|  |  |
| --- | --- |
| CS 340 Final Project | Kelvin Watson  OSU ID 932540242  onid: watsokel |

**Outline**

My database stores data about patients and their medical conditions and medications, medical office assistants, and healthcare providers. It is used to manage and organize patients’ medical appointments, as well as details about their medical history. A database and system such as this could be used in a doctor’s office or other medical practice requiring storage of medical information about patients and staff working at the medical office.

**Database Outline in Words**

The user will be an administrator that works at a medical office. As such, he/she has the ability to add records to every table. The administrator is able to add medical offices assistants (staff) to the Medical Office Assistants relation, patients’ appointments to the appointments relation, patients and their demographic information to the patients relation, patients’ medications to the medications relation, as well as patients’ medical conditions to the medical conditions relation.

Two additional relations, called “Takes” and “Diagnosed” result from the many-to-many relationship between Patient and Medication, and Patient and Medical Condition entities respectively. When the administrator (user) adds patient information into the Patient relation, he/she will also add information into the Medications and Medical Conditions relations as well.

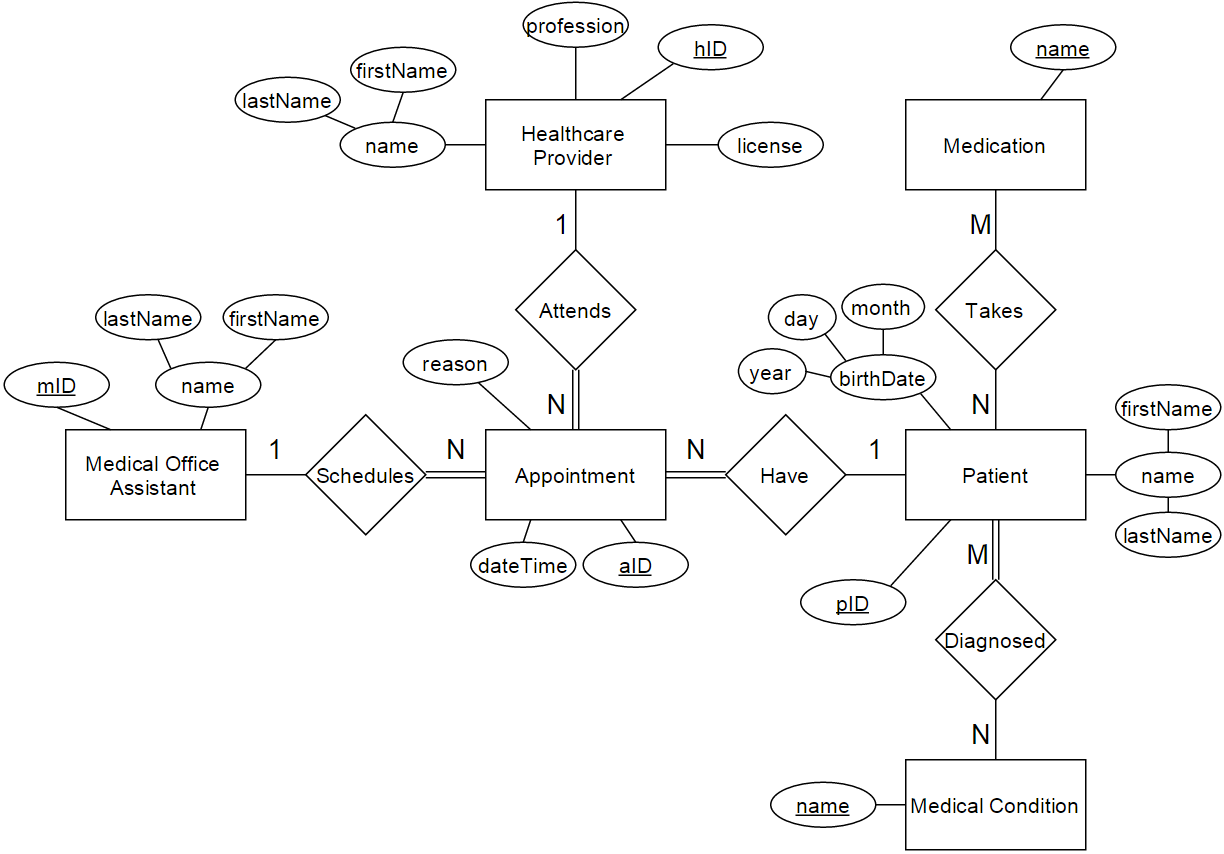
The following describes the database relations and constraints applied to those relations.

