INFO 6210 Data Management and Database Design

Final Report

Stray Animal Rescue Center

Xuan Gao
001220084

Table of Contents

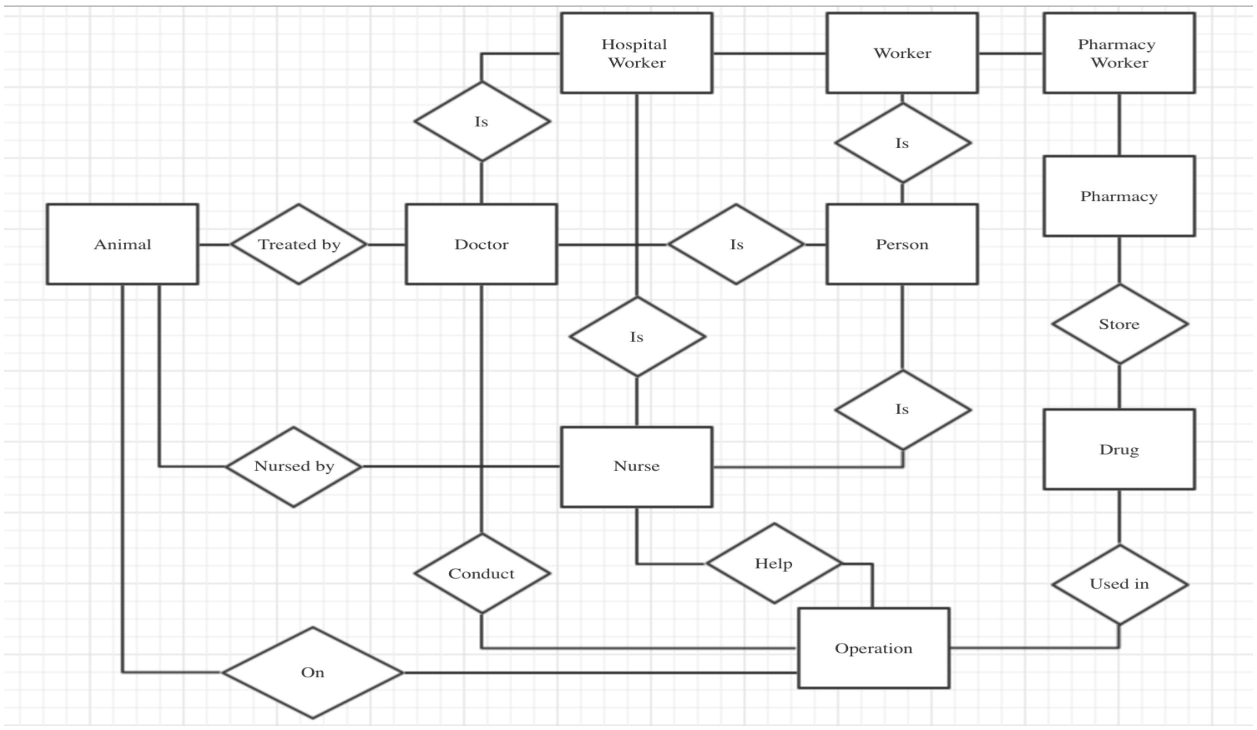
Background

I love animals. I have three dogs in China. They are smart and lovely. However, there are too many stray animals. They need to be treated, recorded, and adapted, so in the final project, I want to make a database system for a stray animal rescue center. It can store the information about stray animals, like its type, age, combined with detailed treatment information, like pharmacy, operation. Also, it can store people's information like doctor, nurse and worker. The database will meet the 3 NF. I will define triggers, views, stored procedures for a better use. It likes a hospital that can treat animals.

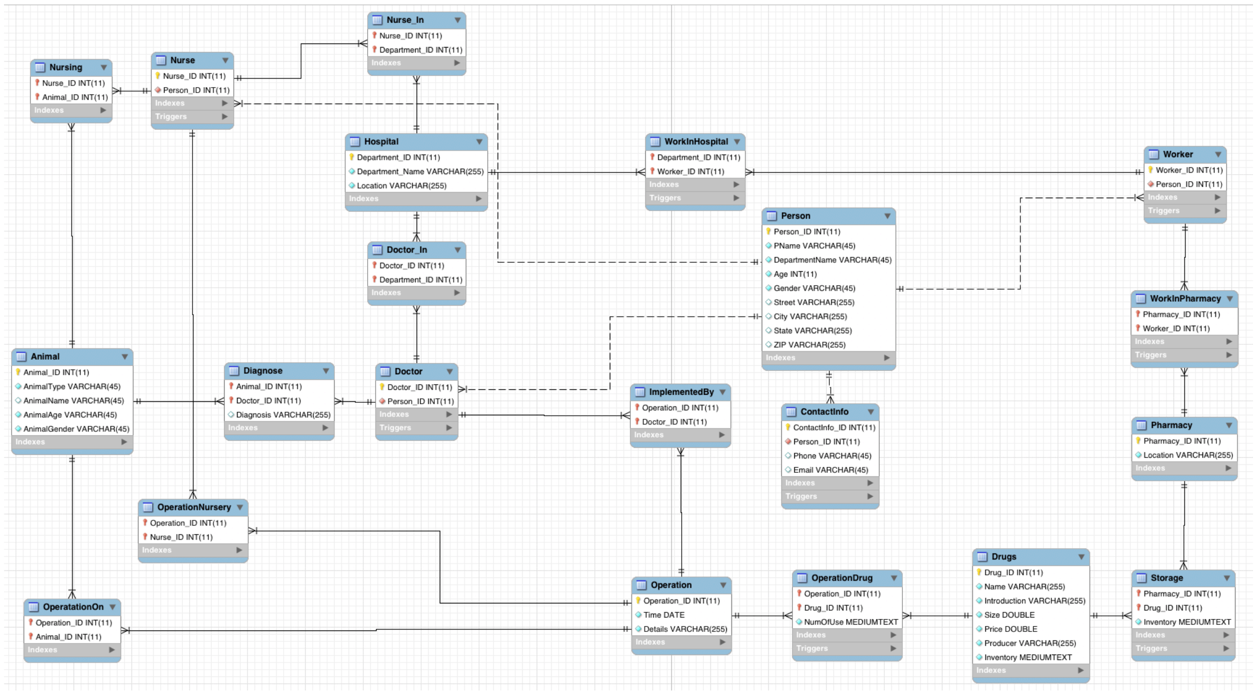
Requirement Analysis

It is required that the system can be put into practical use and meet the basic functional requirements. It requires high reliability, safety and maintainability, and has high portability.
The specific requirements are as follows:
(1) basic information query
The inquiry mainly refers to the inquiries about the information of the doctor, nurse and the animal.
(2) hospital and animal relationship
Like doctor will diagnose animals, nurse will care animals, also they will do operation for animals
(3) medical relationship
In the operation, they need medical supply, and medicine will store in pharmacy.
(4) people management,
We should store people information and divide them into different department

