Drug Store Chain Database Design

Sophic LLC

Lead Architects – Andrew Paulino and Brandon Mccarthy-Santos

Date Published – 1/23/2022 – Updated 2/10/2022

Sophic LLC

Database Presented – and solution proposed

Given the prompt design and schema we evolutionized a relational database with efficiency and clarity to the end user.

Some important considerations made…

- ☐ **Design with security in mind, separation between highly sensitive PII data and separation between tables has been made.**

- ☐ **Design relational design between pharmaceutical companies and data received from hospital**

- ☐ **Design a schema that would be efficient for the end user to use with instant query lookups.**

Given the proposition of the schema we really want to look at how we can make the data efficient and also the use case for example queries easy to understand. With that being said here is the base idea of the data flow and schema.

Here is a decision log to really reflect some of the requirements and design choices.

1. We decided to keep patient names off of the patient_rx, using foreign keys uniquely identifying the patients as any pertinent third parties acquiring information about a prescription would need to have access to the patient table to access the patients PII.

2. This same design mechanic was also used to mask features about the doctor.

3. We designed a pharmaceutical table to hold company information – and this will hold a relational point for other junction tables in the schema.

4. We designed a designated “drug” table to hold meta data for drugs including which manufacturing pharmaceutical company it came from.

5. We designed a “Pharmacy” table to hold all pharmacy data relevant to each specific pharmacy.

6. To handle all pricing for several drugs and pharmacy concerns, we created a “price_listing” table to contain the drug as referenced and the pharmacy it is in as well as that price with it.

7. To handle the mounting concerns around creating and tracking prescriptions – we decided to create two tables to hold these points of data.
    a. Table one would be “patient_rx” which would contain a pending prescription for a user.
    b. Table two would contain “patient_rx_filled” which would basically hold the idea of when and IF the patient filled their prescription. This will also contain metadata where they filled it and by which pharmacy

8. To deal with contracts between pharmacy and pharmaceutical companies, we decided it would be best to create a “contract” table which would hold references to “pharmacy” table and “pharmaceutical” table.

