**Online Pharmacy**

**System Requirements**

1. customers can fulfill prescriptions
2. employees can register new customers
3. employees can register new doctors
4. employees can register new medications
5. employees can restock medications
6. customers can view current prescriptions

**Database requirements and queries to be executed by different users**

The online pharmacy allows for doctors to send in prescriptions and for customers to fulfill those prescriptions.

**The Employee**:

Will need to insert new customers, doctors, medications and restock orders. They will also need to select from those tables. They will also need to be able to delete customers, medications from our offerings, drop doctors from our program and cancel orders.

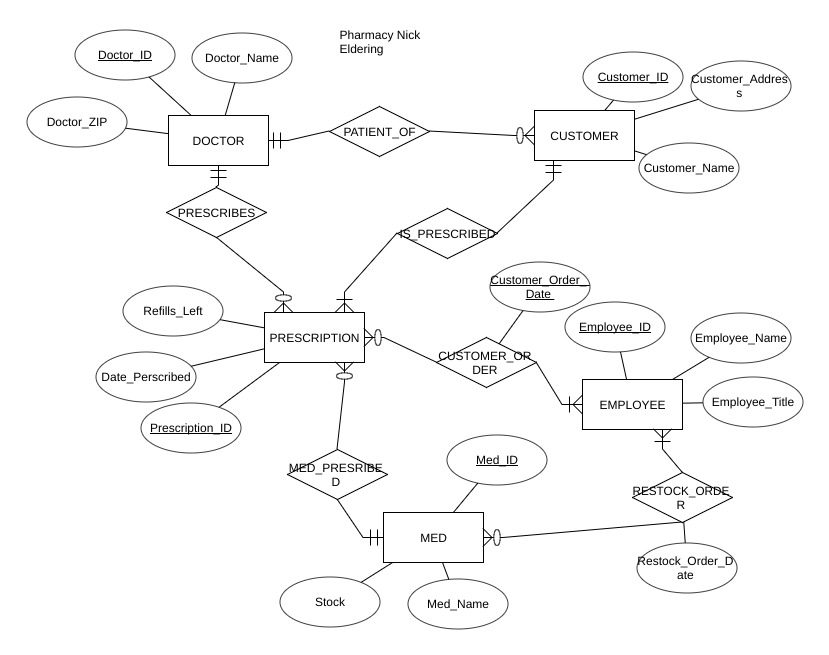
**The Customer:**

Will need to be able to select all their prescriptions, current orders, the stock left for each medication. They will also insert into the customer orders table when fulfilling a prescription. They will also need to be able to delete / cancel an order and change doctors.

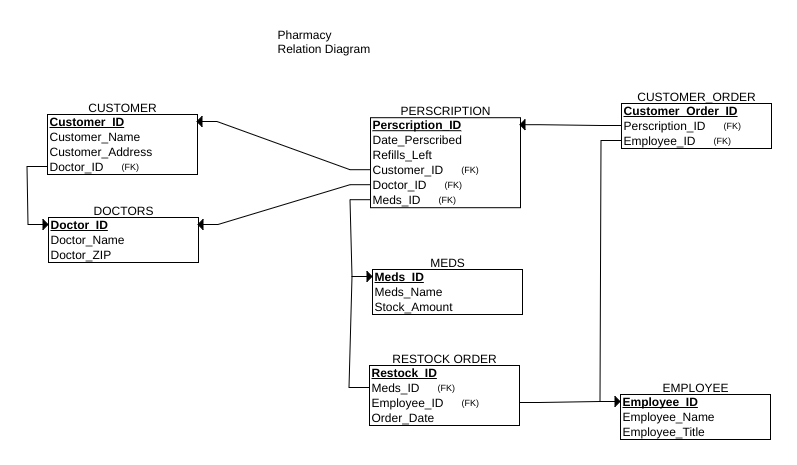
**The Doctor:**

will be able to see (select) all their patients, all the prescriptions they have written, the types of medications the pharmacy can provide (if they don’t have it in their offerings we can’t write a prescription for it and send it to the pharmacy).

ERD



RELATIONAL DIAGRAM:



NORMALIZATION

3-n Normalization

It is in first normalized form because it contains only atomic values and does not have repeating groups.

All non key attributes are fully dependant on the key, and since they are only id’s of other tables or unique to the table

It has no composite keys and their are no transitive dependencies the only tuples that reference other tables are the foreign keys and they connect the two tables so are necessary. In order to get any information from the foreign.

INDEXES

I chose to index prescriptions on doctor\_ID so doctors could see all prescriptions they have written.

I indexed prescriptions on Customer\_ID so that customers can search for all their prescriptions quickly.