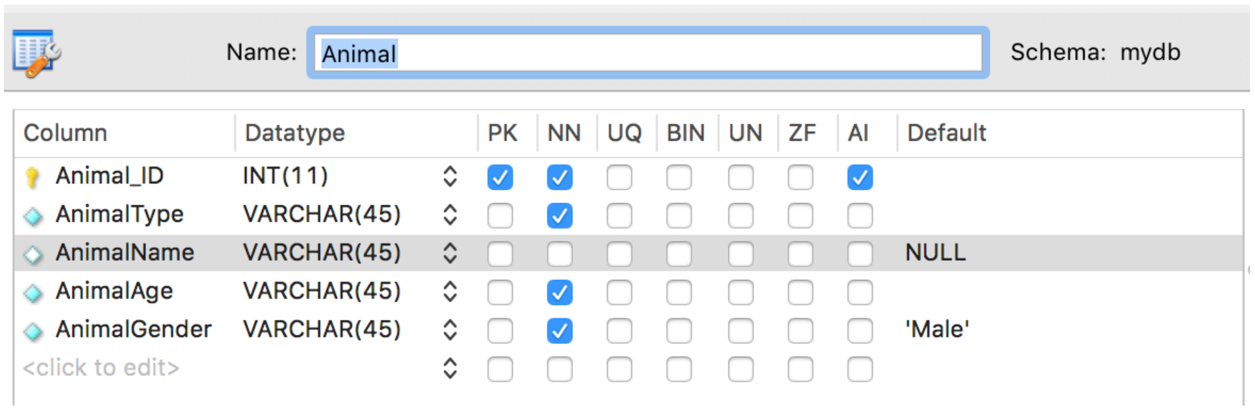
ER Diagram



EER Diagram



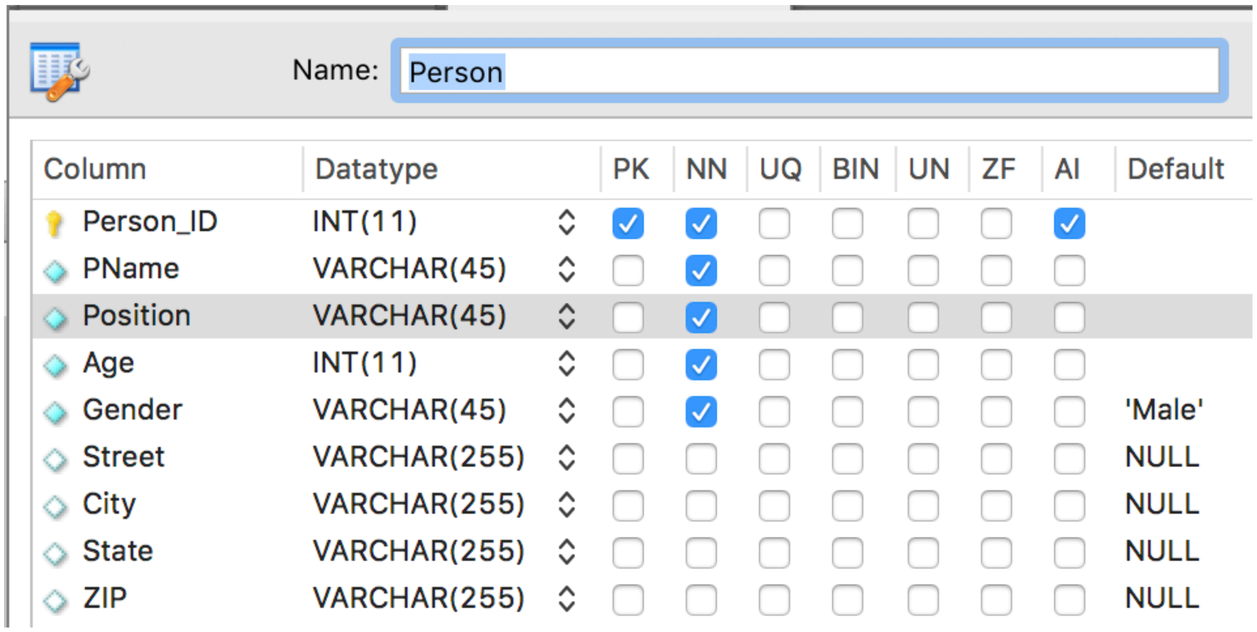
Physical Structure Design



Name: Animal   Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Animal_ID	INT(11)	✓	✓					✓	
AnimalType	VARCHAR(45)		✓						
AnimalName	VARCHAR(45)								NULL
AnimalAge	VARCHAR(45)		✓						
AnimalGender	VARCHAR(45)		✓						'Male'
<click to edit>									

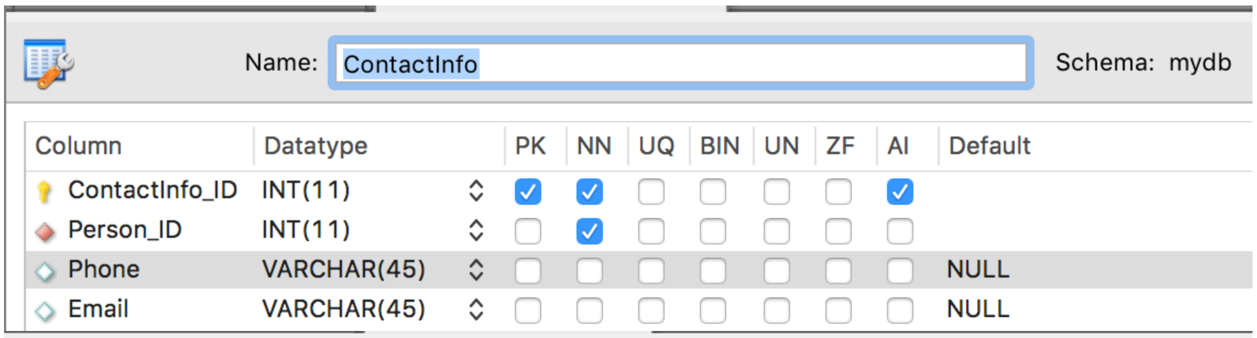
Basic information about animals.



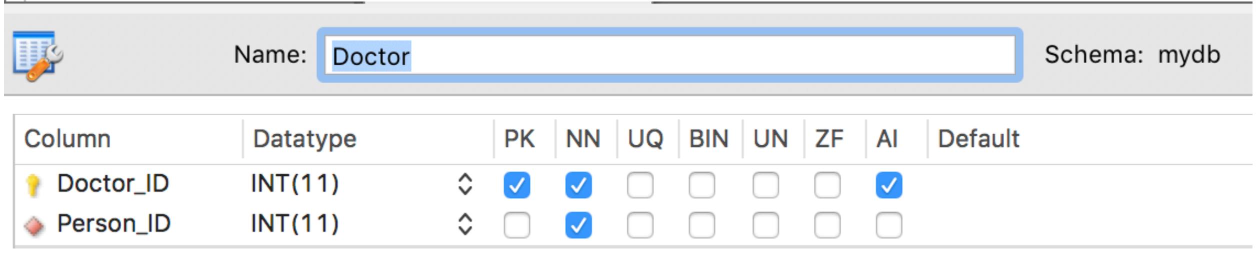
Name: Person

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Person_ID	INT(11)	✓	✓					✓	
PName	VARCHAR(45)		✓						
Position	VARCHAR(45)		✓						
Age	INT(11)		✓						
Gender	VARCHAR(45)		✓						'Male'
Street	VARCHAR(255)								NULL
City	VARCHAR(255)								NULL
State	VARCHAR(255)								NULL
ZIP	VARCHAR(255)								NULL

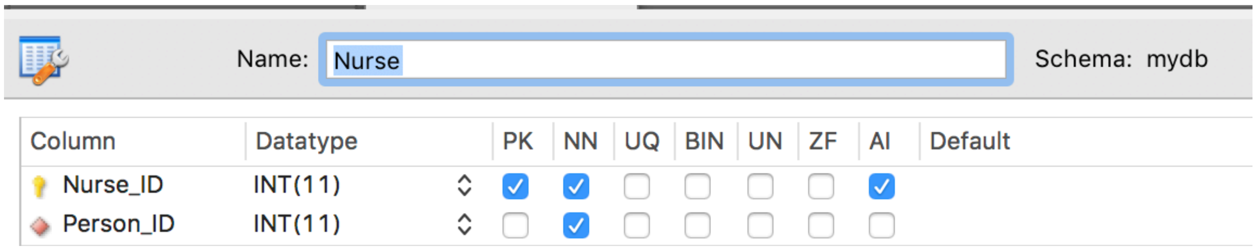
Person is the super type, that all people belongs to person table.
To satisfy 3NF, people can have several contact info, so there is another ContactInfo table.


Name: ContactInfo Schema: mydb

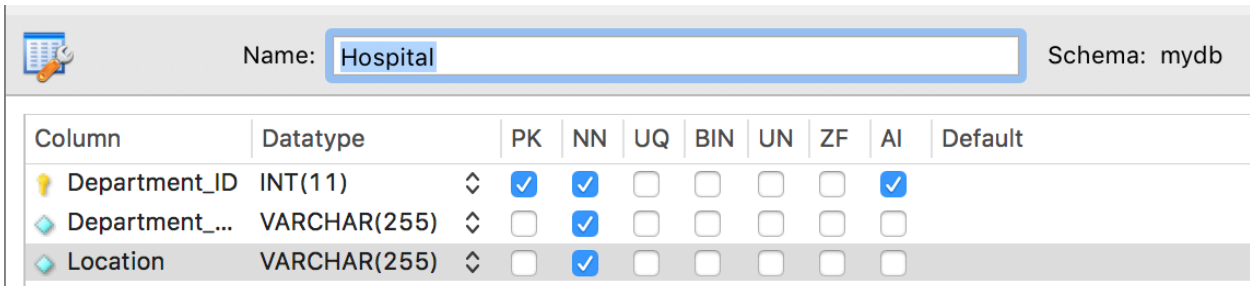
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
ContactInfo_ID	INT(11)	✓	✓					✓	
Person_ID	INT(11)		✓						
Phone	VARCHAR(45)								NULL
Email	VARCHAR(45)								NULL


Name: Doctor Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Doctor_ID	INT(11)	✓	✓					✓	
Person_ID	INT(11)		✓						

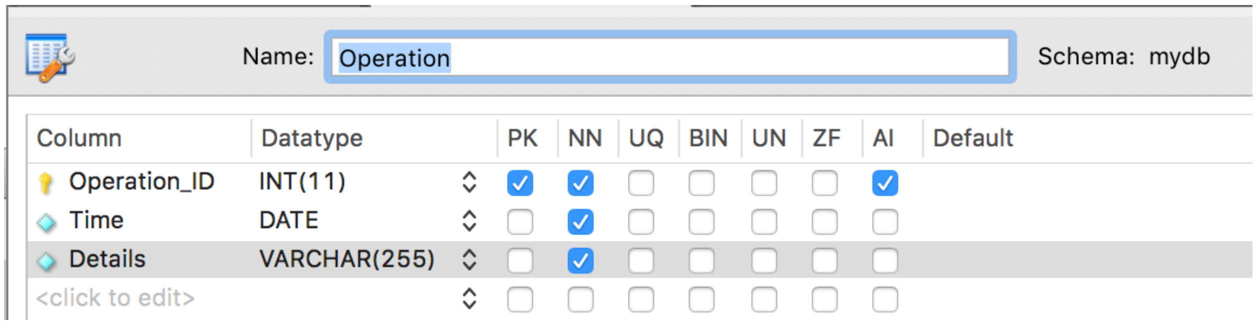

Name: Nurse Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Nurse_ID	INT(11)	✓	✓					✓	
Person_ID	INT(11)		✓						

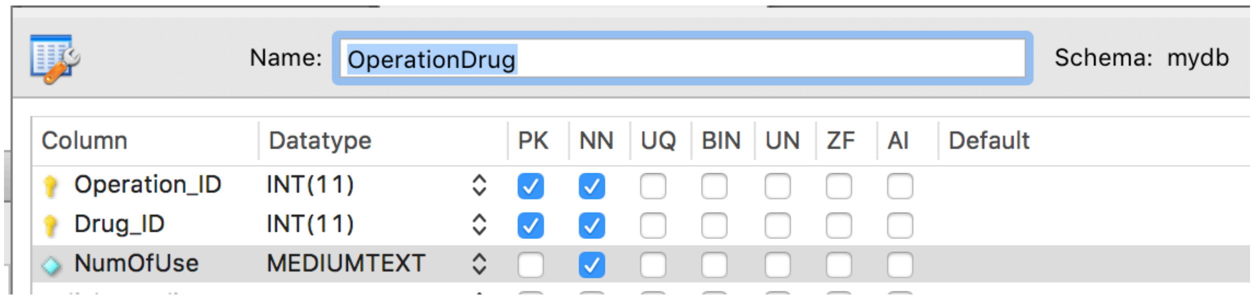

Name: Hospital Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Department_ID	INT(11)	✓	✓					✓	
Department_...	VARCHAR(255)		✓						
Location	VARCHAR(255)		✓						

Doctors and Nurses are in Hospital.

