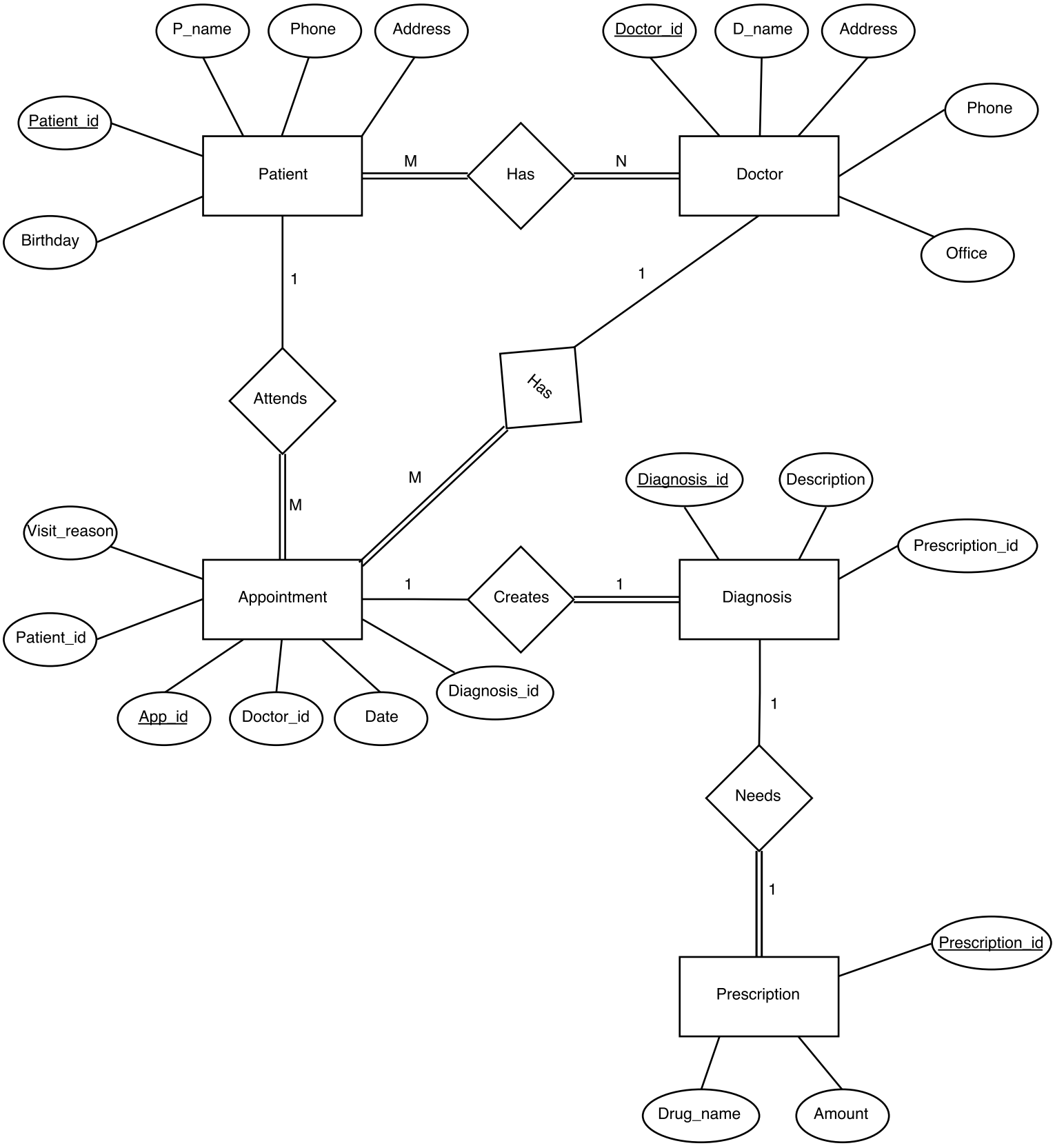
