VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Database Systems (CO2014)

Assignment 1

Data Modelling & Designing

Group 1

Professor:	Phan Trọng Nhân	
Group Members:	Nguyễn Đình Khương Duy	1952207
	Nguyễn Lê Nhật Dương	1952638
	Nguyễn Văn Quốc Chương	1950004
	Trương Đăng Quang	1952940
	Trần Nguyễn Phước Nhân	1952893

HO CHI MINH CITY, October 2021

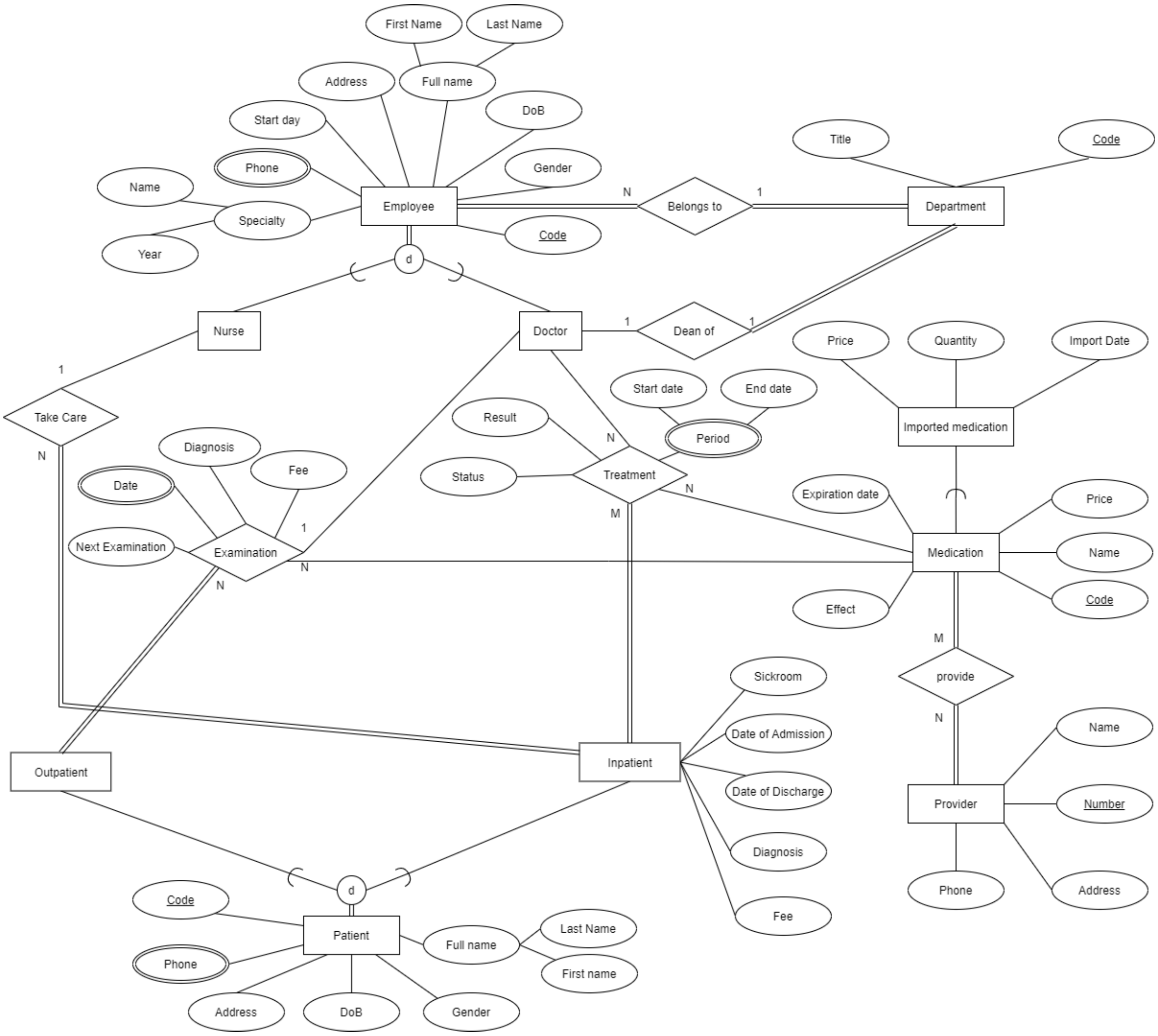
Contents

Member list & Workload

No.	Fullname	Student ID	Workload
1	Nguyễn Đình Khương Duy	1952207	20%
2	Nguyễn Lê Nhật Dương	1952638	20%
3	Nguyễn Văn Quốc Chương	1950004	20%
4	Trương Đăng Quang	1952940	20%
5	Trần Nguyễn Phước Nhân	1952893	20%

1 EER model

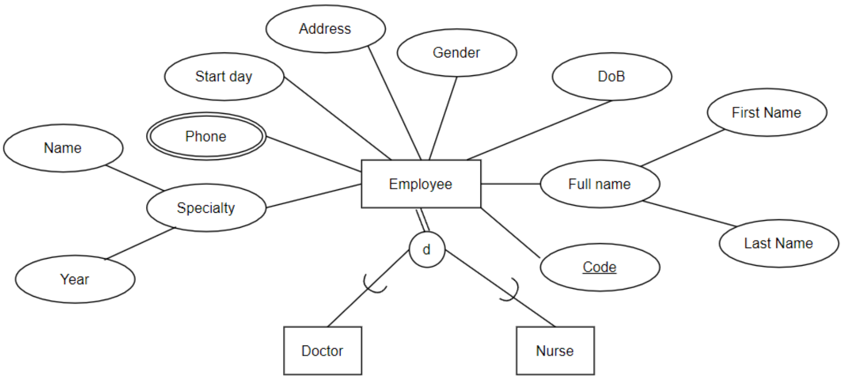
We must list what we understand from the database (the entities, the relationships, constrains, etc). Finally, the our EER model looks like this:



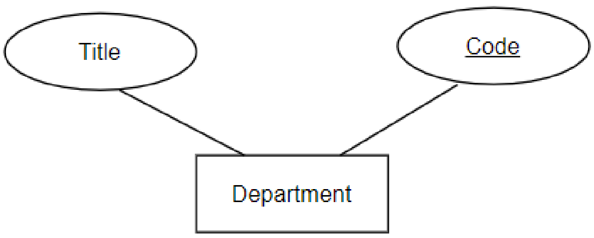
1.1 Entities and attributes, superclass and subclass

Base on the requirements of assignment, with the business description of hospital database, we can determine these entities following:

- `Employee` (superclass) including: **a unique code**, full name consisting of first name and last name, date of birth, gender, address, start date (first day of work), phone number(s), and speciality with its related name and degree's year.
- `Doctor` (subclass of employee)
- `Nurse` (subclass of employee)