I indexed customers on Doctor\_ID to help when doctors need to see all their patients

CREATE INDEX docWrote  
ON prescription (Doctor\_ID);  
  
Create Index custPrescriptions  
on prescription (Customer\_ID);  
  
create index patientOf  
on customer (Doctor\_ID);

APPLICATION DESIGN

The application is primarily for pharmacy employees as they have the most options with this data set. For contrast - doctors only need to be able to write customer prescriptions and see their patients, and Customers only need to see their prescriptions our stock and then create orders. Employees need to be able to do everything from registering new employees/ customers/ doctors/ meds, to restocking and shipping out customer orders. I created popup menus because primary keys are the primary way of entering data into the system. This way an employee can look up an ID without changing pages.

CONSTRAINTS  
There is a constraint on prescriptions so when a customer is dropped from the system all their prescriptions are dropped from the system (we still send out their last order if they have one).  
This required a transaction to ensure integrity.  
  
  
 $this->db->beginTransaction();  
 $statement = $this->db->prepare('DELETE FROM `customer` WHERE Customer\_ID = :custID');  
 $statement->bindParam(':custID', $custID);  
 $statement->execute();  
 $state = $this->db->prepare('DELETE FROM `prescription` WHERE Customer\_ID = :custID');  
 $state->bindParam(':custID', $custID);  
 $state->execute();  
 $this->db->commit();  
  
  
DATABASE INTEGRITY  
Integrity is maintained via foreign key constraints:  
You can’t write a prescription without a valid customer\_ID, med\_ID, doctor\_ID  
You can’t register a customer without a valid doctor.  
When you delete a customer we also delete all their prescriptions in our system (via a transaction).

REQUIRED TRANSACTIONS

When a customer places an order we need to put that process in a transaction so that it doesn’t happen that we create the order before checking if they have any prescription refills left. Also we need to delete a prescription when it has no refills left. So we need a transaction to make sure we subtract from our inventory only if we have enough, and we only want to delete a prescription at 0 if we know the whole order will go through.

Delete customer then delete all that customers prescriptions transaction:

$this->db->beginTransaction();  
 $statement = $this->db->prepare('DELETE FROM `customer` WHERE Customer\_ID = :custID');  
 $statement->bindParam(':custID', $custID);  
 $statement->execute();  
 $state = $this->db->prepare('DELETE FROM `prescription` WHERE Customer\_ID = :custID');  
 $state->bindParam(':custID', $custID);  
 $state->execute();  
 $this->db->commit();

ERROR CHECKING

The error checking is done via primary keys and not null restrictions. You can’t create a prescription without a med\_ID doc\_ID and cust\_ID: if you don’t have any one of those you are not able to create a prescription. In those tables we have type restrictions.

SCREENSHOTS (Clerk interface)

SQL

Create Tables  
  
CREATE TABLE `prescription` (  
 `Prescription\_ID` int(11) NOT NULL AUTO\_INCREMENT,  
 `Doctor\_ID` int(11) NOT NULL,  
 `Customer\_ID` int(11) NOT NULL,  
 `Med\_ID` int(11) NOT NULL,  
 `Date\_Writen` datetime DEFAULT CURRENT\_TIMESTAMP,  
 `Refill\_Amount` int(11) NOT NULL DEFAULT '1',  
 PRIMARY KEY (`Prescription\_ID`),  
 KEY `Doctor\_ID` (`Doctor\_ID`),  
 KEY `Customer\_ID` (`Customer\_ID`),  
 KEY `Med\_ID` (`Med\_ID`),  
 CONSTRAINT `Prescription\_ibfk\_1` FOREIGN KEY (`Customer\_ID`) REFERENCES `customer` (`Customer\_ID`),  
 CONSTRAINT `Prescription\_ibfk\_2` FOREIGN KEY (`Med\_ID`) REFERENCES `med` (`Med\_ID`),  
 CONSTRAINT `Prescription\_ibfk\_3` FOREIGN KEY (`Doctor\_ID`) REFERENCES `doctor` (`Doctor\_ID`)  
) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=latin1  
  
  
create table doctor (  
 Doctor\_ID int not null AUTO\_INCREMENT,  
 Doctor\_Name varchar(255) not null,  
 Doctor\_ZIP int,  
 PRIMARY KEY (Doctor\_ID))  
   
   
   
 CREATE TABLE `customer` (  
 `Customer\_ID` int(11) NOT NULL AUTO\_INCREMENT,  
 `Customer\_Name` varchar(255) NOT NULL,  
 `Doctor\_ID` int(11) NOT NULL,  
 `Cust\_Address` varchar(255) NOT NULL,  
 PRIMARY KEY (Customer\_ID),  
 CONSTRAINT `fks\_customers`  
 FOREIGN KEY (Doctor\_ID) REFERENCES doctor(Doctor\_ID)  
) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=latin1  
  
