Data Design for Drug Store Chain

Humberto Plaza
Fernando Madrigal
Daniel Gopar

1. Introduction

The goal of our data design project is to construct a database that represents the the entity and relationship sets of a drug store chain. In order for our team to be able to construct such a database, our team had to first develop an understanding of how a drug store chain operates. By conducting some research, and determining probabilities, our team was able to pinpoint the types of data needed to be stored in the database. Therefore, building a data design project that models a drug store chain.

Since the drug store chain is growing, a relational database design was created in order to store and retrieve information more efficiently within the chain itself. Some of the data included in the database can retrieved data such as: prescriptions prescribed by a doctor to a patient(s), the contracts amongst pharmaceutical companies and pharmacies, doctors, patients, drugs, pharmacies, and pharmaceutical companies within the drug store chain. Our report provides a summary of the requirements; such as the type of data being store, a conceptual design(ER diagram) that illustrates the relationships among entities, a database's schema written in SQL that represents the structure of each entity, and sample queries and outputs from SQL that will demonstrate the efficiency of our design.

2. Requirements

Here we list a summary of the data our team determined would help build the structure of our database implementation. Some of the data we took into consideration includes the following:

- Each patient is uniquely identified by an ID field, and each patient has a full name(first and last name), along with age and address. Every patient has a primary doctor.
- Each doctor is also uniquely identified by an ID field. Each doctor also has a first and last name field. Each doctor has a field that represents the years of experience, and a field that states the doctor's speciality (e.g., Surgeon). Every doctor has at least one patient

- A pharmaceutical company is identified by an ID. Each pharmaceutical company has a name, address and phone number associated with it. Pharmaceutical companies makes many drugs. Also, pharmaceutical companies can contract with many pharmacies.
- Pharmacies are identified by their ID field. Each pharmacy also has a name, address and phone number associated with it. Different pharmacies can sell different drugs at different prices. Pharmacies can contract with many pharmaceutical companies.
- Drugs are identified by their ID. Each drug has a name and a formula associated with it.

3. Conceptual Design/Design Issues

We first began developing a conceptual design using an Entity-Relationship model. The ER diagram below demonstrates the identities and entities of the relationships within the pharmacies, drugs, patients, and doctors. We will explain the main features in our ER diagram and show how they relate to the requirements from part two.

Our team came up with 5 different entities that include:

1. Patient
2. Doctor
3. Pharmaceutical Company
4. Pharmacy
5. Drug

As represented in Figure 1 below, we have the "Patient" entity set that indicates that the ID is the primary key. The other attributes in the entity set shows the patient's name(first and last name), age, and their respective address. Another entity set that we added is the "Doctor" entity set, which we set the doctor's ID as the primary key and the rest of the attributes indicates the Doctor's full name (first and last name), years of experience, and the field that the doctor specializes in. We decided that the relationship set between the two entities set would be called the primary doctor relationship. We decided that the mapping cardinality between the patient and the doctor is going to be many-to-one. Represented in Figure 1, a line with a right arrow pointing towards the Doctor entity signifies that every patient has at least one primary doctor.

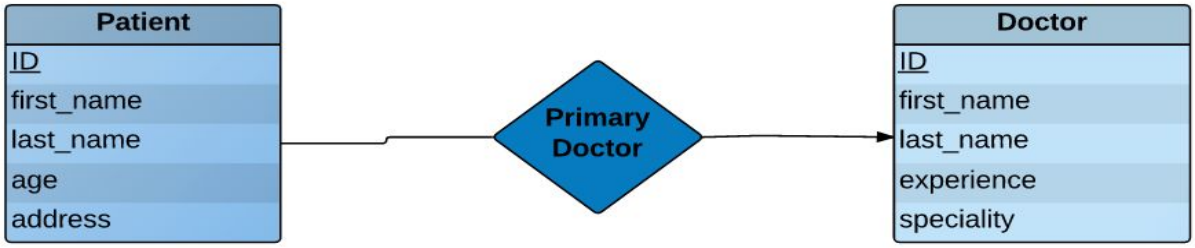


Figure 1.

