Drug Store Database Design

Project 2 - 05.25.2021

Authors:

Sarom Thin,
Mehar Rekhi
Rahul Khurana,
Jordan Cruz-Goodwin

Introduction

We have been tasked with designing a relational database for an up and coming drug store chain. There are different types of data provided by the client that will be used to create a manageable system. The data provided includes information on Patients, Doctors, Pharmacies, Drug prescriptions, and Pharmaceutical companies. Each patient is linked to a doctor whom they have visited a certain amount of times. The doctor writes prescriptions on a certain date and for a certain amount (quantity). The prescription is then used to obtain the prescribed drug, which has a trade name and formula. The drug is tracked by the amount sold and who produces the drug. The amount sold and price for the drug is determined by which pharmacy carries and sells a particular drug. The pharmaceutical company, who produces the drug, creates contracts with various pharmacies to allow the selling of the produced drugs. The contract will contain start/end dates, the contents of the contract, and who supervised this agreement.

The objective is to create a relational database for the drug store chain to centrally control the information of all the Patients, Doctors, Pharmacies, Drug prescriptions, and Pharmaceutical companies. The database will provide access to this information for possible patient history reports, drug sales information, and pharmacy prescription filling reports. With a centralized system, queries would be filled in a moment's notice.

ER Model

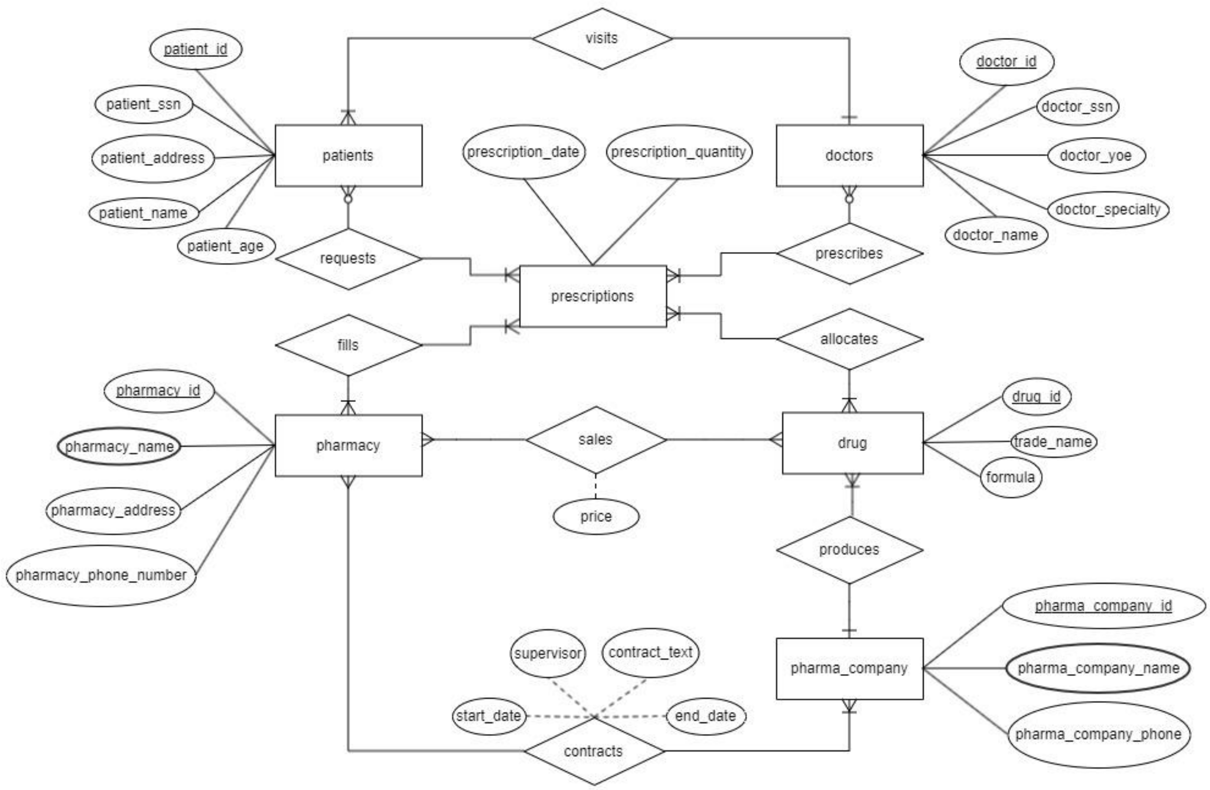

Fig.1: ER Diagram

- For the initial part of the spec, ensuring that each patient has an identifying SSN, name, age, and address, we created a Patient table, using a primary key of a generated auto-incremented Patient ID, using their SSN, name, address, and age as mandatory fields. It was assumed that all fields were mandatory to create a complete profile of each patient for their medical records.