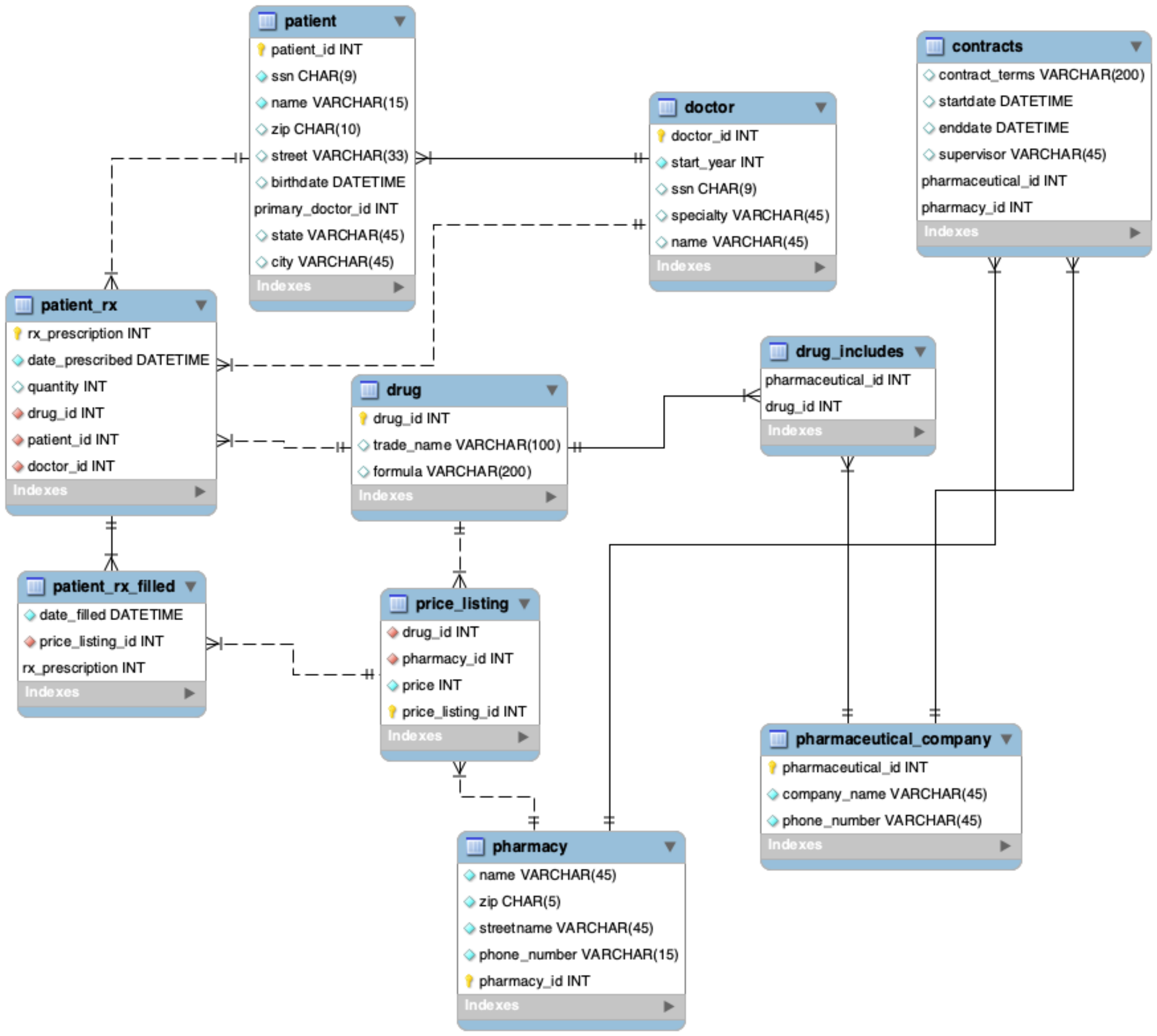
Handling normalizing data

In this section, we wanted to highlight how we handle some normalizing patterns and efficiently handle data transfer between several tables.

We were able to normalize the database in separate tables depending on their particular function. Patient table stores PII as well as an unique id to be referenced in other tables such as the patient_rx table, which will be publicly accessible to pharmacies. By structuring the patient id in this way there exists a potential layer of encapsulation.

We seperated the table for when prescriptions are filled, the price listing for drugs as well as an includes table which would link a drug to their manufacturer.

Schema Design and Diagram



Common Use Cases

Using this instance of our database design relevant questions that management would perhaps want to know could be executed. Such examples are as follows.

Which pharmaceutical company is filling the most prescriptions?

Here we can use the price_listing table combined with prescription filled tables to locate the pharmacy id that has the most occurrences ergo the most prescriptions.

```
select pharmacy_id
from price_listing natural join patient_rx_filled
where pharmacy_id = ( select max(pharmacy_id) from price_listing natural
join patient_rx_filled
group by pharmacy_id);
```

A doctor is accused of writing fraudulent scripts. Make a list of all his patients and the drugs they were prescribed so they may be questioned regarding their prescriptions.

```
select doctor_id, patient_id, drug_id, trade_name
from patient_rx natural join patient_rx_filled natural join price_listing
natural join drug
where doctor_id='DR33313'
order by doctor_id;
```

A popular pharmacy has declared bankruptcy, making a list of all the patients of that pharmacy and a list of alternative pharmacies in the same zip code.

We accomplish this task with two separate queries, first for the patients of the specific pharmacy we desire, and the second for any new pharmacies in the same zip code.

```
select patient.patient_id, patient.name, pharmacy.name, patient.zip,
pharmacy.zip, price_listing.pharmacy_id
from pharmacy natural join price_listing natural join patient_rx_filled
natural join patient_rx join patient on patient.zip = pharmacy.zip
where pharmacy.zip = "90210"
and
pharmacy_id in (select pharmacy_id from price_listing natural join
patient_rx_filled);
```

A drug has been recalled due to a manufacturing problem, creating a list of patients and their doctors that had their script filled in the last year.

– Using the patient relationship table along with the patient rx, patient rx filled and price listing table we can gather the drug name, time filled as well as patient and doctor ids, along with filtering the data to specific dates.

```
select patient.name, patient.patient_id, doctor.doctor_id, doctor.name
from patient natural join patient_rx natural join patient_rx_filled natural
join doctor
where date_filled > '2015-06-10'
order by patient_id;
```

The new regional manager of the pharmacy you work for has decided not to renew any existing contracts to negotiate for better terms. Make a list of all the contract ids for pharmacies in the 90210 zip code that have an end date in 2022.