As represented in Figure 2 below, we have Pharmacy entity that indicates that the ID is the primary key. The reason that we decided to use an ID as the primary key for the "Pharmacy" entity is to easily distinguish an instance of a pharmacy within a chain of pharmacies with the same name. The rest of the attributes in the Pharmacy entity shows the name, address, and the phone number of the patients in the Pharmacy database. The next entity that we added in our ER diagram is the "Drug", which we set the ID as the primary key. The rest of the attributes of the "Drug" entity shows the name of the drug and the formula. What we decided that our relationship set of our two entities is "Makes". We decided that the relation between the two entities "Pharmacy" and "Drug" . Our mapping cardinality represents many-to-many which shows in Figure 2. A partial participation from "Pharmacy" to makes and a total participation from makes to "Drug" entity, concluding that many pharmacies makes many drugs.

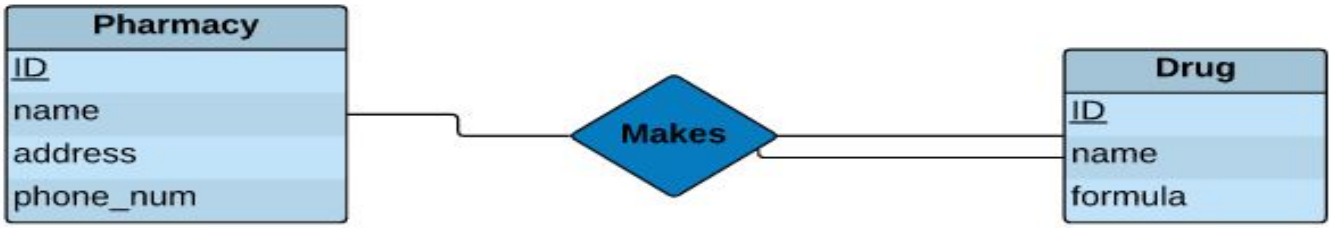


Figure 2.

In Figure 3 below, we gathered the entity sets "Doctor", "Patient", and "Drug" to conclude a relationship between the three of them. In a real world scenario, a doctor prescribes certain drugs to certain patients. To model the relationship set between a doctor, patient, and drug we used a relationship set called "Prescription". This

relationship set has attributes such as the date, that represents the date the prescription was prescribed, and the quantity that was prescribed.

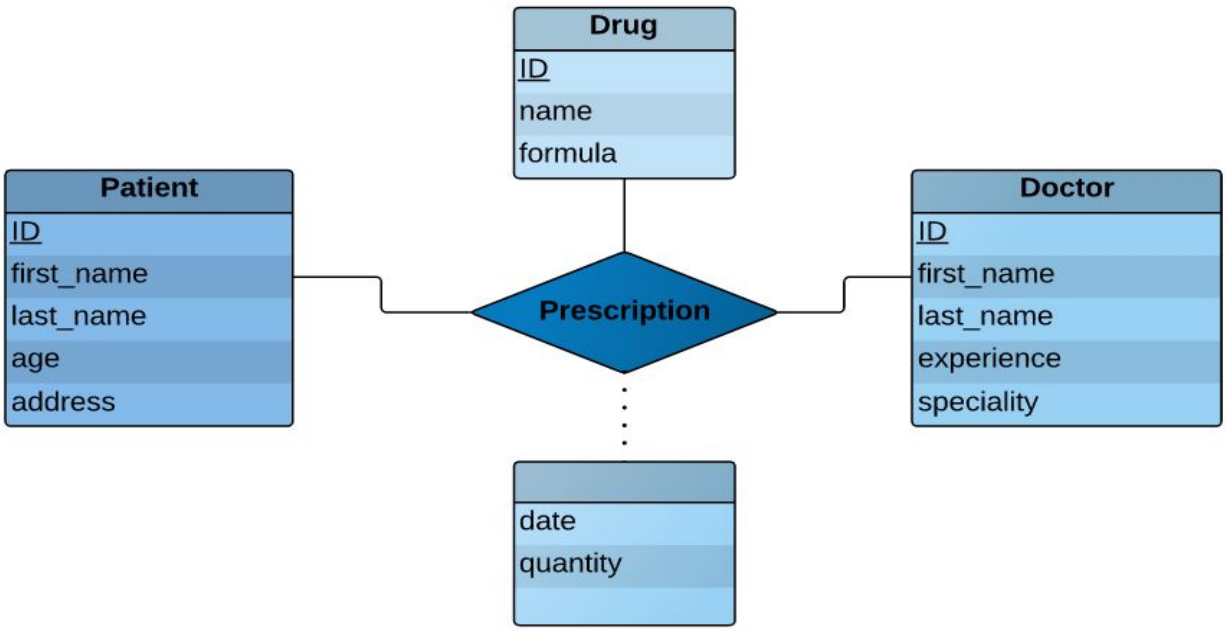


Figure 3.