- As each doctor also required a name, SSN, specialty, and years of experience, a separate Doctor table was created along similar lines, likewise with all fields assumed as mandatory. The patient's primary physician was identified with a foreign key referencing the doctor ID.

- For each pharmaceutical company, we again created a pharmaceutical company ID as the primary key and stored the name and phone number in this table.

- We also created a drug table with its own drug ID that referenced the pharmaceutical company's ID as a foreign key, along with its trade name and formula. To ensure that the specification on the pharmaceutical company being deleted, a “ON DELETE CASCADE” setting was set on the foreign key for the drug table.

- For the pharmacy, a pharmacy table was created, each with an ID, name, address, and phone number. Sale prices are tracked in a separate Sales page that matches the drug ID and pharmacy name combined as a primary key to a price.

- As doctors prescribe drugs to patients in a many to many relationships, a prescriptions table was created. While the use of a doctor/drug/patient combination as a primary key was considered, we opted to create a prescription ID that auto increments. This was then tied to the requisite information of the doctor, patient, drug, date, and quantity.

- As pharmaceutical companies also have long-term contracts with pharmacies on a many to many basis, a contracts table was created, storing a start date, end date, the text of the contract, and a supervisor for the contract. As no additional information regarding the supervisor was mentioned in the specification, it was assumed that the supervisor’s name was sufficient, and the supervisor was stored as a varchar.

