WolfHospital Management System

For WolfHospital

CSC 440 Database Management Systems
Project Report 2

Team A
Renata Zeitler, Andy Zhang, Nicole Hlebak, Alex Sawyer
October 18th, 2017

Assumptions:

1. Nurses, Receptionists, and Admins do not have professional titles.
2. We are not going to check duplicate SSN for patients.
3. No patient will start multiple treatments on the same day- only the day/month/year is recorded.
4. Patients can reserve a specific ward/bed, but they must tell the receptionist at check-in.
5. To check into a ward, SSN is not required since it is optional patient data; however, patientID is.
6. Each type of fee is the same for every patient (for example, a registration fee is always the same amount).
7. The nurse in charge of each ward attends to the patients in the ward. Nurses do not attend to or treat patients outside of the ward they are assigned to.
8. Only one doctor will be performing a test.
9. Patients are limited to one ward/bed per visit; this will not change throughout their visit.
10. A test's name will not determine its cost, as variations from test to test could cause a fluctuation in the test's price.
11. A reserved ward for a patient will be given a start date that is in the future, for when the patient is supposed to arrive.
12. The ward usage percentage represents the current percentage of the word that is occupied.
13. The operation "report the medical history for a given patient and for a certain time period (month/year)" was interpreted as give a patient's medical history for a certain amount of time.

1. Global Relational Database Schema:

StaffMembers(<u>StaffID</u>, Name, Age, Gender, Department, Phone, Address)

StaffID → StaffID, Name, Age, Gender, Department, Phone, Address holds because each StaffID is unique, and identifies an individual that has a name, age, gender, department, phone number, and address. If we were to try and take any combination of other attributes, it would severely limit the possibilities our database could have. For example, many people have the same age, or two coworkers could be roommates and have the same phone number and address. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

Doctors(ProfessionalTitle, <u>StaffID</u>)

StaffID → StaffID, ProfessionalTitle holds since this relation only has two attributes, it is in BCNF and therefore 3NF. Professional title cannot be used as a key because multiple doctors can have the same title, but it cannot indicate a doctor's unique staff ID.

Nurses(<u>StaffID</u>)

StaffID → StaffID is in 3NF because it is in BCNF, as there is only one attribute in this functional dependency, and the one attribute is the key.

Receptionists(<u>StaffID</u>)

StaffID → StaffID is in 3NF because it is in BCNF, as there is only one attribute in this functional dependency, and the one attribute is the key.

Admins(<u>StaffID</u>)

StaffID → StaffID since this relation contains only one attribute, which is the key, it is in BCNF and therefore 3NF.

Patients(<u>PatientID</u>, SSN, Name, Dob, Gender, Phone, Addr, Status)

PatientID → PatientID, SSN, Name, Dob, Gender, Phone, Addr, Status holds because each patient with all of their respective attributes has a unique patient ID (PatientID).

SSN → PatientID, SSN, Name, Dob, Gender, Phone, Addr, Status holds because an SSN, if present, maps to exactly one patient and therefore determines all of their attributes.

Because the left hand side of each of these functional dependencies is a superkey, this relation is in BCNF and therefore 3NF. There are no other functional dependencies because no combination of the other attributes is sufficient to determine a unique patient. It's possible for two different patients to have identical values for each of these attributes.

MedicalRecords(PatientID, StartDate, EndDate, Treatment, Diagnosis, Prescription)

PatientID, StartDate → PatientID, StartDate, EndDate, Treatment, Diagnosis, Prescription holds because each medical record with all their respective attributes is unique for a patient id and start date. PatientID and StartDate together are a key and determine all the other attributes.

Because the left-hand side of this is a superkey, this relation is in BCNF and therefore 3NF. There are no other functional dependencies because there are no other combinations of the other attributes that is sufficient to determine a unique medical record.

BillingAccounts(BID, PayerSSN, payerAddr, PaymentMethod, CardNumber, Insurance)

BID → BID, PayerSSN, payerAddr, PaymentMethod, CardNumber, Insurance holds because the each billing account with all its attributes is unique to a specific billing id. Since the left-hand side is a superkey, this relation is in BCNF and therefore in 3NF. There are no other combination of attributes that are sufficient to determine a unique billing account. It is possible for patients to have identical values for all of the other attributes.

BilledFor(AccountID, StartDate, PatientID)

AccountID, StartDate, PatientID → AccountID, StartDate, PatientID holds because the relationship “billed for” is connected to two entities, one which is a weak entity set. This relationship requires the keys for all entities it is connected to, which would be the three we have listed. Without any of the three listed in the key, we can not appropriately describe the relationship; for instance, if the PatientID is not listed then we cannot identify who the billing account is associated with. Because the left hand side contains all attributes, it is in BCNF (and thereby 3NF).

Fees(FeeID, Amount, Description)

FeeID → FeeID, Amount, Description is true since each fee has a unique id (FeeID) that determines the amount and the description.

