
2	Jane Albert	210000	degree	female
3	Emily Blake	95000	diploma	female

Description

This view displays the staff members who have either a degree or diploma as their qualification, along with their salary and gender.

Assignment 5

Query 1

Code

```
#!/bin/sh
echo "set linesize 200
select room.room_number, count(staff.emp_id)
from staff,room
where exists
       (select uses.emp_id, governs.emp_id,equipment.room_number, governs.room_number
        from uses, governs, room, equipment
        where   uses.equipment_id = equipment.equipment_id
              AND governs.room_number =equipment.room_number
              AND ((staff.emp_id = uses.emp_id) OR (staff.emp_id = governs.emp_id)))
       group by room.room_number;
exit" | sqlplus64 -s  "USER/PASS@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"
```

Output

```
ROOM_NUMBER COUNT(STAFF.EMP_ID)
----------- -------------------
          1                   2
          2                   2
```

Description

This query is checking for how many employees are working in each of the rooms by showing the room number and amount of people in each room.

Query 2

Code

```
#!/bin/sh
echo "set linesize 200
SELECT MIN(salary), MAX(salary), AVG(salary)
From staff;
exit" | sqlplus64 -s  "USER/PASS@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"
```

Output

```
MIN(SALARY) MAX(SALARY) AVG(SALARY)
----------- ----------- -----------
      65000      210000      130000
```

Description

This is an aggregate function query that finds the minimum salary, maximum salary, and the average salary of all the staff members.

Query 3

Code

```
#!/bin/sh
echo "set linesize 200
SELECT equipment_id AS EQP_ID, emp_id AS STAFF_ID, staff_name AS STAFF_NAME
FROM equipment, staff
WHERE EXISTS
    (SELECT *
    FROM uses
    WHERE staff.emp_id = uses.emp_id);
exit" | sqlplus64 -s  "USER/PASS@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"
```

Output

```
    EQP_ID   STAFF_ID STAFF_NAME
---------- ---------- -------------------------
         1          3 Michael Toon
```

Description

This query identifies which staff member is using which equipment by listing the equipment id, staff id of the member using the equipment, and the staff member's name.

Query 4

Code

```
#!/bin/sh
echo "set linesize 200
SELECT patient.patient_id AS ID, p_name AS PeopleInToday
FROM patient
WHERE EXISTS
    (SELECT *
    FROM appointment
    WHERE patient.patient_id = appointment.patient_id
          AND app_date = '2020-09-05 12:00:00')
UNION
SELECT staff.emp_id, staff_name
FROM staff
WHERE EXISTS
    (SELECT *
    FROM schedule
    WHERE staff.emp_id = schedule.emp_id
          AND start_time = '2020-09-05 08:00:00');
exit" | sqlplus64 -s  "USER/PASS@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"
```

Output

	ID	PEOPLEINTODAY
1	2	David James
2	2	Jane Albert

Description

Shows the people coming into the hospital on a specific date. The quotes of the timestamp were not formatting correctly at the time of submission.

Query 5

Code

```
#!/bin/sh
echo "set linesize 200
(select staff_name from staff)
minus
(select staff_name from schedule, staff
where staff.emp_id = schedule.emp_id);
exit" | sqlplus64 -s  "USER/PASS@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"
```

Output

```
STAFF_NAME
--------------------------
Emily Blake
Lydia Chen
Michael Toon
```

Description

This query identifies the staff members who currently do not have a schedule.

Unix Shell Menu

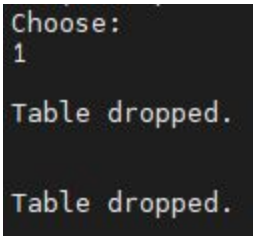
```
|                    Oracle All Inclusive Tool
|
|                       Main Menu - Select Desired Operation(s):
|
|        <CTRL-Z Anytime to Enter Interactive CMD Prompt>
|
------------------------------------------------------------
----
  M)  View Manual

  1)  Drop Tables
  2)  Create Tables
  3)  Populate Tables
  4)  Query Tables

  X)  Force/Stop/Kill Oracle DB

  E)  End/Exit
Choose:
```

Drop Table

Code

```
#!/bin/sh
echo "set linesize 200
drop table fake_table_1;
drop table fake_table_2;
exit" | sqlplus64 -s  "mratnaya/06273101@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"
```

Output

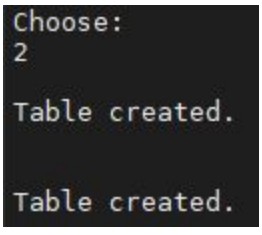


Create Table

Code

```
#!/bin/sh
echo "set linesize 200
create table fake_table_1(
fake1 number primary key,
fake2 varchar(5)
);
create table fake_table_2(
fake3 number primary key,
fake4 varchar(5)
);

exit" | sqlplus64 -s  "mratnaya/06273101@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"
```

Output



Populate Table

Code

```
#!/bin/sh
echo "set linesize 200
insert into fake_table_1 values(1, 'ABC');
insert into fake_table_1 values(2, 'DEF');
insert into fake_table_2 values(3, 'GHI');
insert into fake_table_2 values(4, 'JKL');
exit" | sqlplus64 -s  "mratnaya/06273101@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"
```

Output

```
Choose:
3

1 row created.


1 row created.


1 row created.


1 row created.
```

Query Table

Code

```
#!/bin/sh
echo "set linesize 200
select * from fake_table_1;
select * from fake_table_2;
exit" | sqlplus64 -s  "mratnaya/06273101@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))"
```