Relational Schema

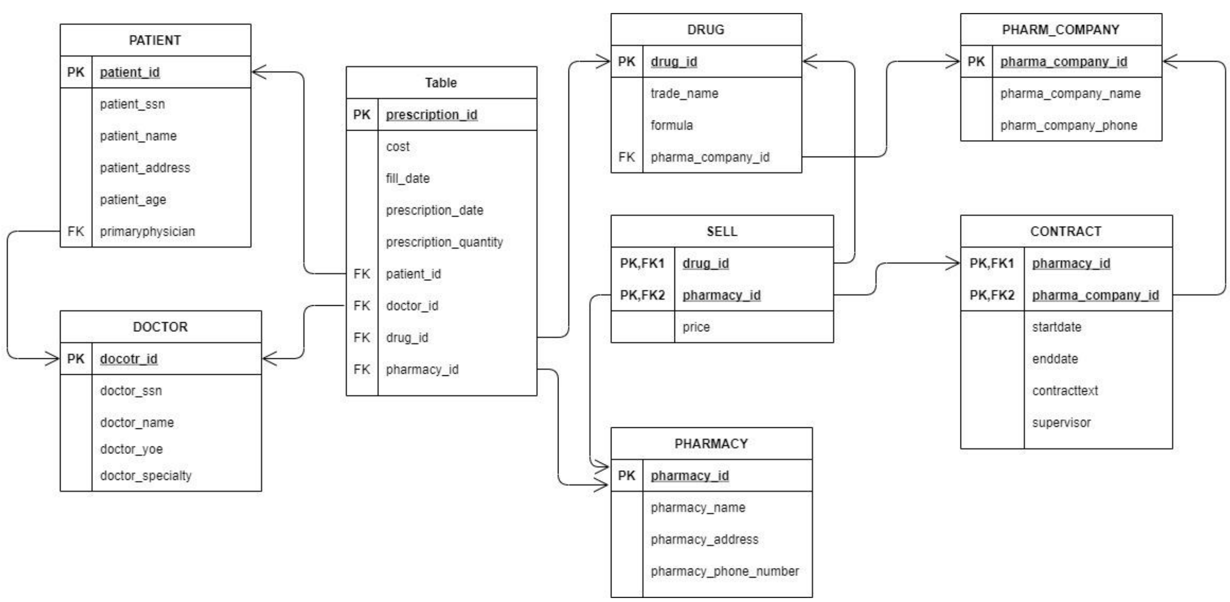

Fig.2: Relation Schema

SQL Code

```sql
-- create the database
DROP DATABASE IF EXISTS project2;
CREATE DATABASE project2;

-- select the database
USE project2;


-- Creating tables for database setup
Create TABLE doctors
(
  doctor_id             int             Primary Key     AUTO_INCREMENT,
  doctor_ssn            int             NOT NULL,
  doctor_name           VarChar(45)     NOT NULL,
  doctor_specialty      VarChar(45)     NOT NUll,
  doctor_yoe            INT             NOT NULL
);

Create Table pharmacy
(
  pharmacy_id           int             Primary Key     AUTO_INCREMENT,
  pharmacy_name         VarChar(45)     UNIQUE,
  pharmacy_phone_number VarChar(10)     NOT NULL,
  pharmacy_address      VarChar(90)     NOT NULL
);

Create Table pharma_company
(
  pharma_company_id     int            Primary Key      AUTO_INCREMENT,
  pharma_company_name   VarChar(60)    NOT NULL,
  pharma_company_phone  VarChar(10)    NOT NULL
);

CREATE TABLE patients
(
  patient_id            int             Primary Key     AUTO_INCREMENT,
  patient_ssn           int             NOT NULL,
  patient_name          VarChar(45)     NOT NULL,
  patient_address       VarChar(90)     NOT NULL,
  patient_age           int             NOT NULL,
  primary_doctor_id     INT             NOT NULL,
  CONSTRAINT patients_fk_doctors
    FOREIGN KEY (primary_doctor_id)
    REFERENCES doctors(doctor_id)
);

Create Table drugs
(
  drug_id               int             Primary Key     AUTO_INCREMENT,
  pharma_company_id     int             NOT NULL,
  trade_name            VarChar(45)     NOT NULL,
  formula               LONGTEXT        NOT NULL,
  CONSTRAINT drugs_fk_pharma_company
    FOREIGN KEY (pharma_company_id)
    REFERENCES pharma_company(pharma_company_id)
    ON DELETE CASCADE
);
```

```sql
CREATE TABLE prescriptions
(
  prescription_id        int            Primary Key     AUTO_INCREMENT,
  patient_id             int            NOT NULL,
  doctor_id              int            NOT NULL,
  drug_id                int            NOT NULL,
  pharmacy_id            int            NULL,
  prescription_quantity  int            NOT NULL,
  prescription_date      DATE           NOT NULL,
  fill_date              DATE           NULL,
  cost                   DOUBLE         NULL,
  CONSTRAINT prescriptions_fk_patients
    FOREIGN KEY (patient_id)
    REFERENCES patients(patient_id),
  CONSTRAINT prescriptions_fk_doctors
    FOREIGN KEY (doctor_id)
    REFERENCES doctors(doctor_id),
  CONSTRAINT prescriptions_fk_drugs
    FOREIGN KEY (drug_id)
    REFERENCES drugs(drug_id),
  CONSTRAINT prescriptions_fk_pharmacy
    FOREIGN KEY (pharmacy_id)
    REFERENCES pharmacy(pharmacy_id)
);

Create Table sales
(
  pharmacy_id            int            NOT NULL,
  drug_id                int            NOT NULL,
  price                  DECIMAL(10)    NOT NULL,
  PRIMARY KEY (pharmacy_id, drug_id),
  CONSTRAINT pharmacy_drug_fk_pharmacy
    FOREIGN KEY (pharmacy_id)
    REFERENCES pharmacy(pharmacy_id),
  CONSTRAINT pharmacy_drug_fk_drug
    FOREIGN KEY (drug_id)
    REFERENCES drugs(drug_id)
);

Create Table contracts
(
  pharmacy_id            int            NOT NULL,
  pharma_company_id      int            NOT NULL,
  supervisor             VarChar(45)    NOT NULL,
  start_date             DATE           NOT NULL,
  end_date               date           NOT NULL,
  contract_text          LONGTEXT       NOT NULL,
  PRIMARY KEY (pharmacy_id, pharma_company_id),
  CONSTRAINT contracts_fk_pharmacy
    FOREIGN KEY (pharmacy_id)
    REFERENCES pharmacy(pharmacy_id),
  CONSTRAINT contracts_fk_company_id
    FOREIGN KEY (pharma_company_id)
    REFERENCES pharma_company(pharma_company_id)
);
```

