Sif Islam
Database Systems CSci 126 (Fresno State)
Prof. David Ruby
Date: 05/05/2017

A database management system for Fresno Hospital

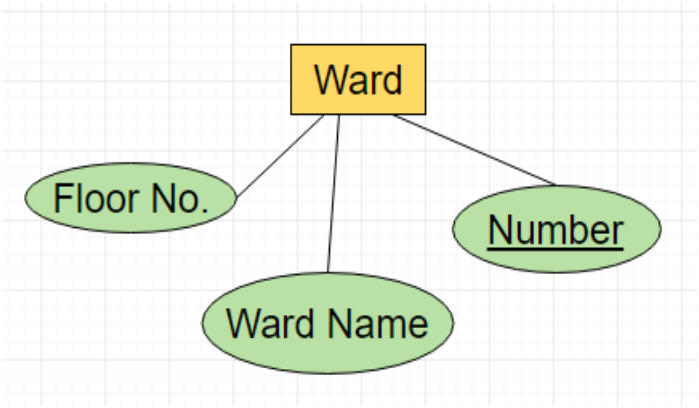
This database is built to work with any hospital system. Currently we will work with a Fresno hospital as a part of initial implementation. Please remember, that this is not a final database system, it has the potential to grow into a vast database system. Thus, more and more features can easily be added to it, and it is not limited to its current capabilities. This database will currently take care of the basic requirement of a general hospital system. I will be able to keep records of wards, doctors, nurses, patients, diagnosis that patient receives, payment made by patients, appointments by patients and many others.

Part 1. Entities in Fresno Hospital Domain Description

- Ward
- Employee
    - Doctor
    - Nurse
- Research Lab
- Patient
- Admitted patient
- Appointment
- Diagnosis
- Payment

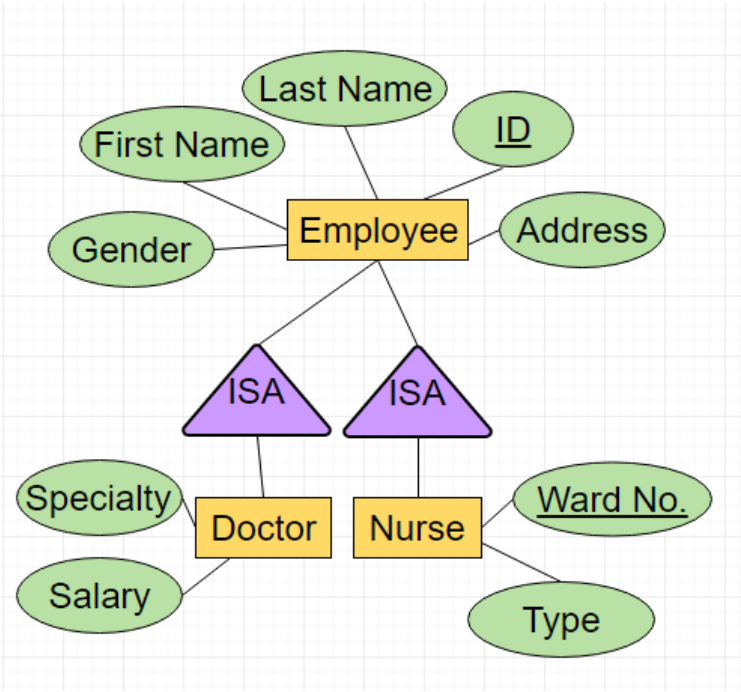
Ward Domain

- Entity Set: Wards
- Entity: Ward
- Attributes: A ward has…
  - Number (Primary key)
  - Floor No.
  - Ward name



Employee Domain

- Entity Set: Employees
- Entity: Employee
- Attributes: An Employee has…
  - ID (Primary key)
  - First Name
  - Last Name
  - Gender
  - Address



Doctor Domain

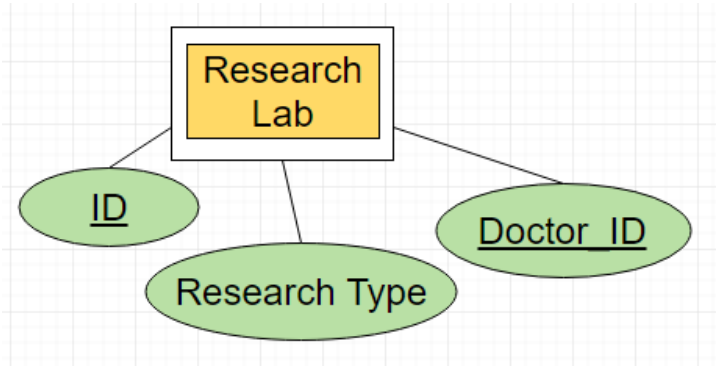
- Attributes: A Doctor has all employee attributes plus
  - Specialty
  - Salary

Nurse Domain

- Attributes: A Nurse has all employee attributes plus
  - Specialty
  - Ward No. (Foreign key)

