Group: Cheuk On Yim, Paola Torres, Matthew Stoney, Wicaksa Munajat

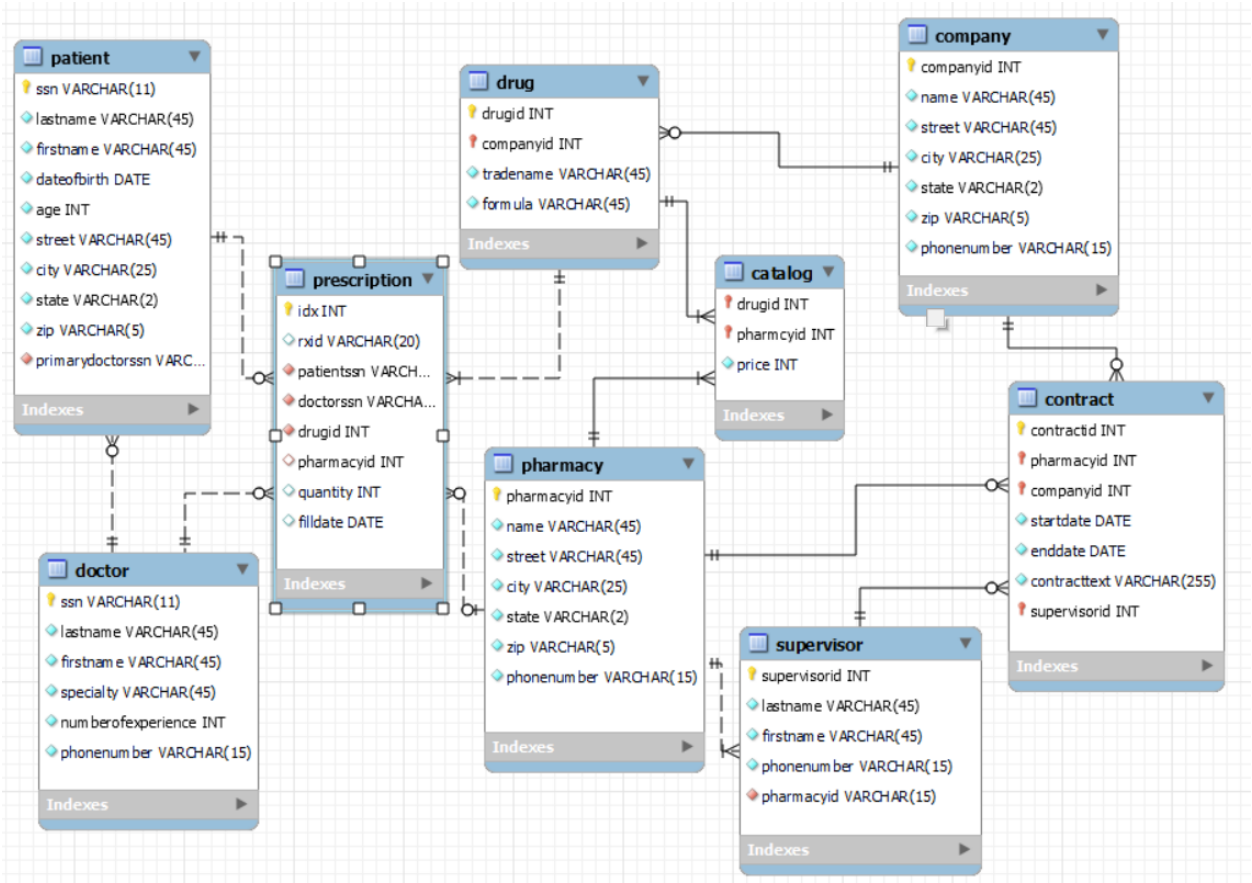
Course/Year: CST 363 CSUMB Summer ‘21

Assignment: Databases – Project 2 Java Web Programs and DB (Updated Report)

Pharmaceutical Relational Database Design

Introduction

Our consulting team has been tasked with designing a relational database for a growing drug store chain. This relational database will assist the drug store chain in managing different aspects of the pharmaceutical business including patient information, doctor information, pharmaceutical company information, pharmacy information, drug information, and prescription information. Different attributes within these entities will be created and cardinality relationships between them will be drawn in order to create an easy to access query. Overall, this database can improve the efficiency, accuracy, and safety of a drug store chain. In addition, it can increase the reliability of storing and retrieving pharmacy records and provide better service to the customers. Below is our ER diagram followed by paragraphs describing the design choices for each entity.