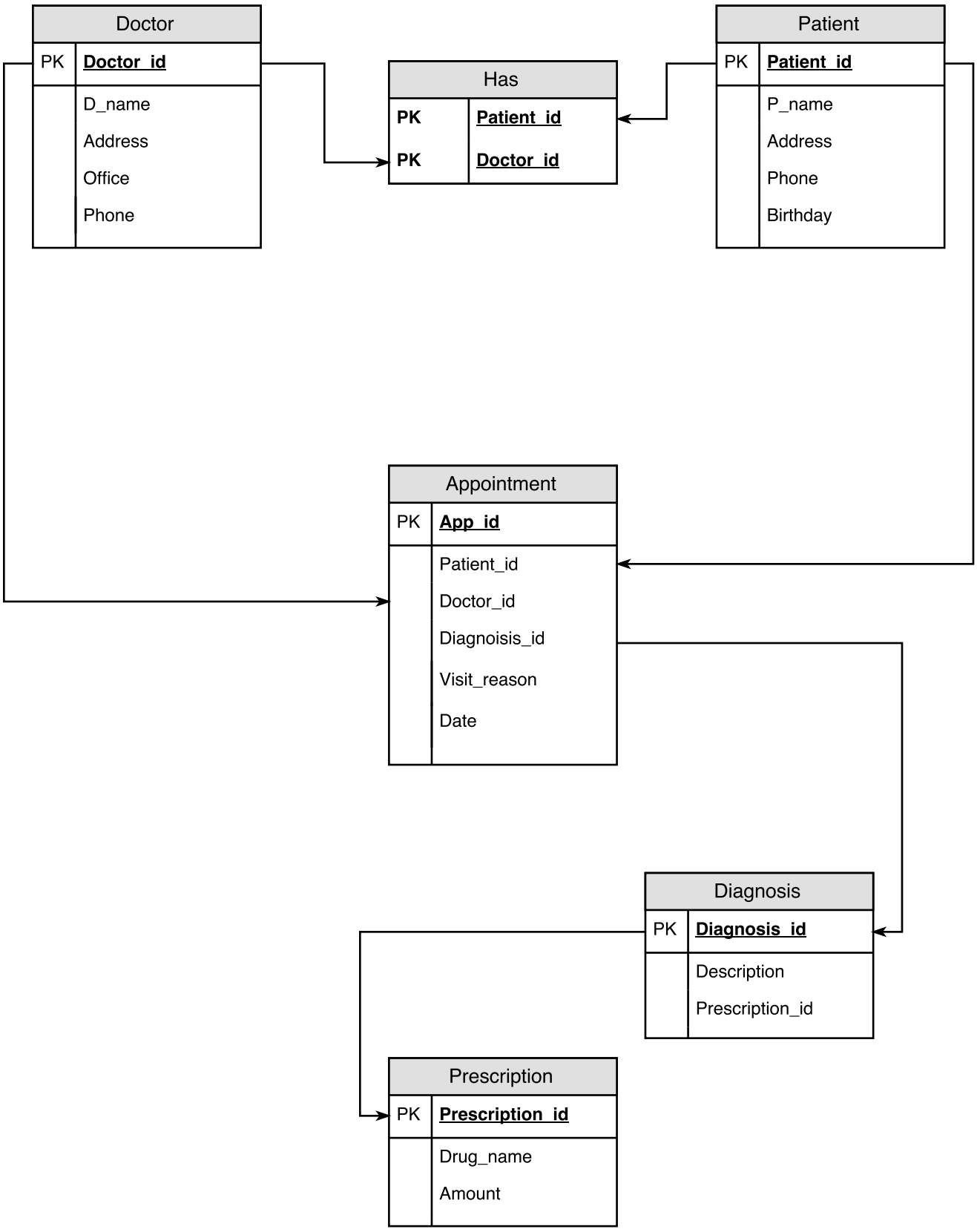
Thad Sauter
3/18/18
CS 340