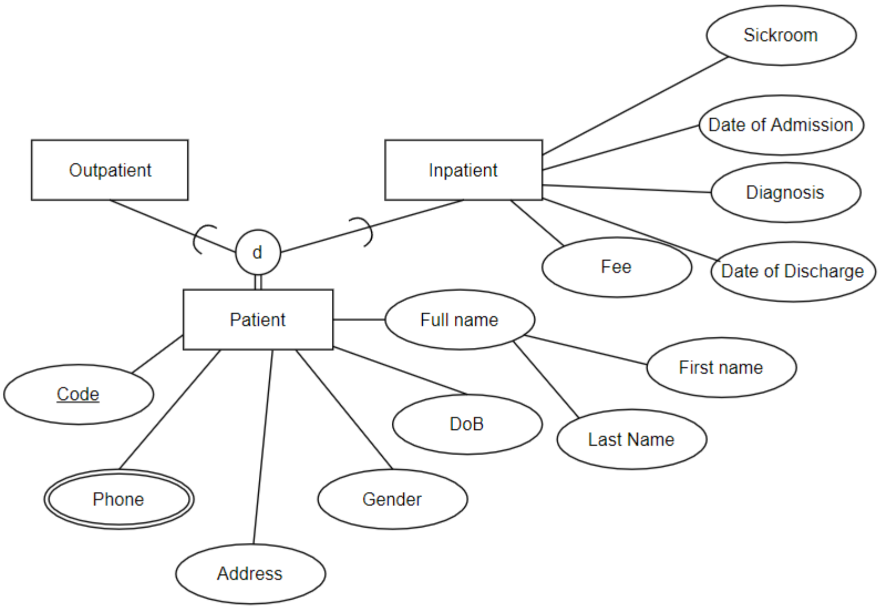


- `Department` : **a unique code**, a title, and a dean who is a doctor. The employees have to belong to a specific department. A department has at least one or many employees. The dean must hold a specific speciality and has had more than 5 years of experience since the date he or she was awarded the speciality degree.

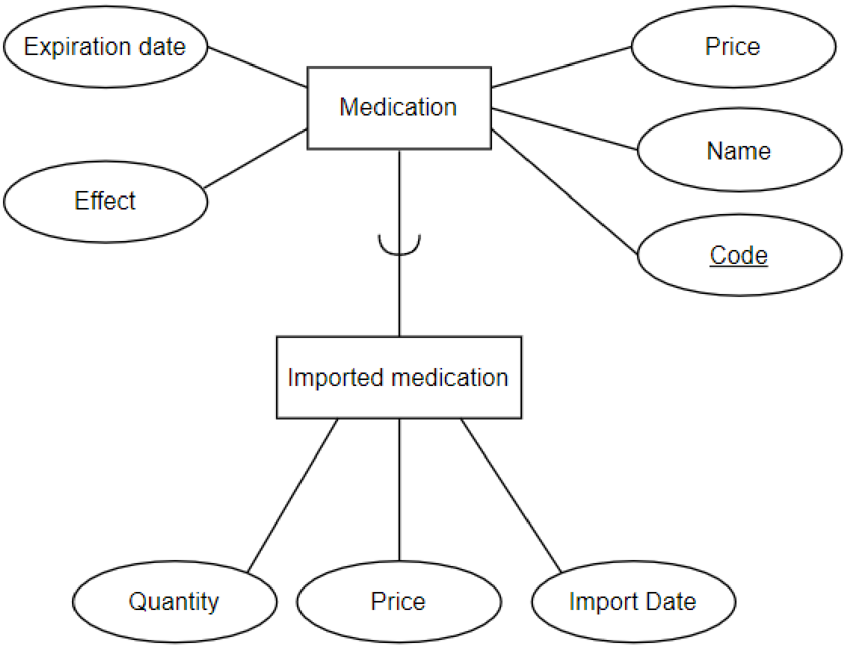


Note: we will explain about a dean and it's relations, conditions later

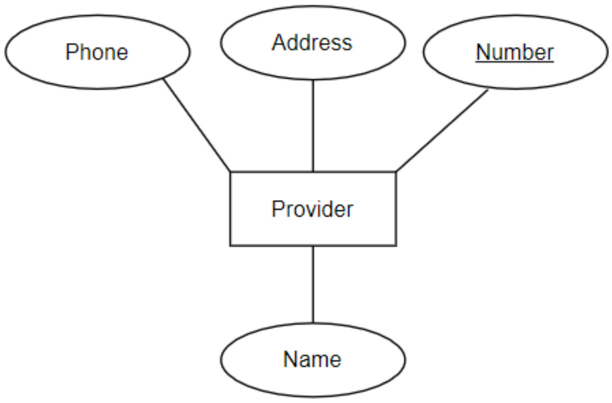
- `Patient` (superclass): provide with the hospital their information such as full name (first name and last name), date of birth, gender, address, and phone number. Patients are divided into two types: outpatients and inpatients. The hospital also wishes to use the first two characters to determine the patient type by **the unique code**.
- `Outpatient` (subclass of patient) : the unique code for him or her starts with "OP," which is then followed by 9 digits such as "OP000000001".
- `Inpatients` (subclass of patient) : the unique code for him or her starts with "IP," which is then followed by 9 digits such as "IP000000001". For inpatients, some information is added such as: date of admission, caring nurse, diagnosis, sickroom, date of discharge, and fee.