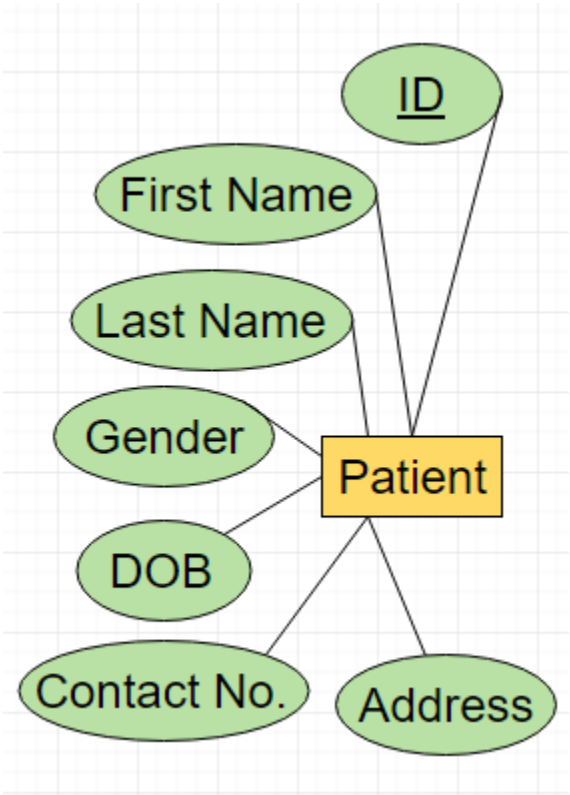
Research Lab Domain

- Entity Set: Research labs
- Entity: Research lab
- Attributes: A Research lab has…
  - ID (Primary key)
  - Research Type
  - Doctor_ID (Foreign key)



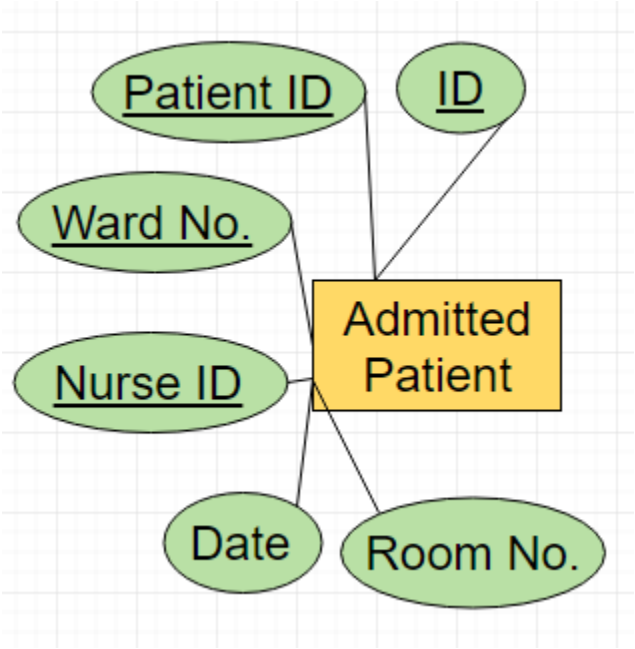
Patient Domain

- Entity Set: Patients
- Entity: Patient
- Attributes: A Patient has…
  - ID (Primary key)
  - First Name
  - Last Name
  - Gender
  - DOB
  - Contact No.
  - Address



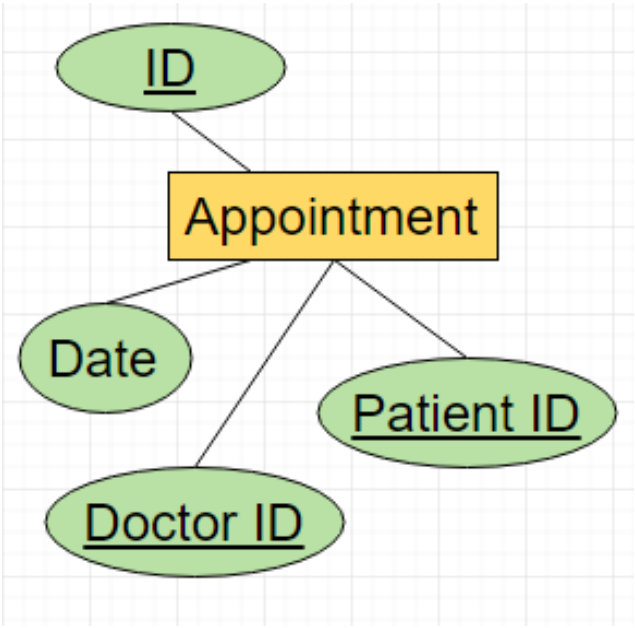
Admitted Patient Domain

- Entity Set: Admitted Patients
- Entity: Admitted Patient
- Attributes: An Admitted has…
    - ID (Primary key)
    - Patient ID (Foreign Key)
    - Nurse ID (Foreign Key)
    - Date
    - Room No.