Fig.3 medical_world.sql

Checks for Normalized Design

- As part of our analysis of the normalization of this design, we first went through the primary forms of normalization - in this exercise, we checked for 1NF, 2NF, and 3NF.
- For 1NF, we did not significantly concern ourselves, since we are using a relational database, which enforces 1NF by nature.
- For 2NF, we are attempting to determine if there are any columns dependent on a partial part of the primary key. Since the majority of our tables were single-column primary key tables, we excluded all but the “Sell” table and the “Contracts” table, since those were the only two that had multi-column tables.
    - For the case of the “Sell” table, which determined pricing for each drug on a per-pharmacy basis, there is only one field - the price field - which we needed to analyze. It was quickly determined that both the drug ID and the pharmacy ID are required to determine the price, since the same drug might have different prices at different pharmacies, and the same pharmacy can sell different drugs at different prices.
    - For the contracts table, the start date, end date, contract text, and supervisor clearly all depend on the individual contract, which is also uniquely dependent on the combination of the pharmacy and the pharmaceutical company. Ergo, we deemed the 2NF satisfied.
- For the 3NF, we checked to see if there were any transitive functional dependencies - identifying if any column was dependent on a column that was not the primary key.
    - For the Patients table, it was immediately clear that the SSN could uniquely determine all aspects of the patient’s information (Name, address, age, doctor), however, we considered it a poor choice to use or pass such a sensitive field as the primary key for identification of a patient. As the patient ID could *also* uniquely identify all of the other fields without these consequences, and there were no other violations of the 3NF, we considered the non-normalized nature of this field acceptable.
    - The Doctor table has a similar case with the 3NF being potentially violated with SSN, but a similar argument came to bear. All other fields had no functional dependency on each other.
    - The prescription table cleanly matched the 3NF, as the date, quantity, patient ID, doctor ID, and drug ID all clearly have no dependency except on the primary key.
    - The drug table also cleanly matched the 3NF, as the trade name, formula, and pharmaceutical company were all solely dependent on the primary key of the drug ID.
    - The Sell table was immediately excluded from analysis as it only had a single field other than the composite primary key.
    - The pharmacy table was deemed acceptable within the 3NF as the pharmacy name, address, and phone number were all uniquely dependent on the primary key. While an argument could be made that the phone number or address might be used to identify the other fields, the dependency is not strong or reliable

enough (potentially multiple pharmacies at the same address in an edge case, potentially reuse of a name or corporate phone number for a chain pharmacy), and thus we considered this table acceptable within the 3NF.

- A similar story was true for the pharmaceutical companies table - the pharmaceutical company ID was the only field capable of completely and unambiguously identifying the other fields.
- For the contracts, the situation was likewise clear cut - the start date, end date, contract text, and supervisor were all clearly independent of each other and only dependent on the composite primary key.

SQL Queries

Below are queries we believed that a manager of this drug chain would have for our database:

- An important query a store manager might have is which patient has more than one prescribing doctor. This would help differentiate which doctor prescribes which drug to which patient.
  - What patients are being prescribed by more than one doctor?

```
-- Display patients details who has more than one prescribing doctors
SELECT prescriptions.patient_id, patient_name
FROM patients, prescriptions
WHERE prescriptions.patient_id=patients.patient_id
GROUP BY patient_id
HAVING COUNT(prescriptions.patient_id) > 1;

```

- A store manager would need to know the surrounding competition's price for each drug they were filling. This would help provide a competitive edge and offer a lower average price than other drug stores.
  - What would each pharmacy's average price be for each drug?

```

-- Display average price by drug for pharmacy_name
SELECT d.trade_name as DRUG, p.pharmacy_name as PHARMACY, round(avg(s.price)) as AVERAGEPRICE
FROM drugs d
INNER JOIN sales s ON d.drug_id = s.drug_id
INNER JOIN pharmacy p ON s.pharmacy_id = p.pharmacy_id
GROUP BY d.trade_name;
```

- Another query a store manager would have would be what doctors have more experience than compared to another doctor. This would help understand the experience behind the doctor who is writing the prescription.
    - What doctors have more experience than doctor ‘john doe’?

```
-- Display doctors with more years of experience than "Dr.JohnDoe"
SELECT d2.doctor_name
FROM doctors d1
    JOIN doctors d2
    ON d2.doctor_yoe > d1.doctor_yoe
WHERE d1.doctor_name = "Dr.JohnDoe";
```

