Project 2 Part 1

4/25/2020

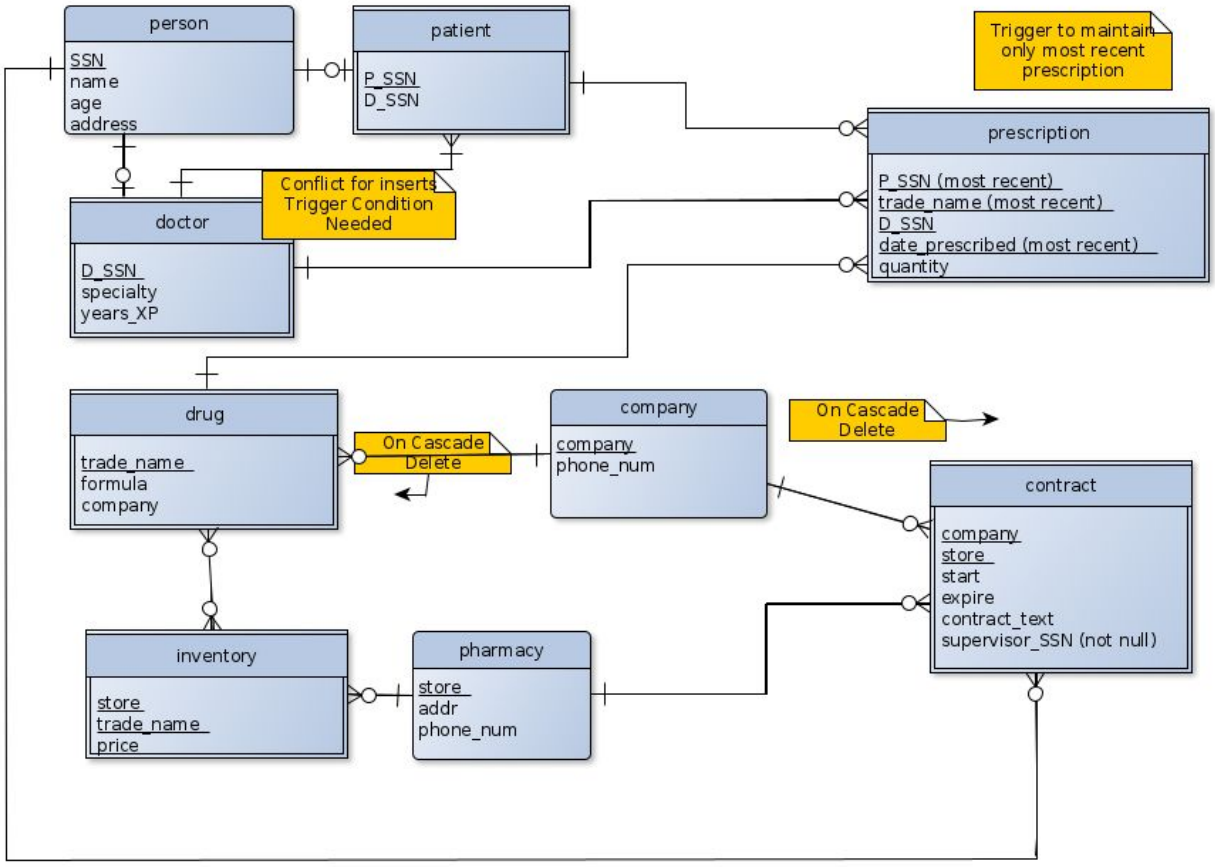
Timothy Hanneman. Federico Rubino

Intro

The goal of this project was to develop a relational database that would function for a drug store chain. The requirements that were presented included the need for interacting with multiple doctors, patients, drug companies, and pharmacies. The database needs to be flexible enough to scale as the drug store chain opens new locations and interacts with doctors and patients that may change over time. The contracts with pharmaceutical companies also require some flexibility as contracts and contract supervisors change. The goal of the model design was specifically for use in a relational database, in particular H2. We hereby present our consultation result to our supervisor the professor & the T.A. for review.

While constructing the ER model we had some major findings. What was the most obvious was the fact that every object in the database was able to be distilled into smaller chunks. For example, creating a person table so as to supply the patient class, the doctor class, and the supervisor field with a relation that they all had in common. These relationships allow us to streamline and normalize the database efficiently. Another finding that we had was the appearance of constraints/trigger statements that were important for the performance of the database. Statements such as “On Cascade Delete” are important to inform the database to delete all occurrences of a table object if the parent dataset has been deleted.

ER Model



The Entity-Relationship model was constructed by going through the information provided. Each statement was divided out into the smallest functional parts. This revealed that there were three primary entities on which everything else would depend: pharmacies, drug companies, and a person table. The person table was abstracted from the need for three types of people: doctors, patients, and contract supervisors. All other entity relationships would have some dependence on the principle tables. The conceptual model is able to capture the major database requirements. It does not include a full listing of all the constraints and data types. Further information on the exact restraints is provided below in the schema model.

Several design issues arose while creating the model. This can be seen by the yellow notes in the model above. One issue was a contradiction between the creation of doctor and patient records. We aren't quite sure how this issue will be managed yet, but believe it can be resolved through a trigger. Another issue is only keeping track of the most recent prescriptions, this will be able to be handled with a trigger action on the creation of a new prescription. The last issue that arose was the need for cascading deletes on the dissolution of pharmaceutical companies.