As represented in Figure 4 below, we have 2 entities sets which are "Pharmacy" and "Drug" that share a relationship. We decided that the relationship set for the two entities would be called "Sells" which demonstrate that different can pharmacies sell different drugs. The relationship set includes the attribute "price" to indicate the price in which the pharmacy sold it at. We decided that the mapping cardinality between the "Pharmacy" and "Drug" entities is many-to-many, which is illustrated in Figure 4 below, a single line from "Pharmacy" to "Drug".

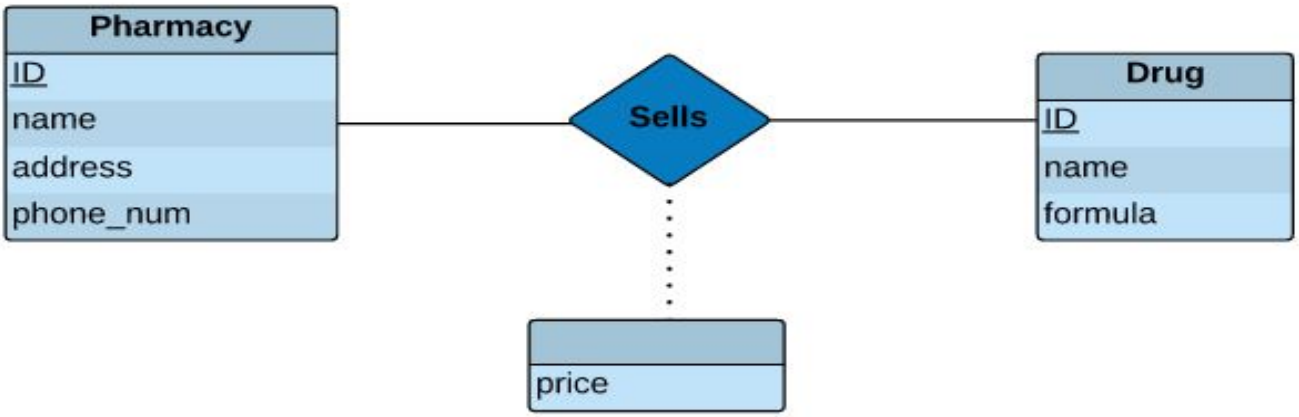


Figure 4.

Figure 5 below demonstrates the relationship between Pharmacy and Pharmaceutical Company entities. The relation between the two entities is represented as a "Contract". The relationship set has a start_date, end_date, text, and manager attributes associated with it. The mapping cardinality is said to be many-to-many because pharmacies can have contracts with many different pharmaceutical companies and pharmaceutical companies can also have contracts with many different pharmacies.