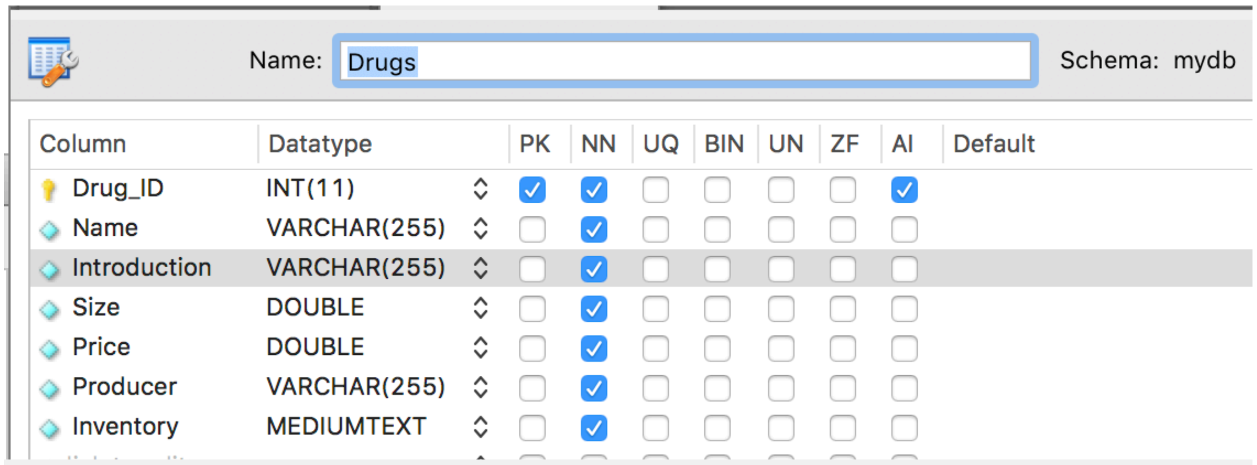

Name: Operation Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Operation_ID	INT(11)	✓	✓					✓	
Time	DATE		✓						
Details	VARCHAR(255)		✓						
<click to edit>									



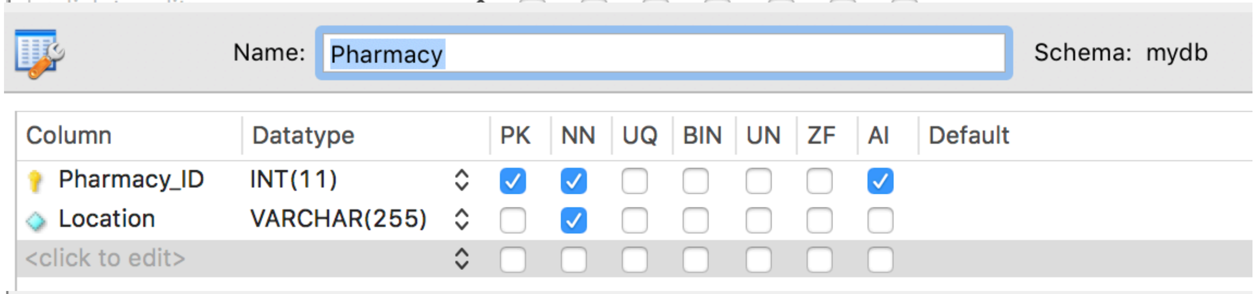
Name: OperationDrug Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Operation_ID	INT(11)	✓	✓						
Drug_ID	INT(11)	✓	✓						
NumOfUse	MEDIUMTEXT		✓						



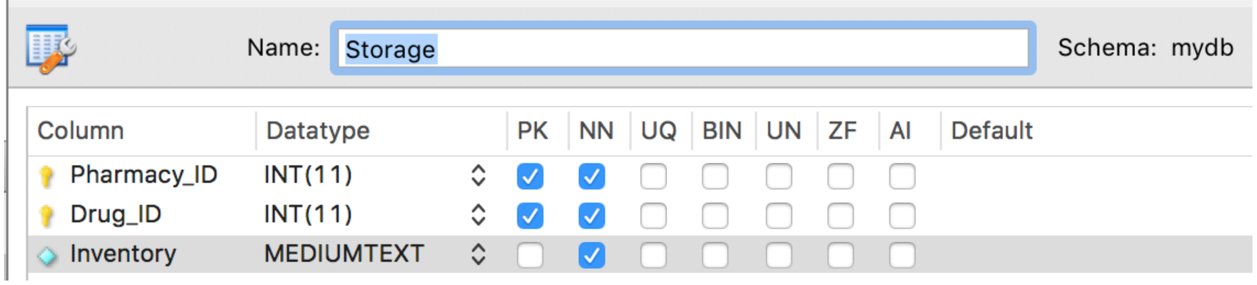
Name: Drugs Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Drug_ID	INT(11)	✓	✓					✓	
Name	VARCHAR(255)		✓						
Introduction	VARCHAR(255)		✓						
Size	DOUBLE		✓						
Price	DOUBLE		✓						
Producer	VARCHAR(255)		✓						
Inventory	MEDIUMTEXT		✓						



Name: Pharmacy Schema: mydb

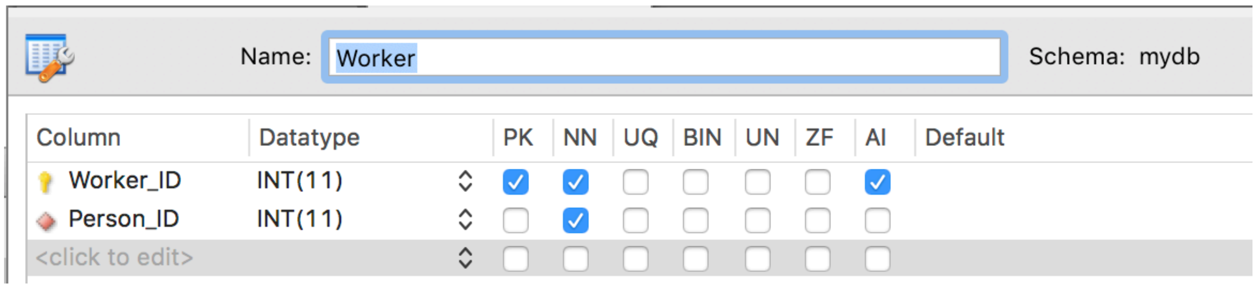
Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Pharmacy_ID	INT(11)	✓	✓					✓	
Location	VARCHAR(255)		✓						
<click to edit>									



Name: Storage Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Pharmacy_ID	INT(11)	✓	✓						
Drug_ID	INT(11)	✓	✓						
Inventory	MEDIUMTEXT		✓						

Drugs are stored in pharmacy, and there is a table to record its storage quantity.

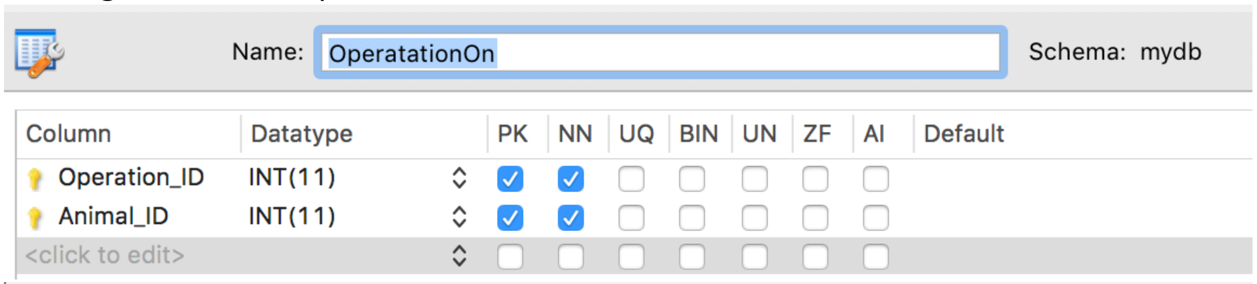


Name: Worker Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Worker_ID	INT(11)	✓	✓					✓	
Person_ID	INT(11)		✓						
<click to edit>									

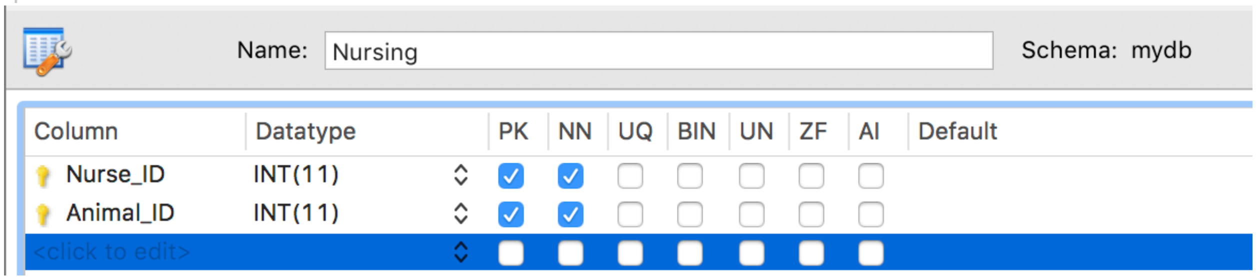
There are two kinds of worker: work in hospital, like doctors and nurses, and work in pharmacy.