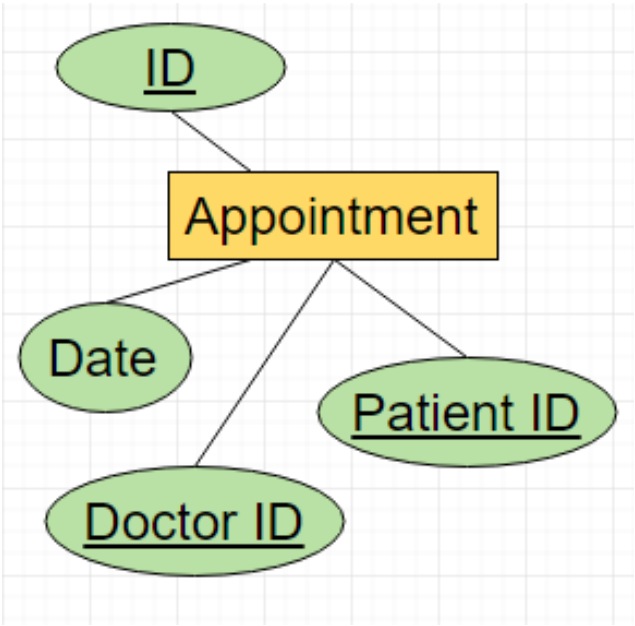
Appointment Domain

- Entity Set: Appointments
- Entity: Appointment
- Attributes: An Appointment has…
    - ID (Primary key)
    - Patient ID (Foreign Key)
    - Doctor ID (Foreign Key)
    - Date



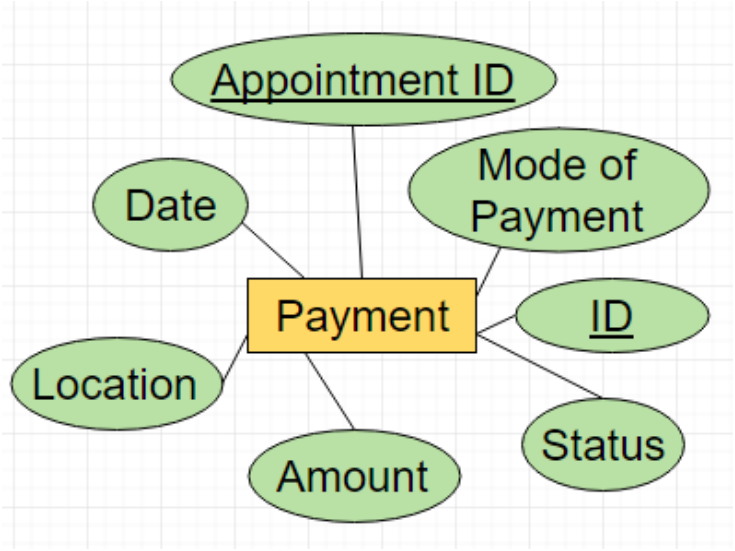
Diagnosis Domain

- Entity Set: Diagnosis
- Entity: Diagnosis
- Attributes: A Diagnosis has…
  - ID (Primary key)
  - Patient ID (Foreign Key)
  - Doctor ID (Foreign Key)
  - Date