- There may be a time where a store manager would like to know the number of varieties of a product from other stores. This can help a store manager to figure out if they need to expand their line up or not.
    - What is the number of types of products being sold at each store?

```
SELECT pharmacy_name, COUNT(drug_id) AS Num_Types_Of_Drugs
FROM pharmacy, sales
WHERE sales.pharmacy_id=pharmacy.pharmacy_id
GROUP BY pharmacy.pharmacy_id;
```

- For this database, a supervisor attribute was added to the contracts table. The supervisor will oversee the contract between the pharmacy and the pharmaceutical company. Since there exists a supervisor then most likely the manager of the store does not need to consult with the pharmaceutical company directly. So if there is a concern or question with a particular product. They only need to concern themselves with the supervisor of that product of a certain store.
    - What is the name of the supervisor who oversees this particular drug at this store?

```
SELECT pharmacy_name, trade_name, supervisor
FROM sales, pharmacy, contracts, drugs
WHERE sales.pharmacy_id=pharmacy.pharmacy_id
    AND sales.pharmacy_id=contracts.pharmacy_id
    AND sales.drug_id=drugs.drug_id
    AND drugs.pharma_company_id=contracts.pharma_company_id
GROUP BY pharmacy_name, trade_name;
```

Java Spring Webpages

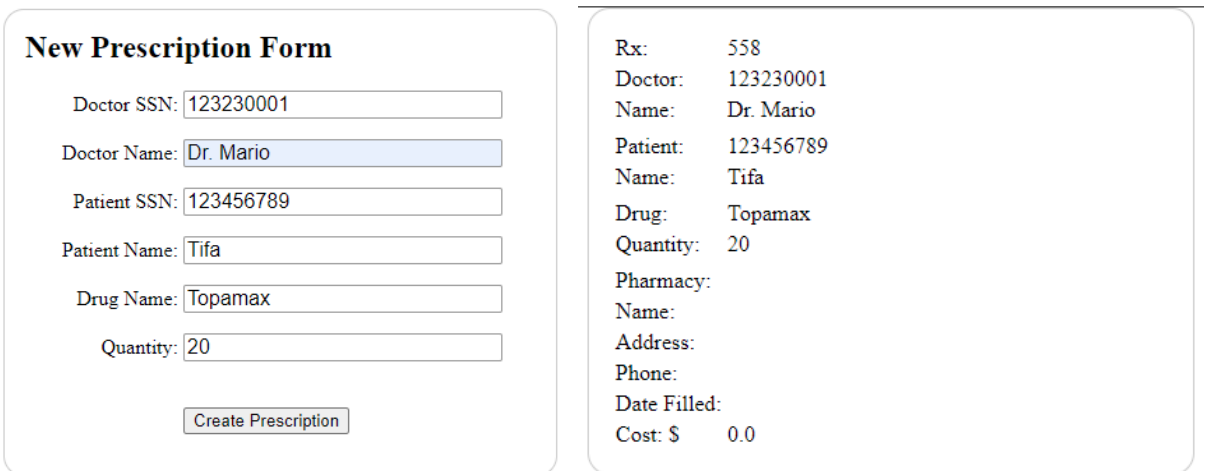

Fig.4 Write new Prescriptions
(For Doctors Only)