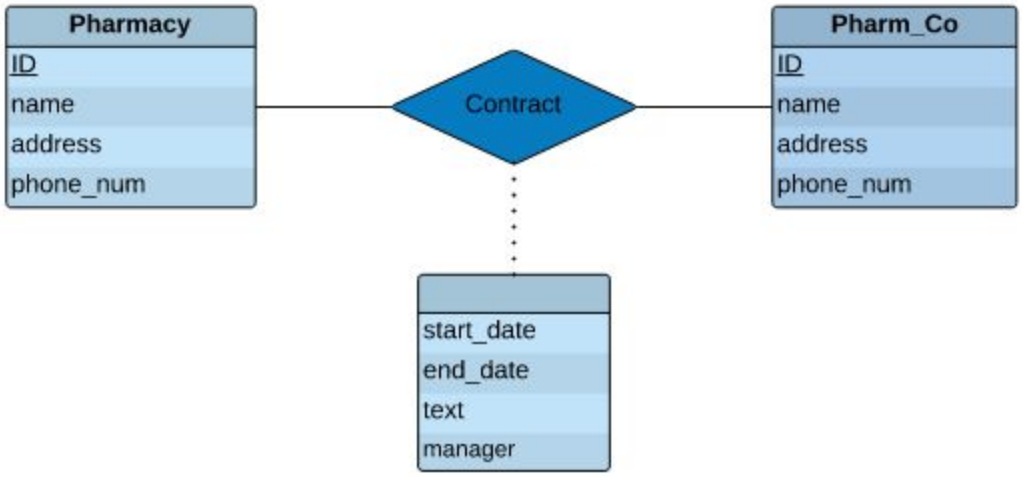
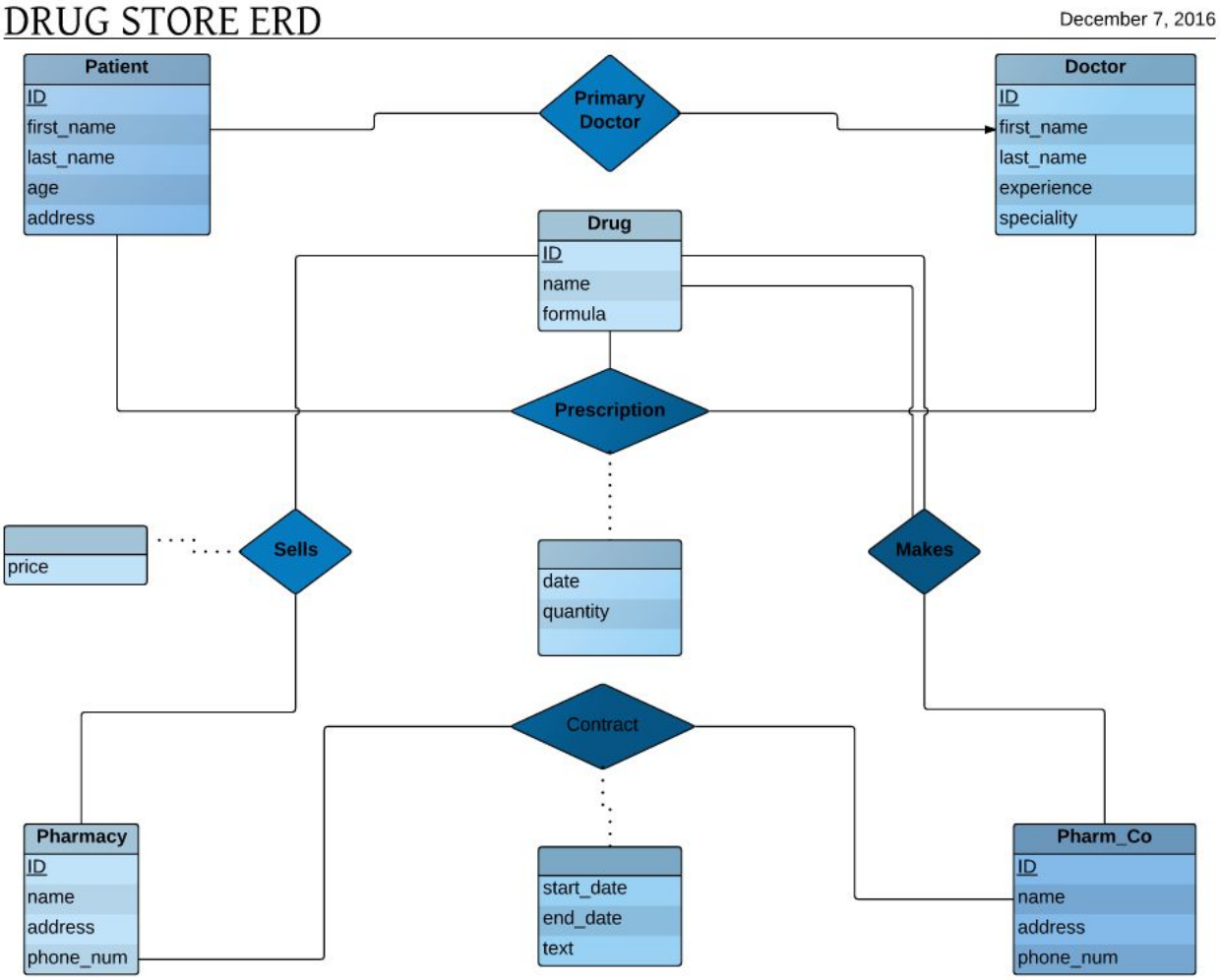


Figure 5.