Medical Database Outline

I chose to design a small medical database, which includes the entities, doctors, patients, appointments, diagnoses, and prescriptions. Starting with the doctor entity, doctors have five attributes, Doctor_id, D_name, Phone, Address, and Office. Doctor_id is the primary key (auto_incrementing) in this entity and none of the attributes may be null. Moving on, the patient entity has the attributes, Patient_id, P_name, Birthday, Phone, Address. Patient_id is the primary key (auto-incrementing) and none of the attributes may be null. From these entities comes the "has" relationship. I defined this relationship as many to many, meaning that a patient can have many doctors, and a doctor can have many patients. Patients may exist without current doctors, and doctors may exist that are not currently seeing any patients. The next entity, appointment, has the attributes App_id, Patient_id, Doctor_id, Date, and Diagnosis_id. App_id is the auto-incrementing primary key and Patient_id and Doctor_id are the foreign keys that reference the patient entity and doctor entity, respectively. Also, none of the values may be null. The next relationship, attends, exists between the doctor, patient, and appointment. We say that if an appointment exists, it must be attended to by exactly one patient and one doctor. However, patients and doctors may themselves have many different appointments, or no appointments at all. The next entity, diagnosis, has the attributes Diagnosis_id, Description, App_id, and Prescription_id. Diagnosis_id is the auto-incrementing primary key and Prescription_id and App_id are the foreign keys which reference the Prescription_id from the prescription entity and the App_id from the Appointment entity. The Prescription_id has the ability to be null because it is not necessary that every diagnosis have a prescription. We can say that an appointment creates at most one diagnosis and a diagnosis must be created from one appointment. It is possible to have an appointment that does not create a diagnosis. The last entity, prescription, has the attributes Prescription_id, Drug_name, and Amount. Prescription_id is the auto-incrementing primary key and none of the attributes may be null. The last relationship, needs, exists between the entities, diagnosis and prescription. A diagnosis may have at most one prescription (meaning it can have one prescription, but does not have to have one). A prescription must belong to one diagnosis (drugs are not prescribed unless a valid diagnosis is present).

A couple more notes about my project. In the "Has" many to many relationship, both foreign keys are set to cascade on delete. This is so if either a patient or doctor is deleted that exists in a relationship in the "Has" table, all of those subsequent relationships will also be deleted. This is the same for the "Appointment" entity. If a patient or doctor is deleted that has an appointment, that appointment will also be deleted. However, there is also a Diagnosis_id (foreign key) in the "Appointment" table which is set to null if a Diagnosis is deleted. Similarly, in the "Prescription" table, if the Diagnosis is deleted, then the Prescription is also deleted. However, a Prescription can be deleted from a Diagnosis without anything happening.


P_name
Phone
Address
Patient_id
Patient
Birthday
M
Has
N
Doctor_id
D_name
Address
Phone
Doctor
Office
1
1
Attends
Has
M
M
Diagnosis_id
Description
Visit_reason
Appointment
1
Creates
1
Diagnosis
Prescription_id
Patient_id
App_id
Doctor_id
Date
Diagnosis_id
1
Needs
1
Prescription_id
Prescription
Drug_name
Amount


Doctor
PK
Doctor_id
D_name
Address
Office
Phone
Has
PK
Patient_id
PK
Doctor_id
Patient
PK
Patient_id
P_name
Address
Phone
Birthday
Appointment
PK
App_id
Patient_id
Doctor_id
Diagnoisis_id
Visit_reason
Date
Diagnosis
PK
Diagnosis_id
Description
Prescription_id
Prescription
PK
Prescription_id
Drug_name
Amount

