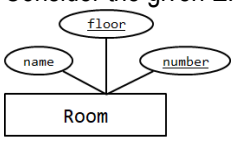
CS2102 Database Systems

Semester 1 2019/2020

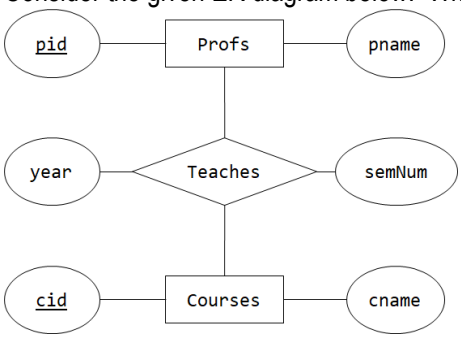
Tutorial 04 (*Selected Answers*)

Quiz

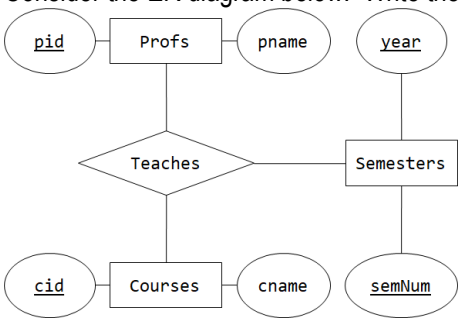
1. Consider the given ER diagram below. Write the SQL code to create the schema.



2. Consider the given ER diagram below. Write the SQL code to create the schema.



3. Consider the ER diagram below. Write the SQL code to create the schema.



4. You are creating a simple search engine. Your search engine works on a following schema `R(phrase, result)`. You are tasked to search for all the result that contains the phrase `'base'` anywhere in the phrase including the middle of the phrase.
5. Consider the following two schemas `R(a,b)` and `S(c,d)`. You are to find all the distinct pairs `(a,c)` from `R` and `S` that satisfies a certain condition `cond`. Is the `DISTINCT` keywords necessary to ensure that the result will always be a distinct pairs `(a,c)`? An example SQL query is shown below.

```
SELECT _________ a,c FROM R, S WHERE cond.
```

6. Consider the definition of full outer join. Which of the following SQL query is equivalent to $\pi_{a,b}(R \leftrightarrow_c S)$?
   a) `SELECT a,b FROM R FULL JOIN S ON c;`
   b) `SELECT a,b FROM R LEFT JOIN S ON c UNION SELECT a,b FROM R RIGHT JOIN S ON c;`
   c) `SELECT a,b FROM R RIGHT JOIN S ON c UNION SELECT a,b FROM R LEFT JOIN S ON c;`
   d) `SELECT a,b FROM ( R LEFT JOIN S ON c ) RIGHT JOIN S ON c;`
   e) `SELECT a,b FROM R LEFT JOIN ( S RIGHT JOIN S ON c ) ON c;`
   f) `SELECT a,b FROM R RIGHT JOIN S ON c UNION SELECT a,b FROM R LEFT JOIN S ON c UNION R JOIN S ON c;`
7. You are given a simplified map where a city is represented as a square. The city hall is located right at the center of the city. We say that the city is at coordinate `(0,0)` since your schema is `Map(x_coord, y_coord)`.
   We use the notation `(+,-)` to denote that the `x_coord` is positive value and `y_coord` is negative value. We change the sign symbol correspondingly. We say that a house is in the first quadrant (`Q1`) if its coordinate is in `(+,+)`, second quadrant (`Q2`) if its coordinate is in `(-,+)`, third quadrant (`Q3`) if its coordinate is in `(-,-)`, and the fourth quadrant (`Q4`) if its coordinate is in `(+,-)`. You are guaranteed that there will be no houses on `x_coord = 0` or `y_coord = 0` as they are the two main roads on the city.
   Write SQL query to output the quadrants of each set of coordinates in the `Map`.

Tutorial Questions *[Discussion: 8(a), 8(b), 8(c), 8(d), 9]*

Questions 8-9 uses the following database schema as Tutorial 03 Question 07. They are reproduced here for convenience.