**Database Relations, Relationships, and Constraints**

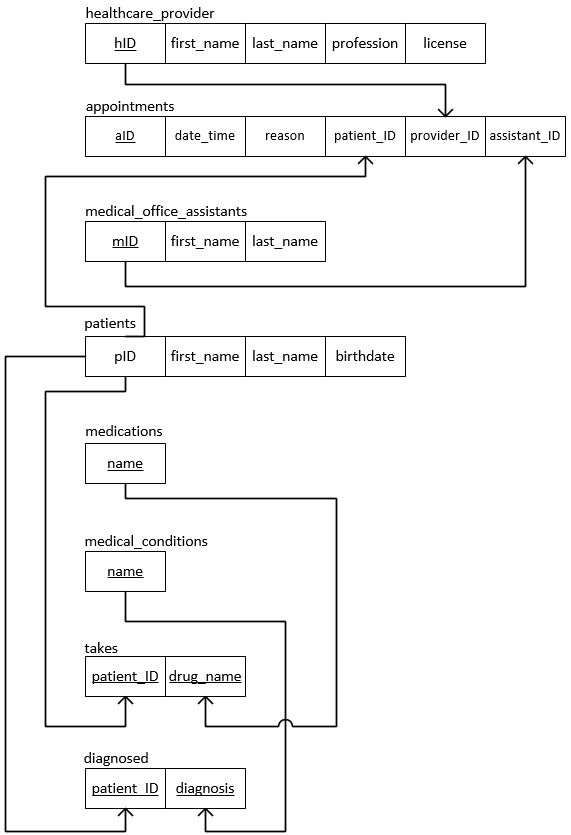
As expected, all entities will have their own tables in the database. However, there are two additional relations that result from many-to-many relationships between entities. The Takes relation is the result from the many-to-many relationship between Patient and Medication, and the Diagnosed relation is the result from the many-to-many relationship between Patient and Medical Conditions. Constraints of each relation are described below:

1. Medical Office Assistants:
   1. Attributes: Medical office assistants are the users that work in the medical clinic. They manage patients’ medical appointments. They will have internal ID’s (mID) and names.
   2. Cardinality Constraint: Each medical office assistant can approve many appointments, so this is a one-to-many relationship.
   3. Not null: The mID column cannot be null.
   4. Primary key constraint: The mID attribute is the primary key as it uniquely identifies each medical office assistant record.
   5. Participation Constraint: A particular medical office assistant is not required to schedule appointments for any patients. As such, the medical office assistant’s participation in this relationship is partial.
   6. Referential Constraint: Because each Medical Office Assistant can schedule multiple appointments, its relationship to Appointment entity is a one-to-many relationship. As a result, the Medical Office Assistants’ mID attribute is a foreign key in the Appointments relation in the database. Each appointment must be associated with a medical office assistant, as the medical office assistant is responsible for scheduling the appointment.
   7. Foreign Key Constraints:
      1. ON UPDATE CASCADE: Once an appointment is set, updating his/her information should not impact the scheduled appointment. If for some reason the mID of a record in the Medical Office Assistant table is altered, it should be reflected in the Appointments table.
      2. ON DELETE NO ACTION: A medical office assistant cannot be deleted as they are tied to an appointment and cannot be set to null in the appointments table. This is because the primary keys mID cannot be null.
2. Healthcare Providers
   1. Attributes: A healthcare provider treats patients. They will have an internal ID (hID), names, professions (physician, dentist, dietician, pharmacist, physiotherapist etc.) and license numbers.
   2. Cardinality Constraint: Each healthcare provider can attend multiple appointments, so this is a one-to-many relationship.
   3. Participation Constraint: Each healthcare provider can attend multiple appointments, but is not required to attend any appointment. As such, the healthcare provider’s participation in the “attends” relationship is partial.
   4. Referential constraint: As mentioned above, although a healthcare provider is not required to attend appointments, it is possible that he/she attends multiple appointments. This is a one-to-many relationship with the Appointments entity. As a result, the healthcare provider’s hID attribute is a foreign key in the Appointments table.
   5. Foreign Key Constraints:
      1. ON UPDATE CASCADE: If for some reason the hID of a record in the Healthcare Provider table is altered, it should be reflected in the Appointments table.
      2. ON DELETE CASCADE: If a healthcare provider no longer works at the medical office, then his/her appointments should be automatically canceled.
3. Patients
   1. Attributes: Patients have an internal ID for the medical office (pID), names and birthdates.
   2. Cardinality constraint: Each patient may have multiple scheduled appointments, so this is a one-to-many relationship. Each patient may have multiple medical conditions, and each medical condition can be associated with multiple patients, so this is a many-to-many relationship. Each patient may also take multiple medications, and each medication can be associated with multiple patients, so this is a many-to-many relationship as well.
   3. Participation Constraints:
      1. Patients are required to have at least one medical condition to be seen in this medical office. As such, the participation of the Patient entity in the Diagnosed relationship with the Medical Condition entity is total.
      2. Patients may take multiple medications, but they are not required to be on any medications. As such, the participation of the Patient entity in the Takes relationship is partial.
   4. Referential Constraint: The Patient entity has many-to-many relationships with the Medication and the Medical Condition entities. Because of this, Patients’ pID’s serve as foreign keys in the Takes and Diagnosed relations in the database.
   5. Foreign Key Constraints:
      1. ON UPDATE CASCADE: If a patient’s pID is altered, it should be reflected in the Takes and Diagnosed tables that reference this pID.
      2. ON DELETE CASCADE: If a Patient were to be deleted, the records that reference the pID in the Takes and Diagnose tables should be deleted.
4. Appointments
   1. Patients will have medical appointments. Attributes include date, time, and reason for the appointment.
   2. These are medications that patients take. They will have names and national drug codes.
   3. Cardinality constraint: Each appointment is approved by only one medical office assistant, but each medical office assistant may approve multiple appointments, so appointment is in a many-to-one relationship with medical office assistant. Each appointment can be attended by only one healthcare provider, but each provider can attend multiple appointments. Thus, appointments is in a many-to-one relationship with healthcare providers. Each appointment can be attended by only one patient, but each patient may have multiple scheduled appointments. As such, appointment is in many-to-one relationship with patient.
   4. Participation Constraint: Each appointment must be associated with a medical office assistant, as the medical office assistant is responsible for scheduling the appointment.