This relation is in BCNF and therefore 3NF because the left hand side of the only functional dependency is a superkey. Amount → FeeID and Amount → Description do not hold since two different fees may be charged the same amount. Description → FeeID and Description → Amount do not hold because two different fees may have the same description. Amount, Description → FeeID does not hold because two different fees may have the same amount and have the same description.

Owes(FeeID, BillingAccountID)

FeeID, BillingAccountID → FeeID, BillingAccountID is in BCNF and therefore 3NF because it only contains two attributes, which are in the superkey.

Ward(<u>WardNumber</u>, ResponsibleNurseID, Capacity, ChargesPerDay)

WardNumber → WardNumber, ResponsibleNurseID, Capacity, ChargesPerDay holds because a ward number is used to uniquely identify a ward; no other attributes can be used in combination to do this since multiple wards can charge the same amount per day, or hold the same capacity. Because the left hand side is a super key, it is in BCNF (and thereby 3NF).

PatientResidesIn(BedNumber, <u>StartDate</u>, <u>PatientID</u>, <u>WardNumber</u>)

StartDate,PatientID, WardNumber → StartDate, PatientID, WardNumber, BedNumber holds because StartDate, PatientID, and WardNumber make up the key for this relation.

WardNumber, BedNumber → StartDate, PatientID, WardNumber, BedNumber holds because only one patient may occupy a given bed in a given ward.

Since all left-hand sides are superkeys, this relation is in BCNF and therefore in 3NF. BedNumber → PatientID does not hold because there could be a bed by the same number in another ward that would be holding another patient. BedNumber → WardNumber does not hold because there can be a bed by the same number in multiple wards. BedNumber → StartDate does not hold because there can be the same BedNumber with different StartDates (one person is in the bed at one time, then leaves and the same bed is filled by another person at another time).

Treats(<u>DoctorID</u>, <u>PatientID</u>)

DoctorID, PatientID → DoctorID, PatientID holds because there are only two attributes and are both in the super key, so it is in BCNF and therefore 3NF.

Tests(<u>TestID</u>, Name, Results, Cost, StartDate, PatientID, DoctorID)

TestID → TestID, Name, Results, Cost, StartDate, PatientID, DoctorID holds within 3NF because each test will have unique test id,which will be used to identify the rest of the attributes of that test, which makes TestID a key, and a superkey. Since the left hand side of the FD is a superkey, then this FD holds in 3NF. None the other attributes within this schema will be used to identify any of the other attributes.

Name → does not hold because tests name will not determine the other attributes
Results → Does not hold because there can be different tests with the same results.
Cost → Does not hold because there could could be multiple tests that cost the same.
StartDate → Does not hold because multiple tests could have the same start date.
PatientID → does not hold because a a patient ID could not be used to determine the other attributes of a test
DoctorID- does not hold because a doctor ID could not be used to determine the other attributes of a test.

2. Design for Global Schema:

Design decision for global schema:

The entity sets in our diagram were made into relations, with their same respective attributes for StaffMembers, Patients, Fees, and BillingAccounts. The entity sets that are subsets of StaffMembers were made into relations based on the E/R viewpoint to avoid redundancy and save table space.

We combined the many-one relationships into attributes by making the key of the *one* as an attribute of the *many*. This reduces redundancy and decreases the overhead that many tables cause. It also makes queries quicker. (Note that the many-one relationship PatientResidesIn was made its own relation because it has an attribute).

Other relationships have each been turned into relations in our schema. Their attributes in the schema are the keys of the entities they represent. BilledFor and PatientResidesIn also have the additional attribute of PatientID, which they get from their connection to MedicalRecords.

StaffMembers(StaffID, Name, Age, Gender, Department, Phone, Address)

StaffID is the primary key

Name, age, gender, department, phone, and address are not allowed to be null

Doctor(ProfessionalTitle, StaffID)

StaffID is the primary key

Professional Title is not allowed to be null

Nurse(StaffID)

StaffID is the primary key

Receptionist(StaffID

StaffID is the primary key

Admin(StaffID)

StaffID is the primary key

Patients(<u>PatientID</u>, SSN, Name, Dob, Gender, Phone, Addr, Status)

PatientID is the primary key

SSN is allowed to be null, in case a patient did not wish to disclose their SSN. A null value in this case would represent a non disclosed SSN.

Name, DOB, gender, phone, addr, and status are not allowed to be null.

MedicalRecords(<u>PatientID</u>, <u>StartDate</u>, EndDate, Treatment, Diagnosis, Prescription)

PatientID is one of two primary keys

StartDate is the second of two primary keys

EndDate is allowed to be null, as ongoing treatment for a patient would mean there is no possible end date for the patient. Therefore, a null end date means that the patient is still receiving treatment in the hospital.

Treatment is allowed to be null, as a patient could be checked into the hospital, but has not yet received any care. Therefore, a null treatment means that the patient has not yet received any care.

Diagnosis is allowed to be null, as a patient could have received treatment and tests, but their illness is yet unknown. Therefore, a null diagnosis means a diagnosis has not yet been concluded for the patient.

Prescription is allowed to be null, as a patient might not need any prescription, or is still undergoing treatment, and does not need one yet. Therefore, a null prescription signifies a patient does not require any prescription at this time.