Output

```
Choose:
4

     FAKE1 FAKE2
---------- -----
         1 ABC
         2 DEF


     FAKE3 FAKE4
---------- -----
         3 GHI
         4 JKL
```

Assignment 6

Staff

{emp_id} → staff_name, gender, salary, qualifications, phone_number, address, special

Billed

{bill_id} → patient_id, procedure_id, medicine_id, app_id

Appointment

{a_id} → app_date, patient_id

Patient

{patient_id} → p_name, gender, admission_date, discharge_date, address, phone_number

Schedule

{s_id} → emp_id, start_time, end_time

Equipment

{equipment_id} → room_number, price, description

Medical_record

{record_number} → patient_id, patient_info

Medical_entry

{entry_number, record_number} → entry_number, record_number, entry_description

Medicine

{m_id} → price, dosage, m_name

Pro_cedure

{p_id} → p_type, price

Insurance

{insurance_number} → provider_name, phone_number, address

Claim

{claim_number} → bill_id

Room

{room_number} → room_type

Uses

{equipment_id, emp_id} → equipment_id, emp_id

Follow

{emp_id, s_id} → emp_id, s_id

Attends

{emp_id, patient_id} → emp_id, patient_id

Governs

{emp_id, room_number} → emp_id, room_number

Handles

{emp_id, appointment_id} → emp_id, appointment_id

Assignment 7

Normalizations

All tables listed below -excluding the Medical Record table which shows our normalizations process- have been normalized to the 3rd normal form.

Each table follows the first normal form as each table only has single (atomic) valued attributes and columns, the values stored in each column is of the same domain or type, each column in a table has a unique name, and the order in which the data is stored does not matter.

Each table follows 2nd normal form as the tables are already in 1st normal form, and there is no partial dependency in any table. Meaning there are no non-primary key attributes that are dependent on a subset of the primary key. As seen in our tables all attributes are dependent on the primary key and since we have no composite keys partial dependencies are almost impossible.

Each table follows 3rd normal form as the tables are already in 2nd normal form, and there are no transitive dependencies in any table. Meaning that all the attributes of a table are functionally dependent on solely the primary key. As seen in our tables each attribute depends only on the primary key of its table and there are no implicit dependencies where C depends on B which depends on A meaning C depends on A.

Normalization Process

Example of Step by Step Verification

The following table does not satisfy 1NF.

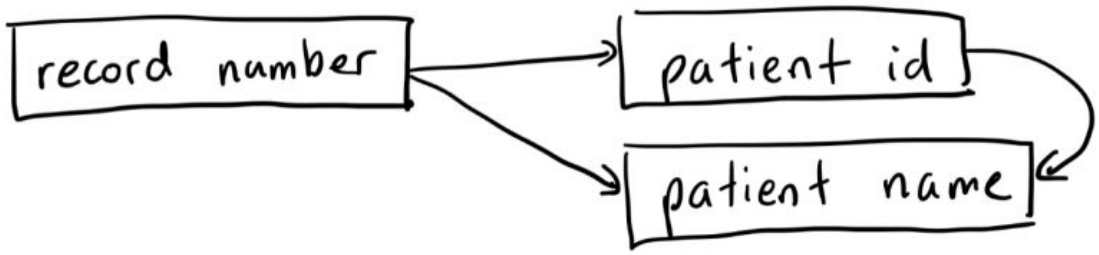
Medical Record			
record number	patient id	patient name	entries
1	1	Sara	check-up, broken nose
2	3	Joe Lad	check-up
3	2	David James	withdrawl

So we created separate tables for the entries attribute. This resulted in the Medical record table being 1NF since it has atomic values.

Medical Record		
record number	patient id	patient name
1	1	Sara
2	3	Joe Lad
3	2	David James

Medical Entries		
entry number	record number	description
1	1	check-up
2	1	broken nose
1	2	check-up
1	3	withdrawl

Checking the Medical Record table, every non-key attribute is fully functionally dependent on the primary key **record number** as shown in the diagram below.

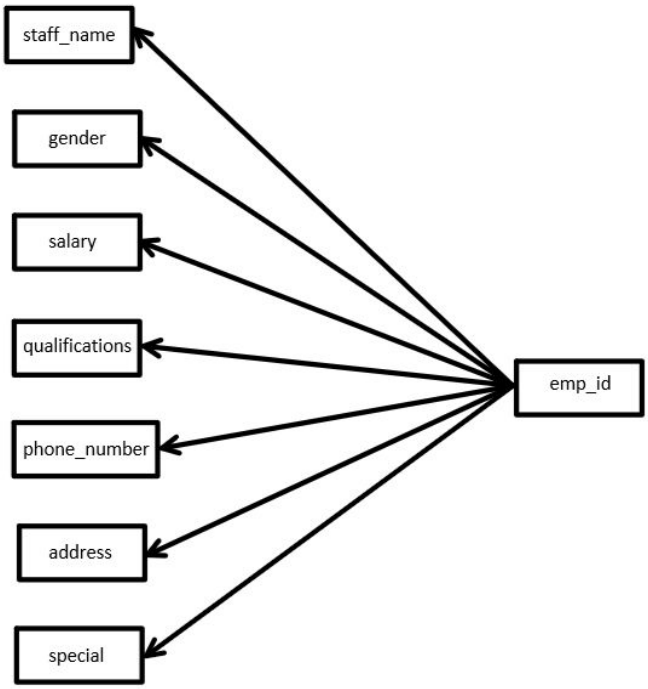


Checking for 3NF, since there is a transitive dependency on the primary key. To be specific, **patient name** is transitively dependent on the primary key **record number**, through **patient id**. Therefore we take out the **patient name** attribute because it can be found from the Patient table using the **patient id**. This results in a 3NF table as shown below with the table and diagram with all non-key attributes non-transitively dependent on the primary key.