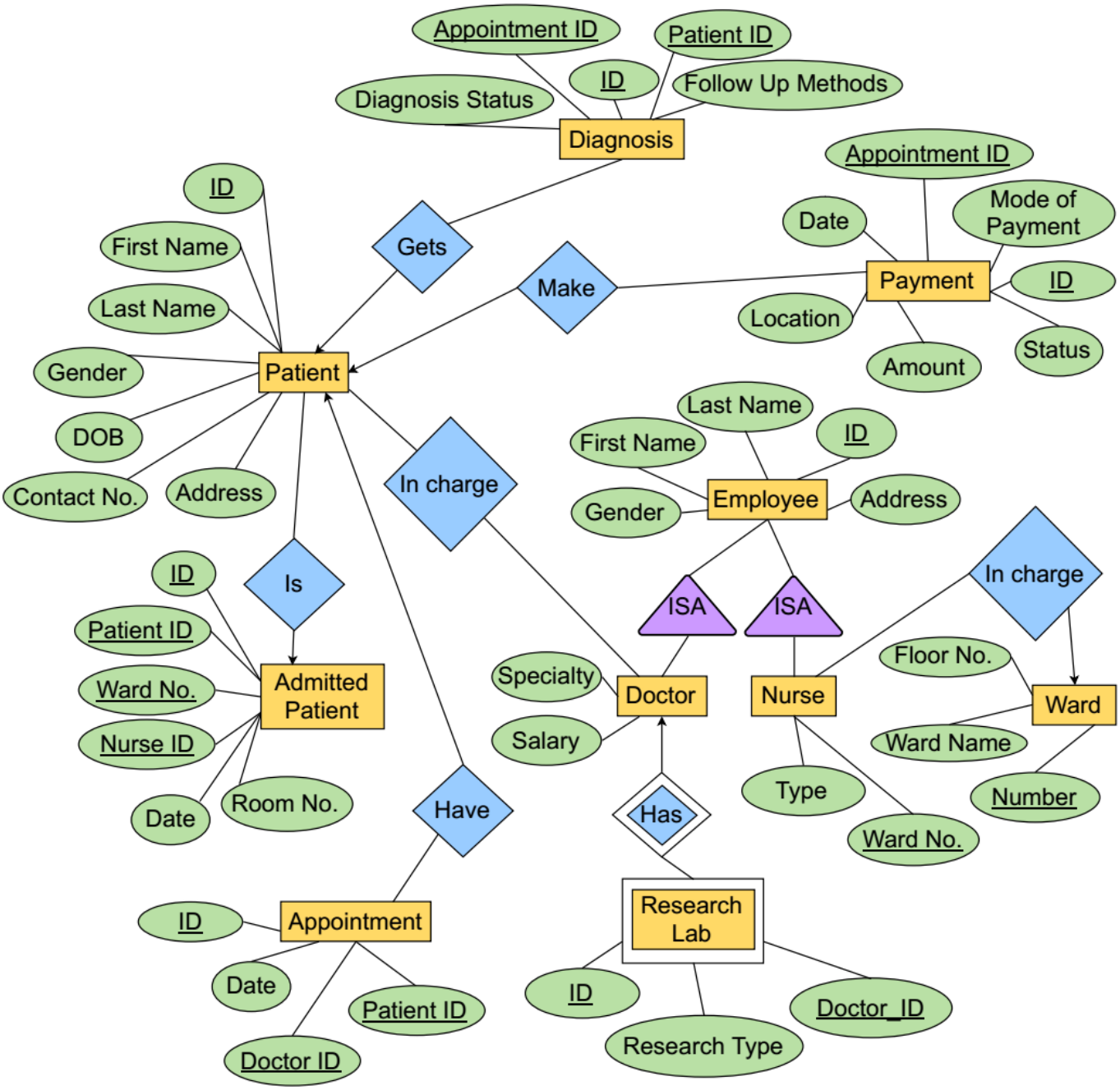


Payment Domain

- Entity Set: Payments
- Entity: Payment
- Attributes: A Payment has…
  - ID (Primary key)
  - Appointment ID (Foreign Key)
  - Mode of Payment
  - Status
  - Amount
  - Location
  - Date



Part 2. Database Design



Relationships:

Domain 1	Domain 2	Relationship	Note
Nurse	Ward	One-to-many	One ward can have multiple nurse, but one nurse can only belong to a single ward
Patient	Diagnosis	One-to-many	One patient can have multiple diagnosis
Patient	Payment	One-to-many	One patient can have multiple payments
Patient	Admitted Patient	One-to-one	One patient cannot have multiple admission at a time
Patient	Appointment	One-to-many	One patient can have multiple appointments
Doctor	Patient	Many-to-many	One patient can have multiple doctors, and one doctor can also have multiple patients
Doctor	Research Lab	Weak-Entity Set	Doctors may belong to a research lab

Part 3: Database Schema/Normal Forms

Ward Domain will have the following SQL INSERT statement

```
CREATE TABLE Ward(
      Ward_No VARCHAR(8) NOT NULL PRIMARY KEY,
      Ward_Name VARCHAR(20) NOT NULL,
      Floor_No VARCHAR(1) NOT NULL
);
```

Sample looks like this:

Ward_No	Ward_Name	Floor_No
W001	Meternity Ward	3
W002	Psychiatric Ward	4
W003	Accident Ward	1
W004	General Ward	2

The schema is **free** from violations for:

– 3rd Normal form (all column depends on Ward_No column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

Doctor Domain will have the following SQL INSERT statement

```
CREATE TABLE Doctor(
      Doctor_ID VARCHAR(8) NOT NULL PRIMARY KEY,
      Doctor_First_Name VARCHAR(20) NOT NULL,
      Doctor_Last_Name VARCHAR(20) NOT NULL,
      Doctor_Gender VARCHAR(1) NOT NULL,
      Doctor_Specialty VARCHAR(20) NOT NULL,
      Doctor_Address VARCHAR(20) NOT NULL,
      Doctor_Salary VARCHAR(10) NOT NULL
);
```

Sample looks like this:

Doctor_ID	Doctor_First_Name	Doctor_Last_Name	Doctor_Gender	Doctor_Type	Doctor_Address	Doctor_Salary
D001	Jacob	Silva	M	Cardiologist	Kansas City, Kansas	197000
D002	Alexander	Turner	M	Hematologist	Denver, Colorodo	162000
D003	Harper	Phillips	F	Hospitalist	Queens, New York	292000
D004	Charlotte	Campbell	F	Medical Geneticist	Fresno, California	266000
D005	Kim	Campbell	M	Gynecologist	SF, California	219000
D006	Emily	Parker	F	Neonatologist	LA, California	162000
D007	Michael	Evans	M	Nephrologist	Austin, Texas	431000
D008	Mia	Edwards	F	Pediatrician	Phoenix, Arizona	225000
D009	Benjamin	Collins	M	Pathologist	Tampa, Florida	262000
D010	Abigail	Stewart	F	Psychiatrist	Wichita, Kansas	105000

The schema is **free** from violations for:

– 3rd Normal form (all column depends on Doctor_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

Nurse Domain will have the following SQL INSERT statement

```
CREATE TABLE Nurse(
      Nurse_ID VARCHAR(8) NOT NULL PRIMARY KEY,
      Nurse_First_Name VARCHAR(20) NOT NULL,
      Nurse_Last_Name VARCHAR(20) NOT NULL,
      Nurse_Gender VARCHAR(1) NOT NULL,
      Ward_No VARCHAR(8) NOT NULL,
      Nurse_Type Varchar(20),
    FOREIGN KEY (Ward_No) REFERENCES Ward(Ward_No)
)
```