BillingAccounts(<u>BID</u>, PayerSSN, payerAddr, PaymentMethod, CardNumber, Insurance)

BID is the primary key

PayerSSN, payerAddr,and PaymentMethod are not allowed to be null.

Insurance is allowed to be null, in case a patient, or the person who is paying for the patient's treatment, does not have insurance, and is paying out of pocket. A null value in this case would represent that no insurance company was involved with the payment for this patient. CardNumber can be null, for instance, if the PaymentMethod is cash.

BilledFor(<u>AccountID</u>, <u>StartDate</u>, <u>PatientID</u>)

AccountID is one of three primary keys

StartDate is the second of three primary keys

PatientID is the third of three primary keys

Fees(<u>FeeID</u>, Amount, Description)

FeeID is the primary key

Amount and Description are not allowed to be null

Owes(<u>FeeID</u>, <u>BillingAccountID</u>)

FeeID is one of two primary keys

BillingAccountID is the second of two primary keys

Ward(<u>WardNumber</u>, ResponsibleNurseID, Capacity, ChargesPerDay)

WardNumber is the primary key
Capacity, and ChargesPerDay are not allowed to be null
ResponsibleNurseID is allowed to be null, in the event that a ward does not yet have a nurse assigned to it (this could be in the event that the ward is temporarily closed. In this case, we still would want to retain the other information about the ward even if there is no responsible nurse).

PatientResidesIn(BedNumber, <u>StartDate</u>, <u>PatientID</u>, <u>WardNumber</u>)

StartDate is one of three primary keys
PatientID is the second of three primary keys
WardNumber is the third of three primary keys
BedNumber is not allowed to be null
The combination BedNumber and WardNumber must be unique

Treats(<u>DoctorID</u>, <u>PatientID</u>)

DoctorID is one of three primary keys
PatientID is the second of three primary keys, not null

Tests(<u>TestID</u>, Name, Results, Cost, StartDate, PatientID, DoctorID)

TestID is the primary key
Name, Cost, StartDate, PatientID, and DoctorID are not allowed to be null.
Results is allowed to be null, as a test could have been entered into the system, but the results of that test are not yet available. Therefore, a null results means that the results of that test have not been returned yet.