   5. Referential Integrity Constraints: Each appointment will have as three foreign keys which reference the Patients, Medical Office Assistants and the Healthcare Providers tables.
   6. Foreign Key Constraints:
      1. Patient pID foreign key:
         1. ON UPDATE CASCADE: If, for some reason, a patient’s ID is updated, it should be updated in the Appointments table as well. However, modifying a patient’s ID once assigned by auto-increment will not be allowed in this implementation of the database as it does not make sense to change this internal ID kept by the medical office.
         2. ON DELETE CASCADE: If a patient record is deleted, his/her appointment records should also be deleted.
      2. Medical Office Assistant mID foreign key:
         1. ON UPDATE CASCADE: If, for some reason, a medical office assistant’s ID is updated, it should be updated in the Appointments table. However, modifying a medical office assistant’s auto-incremented ID will not be allowed in this implementation as it makes little sense to alter this internal ID kept by the medical office.
         2. ON DELETE SET NULL: If a medical office assistant is deleted, then the corresponding row in the Appointments table should show NULL for the medical office assistant’s ID field. Regardless of whether the medical office assistant still works at the office or not, the appointment was scheduled and should still be assigned to a patient.
      3. Healthcare Provider hID foreign key:
         1. ON UPDATE CASCADE: If, for some reason, a healthcare provider’s ID is altered, it should be updated in the Appointments table. However, modifying a healthcare provider’s ID once assigned via auto-increment will not be permitted in this implementation of the database as it does not make sense to change this internal ID kept by the medical office.
         2. ON DELETE CASCADE: If a healthcare provider is removed from the database, his/her appointments should be cancelled and their dates and times freed for other patients.
5. Medical Conditions
   1. Attributes: These are diseases and conditions that patients have. They will have names.
   2. Cardinality Constraint: Each medical condition can be associated with multiple patients, and each patient can be associated with multiple medical conditions. This is a many-to-many relationship.
   3. Participation Constraint: A medication may or may not be associated with any given patient. However, one medical condition associated could also be associated with many patients. Given that a medical condition may or may not be associated with any given patient on file at this medical office, the participation of the Medical Condition entity in the Diagnosed relationship with the Patient entity is partial.
6. Medications
   1. Attributes: These are medications that patients take. They will have names and national drug codes.
   2. Cardinality Constraint: Each medication can be taken by multiple patients, and each patient can take multiple medications. As such, this is a many-to-many relationship.
   3. Participation Constraint: One medication may be taken by multiple patients. However, any particular medication may not be taken by any patients on file at this medical office. As such, the participation of the Medication entity in the Takes relationship with the Patient entity is partial.
7. Takes
   1. This table is the result of a many-to-many relationship between the Patient and Medication entities.
   2. Referential Constraint: The Takes table contains the pID and drug name as foreign keys from the Patient table and the Medication tables respectively. These foreign keys together serve as the composite primary key for this table which uniquely identifies each patient-medication pair.
   3. Foreign Key Constraints:
      1. Patient pID foreign key:
         1. ON UPDATE CASCADE: If the patient ID is updated, then it should be reflected in the Takes table.
         2. ON DELETE CASCADE: If the patient is deleted, then his/her corresponding records in the Takes table should also be deleted.
      2. Medication name foreign key:
         1. ON UPDATE/DELETE NO ACTION: A drug’s name should never be modified or deleted because this is a product identifier. In order to delete a medical condition associated with a patient, the record in the Diagnosed table should be deleted. In order to delete a medication associated with a particular patient, the record in the Takes table should be deleted.
8. Diagnosed
   1. This table is the result of a many-to-many relationship between the Patient and Medical Condition entities.
   2. Referential Constraint: The Diagnosed table contains the pID and name attributes as foreign keys from the Patient table and the Medical Conditions tables respectively. These foreign keys together serve as the composite primary key for this table.
   3. Foreign Key Constraints:
      1. Patient pID foreign key:
         1. ON UPDATE CASCADE: If a patient ID is updated, it should be reflected in the Diagnosed table.
         2. ON DELETE CASCADE: If a patient is deleted, his/her corresponding records in the Diagnosed table should also be deleted.
      2. Medical Condition name foreign key:
         1. ON UPDATE/DELETE NO ACTION: A medical condition name should not be modified or deleted. In order to delete a medical condition associated with a patient, the record in the Diagnosed table should be deleted.