Sample looks like this:

Nurse_ID	Nurse_First_Name	Nurse_Last_Name	Nurse_Gender	Ward_No	Nurse_Type
N001	Noah	Smith	M	W001	Head
N002	Emma	Williams	F	W001	General
N003	Liam	Johnson	M	W002	Head
N004	Olivia	Jones	F	W003	Head
N005	Jacob	Brown	M	W004	Head
N006	Ava	Miller	F	W002	General
N007	Ethan	Moore	M	W003	General
N008	Isabella	Thomas	F	W004	General
N009	Alexander	Taylor	M	W001	General
N010	Alexander	Hopps	M	W002	General

The schema is **free** from violations for:

– 3rd Normal form (all column depends on Doctor_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

The schema is a violation for:

– 4th Normal Form (Column Nurse_Type has multi value dependencies)

Patient Domain will have the following SQL INSERT statement

```
CREATE TABLE Patient(
      Patient_ID VARCHAR(8) NOT NULL PRIMARY KEY,
      Patient_First_Name VARCHAR(20) NOT NULL,
      Patient_Last_Name VARCHAR(20) NOT NULL,
      Patient_Gender VARCHAR(1) NOT NULL,
      Patient_Date_Of_Birth DATE NOT NULL,
      Patient_Address VARCHAR(20) NOT NULL,
      Patient_Contact_No VARCHAR(24) NOT NULL
);
```

Sample looks like this:

Patient_ID	Patient_First_Name	Patient_Last_Name	Patient_Gender	Patient_Date_Of_Birth	Patient_Address	Patient_Contact_No
P001	James	Bond	M	1990-05-12	Wyoming	(251) 546-9442
P002	Angelina	Jolie	F	1982-04-28	Louisiana	(125) 546-4478
P003	Scarlett	Johansson	F	1982-09-18	Michigan	(949) 569-9244
P004	Anne	Hathaway	F	1986-08-28	Indiana	(226) 906-4478
P005	Johnny	Depp	M	1991-07-18	Idaho	(671) 925-9244
P006	Sherlock	Homes	M	1992-04-15	South Carolina	(630) 446-6451
P007	Kevin	Spacey	M	1983-04-16	Kentucky	(237) 644-4478
P008	Denzel	Washington	M	1985-03-07	Oklahoma	(159) 779-6451
P009	Natalie	Portman	F	1982-04-25	Colorado	(407) 732-4478
P010	Tom	Cruise	M	1986-04-21	Washington	(287) 936-9244

The schema is **free** from violations for:

– 3rd Normal form (all column depends on Patient_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

Admitted Patient Domain will have the following SQL INSERT statement

```
CREATE TABLE Admitted_Patient(
      Admitted_ID VARCHAR(8) NOT NULL PRIMARY KEY,
      Patient_ID VARCHAR(8) NOT NULL,
      Ward_No VARCHAR(8) NOT NULL,
      Nurse_ID VARCHAR(8) NOT NULL,
      Admitted_Date_Time DATETIME NOT NULL,
      Room_No VARCHAR(20) NOT NULL,
    FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),
    FOREIGN KEY (Ward_No) REFERENCES Ward(Ward_No),
    FOREIGN KEY (Nurse_ID) REFERENCES Nurse(Nurse_ID)
)
```

Sample looks like this:

Admitted_ID	Patient_ID	Ward_No	Nurse_ID	Admitted_Date_Time	Room_No
AD001	P003	W001	N009	2017-01-01 09:26:07	201
AD002	P008	W002	N008	2017-01-02 11:25:15	311
AD003	P006	W004	N007	2017-01-05 12:35:14	210
AD004	P001	W002	N001	2017-01-06 11:35:14	302
AD005	P010	W004	N006	2017-01-10 20:15:19	211

The schema is **free** from violations for:

– 3rd Normal form (all column depends on Admitted_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

Appointment Domain will have the following SQL INSERT statement

```
CREATE TABLE Appointment(
      Appointment_ID VARCHAR(8) NOT NULL PRIMARY KEY,
      Patient_ID VARCHAR(8) NOT NULL,
      Doctor_ID VARCHAR(8) NOT NULL,
      Appointment_Date_Time DATETIME NOT NULL,
    FOREIGN KEY (Doctor_ID) REFERENCES Doctor(Doctor_ID),
    FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)
);
```

Sample looks like this:

Appointment_ID	Patient_ID	Doctor_ID	Appointment_Date_Time
A001	P001	D010	2017-01-11 09:30:00
A002	P002	D009	2017-01-11 08:30:00
A003	P003	D005	2017-01-12 10:30:00
A004	P004	D005	2017-01-12 11:30:00
A005	P005	D005	2017-01-13 12:30:00
A006	P006	D007	2017-01-14 13:30:00
A007	P007	D006	2017-01-14 14:30:00
A008	P008	D008	2017-01-14 15:30:00
A009	P009	D002	2017-01-14 16:30:00
A010	P010	D002	2017-01-15 10:30:00
A011	P004	D002	2017-01-20 15:30:00

The schema is **free** from violations for:

– 3rd Normal form (all column depends on Appointment_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