Patient

The creation of this entity is relatively straight forward. Since each patient has a unique SSN, using the SSN (INT) as a primary key allows a uniquely identifiable piece of information to keep track of each patient entity. Each patient also has a last name (lastname VARCHAR(45)) and a first name (firstname VARCHAR(45)) attribute to represent their identity. Date of birth is an attribute of this field since it is a common piece of information that helps with identifying a patient in addition to determining a patient's age. In addition, it is more convenient to store a date of birth rather than an age, because an age requires updates periodically and if this entity expands, it would take more resources to make sure all patient's ages are correct.

A few more attributes such as (street VARCHAR(45), city VARCHAR(25), state VARCHAR(2), zip INT) should also be created in order to store a patient's address. Lastly, a patient requires an attribute (primarydoctorssn INT) to identify the patient's primary doctor in which a patient has to have one and only one primary doctor.

Doctor

Similar to the Patient entity, a social security number (ssn INT) allows for a uniquely identifiable attribute of a doctor which will serve as the primary key. A last name (lastname VARCHAR(45)) and a first name (a firstname VARCHAR(45)) are needed to identify a doctor's full name. The specialty attribute (specialty VARCHAR(45)) will be used to indicate a doctor's specialty. Each doctor also has years of experience (numberofexperience INT) which indicates the number of experience the doctor has and a phone number (phonenumber INT) which facilitates communication with a doctor. A doctor also has a non-identifying relationship with zero or many patients.

Prescription

The prescription entity represents a notice of doctor approval that a patient may purchase a drug from a pharmacy. A prescription is identified by idx INT , which is the primary key. It has foreign key fields (patientssn VARCHAR(11), doctorssn VARCHAR(11), drugid INT, pharmacyid INT). A patient can have zero or many prescriptions. A doctor can prescribe zero or many prescriptions. If a doctor prescribes more than one drug to the same patient, a corresponding number of prescriptions should be created. A drugid attribute indicates the drug of the prescription from the drug entity. A pharmacy id attribute indicates a non-identifying relationship with a pharmacy because a pharmacy can fill zero or many prescriptions. A prescription can be filled by zero or one pharmacy. Because each prescription is unique, it must have exactly one doctor, one patient, one drug and one pharmacy. A quantity attribute (quantity INT) indicates the quantity of the drug that should be filled. A fill date attribute (filldate DATE) indicates the last fill date of the prescription.

Drug

The drug entity contains the identification number attribute (drugid INT), trade name, formula, and company identification number (companyid INT). The attributes drugid and companyid serve as a composite key that uniquely identifies a drug since each drug is made from only one company. Even though a drug's trade name (tradename VARCHAR(45)) is unique to a drug within a company, it is not a good idea to use it as a primary key nor part of the composite key since other companies can have a similarly named if not identically named product.

The drug entity has a one to many relationship with the prescription entity since the prescriptions indicate one particular drug per prescription, but a drug can be in multiple

prescriptions. In the case of the catalog entity, each drug can appear in many different catalogs for different pharmacies, therefore it has a one to many relationship.

Pharmacy

The pharmacy entity represents a retailer of pharmaceutical products. It is uniquely identified by its primary key, pharmacyID (INT type). Other attributes that describe the pharmacy entity include the name, street, city, state, zip and phone number (all VARCHAR type).

The pharmacy entity also has relationships to other entities in the database. It is related to the prescription entity which indirectly connects the pharmacy to the patient, doctor, and drug entities. The pharmacy entity has a zero to many relationship with the prescription entity, while a prescription can only be attributed to one pharmacy.

The pharmacy entity is also connected to the catalog entity. The catalog represents the pharmacy's price index for each drug. So each pharmacy can have one or many catalogues but each catalog has exactly one pharmacy.

The next relationship is between the pharmacy entity and the contract entity. The contract represents a long-term agreement between pharmacies and pharmaceutical companies to manufacture and distribute drug products. Each pharmacy can have zero or many contracts with different companies.

Finally, the pharmacy entity is related to the supervisor entity. The pharmacy appoints a supervisor to oversee each of its contracts with pharmaceutical companies. A pharmacy can have many supervisors, but each supervisor has exactly one pharmacy.

Catalog

Each catalog has two foreign keys that reference the drug and pharmacy entities. Along with this, catalogs also contain a price attribute. Each pharmacy can sell a drug at a different price than its competitors. Catalogs have a one to one relationship with the pharmacy entity because each pharmacy has only one catalog of all the drugs that they sell. Catalogs have a one to many relationship with the drug entity because each drug can appear in more than one catalog since each pharmacy can sell it for a different price. Catalog also has a composite primary key of drug id and pharmacy id since both uniquely identify the price of the drug in the catalog.