Because there will be many-to-many relationship between doctor (nurse) and animal, one doctor(nurse) can treat many animals, and one animal may need many doctors, so there are other tables to show their relationships. Like doctor operation in animals, nurse nursing animals, doctor diagnose animals, nurse nursing animals on operation.



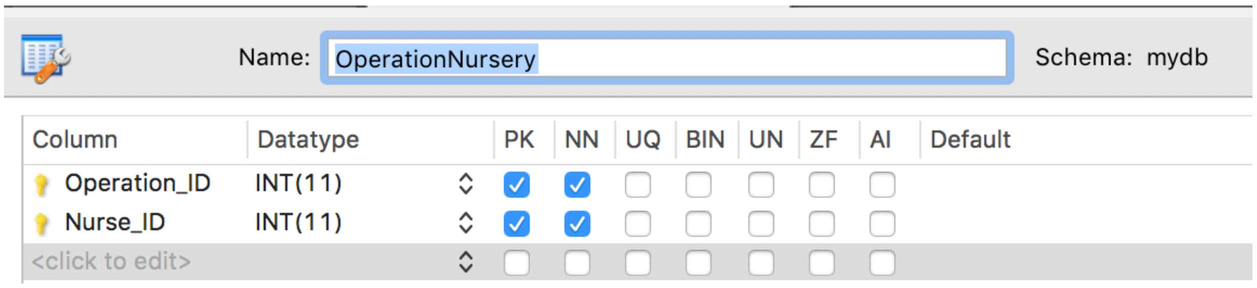
Name: OperatationOn Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Operation_ID	INT(11)	✓	✓						
Animal_ID	INT(11)	✓	✓						
<click to edit>									



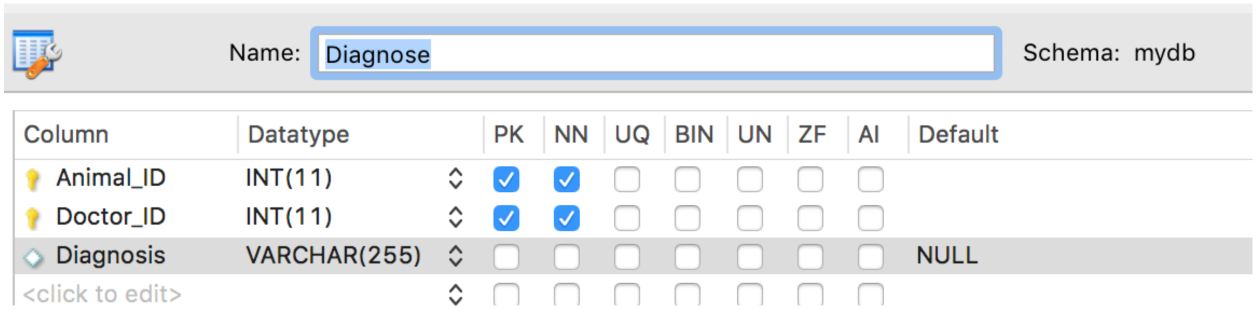
Name: Nursing Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Nurse_ID	INT(11)	✓	✓						
Animal_ID	INT(11)	✓	✓						
<click to edit>									



Name: OperationNursery Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Operation_ID	INT(11)	✓	✓						
Nurse_ID	INT(11)	✓	✓						
<click to edit>									



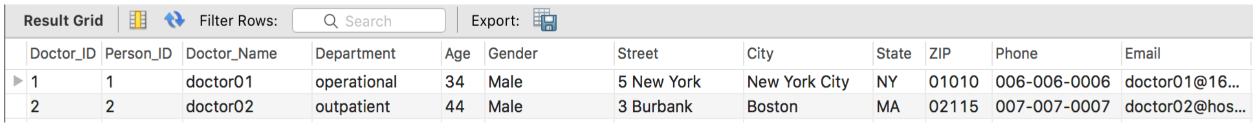
Name: Diagnose Schema: mydb

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
Animal_ID	INT(11)	✓	✓						
Doctor_ID	INT(11)	✓	✓						
Diagnosis	VARCHAR(255)								NULL
<click to edit>									

Views

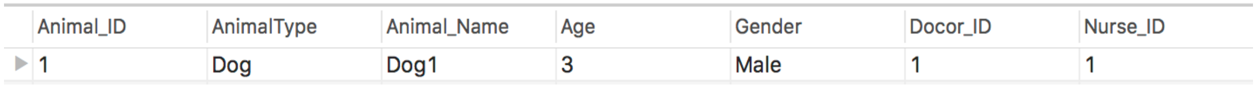
I created 3 views for easier management.

Doctor/Nurse/Worker View to show their whole information



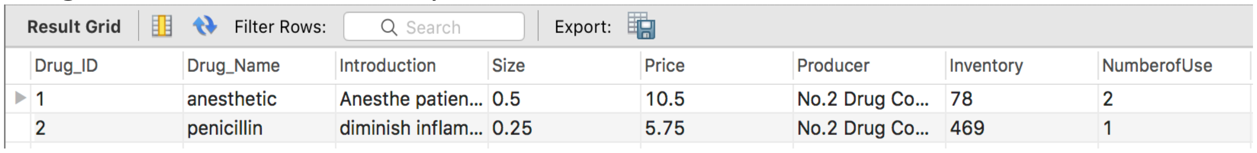
Doctor_ID	Person_ID	Doctor_Name	Department	Age	Gender	Street	City	State	ZIP	Phone	Email
1	1	doctor01	operational	34	Male	5 New York	New York City	NY	01010	006-006-0006	doctor01@16...
2	2	doctor02	outpatient	44	Male	3 Burbank	Boston	MA	02115	007-007-0007	doctor02@hos...

Animal View to show animal's detail information



Animal_ID	AnimalType	Animal_Name	Age	Gender	Docor_ID	Nurse_ID
1	Dog	Dog1	3	Male	1	1

Drug View to show inventory and number of use



Drug_ID	Drug_Name	Introduction	Size	Price	Producer	Inventory	NumberofUse
1	anesthetic	Anesthe patien...	0.5	10.5	No.2 Drug Co...	78	2
2	penicillin	diminish inflam...	0.25	5.75	No.2 Drug Co...	469	1

Triggers

I created several triggers to avoid wrong input/update.

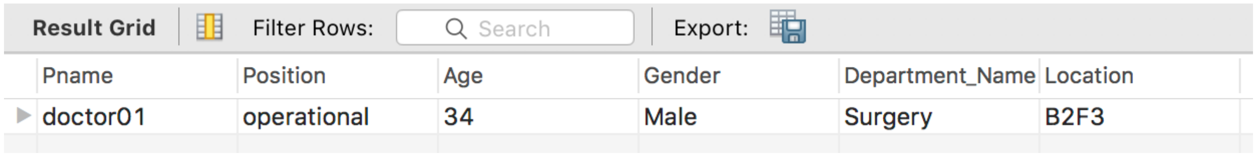
- Before insert/update ContactInfo, check if the email address in a correct form
- Before insert/update WorkinHospital/WorkinPharmacy, check if the worker is correct
- Before insert/update Doctor/Nurse/Worker, check his status that he can only be one position
- Before insert/update Storage, check if new drug storage applies to current inventory

Stored Procedures

I created several stored procedure to manage database easier.

- GetDoctorsDetailswithName(varchar)

call GetDoctorDetailswithName('doctor01');

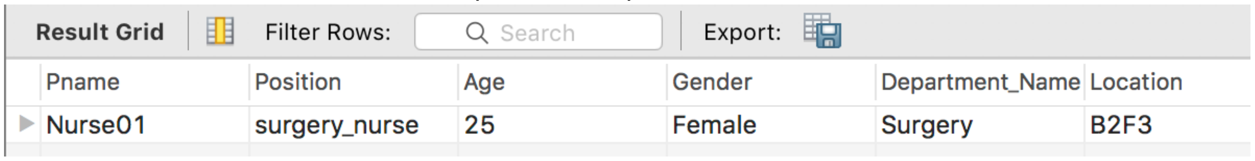


Result Grid | Filter Rows: Search | Export:

Pname	Position	Age	Gender	Department_Name	Location
doctor01	operational	34	Male	Surgery	B2F3

- GetNurseDetailswithName(varchar)

call GetNurseDetailswithName('Nurse01');

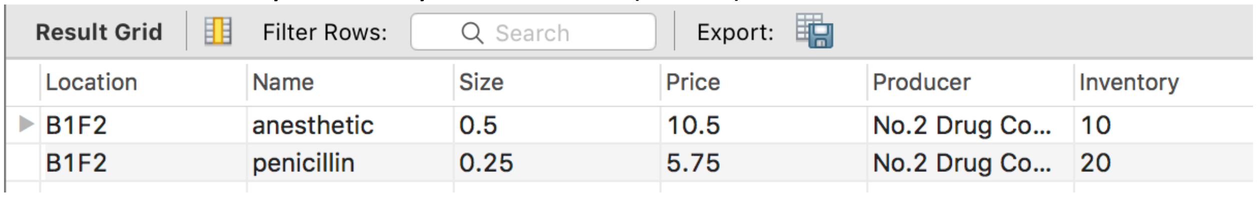


Result Grid | Filter Rows: Search | Export:

Pname	Position	Age	Gender	Department_Name	Location
Nurse01	surgery_nurse	25	Female	Surgery	B2F3

- GetPharmacyInventorywithLocation(varchar)

call GetPharamacyInventorywithLocation('B1F2');