Below is the complete ER diagram:



Complete ER diagram

The parts of the conceptual model that couldn't be put in the ER diagram are the Social Security Numbers (SSN) attributes for the doctor and patient entities. Our team felt that a SSN should not be data that should be seen by everyone. A design issue that our team had was using the attribute "name" in the entity set "Pharmacy" as a primary key. Since pharmacies can change their name, having the "name" attribute as a primary key can cause overhead by having to physically change the name of that pharmacy wherever an instance of that respective pharmacy exists. Our team resolved this by using an ID as a primary key instead. This allows the pharmacy to change its name without having to maintain the database.

4. Database Schema

Beginning with on our conceptual design, our team was able to develop and construct a Entity-Relational diagram. With our ER diagram completed, the next step for our team was to derive a relational schema from the it.

patient(ID, first_name, last_name, age, address)
doctor(ID, first_name, last_name, experience, specialty)
pharm_co(ID, name, address, phome_num)
pharmacy(ID, name, address, phome_num)
drug(ID, name, formula)
prescriptions(doc_id, patient_id, drug_id)
    doc_id -> Foreign key to Doctor table
    patient_id -> Foreign key to Patient table
    Drug_id -> Foreign key to Drug table
contract(pharm_co_id, pharm_id, start_date, end_date, text)
    pharm_co_Id -> Foreign key to Pharm_co table
    Pharm_id ->Foreign key to pharmacy table
sells(pharm_id, drug_id, price)
    Pharm_id -> Foreign key to pharmacy table
    Drug_id -> Foreign key to drug table
primary_doctor (doc_id, patient_id)
    doc_id -> Foreign key to Doctor table
    patient_id -> Foreign key to Patient table
makes (pharm_co_id, drug_id)
    pharm_co_Id -> Foreign key to Pharm_co table
    Drug_id -> Foreign key to drug table

The database schema can be written in the SQL language as follows:

```
create table patient   (
        Id              integer not null primary key,
        first_name      varchar(20) not null,
        last_name       varchar(20) not null,
        age             integer not null check(age > -1), -- Can be a baby
        address         varchar(100)
);

create table doctor  (
```

```
        id                      integer not null primary key,
        first_name      varchar(20) not null,
        last_name       varchar(20) not null,
        experience      integer not null check(experience > -1), -- Can have 0 years
        experience
        speciality      varchar(30)
);

create table pharm_co( --Company that's producing the meds
        id                       integer not null primary key,
        name            varchar(20) not null,
        address         varchar(100) not null,phone_num         varchar(10) not null
);

create table pharmacy( -- Place where it's being sold
        id                      integer  not null primary key,
        name            varchar(20) not null,
        address         varchar(100) not null,
        phone_num       varchar(10) not null

);

create table drug     (
        id              integer not null primary key,
        name            varchar(30),
        formula         varchar(30)
);

create table prescription(
        id                      integer not null primary key,
        date            varchar(19) not null, -- format 'yyyy-MM-dd HH:mm:ss'
        quantity        integer not null,
        doc_id          integer not null,
        patient_id      integer not null,
        drug_id         integer not null,
        foreign key (doc_id)            REFERENCES doctor(id) on delete set null,
        foreign key (patient_id) REFERENCES patient(id) on delete set null,
        foreign key (drug_id)       REFERENCES drug(id) on delete set null
);
```

```
create table contract        (
        id                      integer not null primary key,
        pharm_co_id  integer not null,
        pharm_id      integer not null,
        start_date     varchar(19) not null, -- format 'yyyy-MM-dd HH:mm:ss'
        end_date       varchar(19) not null, -- format 'yyyy-MM-dd HH:mm:ss'
        text             blob not null,
        foreign key (pharm_co_id) REFERENCES pharm_co on delete set null,
        foreign key (pharm_id)       REFERENCES pharmacy on delete set null
);

create table sells     (
        id                      integer not null primary key,
        pharm_id      integer not null,
        drug_id        integer not null,
        price           real not null,
        foreign key (pharm_id) REFERENCES pharm_co(id) on delete set null,
        foreign key (drug_id)     REFERENCES drug(id) on delete set null
);

create table primary_doctor        (
        id                      integer not null primary key,
        doc_id         integer not null,
        patient_id     integer not null,
        foreign key (doc_id) REFERENCES doctor(id) on delete set null,
        foreign key (patient_id) REFERENCES patient(id) on delete set null
);

create table makes  (
        id               integer not null primary key,
        pharm_co_id  integer not null,
        drug_id        integer not null,
        foreign key (pharm_co_id) REFERENCES pharm_co(id) on delete set null,
        foreign key (drug_id) REFERENCES drug(id) on delete set null
);
```