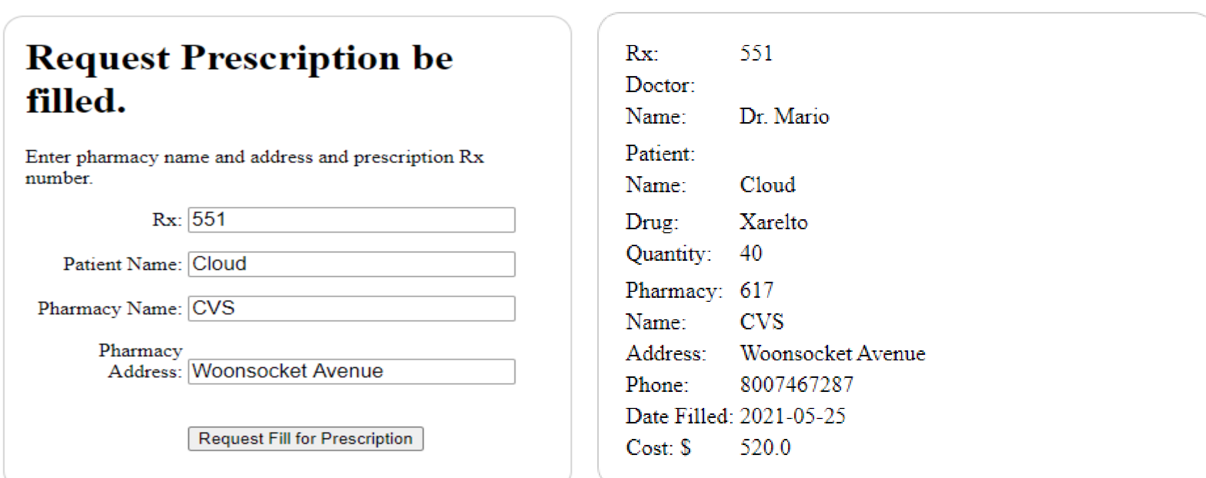

Fig.5 Request a Prescription
(For Patients Only)

Fig.6 Report on Drug Usage
(For Pharmacists Only)

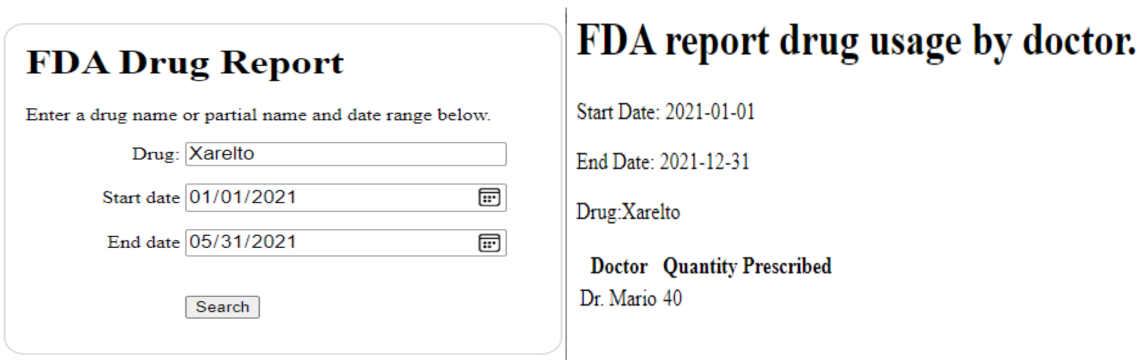

Fig.7 FDA Reporting
(For FDA Only)

Conclusions

To begin this relational database design for an up and coming drug store, we were given a set of information. We took the set of information and discussed the requirements that needed to be met. During the database's developmental process we created a ER Diagram that captured our conceptual model of the list of requirements. There was great emphasis placed on identifying our primary and foreign keys for all entities. After finishing the ER Diagram we mapped it to create a relational database schema. We presented the schema as a diagram and in SQL as CREATE TABLE statements. Next we touched on how we checked our ER Design for a normalized design. We went through the primary forms of normalization and came to the conclusion that our database design was as normalized as we could make it. After agreeing on the design of our database we began testing our database with a test data set. We then began to think about what a store manager of an upcoming drug store would need to query. So we created five SQL queries that we believe would help a store manager. Through this entire design process we learned various things.

The vagueness of the given information taught us the need to communicate with one another, so that the team can move forward on the same page. The ER model design proved to be a necessary fundamental skill. This acted as our blueprint to help make the creation of our schema that much easier. The labeling of our primary and foreign keys made for a smooth hand off between team members who were given different tasks. The schema was fairly straightforward after following the ER Model design. We saved time when it came to normalization, as we took it into account when creating our ER Model and relational schema. Although, we did go through the primary forms of normalization to double check our schema. In the testing portion, we saw the importance of each step of this design process. The more time and effort we put into each step, the following step became that much easier. The final thing we learned was how to look through the eyes of our client. The queries we came up with gave us insight on how important it is to have a manageable database. In order to fill queries and display it in a user friendly manner, there needs to be time and effort made in the database design process. Overall this was a great introduction, and experience in database design that we will take into future endeavors.