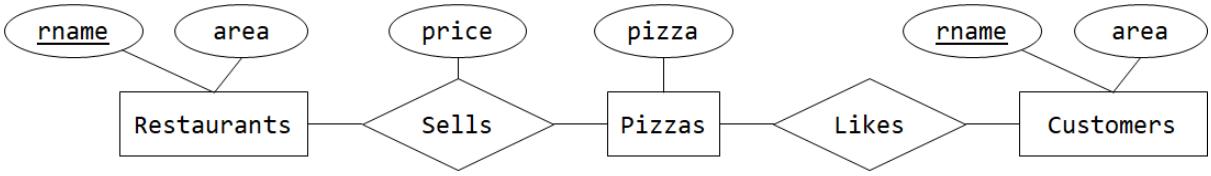
- **Pizzas** (pizza)
- **Customers** (cname, area)
- **Restaurants** (rname, area)
- **Contains** (pizza, ingredient)
- **Sells** (rname, pizza, price)
- **Likes** (cname, pizza)

Pizzas indicates all the pizzas of interest. **Customers** indicates the name and location of each customer. **Restaurants** indicates the name and location of each restaurant. **Contains** indicates the ingredients used in each pizza. **Sells** indicates the pizzas sold by restaurants and their prices. **Likes** indicates the pizzas that customers like.

The following are all the foreign key constraints on the database schema:

- `Contains.pizza` is a foreign key that refers to `Pizzas.pizza`
- `Sells.rname` is a foreign key that refers to `Restaurants.name`
- `Sells.pizza` is a foreign key that refers to `Pizzas.pizza`
- `Likes.cname` is a foreign key that refers to `Customers.pizza`
- `Likes.pizza` is a foreign key that refers to `Pizzas.pizza`

Furthermore, we provide the simplified ER diagram corresponding to the schema above minus `Contains`.



8. Answer each of the following queries using SQL. For parts (a) to (e), remove duplicate records from all query results.
   a) Find all pizzas that 'Alice' likes but 'Bob' does not like.
   b) For each customer, find the pizzas sold by restaurants that are located in the same area as the customer's area. Exclude customers whose associated set of pizzas is empty.
   c) For each customer, find the pizzas sold by restaurants that are located in the same area as the customer's area. Include customers whose associated set of pizzas is empty.
   d) Tutorial 03 Question 07 supposes the existence of relation `Dislikes (cname, pizza)`. The relation indicates the pizzas that customers do not like. The database still also satisfies the following constraint: for every customer $c \in \pi_{\text{cname}}(\text{Customers})$ and for every pizza $p \in \pi_{\text{pizza}}(\text{Contains})$, either $(c, p) \in \text{Likes}$ or $(c, p) \in \text{Dislikes}$ (*in other words, if a customer do not like a pizza, the customer automatically dislike the pizza*). Create the `Dislikes` relation as a `VIEW`.

Solution:

a) SQL Query

```
SELECT pizza FROM Likes WHERE cname = 'Alice'
EXCEPT
SELECT pizza FROM Likes WHERE cname = 'Bob';
```

b) Solution #1:

```
SELECT DISTINCT cname, pizza
FROM Customers NATURAL JOIN Restaurants NATURAL JOIN Sells;
```

Solution #2:

```
SELECT DISTINCT C.cname, S.pizza
FROM ( Customers C INNER JOIN Restaurants R
       ON C.area = R.area )
     INNER JOIN Sells S ON R.rname = S.rname;
```

Solution #3:

```
SELECT DISTINCT C.cname, S.pizza
FROM ( Restaurants R INNER JOIN Sells S
       ON R.rname = S.rname )
     INNER JOIN Customers C ON C.area = R.area;
```

NOTE: the order in which the inner joins are computed does not matter because inner joins are associative operations. Thus, $(C \bowtie_{c_1} R) \bowtie_{c_2} S = C \bowtie_{c_1} (R \bowtie_{c_2} S)$.

c) Solution #1:

```
SELECT DISTINCT cname, pizza
FROM Customers NATURAL LEFT JOIN (Restaurants NATURAL JOIN Sells);
```

Solution #2:

```
SELECT DISTINCT C.cname, S.pizza
FROM   Customers C LEFT JOIN
       ( Restaurants R NATURAL JOIN Sells S)
       ON C.area = R.area;
```

Solution #3:

```
SELECT DISTINCT C.cname, S.pizza
FROM   Customers C LEFT JOIN
       ( Restaurants R INNER JOIN Sells S ON R.rname = S.rname )
       ON C.area = R.area;
```