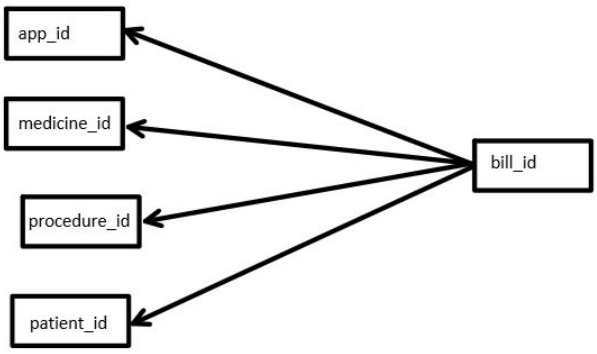
Medical Record	
record number	patient id
1	1
2	3
3	2

Final Tables

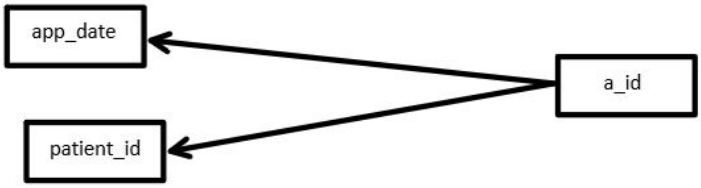
Staff



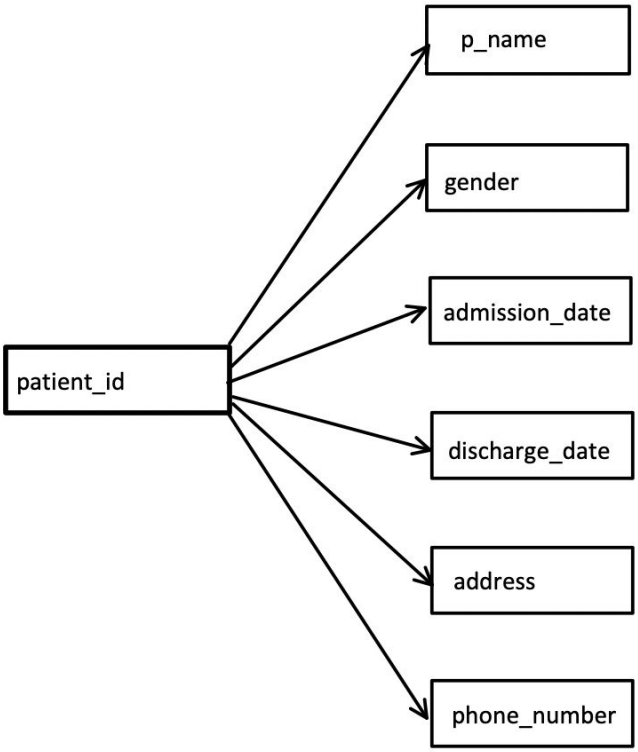
Billed



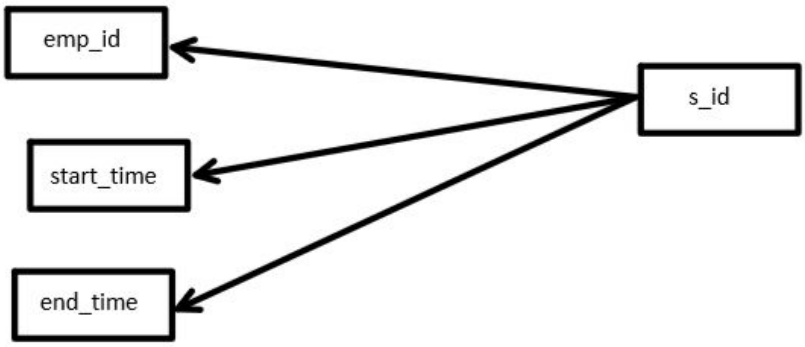
Appointment



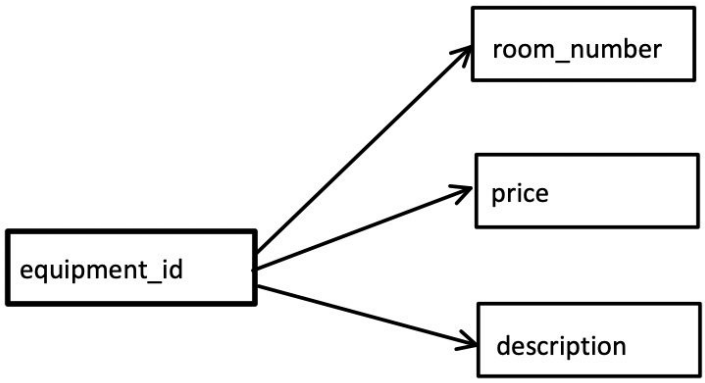
Patient



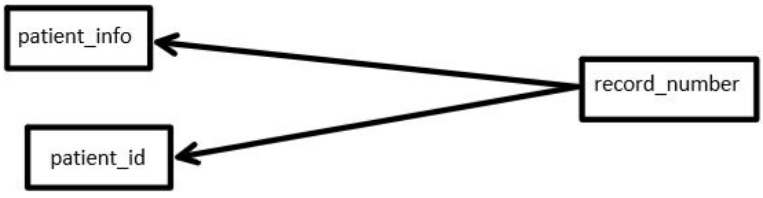
Schedule



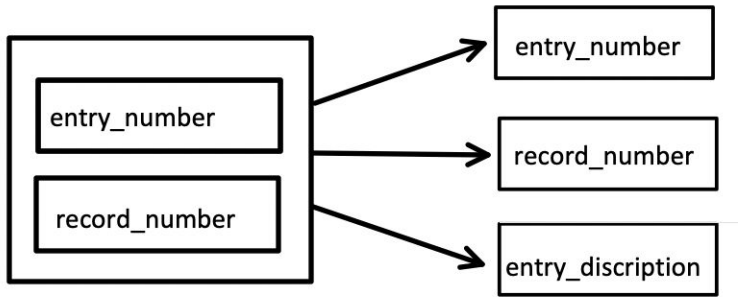
Equipment



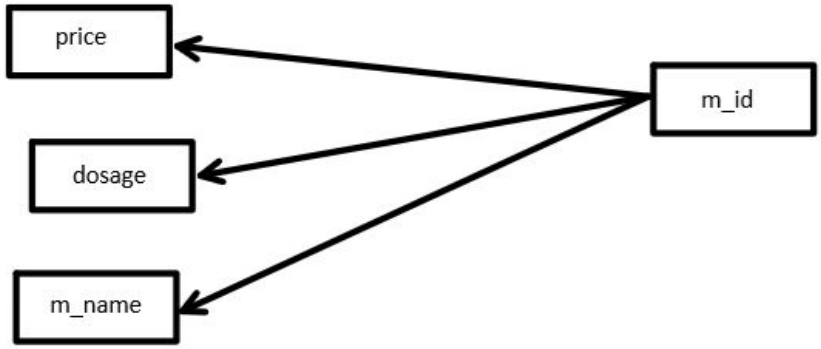
Medical_record



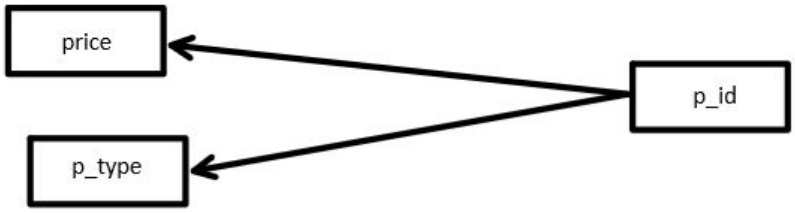
Medical_entry



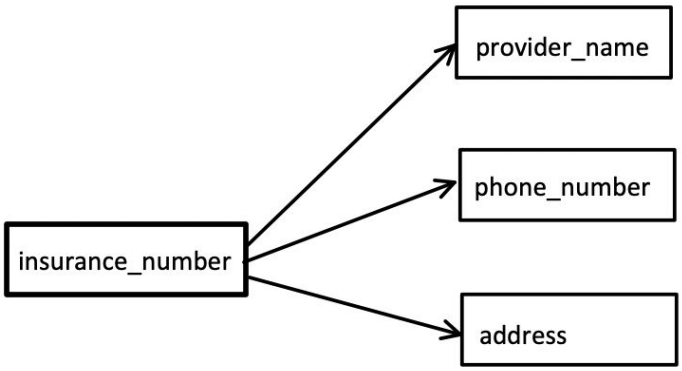
Medicine



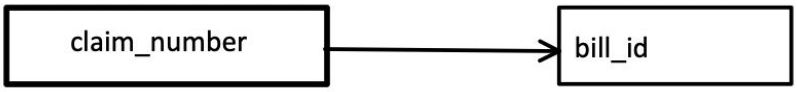
Pro_cedure



Insurance

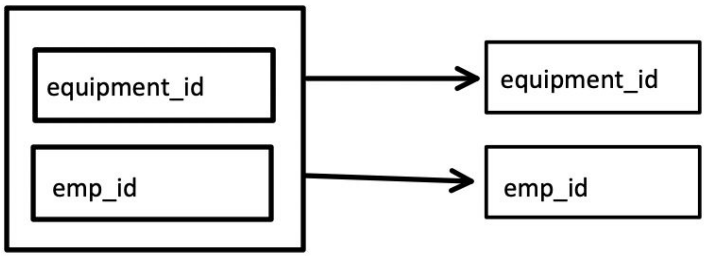


Claim

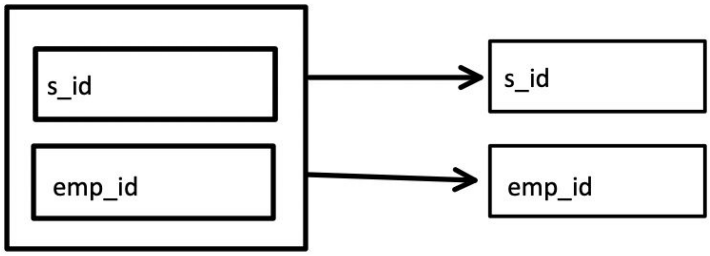