3. Base Relations:

```
CREATE TABLE StaffMembers (
      StaffID    INT,
      Name       VARCHAR(128) NOT NULL,
      Age        INT          NOT NULL,
      Gender     VARCHAR(16)  NOT NULL,
      Department VARCHAR(128) NOT NULL,
      Phone      VARCHAR(16)  NOT NULL,
      Address    VARCHAR(128) NOT NULL,
      PRIMARY KEY(StaffID)
);
```

```
CREATE TABLE Doctors(
      StaffID           INT,
      ProfessionalTitle VARCHAR(128) NOT NULL,
      PRIMARY KEY(StaffID),
      FOREIGN KEY(StaffID) REFERENCES StaffMembers(StaffID)
            ON UPDATE CASCADE
);

CREATE TABLE Receptionists(
      StaffID            INT,
      PRIMARY KEY(StaffID),
      FOREIGN KEY(StaffID) REFERENCES StaffMembers(StaffID)
            ON UPDATE CASCADE
);

CREATE TABLE Nurses(
      StaffID      INT,
      PRIMARY KEY(StaffID),
      FOREIGN KEY(StaffID) REFERENCES StaffMembers(StaffID)
            ON UPDATE CASCADE
);

CREATE TABLE Admins(
      StaffID            INT,
      PRIMARY KEY(StaffID),
      FOREIGN KEY(StaffID) REFERENCES StaffMembers(StaffID)
            ON UPDATE CASCADE
);

CREATE TABLE Patients (
      PatientID INT,
      SSN       VARCHAR(16),
      Name      VARCHAR(128) NOT NULL,
      BirthDate DATE         NOT NULL,
      Gender    VARCHAR(16)  NOT NULL,
      Phone     VARCHAR(16)  NOT NULL,
      Address   VARCHAR(128) NOT NULL,
      Status    VARCHAR(128) NOT NULL,
      PRIMARY KEY(PatientID)
);
```

```
CREATE TABLE MedicalRecords (
      PatientID    INT,
      StartDate    DATE,
      EndDate      DATE,
      Treatment    VARCHAR(128),
      Diagnosis    VARCHAR(128),
      Prescription VARCHAR(128),
      PRIMARY KEY(PatientID, StartDate),
      FOREIGN KEY(PatientID) REFERENCES Patients(PatientID)
            ON UPDATE CASCADE
);

CREATE TABLE BillingAccounts(
      AccountID     INT,
      PayerSSN      VARCHAR(16)  NOT NULL,
      PayerAddress  VARCHAR(128) NOT NULL,
      PaymentMethod VARCHAR(128) NOT NULL,
      CardNumber    VARCHAR(20),
      Insurance     VARCHAR(128),
      UNIQUE(PayerSSN),
      PRIMARY KEY(AccountID)
);

CREATE TABLE BilledFor(
      AccountID INT,
      PatientID INT,
      StartDate DATE,
      PRIMARY KEY(AccountID, PatientID, StartDate),
      FOREIGN KEY(AccountID)
            REFERENCES BillingAccounts(AccountID)
            ON UPDATE CASCADE,
      FOREIGN KEY(PatientID, StartDate)
            REFERENCES MedicalRecords(PatientID, StartDate)
            ON UPDATE CASCADE
);

CREATE TABLE Fees(
      FeeID       INT,
      Amount      DECIMAL(9,2),
      Description VARCHAR(128),
      PRIMARY KEY(FeeID)
);
```

```
CREATE TABLE Owes(
      FeeID     INT,
      AccountID INT,
      PRIMARY KEY(FeeID, AccountID),
      FOREIGN KEY(FeeID)
            REFERENCES Fees(FeeID)
            ON UPDATE CASCADE,
      FOREIGN KEY(AccountID)
            REFERENCES BillingAccounts(AccountID)
            ON UPDATE CASCADE
);

CREATE TABLE Treats(
      DoctorID    INT,
      PatientID   INT,
      Primary Key(DoctorID, PatientID),
      FOREIGN KEY(DoctorID)
            REFERENCES Doctors(StaffID)
            ON UPDATE CASCADE,
      FOREIGN KEY(PatientID)
            REFERENCES Patients(PatientID)
            ON UPDATE CASCADE
);

CREATE TABLE Wards(
      WardNumber          INT,
      Capacity            INT,
      ChargesPerDay       DECIMAL(9,2),
      ResponsibleNurseID  INT,
      PRIMARY KEY(WardNumber),
      FOREIGN KEY(ResponsibleNurseID)
            REFERENCES Nurses(StaffID)
            ON UPDATE CASCADE
);

CREATE TABLE Tests(
      TestID      INT,
      Name        VARCHAR(128) NOT NULL,
      Results     VARCHAR(128),
      Cost        DECIMAL(9,2) NOT NULL,
      StartDate   DATE         NOT NULL,
      PatientID   INT          NOT NULL,
```

```
	DoctorID    INT          NOT NULL,
	PRIMARY KEY(TestID),
	FOREIGN KEY(PatientID, StartDate)
		REFERENCES MedicalRecords(PatientID, StartDate)
		ON UPDATE CASCADE,
	FOREIGN KEY(DoctorID)
		REFERENCES Doctors(StaffID)
		ON UPDATE CASCADE
);

CREATE TABLE PatientResidesIn(
	StartDate   DATE,
	PatientID   INT,
	WardNumber  INT,
	BedNumber   INT,
	PRIMARY KEY(StartDate, PatientID, WardNumber, BedNumber),
	UNIQUE(WardNumber, BedNumber),
	FOREIGN KEY(PatientID, StartDate)
		REFERENCES MedicalRecords(PatientID, StartDate)
		ON UPDATE CASCADE,
	FOREIGN KEY(WardNumber)
		REFERENCES Wards(WardNumber)
		ON UPDATE CASCADE
);

SELECT * FROM StaffMembers;
```

StaffID	Name	Age	Gender	Department	Phone	Address
1	Renata	22	Female	3rd Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
2	Andy	52	Male	1st Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
3	Nicole	34	Female	2nd Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
4	Alex	102	Male	1st Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
5	Ann	22	Female	3rd Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
6	April	18	Female	1st Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
7	Leslie	34	Female	2nd Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
8	Ron	60	Male	1st Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
9	Tom	27	Male	3rd Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
10	Jerry	64	Male	1st Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
11	Donna	39	Female	2nd Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
12	Jean Ralphio	23	Male	1st Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
13	Ben	22	Female	3rd Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
14	Chris	18	Male	1st Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
15	Tina	34	Female	2nd Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604
16	Michael	60	Male	1st Floor	(919) 867-5309	3124 Address Lane, Raleigh, NC, 27604

SELECT * FROM Doctors;

StaffID	ProfessionalTitle
1	Neurosurgeon
2	Rheumatologist
3	Pediatrician
4	Neurosurgeon

SELECT * FROM Receptionists;

StaffID
5
6
7
8

SELECT * FROM Nurses;

StaffID
9
10
11
12

SELECT * FROM Admins;

StaffID
13
14
15
16

SELECT * FROM Patients;

PatientID	SSN	Name	BirthDate	Gender	Phone	Address	Status
1	123-45-6789	Rob Gronkowski	1989-05-14	Male	(919) 999-9999	1234 Address Loop, MA, 01001	Checked In
2	123-45-6780	Philip Fry	1980-02-13	Male	(919) 999-9999	1234 Address Loop, MA, 01001	Processed
3	123-45-6781	Turanga Leela	1987-10-20	Female	(919) 999-9999	1234 Address Loop, MA, 01001	In Surgery
4	NULL	Peter Griffen	1972-08-31	Male	(919) 999-9999	1234 Address Loop, MA, 01001	Checked In

SELECT * FROM MedicalRecords;