Company

A pharmaceutical company contains the company's name, phone number, and address as attributes and is identified by its primary key, companyid. A pharmaceutical company has a one to many relationship with the relation entity drug. Each company can sell many different types of prescription drugs where each prescription drug has a different trade name from the other products that specific company sells. Pharmaceutical companies also have a one to many relationship in the form of contracts with pharmacies. This relationship is also one to many because each pharmaceutical company can have many different contracts with different pharmacies.

Contract

A contract is an entity that links pharmaceutical companies to the pharmacies that sell the drugs. Each pharmaceutical company can have zero or more contracts. Each pharmacy can have zero or more contracts, as well. However, each contract is unique and can have a one to one relationship with both pharmaceutical companies and pharmacies. Each contract must have one supervisor that manages it. Because the contract is an intermediary entity that links

pharmaceutical companies to pharmacies, each contract is identified by a composite key that has a contractID, the pharmacyID of its associated pharmacy, the companyID of its associated pharmaceutical company, as well as the supervisorID of the contract supervisor. Contract entities also have attributes containing the start and end dates (DATE type) of the contract and the actual textual agreement of the contract (VARCHAR type).

Supervisor

The supervisor entity represents the manager of a contract. Each contract is a unique relationship that associates a pharmacy with a pharmaceutical company. The supervisor entity is uniquely identified by the attribute supervisorID (INT type). It also contains the attributes lastname, firstname (VARCHAR type), as well as phonenumber (INT type). The attribute appointedbypharmacyid (INT type) indicates the pharmacy that appoints the supervisor.