> – The data we desire is stored in the contracts and pharmacy tables, we join them together and filter by zip and date.

```
select * from contracts natural join pharmacy
where enddate > '2022-01-01' and zip = '90210'
order by pharmacy_id;
```

Some limitations faced

However, Due to limited funding and staff we could only implement 4 of the requested features, with additional budgets we can complete this project. The milestones we met for this project include.

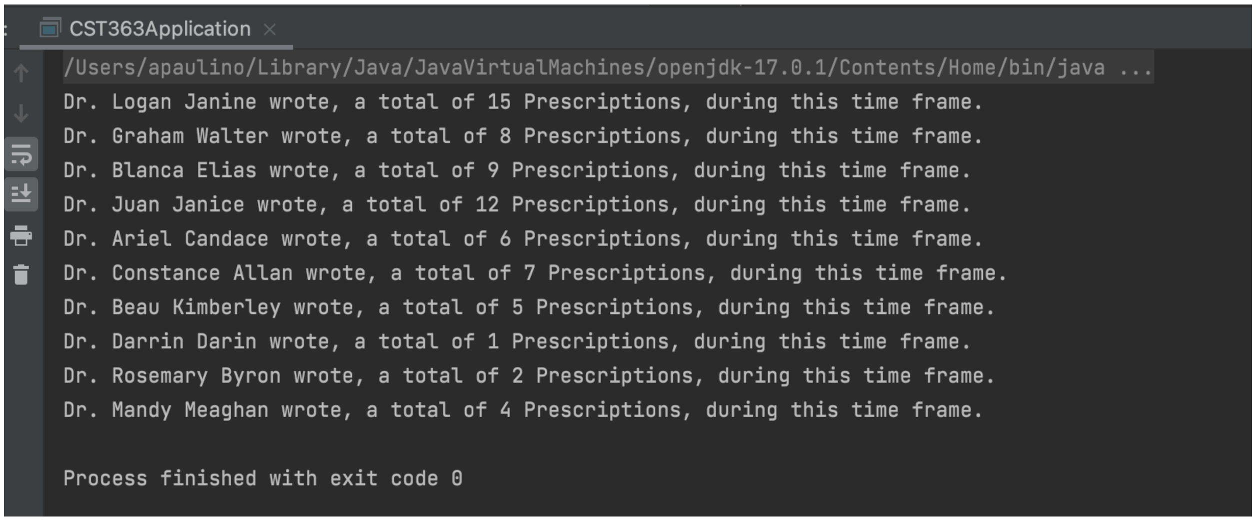
Programs and Applications

Scripting tools and web applications offered…

FDA Reporting tool –

We can utilize our high processing database schema to enable us to search against our prescription data to give accurate time frames of which drugs were sold by which user.

With this you can use this data you can read and send this data officially to the FDA.

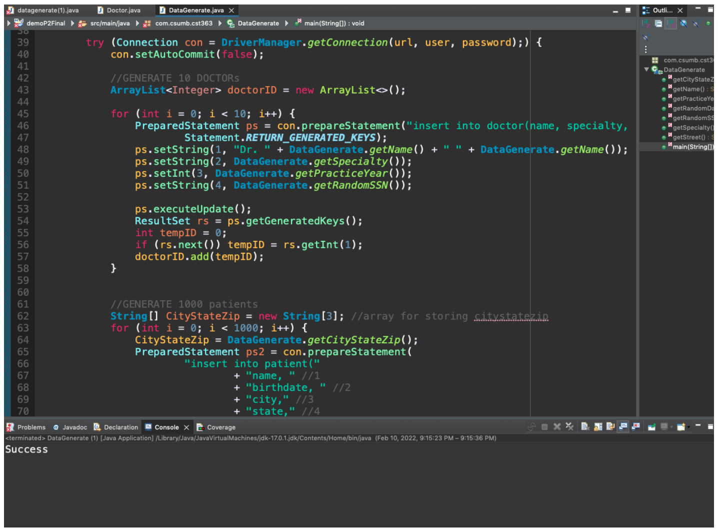


```
/Users/apaulino/Library/Java/JavaVirtualMachines/openjdk-17.0.1/Contents/Home/bin/java ...
Dr. Logan Janine wrote, a total of 15 Prescriptions, during this time frame.
Dr. Graham Walter wrote, a total of 8 Prescriptions, during this time frame.
Dr. Blanca Elias wrote, a total of 9 Prescriptions, during this time frame.
Dr. Juan Janice wrote, a total of 12 Prescriptions, during this time frame.
Dr. Ariel Candace wrote, a total of 6 Prescriptions, during this time frame.
Dr. Constance Allan wrote, a total of 7 Prescriptions, during this time frame.
Dr. Beau Kimberley wrote, a total of 5 Prescriptions, during this time frame.
Dr. Darrin Darin wrote, a total of 1 Prescriptions, during this time frame.
Dr. Rosemary Byron wrote, a total of 2 Prescriptions, during this time frame.
Dr. Mandy Meaghan wrote, a total of 4 Prescriptions, during this time frame.

Process finished with exit code 0
```