CREATE TABLE `med` (  
 `Med\_ID` int(11) NOT NULL AUTO\_INCREMENT,  
 `Med\_Name` varchar(50) NOT NULL,  
 Inventory int(11),  
 PRIMARY KEY (`Med\_ID`)  
) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=latin1  
  
  
  
create table employee (  
 Employee\_ID int not null AUTO\_INCREMENT,  
 Employee\_Name varchar(55) not null,  
 Employee\_Title varchar(55),  
 PRIMARY KEY (Employee\_ID))  
   
   
 CREATE TABLE `restock\_Order` (  
 `Restock\_Order\_ID` int(11) NOT NULL AUTO\_INCREMENT,  
 `Employee\_ID` int(11) NOT NULL,  
 `Med\_ID` int(11) NOT NULL,  
 `Date\_Ordered` datetime DEFAULT CURRENT\_TIMESTAMP,  
 `Order\_Amount` int(11) NOT NULL DEFAULT '1',  
 PRIMARY KEY (`Restock\_Order\_ID`),  
 KEY `Employee\_ID` (`Employee\_ID`),  
 KEY `Med\_ID` (`Med\_ID`),  
 CONSTRAINT `Restock\_ibfk\_1` FOREIGN KEY (`Employee\_ID`) REFERENCES `employee` (`Employee\_ID`),  
 CONSTRAINT `Restock\_ibfk\_2` FOREIGN KEY (`Med\_ID`) REFERENCES `med` (`Med\_ID`)  
) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=latin1  
  
  
  
CREATE TABLE `customer\_Order` (  
 `Customer\_Order\_ID` int(11) NOT NULL AUTO\_INCREMENT,  
 `Employee\_ID` int(11) NOT NULL,  
 `Prescription\_ID` int(11) NOT NULL,  
 `Date\_Ordered` datetime DEFAULT CURRENT\_TIMESTAMP,  
 `Order\_Amount` int(11) NOT NULL DEFAULT '1',  
 PRIMARY KEY (`Customer\_Order\_ID`),  
 KEY `Employee\_ID` (`Employee\_ID`),  
 KEY `Prescription\_ID` (`Prescription\_ID`),  
 CONSTRAINT `CustOrder\_ibfk\_1` FOREIGN KEY (`Employee\_ID`) REFERENCES `employee` (`Employee\_ID`),  
 CONSTRAINT `CustOrder\_ibfk\_2` FOREIGN KEY (`Prescription\_ID`) REFERENCES `prescription` (`Prescription\_ID`)  
) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=latin1  
  
  
LOGIN TABLES  
  
CREATE TABLE `login` (  
 `username` varchar(200) NOT NULL,  
 `password` varchar(200) NOT NULL,  
 PRIMARY KEY (`username`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1  
  
CREATE TABLE `UserName` (  
 `UserNameID` int(9) NOT NULL AUTO\_INCREMENT,  
 `userName` varchar(40) NOT NULL,  
 `pass` varchar(40) NOT NULL,  
 PRIMARY KEY (`UserNameID`)  
) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=latin1  
  
  
CREATE TABLE `users` (  
 `id` int(11) NOT NULL AUTO\_INCREMENT,  
 `username` varchar(50) NOT NULL,  
 `password` varchar(255) NOT NULL,  
 `created\_at` datetime DEFAULT CURRENT\_TIMESTAMP,  
 PRIMARY KEY (`id`),  
 UNIQUE KEY `username` (`username`)  
) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=latin1  
  
//  
  
  
INSERTS  
  
insert into med  
(Med\_Name, Inventory)  
Values  
("Astrovastatin", 14),  
("Levothyroxine", 1),  
("Lisinopril", 2),  
("Prinivil", 3),  
("Metformin", 4),  
("Amlodipine", 5),  
("Simvastatin", 6),  
("Zocor", 7),  
("Metoprolol", 8),  
("Cozaar", 9),  
("Losartan", 10),  
("Amoxicillin", 11),  
("Fluticasone", 12),  
("Tramadol", 13),  
("Carvedilol", 15),  
("Warfarin", 16),  
("Coumadin", 17),  
("Ventolin", 18),  
("Trazodone", 19),  
("Klor-Con", 20)  
  