Diagnosis Domain will have the following SQL INSERT statement

```
CREATE TABLE Diagnosis(
      Diagnosis_ID VARCHAR(8) NOT NULL PRIMARY KEY,
      Appointment_ID VARCHAR(8) NOT NULL,
      Patient_ID VARCHAR(8) NOT NULL,
      Diagnosis_Status VARCHAR(50) NOT NULL,
      Follow_Up_Methods VARCHAR(50) NOT NULL,
      FOREIGN KEY (Appointment_ID) REFERENCES Appointment(Appointment_ID),
      FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)
);
```

Sample looks like this:

Diagnosis_ID	Appointment_ID	Patient_ID	Diagnosis_Status	Follow_Up_Methos
DG001	A001	P001	Coronary Artery Disease	CT Scan
DG002	A002	P002	Stroke	Medication
DG003	A002	P002	Stroke	Bed rest
DG004	A004	P004	Trachea	Injection Course
DG005	A005	P005	Infections	MRI
DG006	A006	P006	Diabetes	Injection Course
DG007	A007	P007	Fever	Medication
DG008	A008	P008	Heart Disease	CT Scan
DG009	A009	P009	Fever	Medication
DG010	A010	P010	Diarrhea	Medication
DG011	A003	P003	Pregnancy	5 Weeks
DG012	A003	P003	High blood pressure	Blood test
DG013	A003	P003	Headache	Medication
DG014	A007	P007	Infections	X-Ray
DG015	A007	P007	Headache	Medication

The schema is **free** from violations for:

– 3rd Normal form (all column depends on Diagnosis_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

NOTE: Though Diagnosis_Status column and Follow_Up_Methodes column have some data that are common, but these fields are text (comment) fields, they can contain anything, nothing (null value) or duplicates.

Payment Domain will have the following SQL INSERT statement

```
CREATE TABLE Payment(
      Payment_ID VARCHAR(8) NOT NULL PRIMARY KEY,
      Appointment_ID VARCHAR(8) NOT NULL,
      Mode_Of_Payment VARCHAR(20) NOT NULL,
      Payment_Date_Time DATETIME NOT NULL,
      Payment_Status VARCHAR(20) NOT NULL,
      Payment_Location VARCHAR(20) NOT NULL,
      Amount INT NOT NULL,
      FOREIGN KEY (Appoinment_ID) REFERENCES Appointment(Appointment_ID)
);
```

Payment_ID	Appointment_ID	Mode_Of_Payment	Payment_Date_Time	Payment_Status	Amount
PAY001	A001	Cash	2017-01-01 05:13:00	PAID	8000
PAY002	A002	Cash	2017-01-05 14:13:00	PAID	4000
PAY003	A003	Debit Card	2017-01-14 09:00:00	DUE	6000
PAY004	A004	Bitcoin	2017-01-01 13:21:00	PAID	10500
PAY005	A005	Credit Card	2017-11-01 09:12:00	DUE	4000
PAY006	A006	Cash	2017-01-09 07:10:00	DUE	4500
PAY007	A007	Debit Card	2017-01-12 18:02:00	PAID	8000
PAY008	A008	Cash	2017-01-11 11:11:00	PAID	5000
PAY009	A009	Credit Card	2017-01-12 14:00:00	PAID	7000
PAY010	A010	Cash	2017-01-14 13:21:00	DUE	4800
PAY011	A001	Check	2017-01-16 12:10:00	PAID	1000

The schema is **free** from violations for:

– 3rd Normal form (all column depends on Payment_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

The schema is a violation for:

– 4th Normal Form (Column Mode_Of_Payment and Payment_Status has multi value dependencies)

Research Lab Domain will have the following SQL INSERT statement

```
CREATE TABLE Research_Lab(
      Research_ID VARCHAR(8) NOT NULL PRIMARY KEY,
      Doctor_ID VARCHAR(8) NOT NULL,
      Research_Type VARCHAR(20) NOT NULL,
      FOREIGN KEY (Doctor_ID) REFERENCES Doctor(Doctor_ID)
);
```

Research_ID	Doctor_ID	Research_Type
R001	D001	Cancer
R002	D003	Cardiology
R003	D004	Gynaecology
R004	D009	Cardiology

The schema is **free** from violations for:

– 3rd Normal form (all column depends on Research_ID column)

– Boyce-Codd Normal Form (All function dependencies have a key on the left-hand side)

– 4th Normal Form (there are no multi value dependencies)

Part 4: Queries

1. A sample select query

- find a patient by address

```
SELECT *
FROM Patient
WHERE Patient_Address='South Carolina';
```

Result:

Patient_ID	Patient_First_Name	Patient_Last_Name	Patient_Gender	Patient_Date_Of_Birth	Patient_Address	Patient_Contact_No
P006	Sherlock	Homes	M	1992-04-15	South Carolina	(630) 446-6451

2. A sample select query with AND clause

- find a specific doctor

```
SELECT *
FROM Doctor
WHERE Doctor_Gender='M' AND Doctor_Last_Name='Campbell';
```

Doctor_ID	Doctor_First_Name	Doctor_Last_Name	Doctor_Gender	Doctor_Type	Doctor_Address	Doctor_Salary
D005	Kim	Campbell	M	Gynecologist	SF, California	219000