Wrong Answer #1:

```
SELECT DISTINCT cname, pizza
FROM Customers NATURAL LEFT JOIN Restaurants NATURAL JOIN Sells;
```

The above query is equivalent to the following query

```
SELECT DISTINCT C.cname, S.pizza
FROM   ( Customers C LEFT JOIN Restaurants R ON C.area = R.area )
       INNER JOIN Sells S ON R.rname = S.rname;
```

While the join does not matter in part (b) due to the associativity of inner joins, the join ordering matters when outer joins are involved as outer joins are generally not associative with inner joins. In particulars,

$$(C \rightarrow_{c_1} R) \bowtie_{c_2} S \neq C \rightarrow_{c_1} (R \bowtie_{c_2} S)$$

To see this, consider a customer tuple $c \in \text{Customers}$ that is not co-located with any restaurants. Therefore, c will appear exactly once in the result of $C \rightarrow_{\text{C.area=R.area}} R$ with the null value for attributes `R.rname` and `R.area`. Due to its null value for `R.rname`, this tuple with c will be a dangling tuple w.r.t. the subsequent inner join with S. Thus c will not appear in the result $(C \rightarrow_{\text{C.area=R.area}} R) \bowtie_{\text{R.rname=S.rname}} S$. In constrast, by performing the outer join after the inner join, c will be preserved in the result of $C \rightarrow_{\text{C.area=R.area}} (R \bowtie_{\text{R.rname=S.rname}} S)$.

You should pay attention to the join ordering when using outer joins!

Wrong Answer #2:

```
SELECT DISTINCT C.cname, S.pizza
FROM   Customers C FULL JOIN
       ( Restaurants R FULL JOIN Sells S ON R.rname = S.rname )
       ON C.area = R.area;
```

By using `FULL JOIN` for both joins, this answer will also preserve dangling tuples from `Sells` and `Restaurants` relations. Thus, the query result could have tuples with a `null` value for `cname`. However, such tuples are not required by the question.
To see this, consider a `Sells` tuple $s \in$ Sells where its restaurant is not co-located with any customer. Thus, s will join with exactly one tuple from Restaurants (say $r \in$ Restaurants), but the resultant (s,r) tuple from the first join computation will not join with any tuple from `Customers`. Thus will produce (`null`, s`.pizza`) in the query result, which is incorrect.
Moreover, if there is some `Restaurants` tuple $r \in$ Restaurants where r is not co-located with any customer and r does not sell any pizza, then this will produce (`null`,`null`) in the query result, which is also incorrect.

The type of join matters! You should use appropriate outer joins to preserve the required dangling tuples only when necessary!

d) SQL Query:

```
SELECT C.cname, P.pizza
FROM   Customers C, Pizzas P
EXCEPT
SELECT * FROM Likes;
```

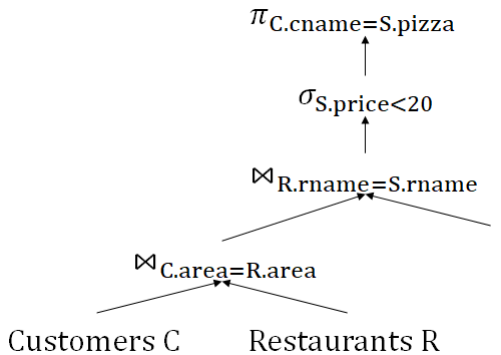
9. Consider the following query. For each customer, find the pizzas sold by restaurants with a price under $20 where the restaurants is located in the same area as the customer's area. *Exclude* customers whose associated set of pizzas is empty.
The following are two possible SQL answers (denoted by Q1 and Q2) for the above query.
Q1:

```
SELECT DISTINCT C.cname, S.pizza
FROM   ( Customers C JOIN Restaurants R ON C.area = R.area )
       JOIN Sells S ON R.rname = S.rname;
WHERE  price < 20;
```