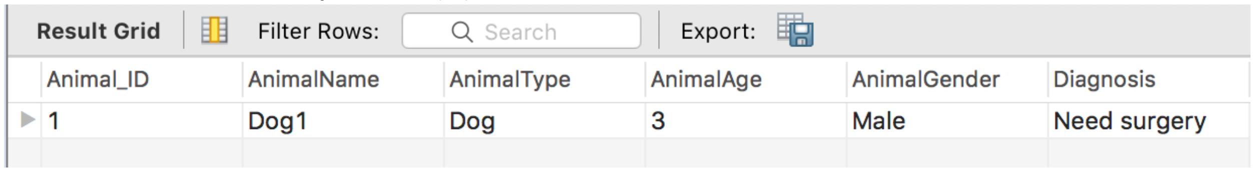


Result Grid | Filter Rows: Search | Export:

Location	Name	Size	Price	Producer	Inventory
B1F2	anesthetic	0.5	10.5	No.2 Drug Co...	10
B1F2	penicillin	0.25	5.75	No.2 Drug Co...	20

- GetAnimalHistorywithID(int)

call GetAnimalHistorywithID(1);

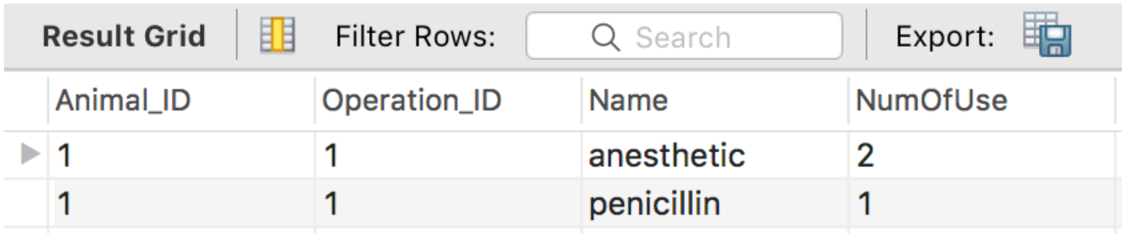


Result Grid | Filter Rows: Search | Export:

Animal_ID	AnimalName	AnimalType	AnimalAge	AnimalGender	Diagnosis
1	Dog1	Dog	3	Male	Need surgery

- GetOperationDrugswithAnimalID(int)

call GetPharamacyInventorywithLocation('B1F2');



Result Grid | Filter Rows: Search | Export:

Animal_ID	Operation_ID	Name	NumOfUse
1	1	anesthetic	2
1	1	penicillin	1

Transactions

When insert/update data, you can use rollback if you insert/update wrong data, and if it is correct, use commit to store data in the database.

Codes

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `mydb` ;
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`Animal`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Animal` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Animal` (
  `Animal_ID` INT(11) NOT NULL AUTO_INCREMENT,
  `AnimalType` VARCHAR(45) NOT NULL,
  `AnimalName` VARCHAR(45) NULL DEFAULT NULL,
  `AnimalAge` VARCHAR(45) NOT NULL,
  `AnimalGender` VARCHAR(45) NOT NULL DEFAULT 'Male',
  PRIMARY KEY (`Animal_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
INSERT INTO `Animal` VALUES (1,'Dog','Dog1',3,'Male'),(2,'Cat','Cat1',1,'Male');

DROP VIEW IF EXISTS `mydb`.`Animal_View` ;
```

```
CREATE VIEW `Animal_View` AS
SELECT  Distinct Animal.Animal_ID Animal_ID, Animal.AnimalType AnimalType,
Animal.AnimalName Animal_Name,Animal.AnimalAge Age,
Animal.AnimalGender Gender,
Diagnose.Doctor_ID Docor_ID, Nursing.Nurse_ID Nurse_ID
FROM Animal, Diagnose, Nursing where
Diagnose.Animal_ID=Animal.Animal_ID AND
Nursing.Animal_ID=Animal.Animal_ID;
SELECT * FROM Animal_View;

-- -----------------------------------------------------
-- Table `mydb`.`Person`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Person` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Person` (
  `Person_ID` INT(11) NOT NULL AUTO_INCREMENT,
  `PName` VARCHAR(45) NOT NULL,
  `Position` VARCHAR(45) NOT NULL,
  `Age` INT(11) NOT NULL,
  `Gender` VARCHAR(45) NOT NULL DEFAULT 'Male',
  `Street` VARCHAR(255) NULL DEFAULT NULL,
  `City` VARCHAR(255) NULL DEFAULT NULL,
  `State` VARCHAR(255) NULL DEFAULT NULL,
  `ZIP` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`Person_ID`))
ENGINE = InnoDB
AUTO_INCREMENT = 4
DEFAULT CHARACTER SET = utf8;

INSERT INTO `Person` VALUES (1,'doctor01','operational',34,'Male','5 New
York','New York City','NY','01010'),(2,'doctor02','outpatient',44,'Male','3
Burbank','Boston','MA','02115'),(3,'Nurse01','surgery_nurse',25,'Female','4
Burbank','Boston','MA','02115'),(4,'Nurse02','outpatient_nurse',33,'Female','5
Burbank','Boston','MA','02115'),(5,'Worker01','phamarcy',45,'Male','360
Huntington','Boston','MA','02115'),(6,'Worker02','department',44,'Female','20
0 Huntington','Boston','MA','02115');
```

```
-- -----------------------------------------------------
-- Table `mydb`.`ContactInfo`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`ContactInfo` ;

CREATE TABLE IF NOT EXISTS `mydb`.`ContactInfo` (
  `ContactInfo_ID` INT(11) NOT NULL AUTO_INCREMENT,
  `Person_ID` INT(11) NOT NULL,
  `Phone` VARCHAR(45) NULL DEFAULT NULL,
  `Email` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`ContactInfo_ID`),
  INDEX `P_ID_idx` (`Person_ID` ASC),
  CONSTRAINT `P_ID`
    FOREIGN KEY (`Person_ID`)
    REFERENCES `mydb`.`Person` (`Person_ID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 5
DEFAULT CHARACTER SET = utf8;


INSERT INTO `ContactInfo` VALUES (1,1,'006-006-
0006','doctor01@163.com'),(2,2,'007-007-
0007','doctor02@hosmail.com'),(3,3,'008-008-
0008','nurse01@gmail.com'),(4,4,'009-009-
0009','nurse02@hotmail.com'),(5,5,'010-010-
0010','worker01@163.com'),(6,6,'011-011-0011','worker02@gmail.com');
#INSERT INTO `ContactInfo` VALUES (8,8,'006-006-0006','doctor01163.com');
#Trigger test, will show wrong email fomat

-- -----------------------------------------------------
-- Table `mydb`.`Hospital`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Hospital` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Hospital` (
  `Department_ID` INT(11) NOT NULL AUTO_INCREMENT,
  `Department_Name` VARCHAR(255) NOT NULL,
  `Location` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`Department_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
SET autocommit=1;
INSERT INTO `Hospital` VALUES (1,'Surgery','B2F3'),(2,'Medicine','B2F2');
delete from hospital where Department_ID=3 or Department_ID=4;
select *from Hospital;
SET autocommit=0;
INSERT INTO `Hospital` VALUES (4,'blabla','B2F1');
select *from Hospital;
rollback;
select *from Hospital;
#INSERT INTO `Hospital` VALUES (3,'blabla','B2F1');
#select *from Hospital;
#commit;
#select *from Hospital;

# test for transaction
-- -----------------------------------------------------
-- Table `mydb`.`Doctor`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Doctor` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Doctor` (
  `Doctor_ID` INT(11) NOT NULL AUTO_INCREMENT,
  `Person_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Doctor_ID`),
  INDEX `Person_ID_idx` (`Person_ID` ASC),
  CONSTRAINT `Person_ID_Doctor`
    FOREIGN KEY (`Person_ID`)
    REFERENCES `mydb`.`Person` (`Person_ID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `Doctor` VALUES (1,1),(2,2);

DROP VIEW IF EXISTS `mydb`.`Doctor_View` ;
CREATE VIEW `Doctor_View` AS
SELECT  Distinct Doctor.Doctor_ID Doctor_ID,Doctor.Person_ID Person_ID,
Person.PName Doctor_Name,Person.Position Position, Person.Age
Age,Person.Gender Gender, Person.Street Street, Person.City City,
Person.State State, Person.ZIP ZIP, ContactInfo.Phone Phone,
ContactInfo.Email Email
FROM Doctor, Person, ContactInfo WHERE
Person.Person_ID=Doctor.Person_ID AND
ContactInfo.Person_ID=Doctor.Person_ID;
SELECT * FROM Doctor_View;

#INSERT INTO `Doctor` VALUES (3,3);
#Trigger test, will show this person is a nurse
-- -----------------------------------------------------
-- Table `mydb`.`Diagnose`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Diagnose` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Diagnose` (
  `Animal_ID` INT(11) NOT NULL,
  `Doctor_ID` INT(11) NOT NULL,
  `Diagnosis` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`Animal_ID`, `Doctor_ID`),
  INDEX `Doctor_ID_idx` (`Doctor_ID` ASC),
  INDEX `Animal_ID_idx` (`Animal_ID` ASC),
  CONSTRAINT `Doctor_ID_Diagnose`
    FOREIGN KEY (`Doctor_ID`)
    REFERENCES `mydb`.`Doctor` (`Doctor_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Animal_ID_Diagnose`
    FOREIGN KEY (`Animal_ID`)
    REFERENCES `mydb`.`Animal` (`Animal_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `Diagnose` VALUES (1,1,'Need surgery'),(2,2,'Diagnosis01');
-- -----------------------------------------------------
-- Table `mydb`.`Doctor_In`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Doctor_In` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Doctor_In` (
  `Doctor_ID` INT(11) NOT NULL,
  `Department_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Doctor_ID`, `Department_ID`),
  INDEX `Dep_ID_idx` (`Department_ID` ASC),
  CONSTRAINT `Dep_ID_DocIn`
    FOREIGN KEY (`Department_ID`)
    REFERENCES `mydb`.`Department` (`Department_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Doc_ID_DocIn`
    FOREIGN KEY (`Doctor_ID`)
    REFERENCES `mydb`.`Doctor` (`Doctor_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
INSERT INTO `Doctor_In` VALUES (1,1),(2,2);