Room

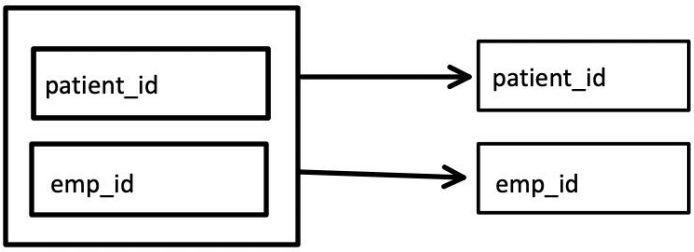
Uses



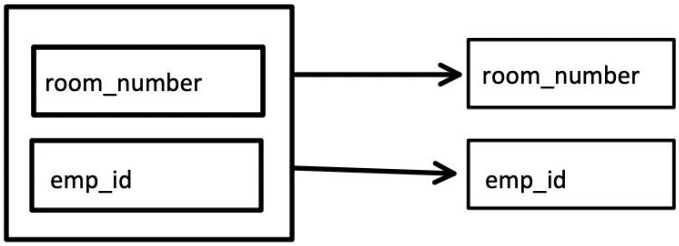
Follow



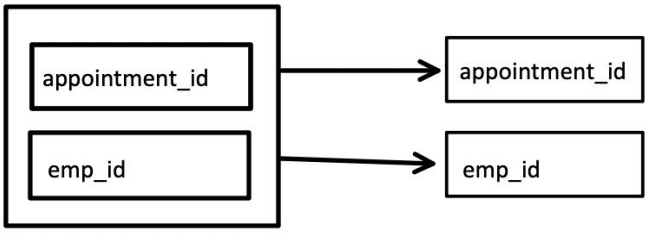
Attends



Governs



Handles



Assignment 8

Normalizations

All tables listed below -excluding the Medical Record table which shows our normalizations process- have been normalized to the 3rd normal form.

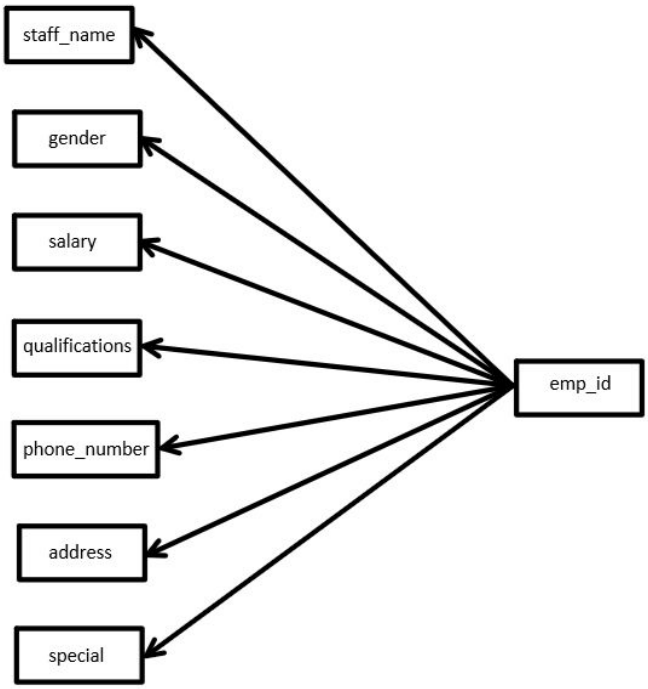
Each table follows the first normal form as each table only has single (atomic) valued attributes and columns, the values stored in each column is of the same domain or type, each column in a table has a unique name, and the order in which the data is stored does not matter.