Data Generating –

To test our prototype we developed an application that would generate valid values to enter into the database. We can generate 10 doctors, 1000 patients, and 5000 prescriptions to test your data against ours. This will improve the development speed to get acquainted to the database.



```java
        try (Connection con = DriverManager.getConnection(url, user, password);) {
            con.setAutoCommit(false);

            //GENERATE 10 DOCTORs
            ArrayList<Integer> doctorID = new ArrayList<>();

            for (int i = 0; i < 10; i++) {
                PreparedStatement ps = con.prepareStatement("insert into doctor(name, specialty,
                        Statement.RETURN_GENERATED_KEYS);
                ps.setString(1, "Dr. " + DataGenerate.getName() + " " + DataGenerate.getName());
                ps.setString(2, DataGenerate.getSpecialty());
                ps.setInt(3, DataGenerate.getPracticeYear());
                ps.setString(4, DataGenerate.getRandomSSN());

                ps.executeUpdate();
                ResultSet rs = ps.getGeneratedKeys();
                int tempID = 0;
                if (rs.next()) tempID = rs.getInt(1);
                doctorID.add(tempID);
            }


            //GENERATE 1000 patients
            String[] CityStateZip = new String[3]; //array for storing citystatezip
            for (int i = 0; i < 1000; i++) {
                CityStateZip = DataGenerate.getCityStateZip();
                PreparedStatement ps2 = con.prepareStatement(
                        "insert into patient("
                                + "name, " //1
                                + "birthdate, " //2
                                + "city," //3
                                + "state," //4
```

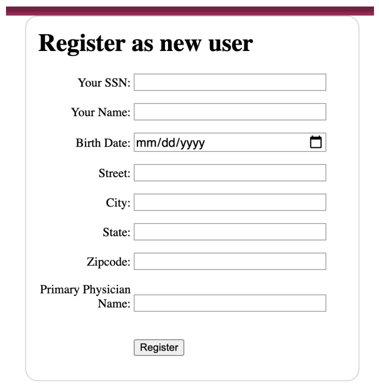
```
<terminated> DataGenerate (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.0.1.jdk/Contents/Home/bin/java  (Feb 10, 2022, 9:15:23 PM – 9:15:36 PM)
Success
```

For the main application of adding patients and editing patient data, we included a video here

Use this URL to access this demo – https://www.youtube.com/watch?v=LJrinaglPd8

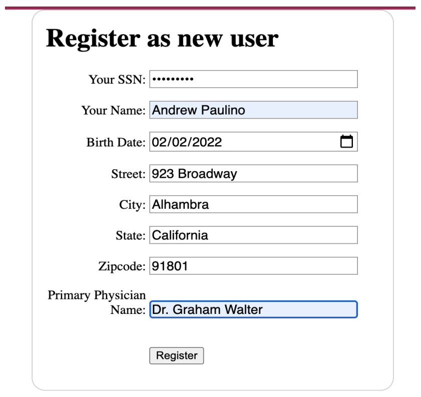
Below we have some screenshots of the application being used

For creating a new patient flow…



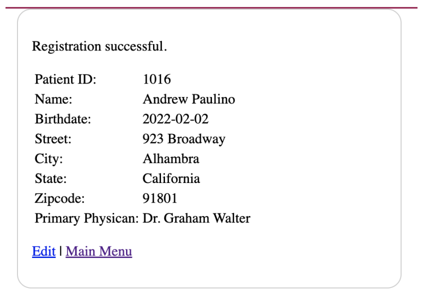
Register as new user

Your SSN:	
Your Name:	
Birth Date:	mm/dd/yyyy
Street:	
City:	
State:	
Zipcode:	
Primary Physician Name:	