- `Medication` (superclass) : the information of a medication is also stored in the database. This information consists of **a unique code** , name of the medication, effects, price, and expiration date.
- `Imported medication` (subclass of Medication) : including imported date, price, and quantity.



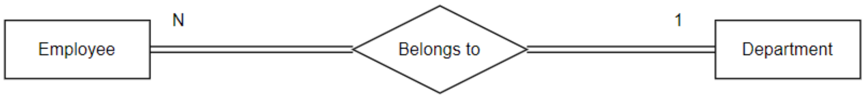
- `Provider` : tracked by its **unique number**, name, address, and phone.



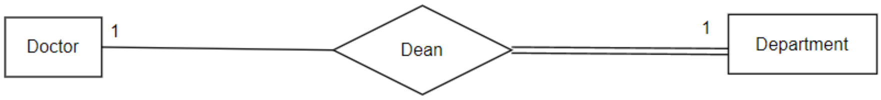
1.2 Relationship, participation constraint and cardinality

Base on requirements of the assignment, and the entities that we listed above, the EER model will have these relationships:

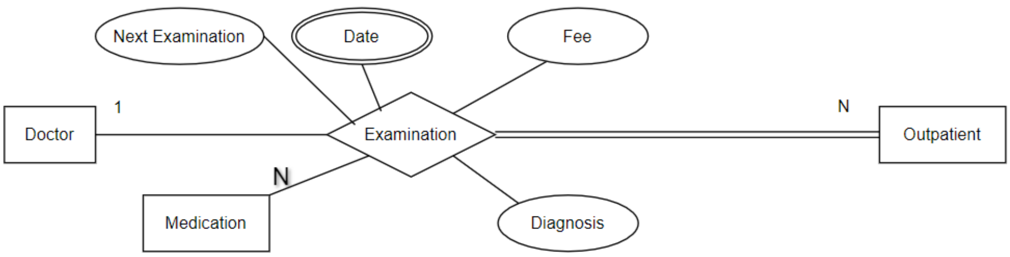
- `Belongs to`: The employees have to belong to a specific department. A department has at least one or many employees.



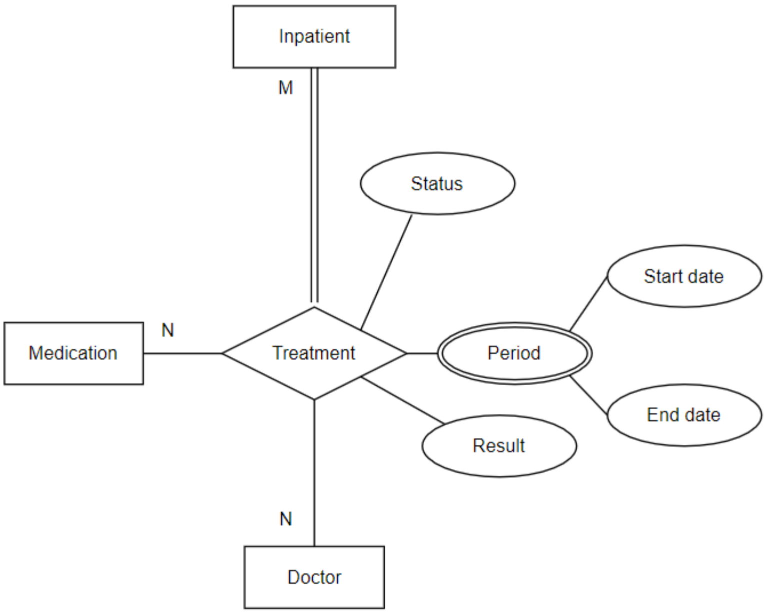
- `Deans`: Each department has a dean who is a doctor.