PatientID	StartDate	EndDate	Treatment	Diagnosis	Prescription
1	2017-10-02	2017-10-07	NULL	NULL	NULL
1	2017-11-23	2017-12-12	Physical Therapy	Slow reflexes	Reflex enhancement
2	2017-10-11	2017-10-21	NULL	hemophiliac	Medicine
3	2017-10-11	NULL	Concussion therapy	Has a concussion	NULL

SELECT * FROM BillingAccounts;

AccountID	PayerSSN	PayerAddress	PaymentMethod	CardNumber	Insurance
200	111-11-1111	1234 Drive Lane, Raleigh, NC, 27602	Credit Card	1234 1234 1234 1234	BCBS
201	222-22-2222	1234 Drive Lane, Raleigh, NC, 27602	Cash	NULL	BCBS
202	333-33-3333	1234 Drive Lane, Raleigh, NC, 27602	Cash	NULL	NULL
203	222-22-2222	1234 Drive Lane, Raleigh, NC, 27602	Check	NULL	BCBS

SELECT * FROM BilledFor;

AccountID	PatientID	StartDate
200	1	2017-11-23
201	1	2017-10-02
202	2	2017-10-11
203	3	2017-10-11

SELECT * FROM Fees;

FeeID	Amount	Description
100	5.00	registration
101	25.00	missed previous appointment
102	20.00	accommodation fee
103	10.00	consultation fee

SELECT * FROM Owes;

FeeID	AccountID
100	200
101	201
102	202
103	203

```
SELECT * FROM Treats;
+----------+-----------+
| DoctorID | PatientID |
+----------+-----------+
|        1 |         1 |
|        2 |         2 |
|        3 |         3 |
|        4 |         4 |
+----------+-----------+

SELECT * FROM Wards;
+------------+----------+--------------+--------------------+
| WardNumber | Capacity | ChargesPerDay | ResponsibleNurseID |
+------------+----------+--------------+--------------------+
|          1 |        1 |         50.00 |                  9 |
|          2 |        2 |         30.00 |                 10 |
|          3 |        4 |         20.00 |                 11 |
|          4 |        2 |         30.00 |                 12 |
+------------+----------+--------------+--------------------+

SELECT * FROM Tests;
+--------+----------------+--------------------+-------+------------+-----------+----------+
| TestID | Name           | Results            | Cost  | StartDate  | PatientID | DoctorID |
+--------+----------------+--------------------+-------+------------+-----------+----------+
|      1 | Blood Test     | NULL               | 20.00 | 2017-10-02 |         1 |        1 |
|      2 | Blood Test     | Blood does not clot | 25.00 | 2017-10-11 |         2 |        3 |
|      3 | Concussion Test | Concussed          | 50.00 | 2017-10-11 |         3 |        2 |
|      4 | Reflexes Test  | No reflexes        | 32.50 | 2017-11-23 |         1 |        4 |
+--------+----------------+--------------------+-------+------------+-----------+----------+

SELECT * FROM PatientResidesIn;
+------------+-----------+------------+-----------+
| StartDate  | PatientID | WardNumber | BedNumber |
+------------+-----------+------------+-----------+
| 2017-10-02 |         1 |          1 |         1 |
| 2017-10-11 |         2 |          1 |         1 |
| 2017-10-11 |         3 |          2 |         1 |
| 2017-11-23 |         1 |          2 |         2 |
+------------+-----------+------------+-----------+
```

4. SQL Queries

4.1

Information Processing:

Enter Patient Info

```
SQL>INSERT INTO Patients VALUES(1, '111-11-1111', 'Bytelie Chaplin',
'2001-10-10', 'Male', '919-555-5555', 'In the past', 'In Ward');
```

```
Query OK, 1 row affected (0.00 sec)
```

Update Patient Info

```
SQL>UPDATE Patients SET Phone='919-555-5405' WHERE PatientID=1;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete Patient Info

```
SQL>DELETE FROM Patients WHERE PatientID=1;

Query OK, 1 row affected (0.01 sec)
```

Enter Staff Info

```
SQL>INSERT INTO StaffMembers VALUES(1, 'Kelly Doctor', 25, 'Female',
'Cardiology', '919-555-8000', '7294 Fake Name Street');

Query OK, 1 row affected (0.00 sec)

SQL>INSERT INTO Doctors VALUES(1, 'Radiologist');

Query OK, 1 row affected (0.00 sec)
```

Update Staff Info

```
SQL>UPDATE StaffMembers SET Department='Radiology' WHERE StaffID=1;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete Staff Info

```
SQL>DELETE FROM StaffMembers WHERE StaffID=1;

Query OK, 1 row affected (0.00 sec)
```

Enter Ward Info

```
SQL>INSERT INTO Wards VALUES(1, 4, 100.00, NULL);

Query OK, 1 row affected (0.01 sec)
```

Update Ward Info

```
SQL>UPDATE Wards SET ChargesPerDay=125.00 WHERE WardNumber=1;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Delete Ward Info

```
SQL>DELETE FROM Wards WHERE WardNumber=1;