Each table follows 2nd normal form as the tables are already in 1st normal form, and there is no partial dependency in any table. Meaning there are no non-primary key attributes that are dependent on a subset of the primary key. As seen in our tables all attributes are dependent on the primary key and since we have no composite keys partial dependencies are almost impossible.

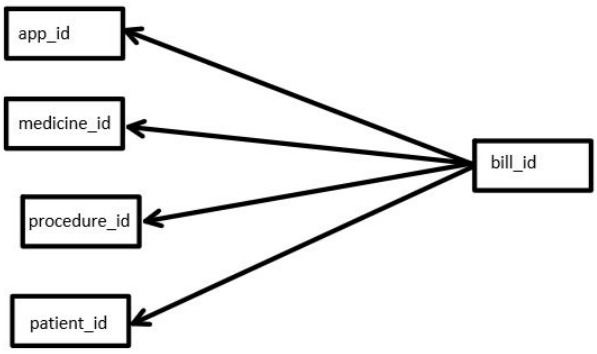
Each table follows 3rd normal form as the tables are already in 2nd normal form, and there are no transitive dependencies in any table. Meaning that all the attributes of a table are functionally dependent on solely the primary key. As seen in our tables each attribute depends only on the primary key of its table and there are no implicit dependencies where C depends on B which depends on A meaning C depends on A.