Relational Schema

The relational schema consists of the “create table” SQL statements that are required to create all of the tables that are needed to follow suit with the ER model. Each line is either composed of the create table command followed by the name of the table or a declaration of the column names, such as SSN in the person table. Every table has a declared primary key and every column has a declared type.

```
create table person(
  SSN integer PRIMARY KEY,
  name varchar(64) NOT NULL UNIQUE,
  age integer,
  address varchar(255));

create table company(
  company varchar(255) PRIMARY KEY,
  phone_num varchar(13) NOT NULL);

create table pharmacy(
  store varchar(255) PRIMARY KEY,
  addr varchar(255) NOT NULL,
  phone_num varchar(13));

create table doctor(
  D_SSN integer PRIMARY KEY,
  specialty varchar(64) NOT NULL,
  years_XP integer);

create table patient(
  P_SSN integer,
  D_SSN integer,
  PRIMARY KEY (P_SSN, D_SSN),
  FOREIGN KEY (P_SSN) REFERENCES person(SSN)ON DELETE CASCADE,
  FOREIGN KEY (D_SSN) REFERENCES doctor(D_SSN)ON DELETE CASCADE);

create table drug(
  trade_name varchar(64) PRIMARY KEY,
  formula varchar(255) NOT NULL,
  company varchar(255) NOT NULL,
  FOREIGN KEY (company) REFERENCES company(company)ON DELETE CASCADE);

create table prescription(
  P_SSN integer,
  trade_name varchar(64),
```

```
  D_SSN integer,
  date_prescribed date,
  quantity numeric(8,4) NOT NULL,
  PRIMARY KEY (P_SSN, trade_name, D_SSN, date_prescribed),
  FOREIGN KEY (P_SSN) REFERENCES patient(P_SSN),
  FOREIGN KEY (trade_name) REFERENCES drug(trade_name),
  FOREIGN KEY (D_SSN) REFERENCES doctor(D_SSN));

create table inventory(
  store varchar(255),
  trade_name varchar(64),
  price numeric(10,2) NOT NULL,
  PRIMARY KEY (store, trade_name),
  FOREIGN KEY (store) REFERENCES pharmacy(store),
  FOREIGN KEY (trade_name) REFERENCES drug(trade_name));

create table contract(
  company varchar(255),
  store varchar(255),
  start date NOT NULL,
  expire date NOT NULL,
  contract_text varchar(255) NOT NULL,
  supervisor_SSN integer,
  PRIMARY KEY (company, store),
  FOREIGN KEY (company) REFERENCES company(company),
  FOREIGN KEY (store) REFERENCES pharmacy(store),
  FOREIGN KEY (supervisor_SSN) REFERENCES person(SSN));
```

Constraints and functional dependencies:

- When a company is deleted -> delete all contracts that are related
- When a company is deleted -> delete all drugs that are related
- When a doctor prescribes a new prescription -> maintain the newest entry

```
CREATE TRIGGER one_prescription AFTER INSERT ON prescription
REFERENCING NEW ROW AS nrow
FOR EACH ROW
WHEN(nrow.trade_name like prescription.trade_name)
BEGIN
DELETE FROM prescription WHERE nrow.trade_name like prescription.trade_name
END;
```

- Trigger to maintain One to One relationship between doctors and patients in conjunction with schema constraints. This trigger must run after each inserted row to function correctly.

```
Delete FROM doctor WHERE doctor.D_SSN =
        (select D_SSN from
                (SELECT COUNT(patient.P_SSN) AS cnt , doctor.D_SSN
                FROM doctor LEFT OUTER JOIN patient
                GROUP BY doctor.D_SSN)
        where CNT < 1)
```

Normalization

The data is already normalized. The tables were created to avoid having multiple dependencies duplicating data. Where data is relied on in other tables will require either cascading delete or updating positions before removal from the database. These are shown in the section above.

SQL Queries

1. How many contracts does a company have?

```
Select Count(*) from company where company  like 'Gilead Sciences';
```

COUNT(*)
1

(1 row, 2 ms)

2. What are the average, min, and max number of prescriptions by person?

```
select avg(cnt) from (Select Count(D_SSN) as cnt from prescription group by P_SSN);
```

AVG(CNT)
2

(1 row, 1 ms)

```
select max(cnt) from (Select Count(D_SSN) as cnt from prescription group by P_SSN);
```

MAX(CNT)
5

(1 row, 4 ms)

```
select min(cnt) from (Select Count(D_SSN) as cnt from prescription group by P_SSN);
```

MIN(CNT)
1

(1 row, 2 ms)

3. How many patients do each of the doctors have?

```
Select person.name, count(patient.P_SSN) from person join patient where patient.D_SSN = person.SSN group by patient.D_SSN
```

NAME	COUNT(PATIENT.P_SSN)
Gaius Plinius Secundus	4
Galen of Pergamum	1
Al-Zahrawi	2
Anthony Fauci	1
Smith	1

(5 rows, 3 ms)

4. How long do the company contracts last?

```
Select company, datediff(day,  start,expire) as days from contract group by company,expire;
```

COMPANY	DAYS
Acura Pharmaceuticals US Inc.	3653
Generic	731
Generic	517
Gilead Sciences	2557
Merck	1643
Nicolas Flamel	226451
biotext	4018

(7 rows, 2 ms)

5. What is the largest number of prescriptions for any given drug?

```
select max(cnt) as CNT from (select count(trade_name) as cnt, trade_name from prescription group by trade_name);
```

CNT
5

(1 row, 3 ms)

Conclusion

We have constructed a database that can be used for the record keeping of a store drug management system. It includes an entity relationship model, with a detailed SQL schema containing the requisite restraints. There is code provided to implement trigger constraints within the database.