-- -----------------------------------------------------
-- Table `mydb`.`Drugs`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Drugs` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Drugs` (
  `Drug_ID` INT(11) NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(255) NOT NULL,
  `Introduction` VARCHAR(255) NOT NULL,
  `Size` DOUBLE NOT NULL,
  `Price` DOUBLE NOT NULL,
  `Producer` VARCHAR(255) NOT NULL,
  `Inventory` MEDIUMTEXT NOT NULL,
  PRIMARY KEY (`Drug_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


INSERT INTO `Drugs` VALUES (1,'anesthetic','Anesthe patient during
operation',0.5,10.5,'No.2 Drug Company','78'),(2,'penicillin','diminish
inflammation',0.25,5.75,'No.2 Drug Company','469'),(3,'plasma type O','plasma
for operation',250,25.25,'No.2 Drug Company','0');

DROP VIEW IF EXISTS `mydb`.`Drug_View` ;
CREATE VIEW `Drug_View` AS
SELECT  Drugs.Drug_ID Drug_ID, Drugs.Name Drug_Name, Drugs.Introduction
Introduction, Drugs.Size Size, Drugs.Price Price,Drugs.Producer Producer,
Drugs.Inventory Inventory, OperationDrug.NumOfUse NumberofUse
FROM Drugs, OperationDrug WHERE Drugs.Drug_ID=OperationDrug.Drug_ID;
SELECT * FROM Drug_View;
-- -----------------------------------------------------
-- Table `mydb`.`Operation`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Operation` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Operation` (
  `Operation_ID` INT(11) NOT NULL AUTO_INCREMENT,
  `Time` DATE NOT NULL,
  `Details` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`Operation_ID`))
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8;

INSERT INTO `Operation` VALUES (1,'2017-01-31','Surgery on patient01');
-- -----------------------------------------------------
-- Table `mydb`.`ImplementedBy`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`ImplementedBy` ;

CREATE TABLE IF NOT EXISTS `mydb`.`ImplementedBy` (
  `Operation_ID` INT(11) NOT NULL,
  `Doctor_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Operation_ID`, `Doctor_ID`),
  INDEX `Doc_ID_idx` (`Doctor_ID` ASC),
  CONSTRAINT `Doc_ID_ImpBy`
    FOREIGN KEY (`Doctor_ID`)
    REFERENCES `mydb`.`Doctor` (`Doctor_ID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `OP_ID_ImpBy`
    FOREIGN KEY (`Operation_ID`)
    REFERENCES `mydb`.`Operation` (`Operation_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
INSERT INTO `ImplementedBy` VALUES (1,1);

-- -----------------------------------------------------
-- Table `mydb`.`MedicalSupply`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`MedicalSupply` ;

CREATE TABLE IF NOT EXISTS `mydb`.`MedicalSupply` (
  `MedicalSupply_ID` INT(11) NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(255) NOT NULL,
  `Usage` VARCHAR(255) NOT NULL,
  `Size` DOUBLE NOT NULL,
  `Producer` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`MedicalSupply_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `MedicalSupply` VALUES (1,'injector','inject',5,'No.1 Medical
Supply'),(2,'scalpel','surgery',25.5,'No.1 Medical Supply');
-- -----------------------------------------------------
-- Table `mydb`.`Nurse`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Nurse` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Nurse` (
  `Nurse_ID` INT(11) NOT NULL AUTO_INCREMENT,
  `Person_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Nurse_ID`),
  INDEX `Person_ID_idx` (`Person_ID` ASC),
  CONSTRAINT `Person_ID_Nurse`
    FOREIGN KEY (`Person_ID`)
    REFERENCES `mydb`.`Person` (`Person_ID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `Nurse` VALUES (1,3),(2,4);
#INSERT INTO `Nurse` VALUES (3,1);
#trigger test

DROP VIEW IF EXISTS `mydb`.`Nurse_View` ;
CREATE VIEW `Nurse_View` AS
SELECT  Distinct Nurse.Nurse_ID Nurse_ID,Nurse.Person_ID Person_ID,
Person.PName Nurse_Name,Person.Position Position, Person.Age
Age,Person.Gender Gender, Person.Street Street, Person.City City,
Person.State State, Person.ZIP ZIP, ContactInfo.Phone Phone,
ContactInfo.Email Email
```

```
FROM Nurse, Person, ContactInfo WHERE Person.Person_ID=Nurse.Person_ID
AND ContactInfo.Person_ID=Nurse.Person_ID;
SELECT * FROM Nurse_View;
-- -----------------------------------------------------
-- Table `mydb`.`Nurse_In`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Nurse_In` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Nurse_In` (
  `Nurse_ID` INT(11) NOT NULL,
  `Department_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Nurse_ID`, `Department_ID`),
  INDEX `Dep_ID_idx` (`Department_ID` ASC),
  CONSTRAINT `Dep_ID_nurIn`
    FOREIGN KEY (`Department_ID`)
    REFERENCES `mydb`.`Department` (`Department_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Nur_ID_nurIn`
    FOREIGN KEY (`Nurse_ID`)
    REFERENCES `mydb`.`Nurse` (`Nurse_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `Nurse_In` VALUES (1,1),(2,2);
-- -----------------------------------------------------
-- Table `mydb`.`Nursing`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Nursing` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Nursing` (
  `Nurse_ID` INT(11) NOT NULL,
  `Animal_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Nurse_ID`, `Animal_ID`),
  INDEX `Nurse_ID_idx` (`Nurse_ID` ASC),
  INDEX `Animal_ID_idx` (`Animal_ID` ASC),
  CONSTRAINT `Nurse_ID_Nursing`
    FOREIGN KEY (`Nurse_ID`)
    REFERENCES `mydb`.`Nurse` (`Nurse_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Animal_ID_Nursing`
    FOREIGN KEY (`Animal_ID`)
    REFERENCES `mydb`.`Animal` (`Animal_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `Nursing` VALUES (1,1),(2,2);
-- -----------------------------------------------------
-- Table `mydb`.`OperatationOn`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`OperatationOn` ;