Query OK, 1 row affected (0.00 sec)
```

Check Ward Availability

```
SQL>SELECT COUNT(*) < (SELECT capacity FROM Wards WHERE WardNumber=1)
AS available FROM PatientResidesIn WHERE WardNumber=1;

+-----------+
| available |
+-----------+
|         1 |
+-----------+
1 row in set (0.00 sec)
```

Assign Ward

```
SQL>INSERT INTO  PatientResidesIn VALUES('2017-10-14', 1, 2, 2);

Query OK, 1 row affected (0.00 sec)
```

Reserve Ward

```
SQL>INSERT INTO  PatientResidesIn VALUES('2017-11-14', 3, 3, 1);

Query OK, 1 row affected (0.00 sec)
```

Release Ward

```
SQL>DELETE FROM PatientResidesIn WHERE PatientID=1;
```

```
Query OK, 1 row affected (0.00 sec)
```

Maintaining Medical Records for each patient:

Add New Medical Record

```
SQL>INSERT INTO MedicalRecords
VALUES(1, '2017-10-02', NULL, 'Physical Therapy', NULL, NULL);

Query OK, 1 row affected (0.00 sec)
```

Update Medical Record

```
SQL>UPDATE MedicalRecords
SET Diagnosis= 'Meningitis'
 WHERE PatientID=1;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Get Medical Record for Patient

```
SQL>SELECT * FROM MedicalRecords WHERE PatientID=1;

+-----------+------------+---------+-----------------+------------+--------------+
| PatientID | StartDate  | EndDate | Treatment       | Diagnosis  | Prescription |
+-----------+------------+---------+-----------------+------------+--------------+
|         1 | 2017-10-02 | NULL    | Physical Therapy | Meningitis | NULL         |
+-----------+------------+---------+-----------------+------------+--------------+
1 row in set (0.00 sec)
```

Add New Test

```
SQL>INSERT INTO Tests
VALUES(1, 'Reflex Test', 'Dulled Reflexes', 35.00, '2017-02-15', 1, 3);

Query OK, 1 row affected (0.00 sec)
```

Update Existing Test

```
SQL>UPDATE Tests SET Cost=40.00 WHERE TestID=1;

Query OK, 1 row affected (0.00 sec)
```

```
Rows matched: 1  Changed: 1  Warnings: 0
```

Maintaining Billing accounts:

Check Bed Availability

```
SQL> SELECT AllBeds.BedNumber FROM (SELECT * FROM PatientResidesIn
WHERE WardNumber=2) Occupied RIGHT OUTER JOIN (SELECT 1 as BedNumber
UNION ALL SELECT 2
UNION ALL SELECT 3
UNION ALL SELECT 4) AllBeds ON Occupied.BedNumber=AllBeds.BedNumber
WHERE Occupied.BedNumber IS NULL
AND AllBeds.BedNumber <= (SELECT capacity FROM Wards WHERE
WardNumber=2);

+-----------+
| BedNumber |
+-----------+
|         1 |
|         2 |
|         4 |
+-----------+
```

Add New Billing Account

```
SQL>INSERT INTO BillingAccounts
VALUES(220, '111-22-3333', '1212 Payer Ln, Raleigh, NC, 27603',
'Personal Check', NULL, NULL);

Query OK, 1 row affected (0.00 sec)

SQL>INSERT INTO BilledFor VALUES(220, 1, '2017-10-02');

Query OK, 1 row affected (0.00 sec)
```

Get Billing Account

```
SQL>SELECT * FROM BillingAccounts WHERE PayerSSN='111-22-3333';
```

```
+-----------+------------+----------------------------------+---------------+------------+-----------+
| AccountID | PayerSSN    | PayerAddress                     | PaymentMethod  | CardNumber | Insurance |
+-----------+------------+----------------------------------+---------------+------------+-----------+
|       220 | 111-22-3333 | 1212 Payer Ln, Raleigh, NC, 27603 | Personal Check | NULL       | NULL      |
+-----------+------------+----------------------------------+---------------+------------+-----------+
1 row in set (0.00 sec)
```

Update Billing Account

```
SQL>UPDATE BillingAccounts
SET Insurance='BCBS'
WHERE AccountID=220;

Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Reports:

Patient medical record history for time(Month/Year)

```
SQL>SELECT * FROM MedicalRecords WHERE PatientID=1 AND StartDate
BETWEEN '2017-10-01' AND LAST_DAY('2017-10-01');
+-----------+------------+---------+-----------------+------------+-------------+
| PatientID | StartDate  | EndDate | Treatment        | Diagnosis  | Prescription |
+-----------+------------+---------+-----------------+------------+-------------+
|         1 | 2017-10-02 | NULL    | Physical Therapy | Meningitis | NULL         |
+-----------+------------+---------+-----------------+------------+-------------+
1 row in set (0.00 sec)
```

Return ward usage

```
SQL>SELECT * FROM PatientResidesIn WHERE WardNumber = 1;
+------------+-----------+------------+-----------+
| StartDate  | PatientID | WardNumber | BedNumber |
+------------+-----------+------------+-----------+
| 2017-10-02 |         2 |          1 |         1 |
+------------+-----------+------------+-----------+
1 row in set (0.00 sec)
```

Return number of patients per month