Register as new user

Your SSN:	•••••••••
Your Name:	Andrew Paulino
Birth Date:	02/02/2022
Street:	923 Broadway
City:	Alhambra
State:	California
Zipcode:	91801
Primary Physician Name:	Dr. Graham Walter



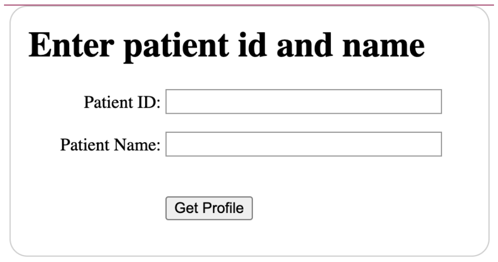
Registration successful.

Patient ID:	1016
Name:	Andrew Paulino
Birthdate:	2022-02-02
Street:	923 Broadway
City:	Alhambra
State:	California
Zipcode:	91801
Primary Physican:	Dr. Graham Walter

Some important notes on using this feature…

- We perform validation against most of our input fields specifically on zip code and birthdate
- Make sure to update development files with the correct password.
- Make sure to pick a doctor that specializes in pediatrics if you are under 18.
- Social Security Number should be formatted as so
  - XXXXXXXXX not ~~XXX-XX-XXXX~~

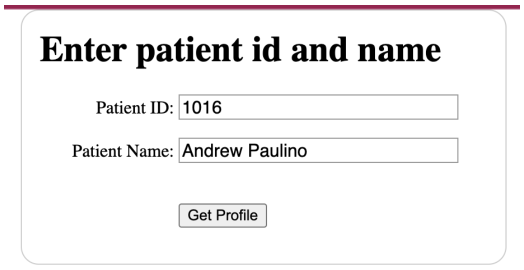
Updating a new patient flow…



Enter patient id and name

Patient ID:

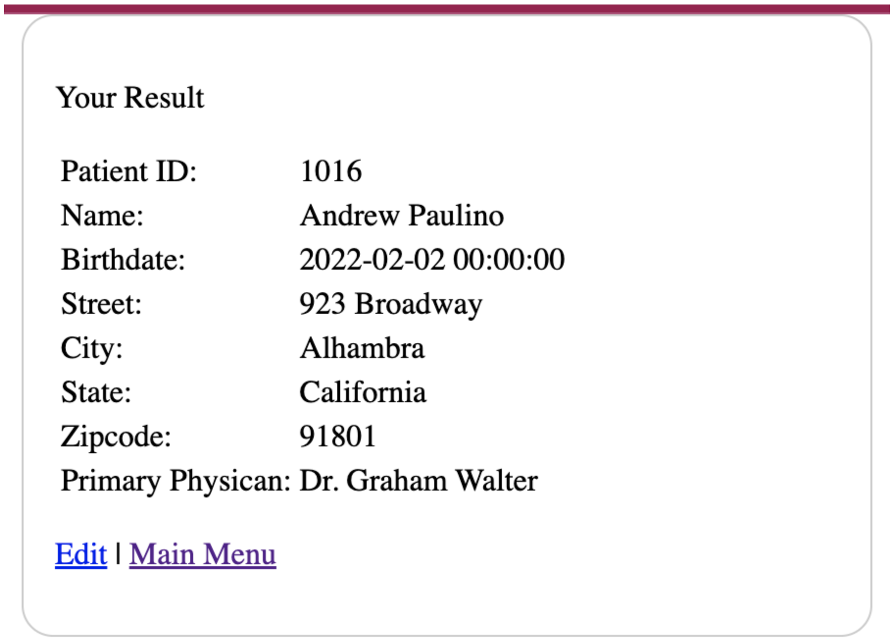
Patient Name:



Enter patient id and name

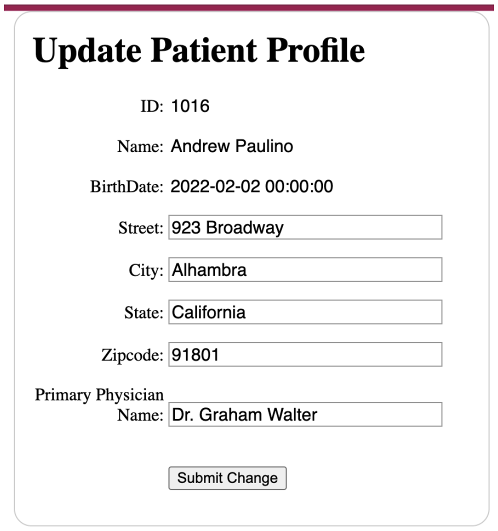
Patient ID: 1016

Patient Name: Andrew Paulino



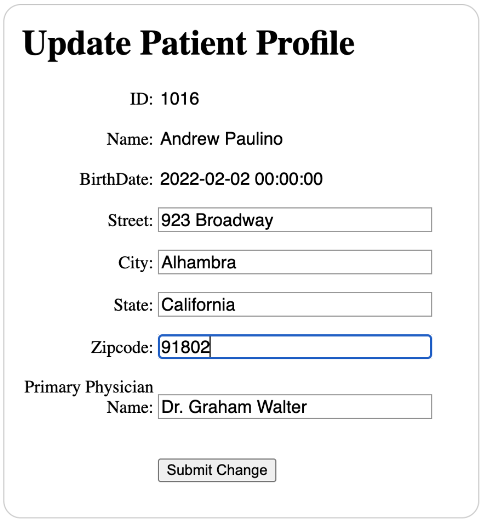
Your Result

Patient ID:	1016
Name:	Andrew Paulino
Birthdate:	2022-02-02 00:00:00
Street:	923 Broadway
City:	Alhambra
State:	California
Zipcode:	91801
Primary Physican:	Dr. Graham Walter



Update Patient Profile

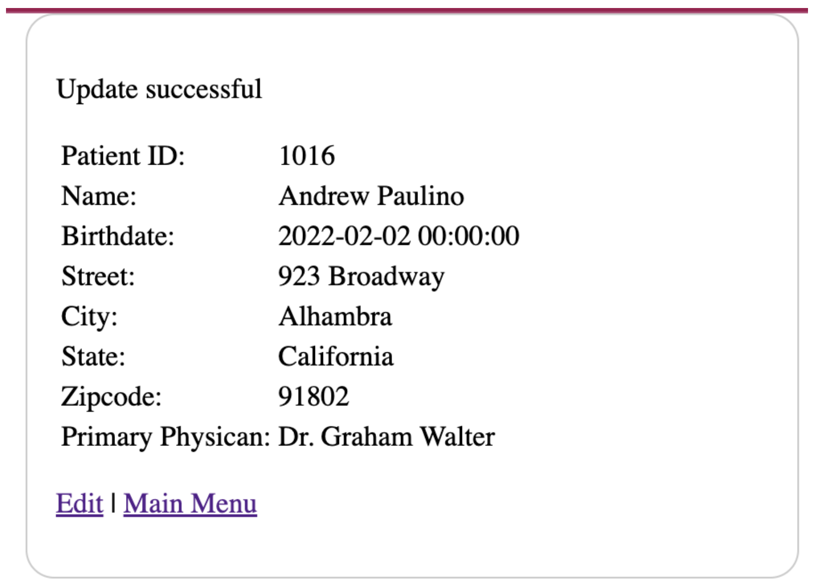
ID:	1016
Name:	Andrew Paulino
BirthDate:	2022-02-02 00:00:00
Street:	923 Broadway
City:	Alhambra
State:	California
Zipcode:	91801
Primary Physician Name:	Dr. Graham Walter



Update Patient Profile

ID:	1016
Name:	Andrew Paulino
BirthDate:	2022-02-02 00:00:00
Street:	923 Broadway
City:	Alhambra
State:	California
Zipcode:	91802
Primary Physician Name:	Dr. Graham Walter

The results after the update…



Update successful

Patient ID:	1016
Name:	Andrew Paulino
Birthdate:	2022-02-02 00:00:00
Street:	923 Broadway
City:	Alhambra
State:	California
Zipcode:	91802
Primary Physican:	Dr. Graham Walter

Edit | Main Menu

Reflection

We learned a lot using and designing the schema using the mySQL workbench, and found a lot of cool ways to really scale up data in an efficient way. I believe that the biggest open concern is using these items in practice and really trying to find optimizations between tables and keys to make good patterns of accessing data. Some applicable questions to ask the client would be how they intend to use the data on a consistent basis, so that views can be established so multiple joins aren’t necessary.