CREATE TABLE IF NOT EXISTS `mydb`.`OperatationOn` (
  `Operation_ID` INT(11) NOT NULL,
  `Animal_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Operation_ID`, `Animal_ID`),
  INDEX `Pat_ID_idx` (`Animal_ID` ASC),
  CONSTRAINT `Op_ID_opOn`
    FOREIGN KEY (`Operation_ID`)
    REFERENCES `mydb`.`Operation` (`Operation_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Pat_ID_OpOn`
    FOREIGN KEY (`Animal_ID`)
    REFERENCES `mydb`.`Animal` (`Animal_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

```
INSERT INTO `OperatationOn` VALUES (1,1);
-- -----------------------------------------------------
-- Table `mydb`.`OperationDrug`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`OperationDrug` ;

CREATE TABLE IF NOT EXISTS `mydb`.`OperationDrug` (
  `Operation_ID` INT(11) NOT NULL,
  `Drug_ID` INT(11) NOT NULL,
  `NumOfUse` MEDIUMTEXT NOT NULL,
  PRIMARY KEY (`Operation_ID`, `Drug_ID`),
  INDEX `Drug_ID_idx` (`Drug_ID` ASC),
  CONSTRAINT `Drug_ID_OpDrug`
    FOREIGN KEY (`Drug_ID`)
    REFERENCES `mydb`.`Drugs` (`Drug_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Op_ID_OpDrug`
    FOREIGN KEY (`Operation_ID`)
    REFERENCES `mydb`.`Operation` (`Operation_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `OperationDrug` VALUES (1,1,'2'),(1,2,'1');
#UPDATE OperationDrug SET NumOfUse ='30' WHERE Drug_ID=1;
#Trigger test, show Not valid drug inventory
-- -----------------------------------------------------
-- Table `mydb`.`OperationNursery`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`OperationNursery` ;

CREATE TABLE IF NOT EXISTS `mydb`.`OperationNursery` (
  `Operation_ID` INT(11) NOT NULL,
  `Nurse_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Operation_ID`, `Nurse_ID`),
  INDEX `Nur_ID_idx` (`Nurse_ID` ASC),
  CONSTRAINT `Nur_ID_OpNur`
    FOREIGN KEY (`Nurse_ID`)
    REFERENCES `mydb`.`Nurse` (`Nurse_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Op_ID_OpNur`
    FOREIGN KEY (`Operation_ID`)
    REFERENCES `mydb`.`Operation` (`Operation_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `OperationNursery` VALUES (1,1);
-- -----------------------------------------------------
-- Table `mydb`.`OperationSupply`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`OperationSupply` ;

CREATE TABLE IF NOT EXISTS `mydb`.`OperationSupply` (
  `Operation_ID` INT(11) NOT NULL,
  `MedicalSupply_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Operation_ID`, `MedicalSupply_ID`),
  INDEX `Med_ID_idx` (`MedicalSupply_ID` ASC),
  CONSTRAINT `Med_ID_OpSup`
    FOREIGN KEY (`MedicalSupply_ID`)
    REFERENCES `mydb`.`MedicalSupply` (`MedicalSupply_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Op_ID_OpSup`
    FOREIGN KEY (`Operation_ID`)
    REFERENCES `mydb`.`Operation` (`Operation_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8;

INSERT INTO `OperationSupply` VALUES (1,1),(1,2);
-- -----------------------------------------------------
-- Table `mydb`.`Pharmacy`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Pharmacy` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Pharmacy` (
  `Pharmacy_ID` INT(11) NOT NULL AUTO_INCREMENT,
  `Location` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`Pharmacy_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `Pharmacy` VALUES (1,'B1F2'),(2,'B2F3');
-- -----------------------------------------------------
-- Table `mydb`.`Storage`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Storage` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Storage` (
  `Pharmacy_ID` INT(11) NOT NULL,
  `Drug_ID` INT(11) NOT NULL,
  `Inventory` MEDIUMTEXT NOT NULL,
  PRIMARY KEY (`Pharmacy_ID`, `Drug_ID`),
  INDEX `Pharmacy_ID_idx` (`Pharmacy_ID` ASC),
  INDEX `Drug_ID_idx` (`Drug_ID` ASC),
  CONSTRAINT `Drug_ID_Storage`
    FOREIGN KEY (`Drug_ID`)
    REFERENCES `mydb`.`Drugs` (`Drug_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Pharmacy_ID_Storage`
    FOREIGN KEY (`Pharmacy_ID`)
    REFERENCES `mydb`.`Pharmacy` (`Pharmacy_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `Storage` VALUES (1,1,'10'),(1,2,'20'),(2,1,'10'),(2,2,'10');

-- -----------------------------------------------------
-- Table `mydb`.`UseByDoc`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`UseByDoc` ;

CREATE TABLE IF NOT EXISTS `mydb`.`UseByDoc` (
  `Doctor_ID` INT(11) NOT NULL,
  `MedicalSupply_ID` INT(11) NOT NULL,
  PRIMARY KEY (`MedicalSupply_ID`, `Doctor_ID`),
  INDEX `Med_ID_idx` (`MedicalSupply_ID` ASC),
  INDEX `Doc_ID_UseByDoc` (`Doctor_ID` ASC),
  CONSTRAINT `Doc_ID_UseByDoc`
    FOREIGN KEY (`Doctor_ID`)
    REFERENCES `mydb`.`Doctor` (`Doctor_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Med_ID_UseByDoc`
    FOREIGN KEY (`MedicalSupply_ID`)
    REFERENCES `mydb`.`MedicalSupply` (`MedicalSupply_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
INSERT INTO `UseByDoc` VALUES (2,1),(1,2);

-- -----------------------------------------------------
-- Table `mydb`.`UseByNurse`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`UseByNurse` ;

CREATE TABLE IF NOT EXISTS `mydb`.`UseByNurse` (
  `Nurse_ID` INT(11) NOT NULL,
  `MedicalSupply_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Nurse_ID`, `MedicalSupply_ID`),
  INDEX `Medic_ID_idx` (`MedicalSupply_ID` ASC),
  CONSTRAINT `Medic_ID_UseByNurse`
    FOREIGN KEY (`MedicalSupply_ID`)
    REFERENCES `mydb`.`MedicalSupply` (`MedicalSupply_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Nur_ID_UseByNurese`
    FOREIGN KEY (`Nurse_ID`)
    REFERENCES `mydb`.`Nurse` (`Nurse_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `UseByNurse` VALUES (1,1),(2,1);
-- -----------------------------------------------------
-- Table `mydb`.`Worker`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Worker` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Worker` (
  `Worker_ID` INT(11) NOT NULL AUTO_INCREMENT,
  `Person_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Worker_ID`),
  INDEX `Person_ID_idx` (`Person_ID` ASC),
  CONSTRAINT `Person_ID_Worker`
    FOREIGN KEY (`Person_ID`)
    REFERENCES `mydb`.`Person` (`Person_ID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `Worker` VALUES (1,5),(2,6);
```

```
#INSERT INTO `Worker` VALUES (3,4);
#trigger test

DROP VIEW IF EXISTS `mydb`.`Worker_View` ;
CREATE VIEW `Worker_View` AS
SELECT  Distinct Worker.Worker_ID Worker_ID,Worker.Person_ID Person_ID,
Person.PName Worker_Name,Person.Position Position, Person.Age
Age,Person.Gender Gender, Person.Street Street, Person.City City,
Person.State State, Person.ZIP ZIP, ContactInfo.Phone Phone,
ContactInfo.Email Email
FROM Worker, Person, ContactInfo WHERE
Person.Person_ID=Worker.Person_ID AND
ContactInfo.Person_ID=Worker.Person_ID;
SELECT * FROM Worker_View;
-- -----------------------------------------------------
-- Table `mydb`.`WorkInHospital`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`WorkInHospital` ;

CREATE TABLE IF NOT EXISTS `mydb`.`WorkInHospital` (
  `Department_ID` INT(11) NOT NULL,
  `Worker_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Department_ID`, `Worker_ID`),
  INDEX `Department_ID_idx` (`Department_ID` ASC),
  INDEX `Worker_ID_idx` (`Worker_ID` ASC),
  CONSTRAINT `Department_ID_InDepartment`
    FOREIGN KEY (`Department_ID`)
    REFERENCES `mydb`.`Department` (`Department_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Worker_ID_InDepartment`
    FOREIGN KEY (`Worker_ID`)
    REFERENCES `mydb`.`Worker` (`Worker_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8;

INSERT INTO `WorkInHospital` VALUES (1,1);
#INSERT INTO `WorkInHospital` VALUES (2,2);
#Trigeer test, will show worker already work in pharmacy

-- -----------------------------------------------------
-- Table `mydb`.`WorkInPharmacy`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`WorkInPharmacy` ;

CREATE TABLE IF NOT EXISTS `mydb`.`WorkInPharmacy` (
  `Pharmacy_ID` INT(11) NOT NULL,
  `Worker_ID` INT(11) NOT NULL,
  PRIMARY KEY (`Pharmacy_ID`, `Worker_ID`),
  INDEX `Pharmacy_ID_idx` (`Pharmacy_ID` ASC),
  INDEX `Worker_ID_idx` (`Worker_ID` ASC),
  CONSTRAINT `Pharmacy_ID_InPharmacy`
    FOREIGN KEY (`Pharmacy_ID`)
    REFERENCES `mydb`.`Pharmacy` (`Pharmacy_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Worker_ID_InPharmacy`
    FOREIGN KEY (`Worker_ID`)
    REFERENCES `mydb`.`Worker` (`Worker_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

INSERT INTO `WorkInPharmacy` VALUES (2,2);
#INSERT INTO `WorkInPharmacy` VALUES (1,1);
#trigger test

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

```
USE `mydb`;

DELIMITER $$

USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`ContactInfo_BEFORE_INSERT` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`ContactInfo_BEFORE_INSERT`
BEFORE INSERT ON `mydb`.`ContactInfo`
FOR EACH ROW
BEGIN
if new.Email not like '%_@%_._%'
then
signal sqlstate value '40001'
set message_text = 'Email address is not valid';
end if;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`ContactInfo_BEFORE_UPDATE` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`ContactInfo_BEFORE_UPDATE`
BEFORE UPDATE ON `mydb`.`ContactInfo`
FOR EACH ROW
BEGIN
if new.Email not like '%_@%_._%'
then
signal sqlstate value '40002'
set message_text = 'Email address is not valid';
end if;
END$$
```

```
USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`Doctor_BEFORE_INSERT` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`Doctor_BEFORE_INSERT`
BEFORE INSERT ON `mydb`.`Doctor`
FOR EACH ROW
BEGIN
if exists (select * from Worker where
new.Person_ID = Worker.Person_ID)
then
signal sqlstate value '40032'
set message_text = 'This person is a worker';
end if;
if exists (select * from Nurse where
new.Person_ID = Nurse.Person_ID)
then
signal sqlstate value '40033'
set message_text = 'This person is a nurse';
end if;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`Doctor_BEFORE_UPDATE` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`Doctor_BEFORE_UPDATE`
BEFORE UPDATE ON `mydb`.`Doctor`
FOR EACH ROW
BEGIN
if exists (select * from Worker where
new.Person_ID = Worker.Person_ID)
then
```

```
signal sqlstate value '40036'
set message_text = 'This person is a worker';
end if;
if exists (select * from Nurse where
new.Person_ID = Nurse.Person_ID)
then
signal sqlstate value '40037'
set message_text = 'This person is a nurse';
end if;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`Nurse_BEFORE_INSERT` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`Nurse_BEFORE_INSERT`
BEFORE INSERT ON `mydb`.`Nurse`
FOR EACH ROW
BEGIN
if exists (select * from Worker where
new.Person_ID = Worker.Person_ID)
then
signal sqlstate value '40039'
set message_text = 'This person is a worker';
end if;
if exists (select * from Doctor where
new.Person_ID = Doctor.Person_ID)
then
signal sqlstate value '40040'
set message_text = 'This person is a doctor';
end if;
END$$


USE `mydb`$$
```

```
DROP TRIGGER IF EXISTS `mydb`.`Nurse_BEFORE_UPDATE` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`Nurse_BEFORE_UPDATE`
BEFORE UPDATE ON `mydb`.`Nurse`
FOR EACH ROW
BEGIN
if exists (select * from Worker where
new.Person_ID = Worker.Person_ID)
then
signal sqlstate value '40042'
set message_text = 'This person is a worker';
end if;
if exists (select * from Doctor where
new.Person_ID = Doctor.Person_ID)
then
signal sqlstate value '40043'
set message_text = 'This person is a doctor';
end if;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`OperationDrug_AFTER_INSERT` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`OperationDrug_AFTER_INSERT`
AFTER INSERT ON `mydb`.`OperationDrug`
FOR EACH ROW
BEGIN
update Drugs
set Drugs.Inventory = (Drugs.Inventory - New.NumOfUse)
where Drugs.Drug_ID = New.Drug_ID;
END$$
```