Relational schema derived from the ER model

```
-- create the database
DROP DATABASE IF EXISTS `drugstore_db`;
CREATE DATABASE `drugstore_db`;
-- select the database
USE `drugstore_db`;
-- -----------------------------------------------------
-- Table `drugstore_db`.`company`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `drugstore_db`.`company` (
  `companyid` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `street` VARCHAR(45) NOT NULL,
  `city` VARCHAR(25) NOT NULL,
  `state` VARCHAR(2) NOT NULL,
  `zip` VARCHAR(5) NOT NULL,
  `phonenumber` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`companyid`)
  );

  -- -----------------------------------------------------
-- Table `drugstore_db`.`doctor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `drugstore_db`.`doctor` (
  `ssn` VARCHAR(11) NOT NULL,
  `lastname` VARCHAR(45) NOT NULL,
  `firstname` VARCHAR(45) NOT NULL,
  `specialty` VARCHAR(45) NOT NULL,
```

```
  `numberofexperience` INT NOT NULL,
  `phonenumber` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`ssn`)
);

-- -----------------------------------------------------
-- Table `drugstore_db`.`pharmacy`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `drugstore_db`.`pharmacy` (
  `pharmacyid` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `street` VARCHAR(45) NOT NULL,
  `city` VARCHAR(25) NOT NULL,
  `state` VARCHAR(2) NOT NULL,
  `zip` VARCHAR(5) NOT NULL,
  `phonenumber` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`pharmacyid`)
  );

-- -----------------------------------------------------
-- Table `drugstore_db`.`drug`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `drugstore_db`.`drug` (
  `drugid` INT NOT NULL,
  `companyid` INT NOT NULL,
  `tradename` VARCHAR(45) NOT NULL,
  `formula` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`drugid`, `companyid`),
  FOREIGN KEY (`companyid`) REFERENCES `drugstore_db`.`company` (`companyid`)
  );

-- -----------------------------------------------------
-- Table `drugstore_db`.`patient`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `drugstore_db`.`patient` (
  `ssn` VARCHAR(11) NOT NULL,
  `lastname` VARCHAR(45) NOT NULL,
  `firstname` VARCHAR(45) NOT NULL,
  `age` INT NOT NULL,
  `street` VARCHAR(45) NOT NULL,
  `city` VARCHAR(25) NOT NULL,
  `state` VARCHAR(2) NOT NULL,
  `zip` VARCHAR(5) NOT NULL,
  `primarydoctorssn` VARCHAR(11) NOT NULL,
  PRIMARY KEY (`ssn`),
  FOREIGN KEY (`primarydoctorssn`) REFERENCES `drugstore_db`.`doctor` (`ssn`)
  );

-- -----------------------------------------------------
-- Table `drugstore_db`.`supervisor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `drugstore_db`.`supervisor` (
  `supervisorid` INT NOT NULL,
  `lastname` VARCHAR(45) NOT NULL,
  `firstname` VARCHAR(45) NOT NULL,
```

```
  `phonenumber` VARCHAR(15) NOT NULL,
  `pharmacyid` INT NOT NULL,
  PRIMARY KEY (`supervisorid`),
  FOREIGN KEY (`pharmacyid`)
      REFERENCES `drugstore_db`.`pharmacy` (`pharmacyid`)
);-- -----------------------------------------------------
-- Table `drugstore_db`.`catalog`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `drugstore_db`.`catalog` (
  `drugid` INT NOT NULL,
  `pharmacyid` INT NOT NULL,
  `price` INT NOT NULL,
  PRIMARY KEY (`drugid`, `pharmacyid`),
  FOREIGN KEY (`drugid`) REFERENCES `drugstore_db`.`drug` (`drugid`),
  FOREIGN KEY (`pharmacyid`) REFERENCES `drugstore_db`.`pharmacy` (`pharmacyid`)
  );

-- -----------------------------------------------------
-- Table `drugstore_db`.`contract`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `drugstore_db`.`contract` (
  `contractid` INT NOT NULL,
  `pharmacyid` INT NOT NULL,
  `companyid` INT NOT NULL,
  `startdate` DATE NOT NULL,
  `enddate` DATE NOT NULL,
  `contracttext` VARCHAR(255) NOT NULL,
  `supervisorid` INT NOT NULL,
  PRIMARY KEY (`pharmacyid`, `companyid`, `contractid`, `supervisorid`),
  FOREIGN KEY (`pharmacyid`) REFERENCES `drugstore_db`.`pharmacy` (`pharmacyid`),
  FOREIGN KEY (`companyid`) REFERENCES `drugstore_db`.`company` (`companyid`),
  FOREIGN KEY (`supervisorid`)REFERENCES `drugstore_db`.`supervisor` (`supervisorid`)
  );

-- -----------------------------------------------------
-- Table `drugstore_db`.`prescription`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `drugstore_db`.`prescription` (
  `rxID` VARCHAR(20),
  `idx` INT NOT NULL AUTO_INCREMENT,
  `patientssn` VARCHAR(11) NOT NULL,
  `doctorssn` VARCHAR(11) NOT NULL,
  `drugid` INT NOT NULL,
  `pharmacyid` INT COMMENT 'Can be null. because a prescription is not filled when
first created.',
  `quantity` INT NULL,
  `filldate` DATE NULL,
  PRIMARY KEY (`idx`),
  FOREIGN KEY (`patientssn`) REFERENCES `drugstore_db`.`patient` (`ssn`),
  FOREIGN KEY (`doctorssn`)REFERENCES `drugstore_db`.`doctor` (`ssn`),
  FOREIGN KEY (`drugid`)REFERENCES `drugstore_db`.`drug` (`drugid`),
  FOREIGN KEY (`pharmacyid`)REFERENCES `drugstore_db`.`pharmacy` (`pharmacyid`)
  );
```

Normalized relational schema

Normalization is the process of organizing data in order to reduce redundancy that can create future problems when inserting, deleting, and updating data. In our design, we tried to see which columns could be normalized. We concluded that the addresses could be split up into another table since there are functional dependencies between the attributes such as knowing the zip code can tell someone about the city and state.

However, we decided to keep this design unnormalized. Normalizing this would most likely lower redundancy and lower space usage but since zip codes, city, and state are mostly static, they won't get updated that much so there would not be a problem with updating. Normalizing will also increase the difficulty in creating queries and increase performance usage since we are creating another relation in our database.

SQL Queries

1). Display the first name as 'PatientFirstName', last name as 'PatientLastName', and age as 'PatientAge' of patients who live in the state of California and who are under the care of a cardiologist with years of experience over 5 years. Sort by number of experiences.

```
SELECT p.firstname as PatientFirstName, p.lastname as PatientLastName, p.age as
PatientAge
FROM patient p, doctor d
WHERE p.State = 'CA' AND d.Specialty = 'Cardiologist' AND p.primarydoctorssn = d.ssn
AND d.NumberOfExperience > 5
ORDER BY d.NumberOfExperience;
```

2). Display the drug id, and the average price of that drug. Sort by INCREASING drug price.

```
SELECT c.DrugID, AVG(c.price)
FROM catalog c
GROUP BY c.DrugID
ORDER BY c.drugID ASC;
```