3. A sample select query with build in SQL function “DATEDIFF”

- find appointment in a given date

```
SELECT *
FROM Appointment
WHERE DATEDIFF(Appointment_Date_Time, '2017-01-14')=0
```

Appointment_ID	Patient_ID	Doctor_ID	Appointment_Date_Time
A006	P006	D007	2017-01-14 13:30:00
A007	P007	D006	2017-01-14 14:30:00
A008	P008	D008	2017-01-14 15:30:00
A009	P009	D002	2017-01-14 16:30:00

4. A sample Query with subquery

- find the phone number of female patients who are suffering from strokes

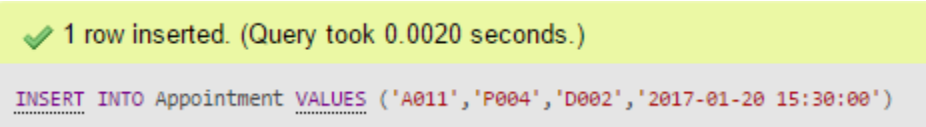
```
SELECT Patient_Contact_No
FROM Patient
WHERE Patient_ID IN
(     SELECT Patient_ID
      FROM Diagnosis
      WHERE Diagnosis_Status like('Stroke')
)
AND Patient_Gender like('F');
```

Patient_Contact_No
(125) 546-4478

5. A sample Insert Query

- create a new appointment

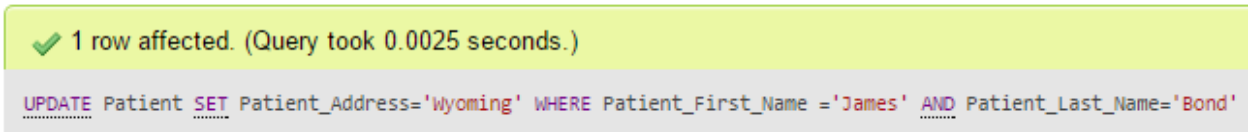
```
INSERT INTO Appointment VALUES
('A011','P004','D002','2017-01-20 15:30:00');
```



6. A sample Update Query

- update the address of a given patient

```
UPDATE Patient
SET Patient_Address='Wyoming'
WHERE Patient_First_Name ='James' AND Patient_Last_Name='Bond';
```



7. A more advanced select Query

- find assigned doctors for a selected patient, and his salary

```
SELECT Doctor_First_Name, Doctor_Last_Name, Doctor_Salary
FROM Appointment a, Doctor d
WHERE
      d.Doctor_ID = a.Doctor_ID
AND
      a.Patient_ID =
      (
      SELECT Patient_ID FROM Patient
      WHERE Patient_First_Name= 'Scarlett'
      AND Patient_Last_Name= 'Johansson'
      )
;
```

Doctor_First_Name	Doctor_Last_Name	Doctor_Salary
Kim	Campbell	219000

8. Another more advanced select Query

- find number of upcoming appointments between dates

```
SELECT count(Appointment_ID) AS Appointments_Upcomming
FROM Appointment
WHERE Appointment_Date_Time
      BETWEEN '2017-01-14 00:00:00'
      AND '2017-01-15 23:59:00';
```

Appointments_Upcomming
5

9. Another more advanced select Query with Aggregation

- find total amount paid by a single patient

```
SELECT SUM(Amount) AS Total_Amount_Paid_By_James_Bond
FROM Payment p, Appointment a
WHERE
      a.Appointment_ID = p.Appointment_ID
AND
      a.Patient_ID = (
            SELECT Patient_ID FROM Patient
            WHERE Patient_First_Name = 'James'
            AND Patient_Last_Name = 'Bond'
      );
```

Total_Amount_Paid_By_James_Bond
9000

10. Another more advanced select Query with Subquery and Order by clause

- find patient with appointment that listed 'fever' as their illness

```
SELECT Patient_First_Name, Patient_Last_Name
FROM Patient
WHERE Patient_ID IN
      (SELECT Patient_ID FROM Appointment WHERE Appointment_ID IN
            (SELECT Appointment_ID
            FROM Diagnosis
            WHERE Diagnosis_Status='Fever')
      )
ORDER BY Patient_ID;
```

Patient_First_Name	Patient_Last_Name
Kevin	Spacey
Natalie	Portman

11. Query with Count clause

- find number of nurse assigned for each ward

```
SELECT Ward_No, COUNT(*) AS Nurse_Count
FROM Nurse
GROUP BY Ward_No
```

Ward_No	Nurse_Count
W001	3
W002	3
W003	2
W004	2

12. Query with Subquery

- find nurses in a certain ward with patients in it

```
SELECT Nurse_First_Name, Nurse_Last_Name
FROM Nurse
WHERE Nurse_ID IN (
      SELECT Nurse_ID FROM Admitted_Patient
      WHERE Ward_No='W001'
);
```

Nurse_First_Name	Nurse_Last_Name
Alexander	Taylor

13. Another Query with Aggregation

- find maximum payment received by the hospital by a single patient

```
SELECT Mode_Of_Payment, Amount AS Max_Payment_Made_By_Patient
FROM Payment
WHERE Amount=(SELECT MAX(Amount) FROM Payment);
```