```
USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`OperationDrug_AFTER_UPDATE` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`OperationDrug_AFTER_UPDATE`
AFTER UPDATE ON `mydb`.`OperationDrug`
FOR EACH ROW
BEGIN
update Drugs
set Drugs.Inventory = (Drugs.Inventory - New.NumOfUse)
where Drugs.Drug_ID = New.Drug_ID;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`OperationDrug_BEFORE_INSERT` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`OperationDrug_BEFORE_INSERT`
BEFORE INSERT ON `mydb`.`OperationDrug`
FOR EACH ROW
BEGIN
declare inventory_remain long;
set inventory_remain = (select Drugs.Inventory from Drugs where
New.Drug_ID = Drugs.Drug_ID);
if(inventory_remain - new.NumOfUse < 0)
then
signal sqlstate value '40007'
set message_text = 'Not valid drug inventory!';
end if;
END$$


USE `mydb`$$
```

```
DROP TRIGGER IF EXISTS `mydb`.`OperationDrug_BEFORE_UPDATE` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`OperationDrug_BEFORE_UPDATE`
BEFORE UPDATE ON `mydb`.`OperationDrug`
FOR EACH ROW
BEGIN
declare inventory_remain long;
set inventory_remain = (select Drugs.Inventory from Drugs where
New.Drug_ID = Drugs.Drug_ID);
if(inventory_remain - new.NumOfUse < 0)
then
signal sqlstate value '40008'
set message_text = 'Not valid drug inventory!';
end if;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`Storage_AFTER_INSERT` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`Storage_AFTER_INSERT`
AFTER INSERT ON `mydb`.`Storage`
FOR EACH ROW
BEGIN
update Drugs
set Drugs.Inventory = (Drugs.Inventory - New.Inventory)
where Drugs.Drug_ID = New.Drug_ID;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`Storage_AFTER_UPDATE` $$
USE `mydb`$$
```

```
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`Storage_AFTER_UPDATE`
AFTER UPDATE ON `mydb`.`Storage`
FOR EACH ROW
BEGIN
update Drugs
set Drugs.Inventory = (Drugs.Inventory - New.Inventory)
where Drugs.Drug_ID = New.Drug_ID;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`Storage_BEFORE_INSERT` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`Storage_BEFORE_INSERT`
BEFORE INSERT ON `mydb`.`Storage`
FOR EACH ROW
BEGIN
declare inventory_remain long;
set inventory_remain = (select Drugs.Inventory from Drugs where
New.Drug_ID = Drugs.Drug_ID);
if(inventory_remain - new.Inventory < 0)
then
signal sqlstate value '40017'
set message_text = 'Not valid drug inventory!';
end if;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`Storage_BEFORE_UPDATE` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
```

```
TRIGGER `mydb`.`Storage_BEFORE_UPDATE`
BEFORE UPDATE ON `mydb`.`Storage`
FOR EACH ROW
BEGIN
declare inventory_remain long;
set inventory_remain = (select Drugs.Inventory from Drugs where
New.Drug_ID = Drugs.Drug_ID);
if(inventory_remain - new.Inventory < 0)
then
signal sqlstate value '40018'
set message_text = 'Not valid drug inventory!';
end if;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`Worker_BEFORE_INSERT` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`Worker_BEFORE_INSERT`
BEFORE INSERT ON `mydb`.`Worker`
FOR EACH ROW
BEGIN
if exists (select * from Doctor where
new.Person_ID = Doctor.Person_ID)
then
signal sqlstate value '40026'
set message_text = 'This person is a doctor';
end if;
if exists (select * from Nurse where
new.Person_ID = Nurse.Person_ID)
then
signal sqlstate value '40027'
set message_text = 'This person is a nurse';
end if;
END$$
```

```
USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`Worker_BEFORE_UPDATE` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`Worker_BEFORE_UPDATE`
BEFORE UPDATE ON `mydb`.`Worker`
FOR EACH ROW
BEGIN
if exists (select * from Doctor where
new.Person_ID = Doctor.Person_ID)
then
signal sqlstate value '40029'
set message_text = 'This person is a doctor';
end if;
if exists (select * from Nurse where
new.Person_ID = Nurse.Person_ID)
then
signal sqlstate value '40030'
set message_text = 'This person is a nurse';
end if;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`WorkInHospital_BEFORE_INSERT` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`WorkInHospital_BEFORE_INSERT`
BEFORE INSERT ON `mydb`.`WorkInHospital`
FOR EACH ROW
BEGIN
if exists (select * from WorkInPharmacy where
new.Worker_ID = WorkInPharmacy.Worker_ID)
```

```
then
signal sqlstate value '40003'
set message_text = 'Worker already work in pharmacy';
end if;
if exists (select * from WorkInHospital where
new.Worker_ID = WorkInHospital.Worker_ID)
then
signal sqlstate value '40013'
set message_text = 'Worker already has a position in another department';
end if;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`WorkInHospital_BEFORE_UPDATE` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`WorkInHospital_BEFORE_UPDATE`
BEFORE UPDATE ON `mydb`.`WorkInHospital`
FOR EACH ROW
BEGIN
if exists (select * from WorkInPharmacy where
new.Worker_ID = WorkInPharmacy.Worker_ID)
then
signal sqlstate value '40004'
set message_text = 'Worker already work in pharmacy';
end if;
if exists (select * from WorkInHospital where
new.Worker_ID = WorkInHospital.Worker_ID)
then
signal sqlstate value '40014'
set message_text = 'Worker already has a position in another department';
end if;
END$$
```

```
USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`WorkInPharmacy_BEFORE_INSERT` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`WorkInPharmacy_BEFORE_INSERT`
BEFORE INSERT ON `mydb`.`WorkInPharmacy`
FOR EACH ROW
BEGIN
if exists (select * from WorkInHospital where
new.Worker_ID = WorkInHospital.Worker_ID)
then
signal sqlstate value '40005'
set message_text = 'Worker already work in department';
end if;
if exists (select * from WorkInPharmacy where
new.Worker_ID = WorkInPharmacy.Worker_ID)
then
signal sqlstate value '40015'
set message_text = 'Worker already has a position in another pharmacy';
end if;
END$$


USE `mydb`$$
DROP TRIGGER IF EXISTS `mydb`.`WorkInPharmacy_BEFORE_UPDATE` $$
USE `mydb`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `mydb`.`WorkInPharmacy_BEFORE_UPDATE`
BEFORE UPDATE ON `mydb`.`WorkInPharmacy`
FOR EACH ROW
BEGIN
if exists (select * from WorkInHospital where
new.Worker_ID = WorkInHospital.Worker_ID)
then
signal sqlstate value '40006'
```

```
set message_text = 'Worker already work in department';
end if;
if exists (select * from WorkInPharmacy where
new.Worker_ID = WorkInPharmacy.Worker_ID)
then
signal sqlstate value '40016'
set message_text = 'Worker already has a position in another pharmacy';
end if;
END$$


DELIMITER ;

DROP PROCEDURE IF EXISTS `mydb`.`GetDoctorDetailswithName`
DELIMITER ;;
CREATE DEFINER=`root`@`localhost` PROCEDURE
`GetDoctorDetailswithName`(
    in firstname varchar(45)
)
begin
    select b.Pname, b.Position, b.Age, b.Gender, d.Department_Name,
d.Location from
   Doctor as a join Person as b on
   a.Person_ID = b.Person_ID
   join Doctor_In as c on
   a.Doctor_ID = c.Doctor_ID
   join Hospital as d on
   c.Department_ID = d.Department_ID
   where b.Pname = firstname ;
end ;;
DELIMITER ;

DROP PROCEDURE IF EXISTS `mydb`.`GetNurseDetailswithName`
DELIMITER ;;
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetNurseDetailswithName`(
    in firstname varchar(45)
)
```

```
begin
    select b.Pname, b.Position, b.Age, b.Gender, d.Department_Name,
d.Location from
   Nurse as a join Person as b on
   a.Person_ID = b.Person_ID
   join Nurse_In as c on
   a.Nurse_ID = c.Nurse_ID
   join Hospital as d on
   c.Department_ID = d.Department_ID
   where b.Pname = firstname;
end ;;
DELIMITER ;

DROP PROCEDURE IF EXISTS `mydb`.`GetPharamacyInventorywithLocation`
DELIMITER ;;
CREATE DEFINER=`root`@`localhost` PROCEDURE
`GetPharamacyInventorywithLocation`(
    in location varchar(255)
)
begin
    select a.Location, c.Name, c.Size, c.Price, c.Producer, b.Inventory from
   Pharmacy as a join Storage as b on
   a.Pharmacy_ID = b.Pharmacy_ID
   join Drugs as c on
   b.Drug_ID = c.Drug_ID
   where a.Location = location;
end ;;
DELIMITER ;
DROP PROCEDURE IF EXISTS `mydb`.`GetOperationDrugswithAnimalID`
DELIMITER ;;
CREATE DEFINER=`root`@`localhost` PROCEDURE
`GetOperationDrugswithAnimalID`(
    in animal_id int
)
begin
    select Animal_ID, a.Operation_ID, d.Name, c.NumOfUse from
   OperatationOn as a join Operation as b join OperationDrug as c on
   a.Operation_ID = b.Operation_ID = c.Operation_ID
   join Drugs as d on
   c.Drug_ID = d.Drug_ID
   where a.Animal_ID = animal_id;
end ;;
DELIMITER ;

DROP PROCEDURE IF EXISTS `mydb`.`GetAnimalHistorywithID`
DELIMITER ;;
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetAnimalHistorywithID`(
    in check_patient_id int
)
begin
   select a.Animal_ID, a.AnimalName, a.AnimalType, a.AnimalAge,
a.AnimalGender, b.Diagnosis from (
   Animal as a
   join Diagnose as b on
   a.Animal_ID = b.Animal_ID
   )
   where
   a.Animal_ID = check_patient_id;

end ;;
DELIMITER ;
```