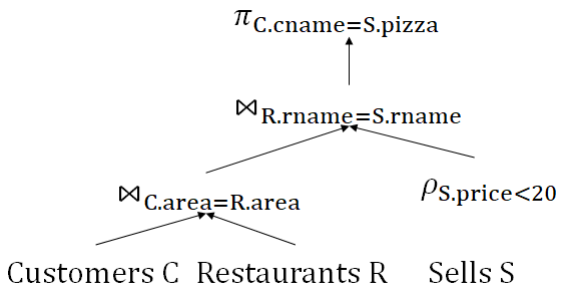
Q2:

```
SELECT DISTINCT C.cname, S.pizza
FROM   ( Customers C JOIN Restaurants R on C.area = R.area )
       JOIN Sells S ON R.rname = S.rname AND price < 20;
```

Observe that Q1 and Q2 are similar except that the predicate on price appears as a selection predicate in Q1 but appears as part of the second join predicate in Q2. The semantics of these two SQL queries are defined by the relational algebra expressions[1] shown below. Is Q1 and Q2 equivalent?



Q1



Q2

[1] For convenience, we abuse the relational algebra notation to use `Customers C` to denote ρ_C(Customers)

Solution:

Both Q1 and Q2 are equivalent queries because by simplification we can get

Q1: $\sigma_{c_1}\left(\left(\mathrm{C} \bowtie_{c_2} \mathrm{R}\right) \bowtie_{c_3} \mathrm{S}\right)$

Q2: $\left(\mathrm{C} \bowtie_{c_2} \mathrm{R}\right) \bowtie_{c_3} \left(\sigma_{c_1}(\mathrm{S})\right)$

By renaming we can get

Q1: $\sigma_{c_1}\left(A \bowtie_{c_3} \mathrm{S}\right)$

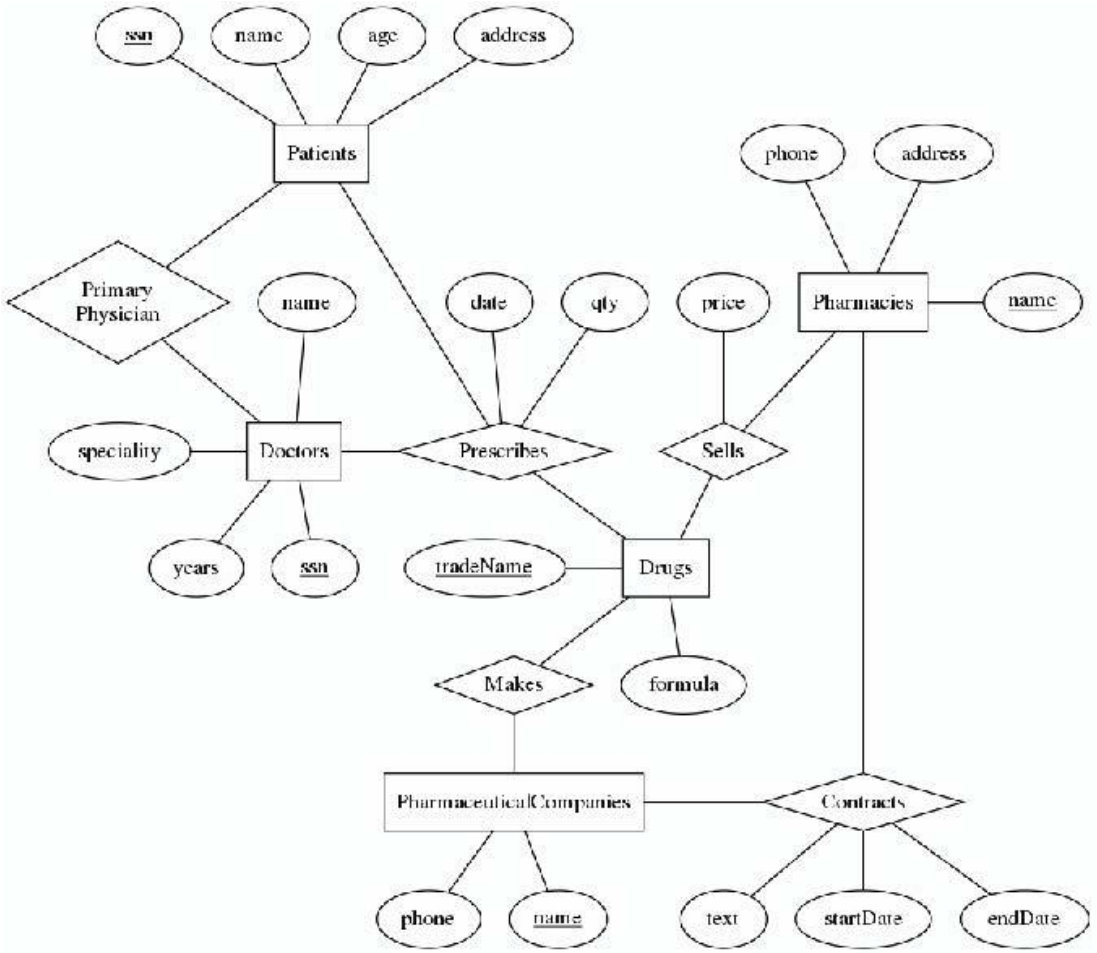
Q2: $A \bowtie_{c_3} \left(\sigma_{c_1}(\mathrm{S})\right)$