Create Table Queries

```
CREATE TABLE Doctor(
        Doctor_id INT NOT NULL AUTO_INCREMENT UNIQUE,
                D_name VARCHAR(45) NOT NULL,
                Phone VARCHAR(45) NOT NULL,
                Address VARCHAR(45) NOT NULL,
                Office VARCHAR(45) NOT NULL,
                PRIMARY KEY (Doctor_id))
               ENGINE = InnoDB;

CREATE TABLE Patient(
  Patient_id INT NOT NULL AUTO_INCREMENT UNIQUE,
          P_name VARCHAR(45) NOT NULL,
          Birthday DATETIME NOT NULL,
          Phone VARCHAR(45) NOT NULL,
          Address VARCHAR(45) NOT NULL,
          PRIMARY KEY (Patient_id))
         ENGINE = InnoDB;

CREATE TABLE Has(
  Patient_id INT NOT NULL,
  Doctor_id INT NOT NULL,
  PRIMARY KEY (Patient_id, Doctor_id),
  CONSTRAINT Patient_id
   FOREIGN KEY (Patient_id)
   REFERENCES Patient (Patient_id)
   ON DELETE CASCADE
   ON UPDATE NO ACTION,
  INDEX Doctor_id_idx (Doctor_id ASC),
  CONSTRAINT Doctor_id
    FOREIGN KEY (Doctor_id)
    REFERENCES Doctor (Doctor_id)
    ON DELETE CASCADE
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
CREATE TABLE Prescription(
  Prescription_id INT NOT NULL AUTO_INCREMENT,
  Drug_name VARCHAR(45) NOT NULL,
  Amount INT NOT NULL,
  PRIMARY KEY (Prescription_id))
ENGINE = InnoDB;

CREATE TABLE Diagnosis(
  Diagnosis_id INT NOT NULL AUTO_INCREMENT,
  Description TEXT NOT NULL,
  Prescription_id INT NULL,
  PRIMARY KEY (Diagnosis_id),
  INDEX Prescription_id_idx (Prescription_id ASC),
  CONSTRAINT Prescription_id
    FOREIGN KEY (Prescription_id)
    REFERENCES Prescription (Prescription_id)
    ON DELETE SET NULL
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE Appointment(
  App_id INT NOT NULL AUTO_INCREMENT,
  Visit_reason TEXT NOT NULL,
  Patient_id INT NOT NULL,
  Doctor_id INT NOT NULL,
  Date DATETIME NOT NULL,
  Diagnosis_id INT,
  PRIMARY KEY (App_id),
  INDEX Doctor_id_idx (Doctor_id ASC),
  INDEX Patient_id_idx (Patient_id ASC),
  INDEX Diagnosis_id_idx (Diagnosis_id ASC),
  CONSTRAINT Patient_id_app
    FOREIGN KEY (Patient_id)
    REFERENCES Patient (Patient_id)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT Doctor_id_app
    FOREIGN KEY (Doctor_id)
    REFERENCES Doctor (Doctor_id)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT Diagnosis_id_app
    FOREIGN KEY (Diagnosis_id)
    REFERENCES Diagnosis (Diagnosis_id)
```

```
    ON DELETE SET NULL
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Data Manipulation Queries

For this part of the project, I wanted to make the queries as realistic as possible. I chose to create queries to find appointments based on the date of the appointment and on the name of the doctor or patient in the appointment. I thought this would most closely mimic how a database such as this one would be used in a medical office.

Find Appointments Based on Date:

```
SELECT * FROM Appointment
    INNER JOIN Patient ON Appointment.Patient_id = Patient.Patient_id
        INNER JOIN Doctor ON Appointment.Doctor_id = Doctor.Doctor_id
            WHERE Date = [dateInput];
```

Find Appointments Based on Name:

```
SELECT * FROM Appointment
    INNER JOIN Patient ON Appointment.Patient_id = Patient.Patient_id
        INNER JOIN Doctor ON Appointment.Doctor_id = Doctor.Doctor_id
            WHERE P_name LIKE [nameInput] OR D_name LIKE [nameInput]
```

Find Appointments Based on Date and Name:

```
SELECT * FROM Appointment
    INNER JOIN Patient ON Appointment.Patient_id = Patient.Patient_id
        INNER JOIN Doctor ON Appointment.Doctor_id = Doctor.Doctor_id
            WHERE (P_name LIKE [nameInput] OR D_name LIKE [nameInput])
                AND Date = [dateInput];
```

Find Appointments to Add a Diagnosis Based on Name:

```
SELECT * FROM Appointment
    INNER JOIN Patient ON Appointment.Patient_id = Patient.Patient_id
        INNER JOIN Doctor ON Appointment.Doctor_id = Doctor.Doctor_id
        LEFT JOIN Diagnosis ON Appointment.Diagnosis_id = Diagnosis.Diagnosis_id
            WHERE P_name LIKE [nameInput] OR D_name LIKE [nameInput];
```

Find Appointments to Add a Prescription Based on Name:

```
SELECT * FROM Appointment
INNER JOIN Patient ON Appointment.Patient_id = Patient.Patient_id
INNER JOIN Doctor ON Appointment.Doctor_id = Doctor.Doctor_id
INNER JOIN Diagnosis ON Appointment.Diagnosis_id = Diagnosis.Diagnosis_id
 LEFT JOIN Prescription ON Diagnosis.Prescription_id = Prescription.Prescription_id
WHERE P_name LIKE [nameInput] OR D_name LIKE [nameInput];
```