- `Examination`: The outpatients can have many examinations with their examining doctor. The hospital needs to store the details of each examination such as: examination date, diagnosis, the next examination date if any, medications, and fee.

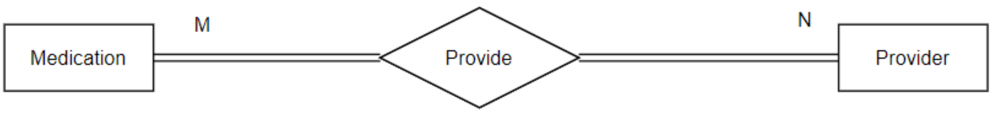


- `Treatment`: a patient can receive treatment from at least one doctor. A doctor can treat many patients at the same time, or sometimes, he has no patients to treat. The hospital needs the details of each treatment such as: treatment period (start date and end date), result, and medications. Here we at the `Status` attribute in order to monitor if the patient is discharged or not, if his or her `Status` is "recovered" then she or he is discharged.



- `Take Care`: Each inpatient is taken care of by a nurse; a nurse can take care of many inpatients at the same time.

- `Provide`: A medication is provided by one or more providers, and one provider may provide many types of medication. Here, we assume that the provider must provide at least 1 medication, therefore, the hospital needs to keep that provider's information.

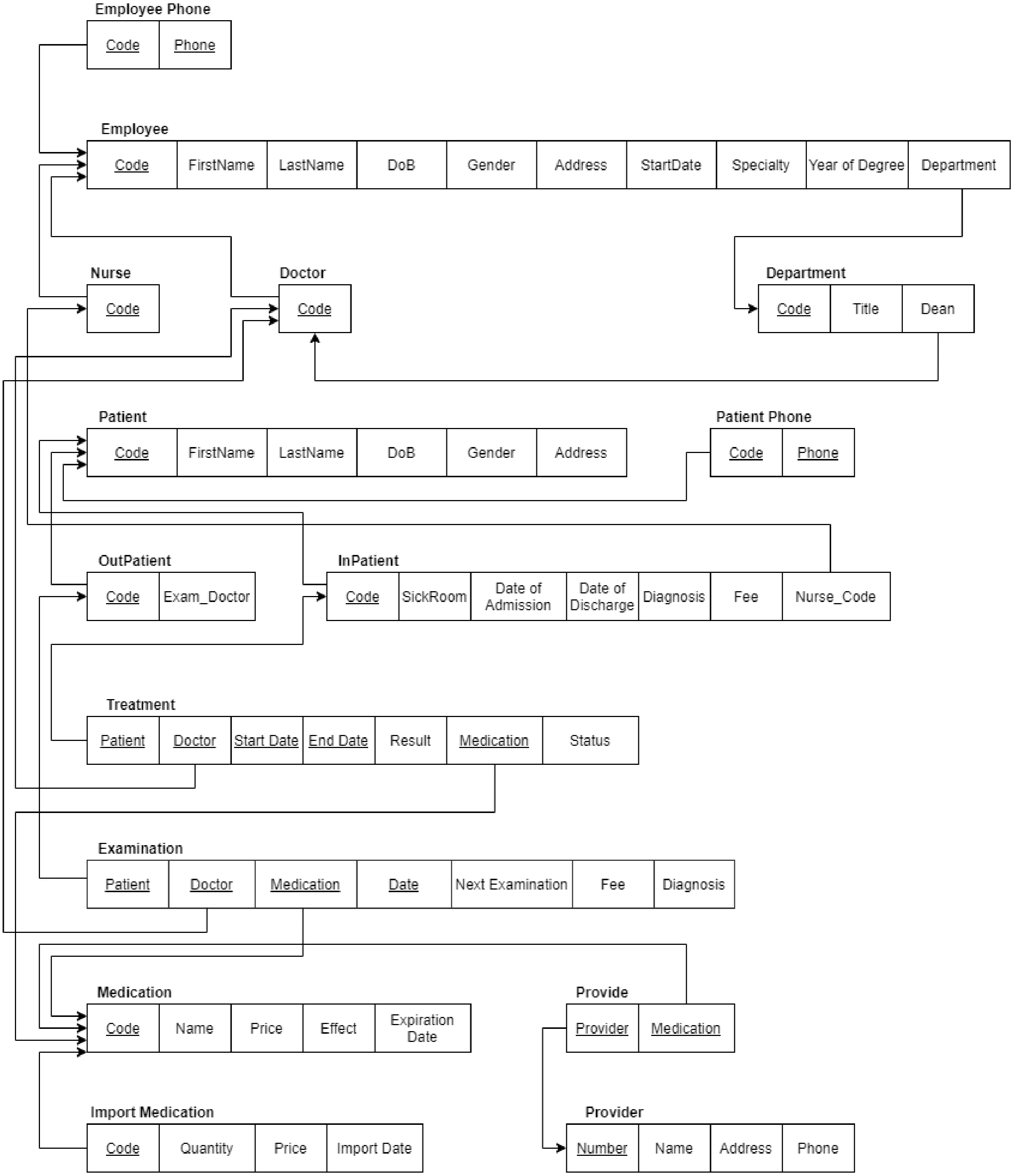


2 Relational database schema

Applying the rules from the book to convert EER to relational schema:

- `Step 1:` Map the strong entity type. For composite variables, we include its components. If it has multiple unique keys then we can choose.
- `Step 2:` Map the weak entity type. The primary key of the weak entity table will be the key attributes of the its owner and its *partial* key if it does possess one.
- `Step 3:` Map 1:1 relationship
- `Step 4:` Map 1:N relationship
- `Step 5:` Map M:N relationship. We must create a new table. The key attribute of that table is the composite of the two tables