Mode_Of_Payment	Max_Payment_Made_By_Patient
Bitcoin	10500

14. Query with Subquery and Aggregation

- count the number of mail patient that are currently admitted in the hospital

```
SELECT COUNT(Admitted_ID) AS Male_Patient_COUNT
FROM Admitted_Patient
WHERE Patient_ID IN(
      SELECT Patient_ID FROM Patient
      WHERE Patient_Gender='M'
);
```

Male_Patient_COUNT
4

15. Query with NOT IN clause

- Find free nurses

```
SELECT Nurse_First_Name, Nurse_Last_Name, Ward_No
FROM Nurse
WHERE Nurse_ID NOT IN (SELECT Nurse_ID FROM Admitted_Patient);
```

Nurse_First_Name	Nurse_Last_Name	Ward_No
Emma	Williams	W001
Liam	Johnson	W002
Olivia	Jones	W003
Jacob	Brown	W004
Alexander	Hopps	W002

16. An advanced DISTINCT Query with Order by clause

- Find doctors who are treating 'Diabetes'

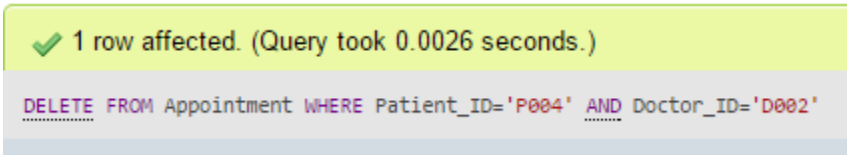
```
SELECT DISTINCT doc.Doctor_ID, Doctor_First_Name, Doctor_Last_Name
FROM Doctor doc, Appointment a, Diagnosis dig
WHERE doc.Doctor_ID = a.Doctor_ID
AND dig.Diagnosis_Status = 'Diabetes'
AND dig.Appointment_ID = a.Appointment_ID
ORDER BY Doctor_ID;
```

Doctor_ID	Doctor_First_Name	Doctor_Last_Name
D007	Michael	Evans

17. Delete Query

- remove an appointment

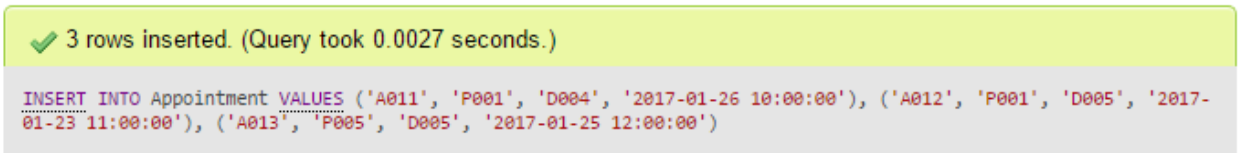
```
DELETE FROM Appointment
WHERE Patient_ID='P004' AND Doctor_ID='D002';
```



18. Insert Query (add multiple appointments)

- create multiple appointments at the same time

```
INSERT INTO Appointment VALUES
('A011', 'P001', 'D004', '2017-01-26 10:00:00'),
('A012', 'P001', 'D005', '2017-01-23 11:00:00'),
('A013', 'P005', 'D005', '2017-01-25 12:00:00');
```



19. Verifying Insert Query

[Appointment count for Patient P001 is now 3]

```
SELECT Patient_ID, COUNT(*) AS Appointment_Count
FROM Appointment
GROUP BY Patient_ID;
```

Patient_ID	Appointment_Count
P001	3
P002	1
P003	1
P004	1
P005	2
P006	1
P007	1
P008	1
P009	1
P010	1

20. Select Query with HAVING keyword

- find Doctors who have multiple appointments

```
SELECT Doctor_ID,COUNT(Appointment_ID) AS Appointment_Count
FROM Appointment
GROUP BY Doctor_ID
HAVING COUNT(Appointment_ID) >= 2;
```

Doctor_ID	Appointment_Count
D002	2
D005	5

Part 5: Application/Use Cases

This database management system is designed to work with any hospital system. It has all the basic requirements to manage Doctors, Nurse, Patients. Including assigning nurses to specific wards and assigning doctors to patients. Doctors are also allowed to have their own (optional) research labs. This system also has the appointment scheduling and payment system build into it. Once a patient is admitted to the hospital, doctors can assign appropriate diagnosis methods that suits the patient.

As mentioned earlier, this is an open-end system, modifications are welcome. Thus, it has the room to grow big and bigger. And once this database grows to its potential, it can not only be used by a hospital system but in variety of other systems. People can perform research based on this database. For example, they can calculate the average cost of a patient visiting a hospital in a certain area, thus enabling them to compare and contrast this data to make a table/graph for some project or paper. This is just one small example; this database is not limited to calculating payments. Not only humans, but AI like IBM Watson can use this database to gather knowledge about the diagnosis of patients and increase its knowledge base allowing it to be more aware. One example for Watson would be to calculate the main type of illness that people get in a certain area. Later, Watson can use this data to determine diagnosis not only based on a person's health but also taking the factor of patient's location, which a real doctor might overlook. These few example proves that this system can be used for border range of tasks, including maintaining a hospital. It will grow bigger, and help may in the process.

Thank you