Final Tables

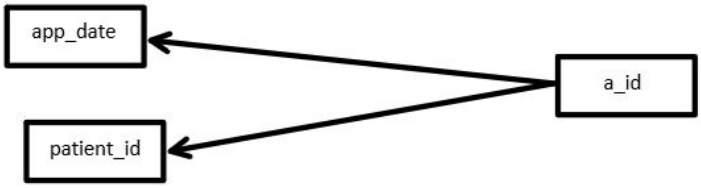
Staff



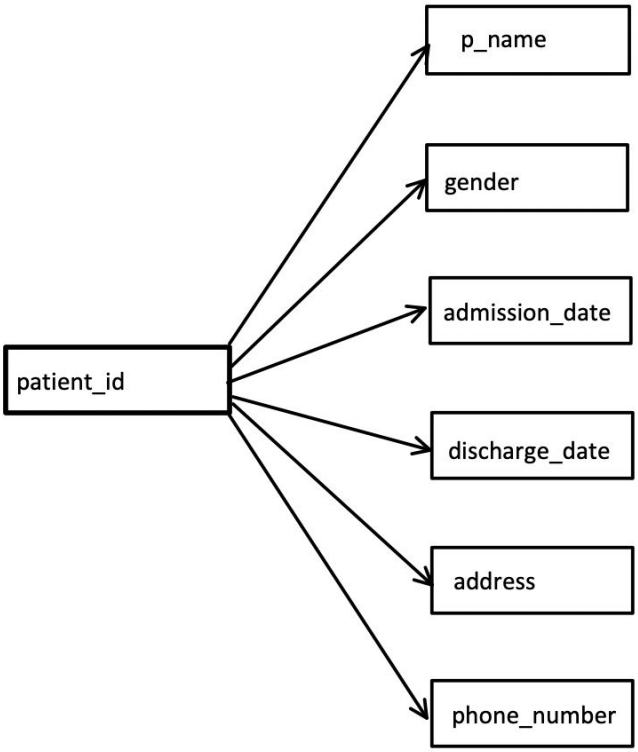
Billed



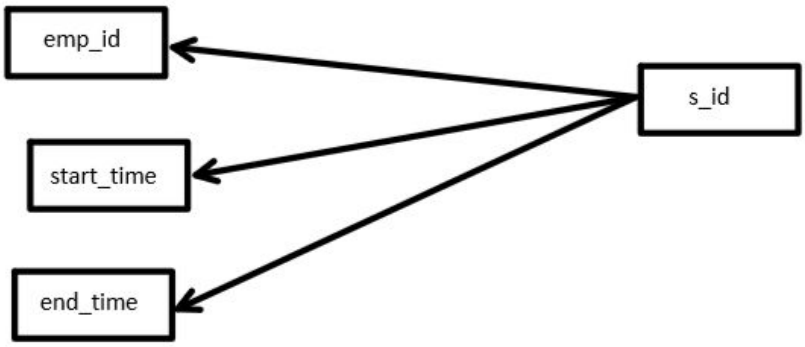
Appointment



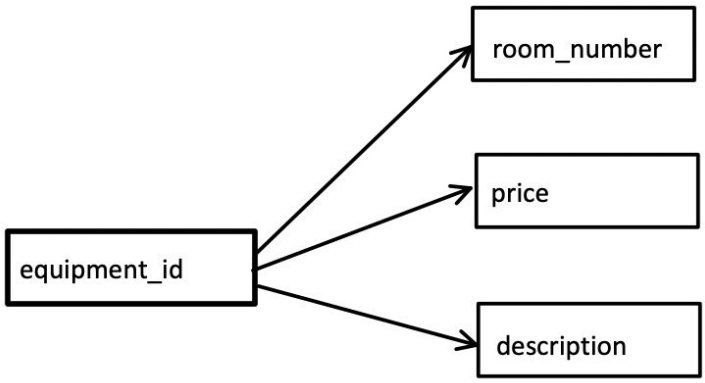
Patient



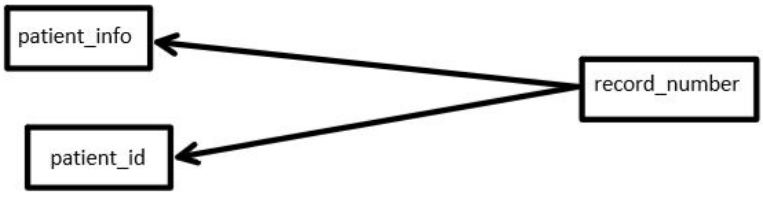
Schedule



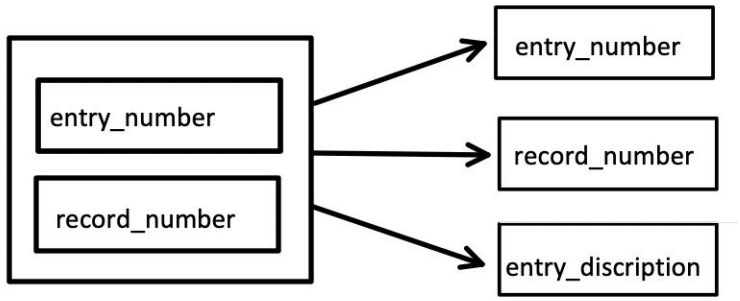
Equipment



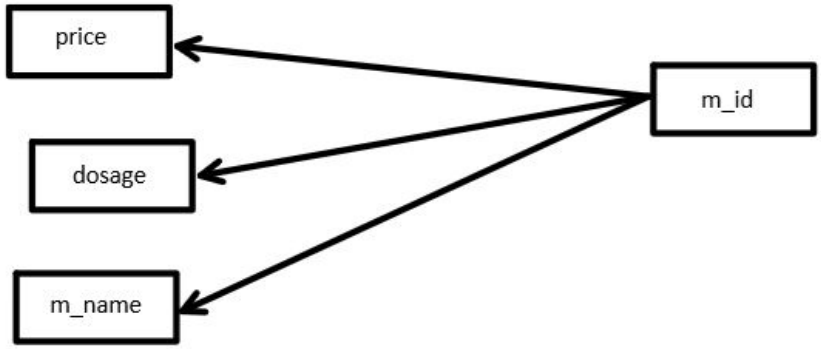
Medical_record



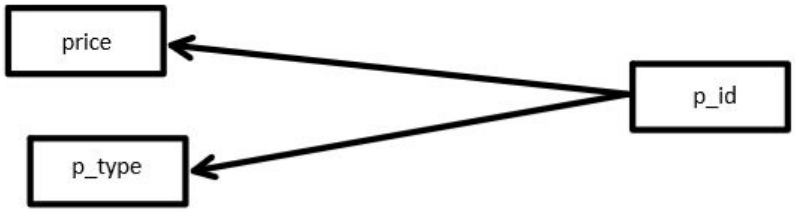
Medical_entry



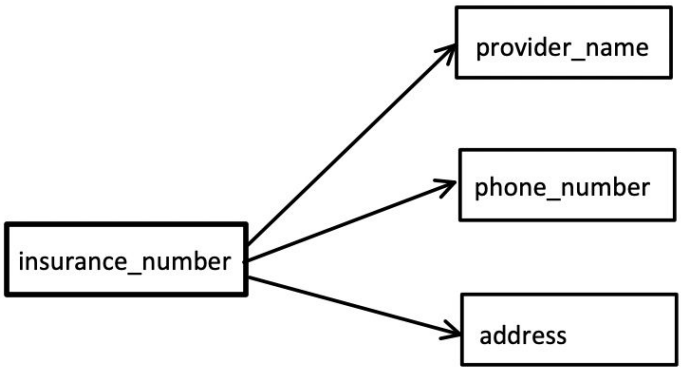
Medicine



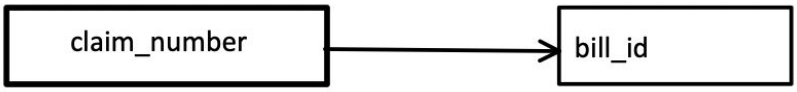
Pro_cedure



Insurance

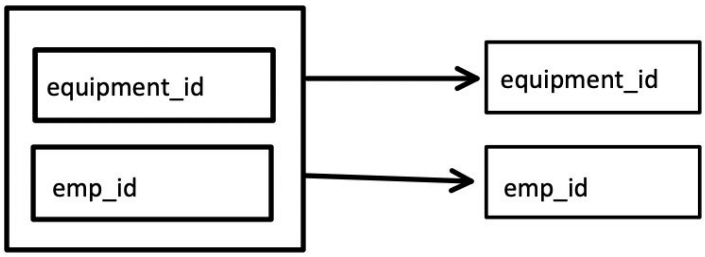


Claim

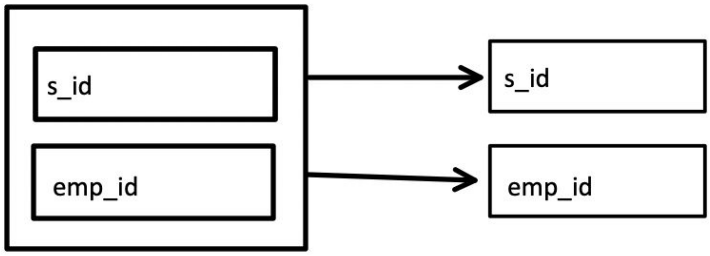


Room

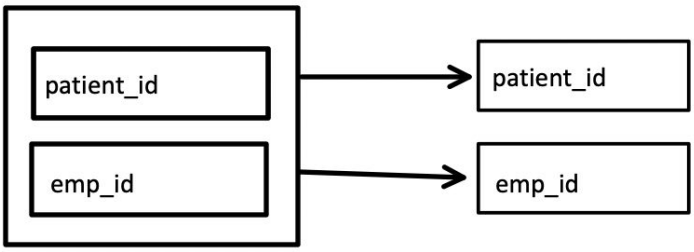
Uses



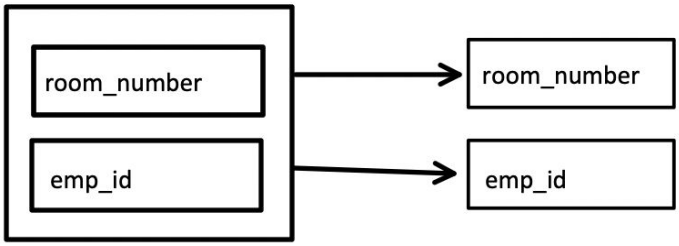
Follow



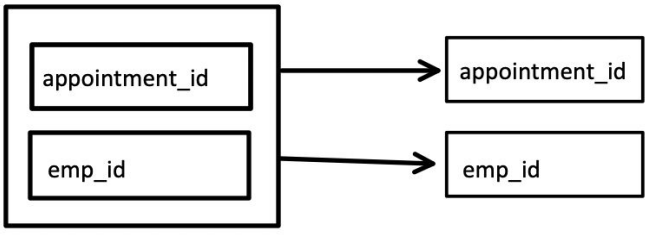
Attends



Governs



Handles



For Medical Record Table:

rn = record number, pid = patient id, pn = patient name, aid = appointment id
ad = appointment date, prid = procedure id, prprice = procedure id

Bernsteins Algorithm for 3NF

rnum	pid	pname	aid	adate	prid	prprice

mr(rn, pid, pn, aid, ad, prid, prp)

Step 1

FD = {rn → pid, pn, aid, ad, prid, prp; pid → pn; aid, prid → prp, ad

Step 2a

rn → pid : rn + = {rn, pn, ad, aid, prid, prp} not redundent ✓
rn → pn : rn + = {rn, pid, aid, ad, prid, prp, pn} redundent ✗
rn → aid : rn + = {rn, pid, pn, ad, prid, prp} ✓
rn → adate : rn + = {rn, pid, pn, aid, prid, prp, ad} ✗
rn → prid : rn + = {rn, pid, pn, aid, ad, prp} ✓
rn → prp : rn + = {rn, pid, pn, aid, prid, prp, ad} ✗
pid → pn : pid + = {pid} ✓
prid, aid → ad : prid, aid + = {prid, aid} ✓
prid, aid → prp : prid, aid + = {prid, aid} ✓