- `Step 6:` Map the multi-value attribute. We create a new table. The primary key is the combination of the multi-variable value and its owner entity's primary key
- `Step 7:` Map higher degree relationships
- `Step 8:` Map the generalization/specialization (8A, 8B, 8C, 8D). 8A : Create many table
  8B : Cannot solve overlap
  8C : Cannot solve the overlap
  8D : Create only one table
- `Step 9:` Map the union type

Finally, we obtain the following relational schema:



3 EER semantic constraint and Data Definition Language solution

3.1 SQL Implementation

From the relational database that we have created before, we are able to put all of the schema into practice. Using SQL (structured query language) as DDL (Data Definition Language) and DML (data manipulation language), we could translate these tables into SQL code.
The Employee table:

```
CREATE TABLE Employee(
Code                        VARCHAR(9),
FirstName                   VARCHAR(40),
LastName                    VARCHAR(40),
DoB                         DATE,
Gender                      CHAR,
Address                     VARCHAR(40),
StartDate                   DATE,
Specialty                   VARCHAR(20),
YearOfDeg                   INT,
Department                  VARCHAR(9),
PRIMARY KEY (Code),
FOREIGN KEY (department) REFERENCES Department(code)
);
```

Following the Employee table, we have the Employee_Phone table, Doctor table and Nurse table.

```
CREATE TABLE Employee_Phone(
Code                        VARCHAR(9),
Phone_Num                   VARCHAR(15),
PRIMARY KEY (Code, Phone_Num),
FOREIGN KEY (Code) REFERENCES Employee(Code)
);
CREATE TABLE Doctor(
Code                        VARCHAR(9),
PRIMARY KEY (Code),
FOREIGN KEY(Code) REFERENCES Employee (Code)
);
CREATE TABLE Nurse(
Code                        VARCHAR(9),
PRIMARY KEY (Code),
FOREIGN KEY(Code) REFERENCES Employee (Code)
);
```