  
Doctor insert  
INSERT INTO doctor  
(Doctor\_Name, Doctor\_ZIP )  
VALUES  
("Dr. Albert", 12345),  
("Dr. Bruel", 42312),  
("Dr. Phill", 21321),  
("Dr. Doolittle", 58493),  
("Dr. Doom", 53423),  
("Dr. Faustus", 94949),  
("Dr. Light", 31233),  
("Dr. Mid-nite", 32321),  
("Dr. Fate", 58488),  
("Dr. Manhattan", 32132),  
("Dr. Octopus", 32112),  
("Dr. Jekyll", 12341),  
("Dr. Lecter", 12345),  
("Dr. Watson", 54124),  
("Dr. Strangelove", 12312),  
("Dr Pyckle", 44444),  
("Dr. Syn", 55555),  
("Dr. Who", 62323),  
("Dr. Kane", 99788),  
("Dr. Sleep", 78978)  
  
INSERT INTO employee  
(Employee\_Name, Employee\_Title )  
VALUES  
("Steve", "Clerk"),  
("Nick", "Manager"),  
("Phill", "Clerk"),  
("Kevin", "Clerk"),  
("Drew", "Manager"),  
("Anna", "Manager"),  
("Mary", "Manager"),  
("Mike", "Clerk"),  
("Francis", "Clerk"),  
("Albert", "Clerk"),  
("Bob", "Manager"),  
("Sarah", "Manager"),  
("Stephan", "Clerk"),  
("Joseph", "Clerk"),  
("Tom", "Clerk"),  
("Kyle", "Clerk"),  
("Kara", "Manager"),  
("Tara", "Clerk"),  
("Beth", "Clerk"),  
("Liz", "Clerk")  
  
  
   
 insert into prescription (Doctor\_ID, Customer\_ID, Med\_ID, Refill\_Amount)  
 Values  
 (1,6,3,1),  
 (2,7,5,4),   
 (3,8,3,3),  
 (4,9,4,10),  
 (5,10,5,12),  
 (6,11,4,20),  
 (7,12,13,11),  
 (8,13,12,12),  
 (9,14,17,3),  
 (10,15,10,2),  
 (11,14,9,5),  
 (12,13,12,2),  
 (13,12,13,1),  
 (14,10,5,7),  
 (15,11,7,2),  
 (16,6,9,4),  
 (17,12,11,3),  
 (18,19,13,9),  
 (19,7,12,48),  
 (20,9,7,3)  
  
  
insert into customer  
(Customer\_Name, Cust\_Address, Doctor\_ID )  
VALUES  
("Steve", "123 Street", 1),  
("Nick", "321 Road" , 2),  
("Phill", "312 Lane", 3),  
("Kevin", "402 Drive", 4),  
("Drew", "32 Lane", 5),  
("Anna", "123 Mockingbird", 5),  
("Mary", "123 Street rd", 6),  
("Mike", "123 Home", 1),  
("Francis", "5233 Main st", 2),  
("Albert", "123 Apt street", 8),  
("Bob", "PO box 12312", 10),  
("Sarah", "california street", 11),  
("Stephan", "123 Streeting", 12),  
("Joseph", "422 Address", 13),  
("Tom", "393 Address", 14),  
("Kyle", "123 Home", 15),  
("Kara", "321 Apt", 16),  
("Tara", "30 2nd", 17),  
("Beth", "32 3rd street",18),  
("Liz", "Main Street", 19)  
  
  
insert into restock\_Order ( Employee\_ID, Med\_ID, Order\_Amount)  
Values  
(12,13,12),  
(6,8,9),  
(7,12,13),  
(7,5,4),  
(11,12,12),  
(9,11,13),  
(7,8,6),  
(8,7,6),  
(7,6,5),  
(5,6,7),  
(5,5,5),  
(8,8,8),  
(6,6,6),  
(7,7,7),  
(0,9,13),  
(11,13,14),  
(8,10,10),  
(5,4,1),  
(10,10,10),  
(12,11,9)  
INE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=latin1  
  
insert into customer\_Order (Employee\_ID, Prescription\_ID, OrderAmount)  
Values  
(12,13,12),  
(6,8,9),  
(7,12,13),  
(7,5,4),  
(11,12,12),  
(9,11,13),  
(7,8,6),  
(8,7,6),  
(7,6,5),  
(5,6,7),  
(5,5,5),  
(8,8,8),  
(6,6,6),  
(7,7,7),  
(0,9,13),  
(11,13,14),  
(8,10,10),  
(5,4,1),  
(10,10,10),  
(12,11,9)  
  
  
  
CREATE INDEX docWrote  
ON prescription (Doctor\_ID);  
  
Create Index custPrescriptions  
on prescription (Customer\_ID);  
  
create index patientOf  
on customer (Doctor\_ID);