Step 2b

prid, aid → adate prid + = {prid}, aid + = {aid} Do not get a date So Fully dependent

prid, aid → prp prid + = {prid}, aid + = {aid} Do not get a date So Fully dependent

Step 3

- rn always part of pkey
- pn, ad, prp never part of pkey
- nothing is neither on the left or right side

Therefore:

rn + = {rn, pid, pn, aid, ad, prid, prp} This is the key

Step 4

rn ⟶ { pid , pn , aid , ad, prid, prp} : R_1 (rn , pid , pn , aid , ad , prid , prp)

pid ⟶ { pn } : R_2 (pid , pn)

prid , aid ⟶ { ad , prp } R_3 (prid , aid , ad , prp)

∴ There is a key in R1 which is rn

BCNF Algorithm

rnum	pid	pname	aid	adate	prid	prprice

mr(rn, pid, pn , aid , ad , prid , prp)

rn ⟶ {pid, pn , aid , ad , prid , prp ; pid}

pid ⟶ {pn}

aid , prid ⟶ {ad, prp}

rn += {rn, pin, pn, aid, ad, prid, prp } is a key

pid += { pid , pn} is not a key so MR is not a BCNF with respect to pid → {pn}

MR11 = {rn, aid , ad, prid, prp, pid }

MR21 = { Pid , Pn} which is BCNF

FD11 = { rn → pid, pn , aid , ad , prid , prp ; pid ; aid , prid → ad, prp }

rn += {pid, pn , aid , ad , prid , prp ; pid } is a key

aid , prid += { aid , ad , prid , prp } is not a key so split MR11 to MR111 and MR112

MR111 = (aid, prid, rn , pid) is BCNF

MR112 = (aid, ad, prid , prp) is BCNF

Final BCNF schema for MR is:

R_1 = (Pid , Pn) MR12

R_2 = (aid , ad , prid , prp) MR112

R_3 = (aid , prid , rn , pid) MR111

Assignment 9

Submitted Files.

Assignment 10

Relational Algebra

Query 1

$$\Pi_{staff\_name,\ emp\_id,\ gender,\ salary,\ qulifications,\ phone\_number,\ address,\ special}\ (staff)$$

Query 2

$$\Pi_{p\_name,\ patient\_id,\ gender,\ admission\_date,\ discharge\_date,\ address,\ phone\_number}\ (patient)$$

Query 3

$$\Pi_{emp\_id,\ patient\_id}\ (attends)$$

Query 4

$$\Pi_{record\_number,\ patient\_id,\ patient\_info}\ (medical\_record)$$

Query 5

$$\Pi_{entry\_number,\ record\_number,\ entry\_description}\ (medical\_entry)$$

Query 6

$$\Pi_{record\_number,\ emp\_id}\ (manages)$$

Query 7

$$\Pi_{room\_number,\ room\_type}\ (room)$$

Query 8

$$\Pi_{equipment\_id,\ room\_number,\ price,\ description}\ (equipment)$$

Query 9

$\Pi_{equipment\_id,\ emp\_id}\ (uses)$

Query 10

$\Pi_{s\_id,\ emp\_id,\ start\_time,\ end\_time}\ (schedule)$

Query 11

$\Pi_{emp\_id,\ s\_id}\ (follow)$

Query 12

$\Pi_{a\_id,\ app\_date,\ patient\_id}\ (appointment)$

Query 13

$\Pi_{emp\_id,\ appointment\_id}\ (handles)$

Query 14

$\Pi_{emp\_id,\ room\_number}\ (governs)$

Query 15

$\Pi_{p\_id,\ p\_id,\ price}\ (pro\_cedure)$

Query 16

$\Pi_{m\_id,\ price,\ dosage,\ m\_name}\ (medicine)$

Query 17

$\Pi_{bill\_id,\ patient\_id,\ procedure\_id,\ medicine\_id,\ app\_id}\ (billed)$

Query 18

$\Pi_{claim\_number,\ bill\_id}\ (claim)$

Query 19

$\Pi_{provider\_name,\ insurance\_number,\ phone\_number,\ address}\ (insurance)$

Query 20

$$\Pi_{record\_number(medical\_record),\ p\_name(patient)}\ (\sigma_{patient\_id(medical\_record) = patient\_id(patient)}\ (medical\_record,\ patient))$$

Query 21

$$\Pi_{s\_id(schedule),\ staff\_name(staff)}\ (\sigma_{emp\_id(schedule) = emp\_id(staff)}\ (schedule,\ staff))$$

Query 22

$$\Pi_{bill\_id(billed),\ claim\_number(claim),\ p\_name(patient)}\ (\sigma_{patient\_id(billed) = patient\_id(patient)\ AND\ patient\_id(billed) = patient\_id(claim)}(billed,\ patient,\ claim))$$

Concluding Remarks

This patient management allows the users such as hospitals and medical centers to efficiently organize and manage patient information, including name, address, medical record, as well as manage the staff and their tasks. This database fully encompases the required information that medical facilities require for their operation and uses all of the SQL database fundamentals and concepts taught in this course such as, normalization, 3rd NF, and BCNF. Overall this has been a very valuable design experience for the team and has helped further our knowledge and understanding of database systems.