On the following database schema, there is a good amount of ‘on delete set null’, the reasoning behind this is that most of these tables would still be valuable even if

some entries are still missing. For example, it would be good to know what drugs a patient has been prescribed to even if the doctor who prescribed them no longer works there or something like that. Another thing that can be found in the schema is there are 2 checks. One of the them is the patient age must be greater than -1, the reason for this is that a baby can be given medication and it doesn't have to be at least 1 year old. The other check is the experience field in the doctor table. No doctor can have negative experience, it doesn't make sense. Another thing to mention is that the phone number fields are 10 characters long, we only include the numbers and the area code, with this we are assuming that all the numbers are from the U.S and follow the traditional format. Last thing to mention is that Date fields all go in the format of ISO 8601, which is the following: 'yyyy-MM-dd HH:mm:ss' since sqlite does not have native date data types.

5. Database Queries

From the database schema that we have illustrated above, we can run certain queries that would answer questions. Here is a couple of questions that could be helpful when having a schema like the one shown above

- What are all the drugs a patient is taking? This is relevant so that a doctor can find out if there any drugs that might conflict with one another before prescribing another one.
- How many prescriptions have all the doctors given? This could be helpful in the way of checking how often each doctor gives prescription.
- What patient has the highest amount of prescriptions? This can be helpful in figuring out what patient are already taking too many prescriptions and should not be given more.
- What pharmacy has sold the highest amount of drugs in terms of revenue? This can be useful for pharmaceutical companies to see which pharmacies they should prioritize in terms of keeping them a happy customer
- What pharmacies are not currently in any contracts? If no contract is being done with a certain pharmacy then a pharmaceutical company can reach out and try to make a deal, making more revenue.

Taking these questions, we can answer them with the following queries:

- What are all the drugs a patent is taking?

```
SELECT name FROM drug WHERE
  id in (SELECT DISTINCT(drug_id) FROM patient AS P, prescription AS Pres ON
```

P.ID = Pres.patient_id WHERE first_name='Daniel');

- How many prescriptions has a certain doctor given?

```
SELECT last_name, COUNT(*) FROM prescription, doctor ON
doctor.id=prescription.doc_id GROUP BY doctor.id;
```

- What patient has the highest amount of prescriptions?

```
SELECT first_name, max(tot) FROM
  (SELECT first_name, COUNT(*) as tot FROM prescription, patient ON
patient.id=prescription.patient_id GROUP BY patient.id);
```

- What pharmacy has sold the highest amount of drugs in terms of revenue?

```
SELECT pharm_id, MAX(tot) FROM (SELECT pharm_id, SUM(price * quantity) AS tot
FROM sells, prescription ON sells.drug_id=prescription.drug_id GROUP BY pharm_id);
```

- What pharmacy are not currently in any contracts

```
SELECT id FROM pharmacy WHERE id NOT IN (SELECT DISTINCT(pharm_id) FROM
contracts)
```

6. Conclusion(Fernando)

We presented our data design for the Drug Store Chain. We focused on constructing a database for a drug store chain to keep track of data such as patients, doctors, the drugs sold by pharmacies, and contracts associated with the pharmaceutical companies. We shown the procedure to make the database possible by building a relation between the five entities. We displayed an ER diagram of the drug store chain to have a clear picture of the relations of the entities presented in the ER diagram. We derived a relational database schema, sample data and sample SQL queries to make sure that our SQL executes perfectly and illustrates that the conceptual design is accurate.