3). Display the quantity per drug

```
SELECT d.drugid, SUM(p.quantity)
```

```
FROM drug d, prescription p
WHERE d.drugid = p.drugid
GROUP BY d.drugid;
```

4). Display the city, state, zip code, and number of pharmacies in zip codes that have at least 2 pharmacies, order by state.

```
SELECT city, state, zip, count(zip)
FROM pharmacy
GROUP BY zip, city, state
having count(zip) >= 2
ORDER BY state;
```

5). Display the contractid that has both its pharmacy and company located in state ‘CA’, also the contract’s supervisorid and the supervisor’s phonenumber.

```
Select c.contractid, s.supervisorid, s.phonenumber
from contract c, pharmacy p, company com, supervisor s
where c.pharmacyid = p.pharmacyid and c.companyid = com.companyid
and p.state = 'CA' and com.state = 'CA' and c.supervisorid = s.supervisorid;
```

Java Web Application and Database

We have also created web pages that serve as a UI in order to create an easier user interaction to insert or pull data from the database. We implemented code that allows:

1). a doctor to write a prescription for a patient for a drug and quantity

2). a patient to request a pharmacy prescription

3). a pharmacy manager to request a report of the quantity of drugs that have been used to fill prescriptions by the pharmacy

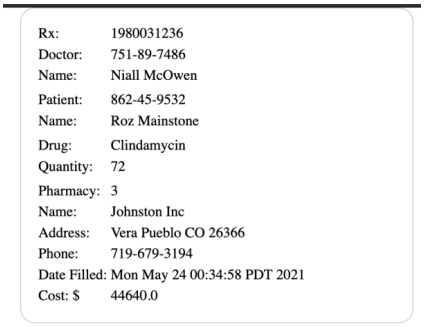
4). an FDA official to look at the quantity of drugs each doctor has prescribed.

Down below are screen shots of each of these examples along with a description of what the screen shot entails.

1. **A doctor prescription:** The doctor has a form that allows them to create a new prescription. Filling out the prescription and clicking create prescription will add a new row into our prescription table and show the user the prescription that was generated.

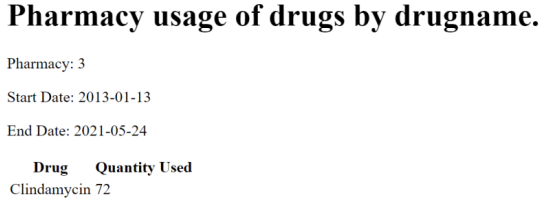
2. **Prescription error:** If there was an error in entering the information (ie. no entry exists in the database for a particular query), it will show the user an error message.

3. **Patient prescription:** The patient also has the ability to create a new prescription. After filling out the form, information will be validated in the backend. If the information was valid, the user will be shown a screen with their prescription information.

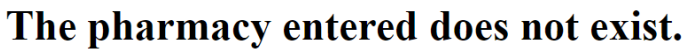


4. **Prescription error:** If the user enters incorrect information, for example RXID, they will be prompted an error screen telling them which error it corresponds to.

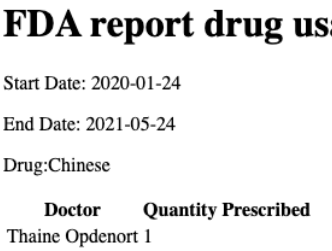
5. **Pharmacy drug report:** Pharmacy can look up a drug report that shows how many drugs have been prescribed by the pharmacy.



6. **Pharmacy drug error:** When the user enters incorrect information, an error page will be prompted.



7. **FDA lookup:** FDA can look up the drug and quantity of drug that each doctor has prescribed.



8. **FDA error:** When the user enters incorrect information, an error page will be prompted.

Conclusions

This project examined the creation of a large and scalable relational database for a pharmaceutical company. We have created a great foundation for a relational database that started from the planning stage of an ER diagram. The database has data pertaining to doctors, patients, companies, pharmacies, prescriptions, drugs, supervisors, and catalog. In order to piece the database together, an ER model was created to outline the relationship between the different entities. After the ER diagram was modeled, we also checked for opportunities to normalize the data. Since ours was structured in an ER diagram, we did not find it necessary to further normalize the data because the design minimized the redundancies in the data. After the normalization step, we created queries that a potential client can perform on the database as example usage cases. These queries were included in both written english and SQL statements. This particular project illustrated the potential situation of a real world client request.

The steps practiced in the assignment covered the database basics that are necessary to creating a concise and practical relational database. Through these steps, we have gained further understanding of the inner working of a relational database and its real world applications.