Next we create Department table that references to Doctor table.

```
CREATE TABLE Department(
Code                        VARCHAR(9),
Title                       VARCHAR(40),
Dean                        VARCHAR(9),
PRIMARY KEY (Code),
FOREIGN KEY (Dean) REFERENCES Doctor(Code)
);
```

Now, we move to the next table which is the `Patient` table.

```
CREATE TABLE Patient(
Code                        VARCHAR(11),
First_Name                  VARCHAR(40),
Last_Name                   VARCHAR(40),
DoB                         DATE,
Gender                      CHAR,
Address                     VARCHAR(40),
PRIMARY KEY (Code)
);
```

As we can see from the schema, there are two tables that reference to the `Patient` table which are `Out_Patient` table and `In_Patient` table.

```
CREATE TABLE Out_Patient(
Code                        VARCHAR(11),
Exam_Doctor                 VARCHAR(9),
PRIMARY KEY (Code),
FOREIGN KEY (Exam_Doctor) REFERENCES Doctor(Code),
FOREIGN KEY (Code) REFERENCES Patient(Code)
);
CREATE TABLE In_Patient(
Code                        VARCHAR(11),
SickRoom                    VARCHAR(9),
Date_of_Admission           DATE,
Date_of_Discharge           DATE,
Diagnosis                   VARCHAR(40),
Fee                         DECIMAL(10,2),
Nurse_Code                  VARCHAR(9),
PRIMARY KEY (Code),
FOREIGN KEY (Nurse_Code) REFERENCES Nurse(Code),
FOREIGN KEY (Code) REFERENCES Patient(Code)
);
```

We also have the `Patient_Phone` table so store the patient's phone if they have many one. Next, we define the table for `Medication` and `Import_Medication`, the `Import_Medication` should reference to the `Medication`.

```
CREATE TABLE Medication(
Code                        VARCHAR(9),
```

```
Name                        VARCHAR(20),
Price                       DECIMAL(5,2),
Effect                      VARCHAR(100),
Expiration_Date             DATE,
PRIMARY KEY (Code)
);
CREATE TABLE Import_Medication(
Code                        VARCHAR(9),
Quantity                    INT,
Price                       DECIMAL(5,2),
Import_Date                 DATE,
PRIMARY KEY(Code),
FOREIGN KEY (Code) REFERENCES Medication(Code)
);
```

The last entity is `Provider`, the `Provider` table will reference to the `Medication` table.

```
CREATE TABLE PROVIDER (
Provider_Number             INT,
Provider_Name               VARCHAR(40),
Address                     VARCHAR(40),
Phone                       VARCHAR(15),
PRIMARY KEY (Provider_Number)
);
```

Finally, we define the schema's relationship. There are the `Provide` table to describe the N-to-M relationship between `Medication` and `Provider`

```
CREATE TABLE PROVIDE (
Provider                    INT,
Medication                  VARCHAR(9),
PRIMARY KEY (Provider, Medication),
FOREIGN KEY (Provider) REFERENCES Provider(Provider_Number),
FOREIGN KEY (Medication) REFERENCES Medication(Code)
);
```

Next the `Examination` table defines the ternary relationship between `Doctor`, `Out_Patient` and `Medication`.

```
CREATE TABLE Examination(
Patient                     VARCHAR(9),
Doctor                      VARCHAR(9),
Medication                  VARCHAR(9),
DateExam                    Date,
NextExam                    Date                    DEFAULT NULL,
Fee                         DECIMAL(10,2),
Diagnosis                   VARCHAR(40),
PRIMARY KEY (Patient, Doctor, DateExam, MedicationMetionMedicaWtionMedicaWtion),
FOREIGN KEY (Medication) REFERENCES Medication(Code),
```

```
FOREIGN KEY (Doctor) REFERENCES Doctor(Code),
FOREIGN KEY (Patient) REFERENCES Out_Patient(Code)
);
```