**ER Diagram**



**Database Schema**



**Table Creation Queries**

CREATE TABLE medical\_office\_assistants(

mID INT( 11 ) NOT NULL AUTO\_INCREMENT PRIMARY KEY ,

first\_name VARCHAR( 255 ) DEFAULT 'Temporary',

last\_name VARCHAR( 255 ) DEFAULT 'Assistant'

) ENGINE = INNODB

CREATE TABLE healthcare\_providers(

hID INT( 11 ) NOT NULL AUTO\_INCREMENT PRIMARY KEY ,

first\_name VARCHAR( 255 ) NOT NULL ,

last\_name VARCHAR( 255 ) NOT NULL ,

profession VARCHAR( 255 ) NOT NULL ,

license INT( 11 ) UNIQUE

) ENGINE = INNODB;

CREATE TABLE patients(

pID INT( 11 ) NOT NULL AUTO\_INCREMENT PRIMARY KEY ,

first\_name VARCHAR( 255 ) NOT NULL ,

last\_name VARCHAR( 255 ) NOT NULL ,

birthdate DATE

) ENGINE = INNODB;

CREATE TABLE appointments(

aID INT( 11 ) NOT NULL AUTO\_INCREMENT PRIMARY KEY ,

date\_time DATETIME NOT NULL ,

reason TEXT,

patient\_ID INT( 11 ) NOT NULL ,

provider\_ID INT( 11 ) NOT NULL ,

assistant\_ID INT( 11 ) NOT NULL ,

FOREIGN KEY ( patient\_ID ) REFERENCES patients( pID )

ON UPDATE CASCADE

ON DELETE CASCADE,

FOREIGN KEY ( provider\_id ) REFERENCES healthcare\_providers( hID )

ON UPDATE CASCADE

ON DELETE CASCADE,

FOREIGN KEY ( assistant\_id ) REFERENCES medical\_office\_assistants( mID )

ON UPDATE CASCADE  
 ON DELETE NO ACTION

) ENGINE = INNODB;

CREATE TABLE medications(

name varchar(255) PRIMARY KEY

)ENGINE=InnoDB;

CREATE TABLE medical\_conditions(

name varchar(255) PRIMARY KEY

)ENGINE=InnoDB;

CREATE TABLE takes(

patient\_ID INT( 11 ) NOT NULL ,

drug\_name VARCHAR( 255 ) NOT NULL ,

FOREIGN KEY ( patient\_ID ) REFERENCES patients( pID )

ON UPDATE CASCADE

ON DELETE CASCADE,

FOREIGN KEY ( drug\_name ) REFERENCES medications( name ) ,

PRIMARY KEY ( patient\_ID, drug\_name )

) ENGINE = INNODB

CREATE TABLE diagnosed(

patient\_ID INT( 11 ) NOT NULL ,

diagnosis VARCHAR( 255 ) NOT NULL ,

FOREIGN KEY ( patient\_ID ) REFERENCES patients( pID )

ON UPDATE CASCADE

ON DELETE CASCADE,

FOREIGN KEY ( diagnosis ) REFERENCES medical\_conditions( name )

ON UPDATE NO ACTION

ON DELETE NO ACTION,

PRIMARY KEY ( patient\_ID, diagnosis )

) ENGINE = INNODB;

-- Disable foreign key checking

SET foreign\_key\_checks = 0;# MySQL returned an empty result set (i.e. zero rows).

-- Drop all of the tables

DROP TABLE IF EXISTS medical\_office\_assistants, patients, appointments, medications, medical\_conditions,takes,diagnosed;# MySQL returned an empty result set (i.e. zero rows).

-- Re-enable foreign key checking

SET foreign\_key\_checks = 1;# MySQL returned an empty result set (i.e. zero rows).

Code Sources:

<http://stackoverflow.com/questions/13865218/jquery-ui-datepicker-set-year-range-dropdown-to-100-years>

<http://code.runnable.com/UdTuotHbZoQNAABq/adding-a-date-picker-to-an-input-form-field-using-jquery>

<https://jqueryui.com/datepicker/>

SELECT \* FROM appointments

INNER JOIN patients ON pID=patient\_ID

INNER JOIN healthcare\_providers ON hID=provider\_ID

LEFT JOIN medical\_office\_assistants ON mID=assistant\_ID

**SELECT** pt.first\_name, pt.last\_name, ap.date\_time, hp.last\_name **AS** MD, ap.reason, oa.last\_name **AS** MOA  
**FROM** appointments ap  
**INNER** **JOIN** patients pt **ON** pID = patient\_ID  
**INNER** **JOIN** healthcare\_providers hp **ON** hID = provider\_ID  
**LEFT** **JOIN** medical\_office\_assistants oa **ON** mID = assistant\_ID