```
SQL>SELECT COUNT(*)
FROM MedicalRecords
WHERE StartDate BETWEEN '2017-10-01' AND LAST_DAY('2017-10-01')
OR EndDate BETWEEN '2017-10-01' AND LAST_DAY('2017-10-01')
OR (EndDate > LAST_DAY('2017-10-01') AND StartDate < '2017-10-01');
+----------+
| COUNT(*) |
+----------+
|        2 |
+----------+
1 row in set (0.00 sec)
```

Return ward usage percentage

```
SQL>SELECT COUNT(*) /
(SELECT capacity FROM Wards WHERE WardNumber=1) as percentage
FROM PatientResidesIn WHERE WardNumber = 1;

+------------+
| percentage |
+------------+
| 0.2500     |
+------------+
```

Return all patient's information of the responsible doctor

```
SQL>SELECT Patients.PatientID, Patients.SSN, Patients.Name,
Patients.BirthDate, Patients.Gender, Patients.Phone, Patients.Address,
Patients.Status
FROM  Treats CROSS JOIN Patients
WHERE Patients.PatientID = Treats.PatientID AND Treats.DoctorID=1;
+-----------+------------+-----------------+------------+--------+-------------+-------------+---------+
| PatientID | SSN        | Name            | BirthDate  | Gender | Phone       | Address     | Status  |
+-----------+------------+-----------------+------------+--------+-------------+-------------+---------+
|         1 | 111-11-1111 | Bytelie Chaplin  | 2001-10-10 | Male   | 919-555-5555 | In the past | In Ward |
|         2 | 111-11-1111 | Chris Hemsworths | 2001-10-10 | Male   | 919-555-5555 | In the past | In Ward |
+-----------+------------+-----------------+------------+--------+-------------+-------------+---------+
2 rows in set (0.00 sec)
```

Return grouped hospital staff

SQL>SELECT * FROM StaffMembers JOIN Nurses WHERE Nurses.StaffID = StaffMembers.StaffID;

```
+---------+--------------+-----+--------+------------+----------------+-------------------------------------+---------+
| StaffID | Name         | Age | Gender | Department | Phone          | Address                             | StaffID |
+---------+--------------+-----+--------+------------+----------------+-------------------------------------+---------+
|       9 | Tom          |  27 | Male   | 3rd Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |       9 |
|      10 | Jerry        |  64 | Male   | 1st Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |      10 |
|      11 | Donna        |  39 | Female | 2nd Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |      11 |
|      12 | Jean Ralphio |  23 | Male   | 1st Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |      12 |
+---------+--------------+-----+--------+------------+----------------+-------------------------------------+---------+
```

4 rows in set (0.00 sec)

SQL>SELECT * FROM StaffMembers JOIN Doctors WHERE Doctors.StaffID = StaffMembers.StaffID;

```
+---------+--------+-----+--------+------------+----------------+-------------------------------------+---------+-------------------+
| StaffID | Name   | Age | Gender | Department | Phone          | Address                             | StaffID | ProfessionalTitle |
+---------+--------+-----+--------+------------+----------------+-------------------------------------+---------+-------------------+
|       1 | Renata |  22 | Female | 3rd Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |       1 | Nuerosurgeon      |
|       2 | Andy   |  52 | Male   | 1st Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |       2 | Rheumatologist    |
|       3 | Nicole |  34 | Female | 2nd Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |       3 | Pediatrician      |
|       4 | Alex   | 102 | Male   | 1st Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |       4 | Nuerosurgeon      |
+---------+--------+-----+--------+------------+----------------+-------------------------------------+---------+-------------------+
```

4 rows in set (0.00 sec)

SQL>SELECT * FROM StaffMembers JOIN Admins WHERE Admins.StaffID = StaffMembers.StaffID;

```
+---------+---------+-----+--------+------------+----------------+-------------------------------------+---------+
| StaffID | Name    | Age | Gender | Department | Phone          | Address                             | StaffID |
+---------+---------+-----+--------+------------+----------------+-------------------------------------+---------+
|      13 | Ben     |  22 | Female | 3rd Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |      13 |
|      14 | Chris   |  18 | Male   | 1st Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |      14 |
|      15 | Tina    |  34 | Female | 2nd Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |      15 |
|      16 | Michael |  60 | Male   | 1st Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |      16 |
+---------+---------+-----+--------+------------+----------------+-------------------------------------+---------+
```

4 rows in set (0.00 sec)

SQL>SELECT * FROM StaffMembers JOIN Receptionists WHERE Receptionists.StaffID = StaffMembers.StaffID;

```
+---------+--------+-----+--------+------------+----------------+-------------------------------------+---------+
| StaffID | Name   | Age | Gender | Department | Phone          | Address                             | StaffID |
+---------+--------+-----+--------+------------+----------------+-------------------------------------+---------+
|       5 | Ann    |  22 | Female | 3rd Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |       5 |
|       6 | April  |  18 | Female | 1st Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |       6 |
|       7 | Leslie |  34 | Female | 2nd Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |       7 |
|       8 | Ron    |  60 | Male   | 1st Floor  | (919) 867-5309 | 3124 Address Lane, Raleigh, NC, 27604 |       8 |
+---------+--------+-----+--------+------------+----------------+-------------------------------------+---------+
```

4 rows in set (0.01 sec)

4.2

Return number of patients per month

1. **SQL>EXPLAIN SELECT COUNT(*) FROM MedicalRecords WHERE StartDate BETWEEN '2017-10-01' AND LAST_DAY('2017-10-01') OR EndDate BETWEEN '2017-10-01' AND LAST_DAY('2017-10-01') OR (EndDate > LAST_DAY('2017-10-01') AND StartDate < '2017-10-01');**

2.