Finally, the `Treatment` table describes the ternary relationship between `Doctor`, `In_Patient` and `Medication`.

```
CREATE TABLE Treatment (
Patient                     VARCHAR(9),
Doctor                      VARCHAR(9),
Medication                  VARCHAR(9),
StartDate                   Date,
EndDate                     Date,
Result                      VARCHAR(40),
--Status is used to check if the patient is discharged or not
Status                      VARCHAR(40),
PRIMARY KEY (Patient, Doctor, StartDate, EndDate, Medication),
FOREIGN KEY (Medication) REFERENCES Medication(Code),
FOREIGN KEY (Doctor) REFERENCES Doctor(Code),
FOREIGN KEY (Patient) REFERENCES In_Patient(Code)
);
```

It is important to note that in the above line of codes, there are some *foreign keys* that may cause errors because they refer to the table that has not been created. For example, the `Employee` table reference to the `Department` table and vice versa. To deal with this type of problem, these constraints can be left out of the initial `CREATE TABLE` statement, and then add later using `ALTER TABLE` statement.

3.2 EED Constraints and Solution

There are some constraints in the database that the EED model cannot describe. This is due to the semantic constraint of the EED model.
In the database description, there is a criteria for a doctor to be the dean of a department.

> The dean must hold a specific speciality and has had more than 5 years of experience since the date he or she was awarded the speciality degree.

We are able to solve this overhead by using the `CHECK` statement in `SQL`.

```
-- First, we add the year of the doctor to the department
ALTER TABLE DEPARTMENT ADD deanYear int;
-- We use ALTER statement to add a constraint
ALTER TABLE DEPARTMENT ADD CONSTRAINT checkYEAR CHECK (deanyear > 5);
```

Next, the hospital wish to manage the patient information easily, so they want a constraint that cannot be described in the EED model too.

> If one is an outpatient, the unique code for him or her starts with “OP,” which is then followed by 9 digits such as “OP000000001.” If one is an inpatient, the unique code for him or her starts with “IP,” which is then followed by 9 digits such as “IP000000001.”

We can circumvent this request by making use of the `LIKE` and `WILDCARD` statement in SQL. And similarly, we must add a `CHECK` statement to each table.

```
-- ADD constraints of patient
ALTER TABLE patient ADD CONSTRAINT checkID CHECK (code LIKE 'OP%' OR
            code LIKE 'IP%');
-- ADD constraint of Out_Patient
ALTER TABLE Out_Patient ADD CONSTRAINT checkID CHECK (code LIKE 'OP%');
-- ADD constraint of In_Patient
ALTER TABLE In_Patient ADD CONSTRAINT checkID CHECK (code LIKE 'IP%');
```

Finally, the last constraint that cannot be described by the EED model is when the hospital would like to keep track of the expired medication in the `Medication` table.

> In case one medication is out-of-date, it will be automatically marked so in the database.

Fortunately, the `SQL` language offers us the concept of `TRIGGER` so that we can get rid of this problem.

```
CREATE TRIGGER check_expiration
AFTER INSERT OR UPDATE OF code, Expiration_Date
ON Medication
FOR EACH ROW
WHEN(EXISTS(SELECT *
            FROM import_medication
            WHERE expiration_date - to_date(current_date) > 0))
            HANDLE_EXPIRATION(code);
```

Here the condition is fired when there is the existence of the case the `expiration_dare - current_date` > 0 (It means that the medication has expired). In this case, we invoke the stored execute `HANDLE_EXPIRATION`, we pass in the code as the parameter.

References

[1] Ramez Elmasri. *FUNDAMENTALS OF Database Systems*. Edinburgh Gate, Harlow, Essex CM20 2JE, England: Pearson, 2016.

[2] Nilesh Shah. *Database Systems Using Oracle A Simplified Guide to SQL and PL/SQL*. Upper Saddle River, NJ 07458: Pearson, 2005.