```
+------+-------------+----------------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table          | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+----------------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | MedicalRecords | ALL  | NULL          | NULL | NULL    | NULL |    4 | Using where |
+------+-------------+----------------+------+---------------+------+---------+------+------+-------------+
```

3. **CREATE INDEX endDateIndex ON MedicalRecords(EndDate);**

4.

```
+------+-------------+----------------+-------+---------------+-------------+---------+------+------+--------------------------+
| id   | select_type | table          | type  | possible_keys | key         | key_len | ref  | rows | Extra                    |
+------+-------------+----------------+-------+---------------+-------------+---------+------+------+--------------------------+
|    1 | SIMPLE      | MedicalRecords | index | endDateIndex  | endDateIndex | 4       | NULL | 4    | Using where; Using index |
+------+-------------+----------------+-------+---------------+-------------+---------+------+------+--------------------------+
```

Get Billing Account

1. **EXPLAIN SELECT * FROM BillingAccounts WHERE PayerSSN='111-22-3333';**
2.

```
+------+-------------+-----------------+------+---------------+------+---------+------+------+-------------+
| id   | select_type | table           | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+------+-------------+-----------------+------+---------------+------+---------+------+------+-------------+
|    1 | SIMPLE      | BillingAccounts | ALL  | NULL          | NULL | NULL    | NULL |    5 | Using where |
+------+-------------+-----------------+------+---------------+------+---------+------+------+-------------+
```

3. **CREATE INDEX ssnBillingIndex ON BillingAccounts(PayerSSN);**
4.

```
+----+-------------+-----------------+------+----------------+----------------+---------+-------+------+-----------------------+
| id | select_type | table           | type | possible_keys  | key            | key_len | ref   | rows | Extra                 |
+----+-------------+-----------------+------+----------------+----------------+---------+-------+------+-----------------------+
|  1 | SIMPLE      | BillingAccounts | ref  | ssnBillingIndex | ssnBillingIndex | 18      | const |  1   | Using index condition |
+----+-------------+-----------------+------+----------------+----------------+---------+-------+------+-----------------------+
```

4.3 Query Correctness:

1.) Select all patient information from a certain doctor.

```
SQL>SELECT Patients.PatientID, Patients.SSN, Patients.Name,
Patients.BirthDate, Patients.Gender, Patients.Phone, Patients.Address,
Patients.Status
FROM  Treats
CROSS JOIN Patients
WHERE Patients.PatientID = Treats.PatientID AND Treats.DoctorID=1;
```

$\pi_{\text{Patients.id, Patients.SSN, Patients.Name, Patients.BirthDate, Patients.Gender, Patients.Phone, Patients.Address, Patients.Status}}((\sigma_{\text{Treats.DoctorID=1}}\ (\text{Patients} \bowtie_{\text{Patients.PatientID=Treats.PatientID}} \text{Treats})))$

Suppose p is any tuple in the patients relation, and t is any tuple in the treats relation, such that the value of p.PatientID and t.PatientID is the same. The combination of the tuples p and t will give all patient information for a patient that was treated by whichever doctor you were examining. This combination will return the patients ID, SSN, Name, BirthDate, Gender, Phone, Address, and Status. By looking at the combination of whether the patient and treats id are the same, we will be given all the patients information whose doctor was the requested doctor. This is exactly what we were required to retrieve.

2). Group all staff together by profession(Nurse, Doctor, etc..)

```
SELECT * FROM StaffMembers JOIN Nurses WHERE Nurses.StaffID =
StaffMembers.StaffID;
```

$\text{StaffMembers} \bowtie_{\text{StaffMembers.StaffID=Nurses.StaffID}} \text{Nurses}$

Suppose s is any tuple in the StaffMembers relation, and n is any tuple in the Nurses relation, such that the value of s.StaffID is equal to n.StaffID. The combination of the tuples s and n will return a single tuple, who is a staff member, and is also a nurse. If we change our assumption to where s' is a table of staff members, and n' is a table of nurses, when we join these tables, and grab all tuples where n'.StaffID and s'.StaffID are equal, we will be given all staff members who are nurses. This will return all staff member information of staff members who are nurses, based on the fact that the staff member ids are the same in the staff member table, and the nurses table. When we do this for each different type of table for each type of staff member, we will be given all the staff members grouped by profession, which is what we exactly wanted.