Then, we simply have to proof $\sigma_{c_1}\left(A \bowtie_{c_3} \mathrm{S}\right) \equiv A \bowtie_{c_3} \left(\sigma_{c_1}(\mathrm{S})\right)$. This can be done by partitioning S into $\sigma_{c_1}(\mathrm{S}) \cup \left(\mathrm{S} - \sigma_{c_1}(\mathrm{S})\right)$. Thus, $A \bowtie_{c_3} \mathrm{S} \equiv \left(A \bowtie_{c_3} \sigma_{c_1}(\mathrm{S})\right) \cup \left(A \bowtie_{c_3} \left(\mathrm{S} - \sigma_{c_1}(\mathrm{S})\right)\right)$. Therefore $\sigma_{c_1}\left(A \bowtie_{c_3} \mathrm{S}\right) \equiv \sigma_{c_1}\left(A \bowtie_{c_3} \sigma_{c_1}(\mathrm{S})\right) \cup \sigma_{c_1}\left(A \bowtie_{c_3} \left(\mathrm{S} - \sigma_{c_1}(\mathrm{S})\right)\right) \equiv A \bowtie_{c_3} \sigma_{c_1}(S)$.

10. The Prescriptions-R-X chain of pharmacies has offered to give you a free lifetime supply of medicine if you design its database. Given the rising cost of health care, you agree. Here's the information that you gather:
    - Patients are identified by an SSN, and their names, addresses, and ages must be recorded.
    - Doctors are identified by an SSN. For each doctor, the name, specialty, and years of experience must be recorded.
    - Each pharmaceutical company is identified by name and has a phone number. For each drug, the trade name, and formula must be recorded. Each drug is sold by a given pharmaceutical company, and the trade name identifies a drug uniquely from among the products of that company. If a pharmaceutical company is deleted, you need not keep track of its products any longer.
    - Each pharmacy has a name, address, and phone number.
    - Every patient has a primary physician. Every doctor is the primary physician of at least one patient.
    - Each pharmacy sells several drugs and has a price for each. A drug could be sold at several pharmacies, and the price could vary from one pharmacy to another.
    - Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, and a patient could obtain prescriptions from several doctors. Each prescription has a date and a quantity associated with it. You can assume that, if a doctor prescribes the same drug for the same patient more than once, only the last such prescription needs to be stored.
    - There is exactly one contract between a pharmacy and a pharmaceutical company if and only if that pharmacy sells some drug that is made by that pharmaceutical company. For each contract, you have to store a start date, an end date, and the text of the contract.

    a) Consider the ER diagram shown on the next page for Prescriptions-R-X. What are the constraints that are not captured by this design? Modify the ER design to capture as many of the constraints as possible.
    b) Translate your ER design in part (a) into a relation schema using SQL (assume reasonable data types for the domain constraints). Your solution should capture as many of the application's constraints as possible. Identify any constraints that are not captured by your relational schema.
    c) How would your design in part (a) change if each drug must be sold at a fixed price by all pharmacies?
    d) How would your design in part (a) change if the design requirements change as follows: If a doctor prescribes the same drug for the same patient more than once, several such prescriptions may have to be stored.
    e) Suppose that pharmacies appoint a supervisor for each contract. There must always be a supervisor for each contract, but the contract supervisor can change over the lifetime of the contract. Supervisors are identified by an SSN, and their start dates must be recorded. Modify your ER design in part (a) to capture this additional requirement.

f) Translate your ER design in part (a) into a relational schema. Identify any constraints that are not captured by your relational schema.

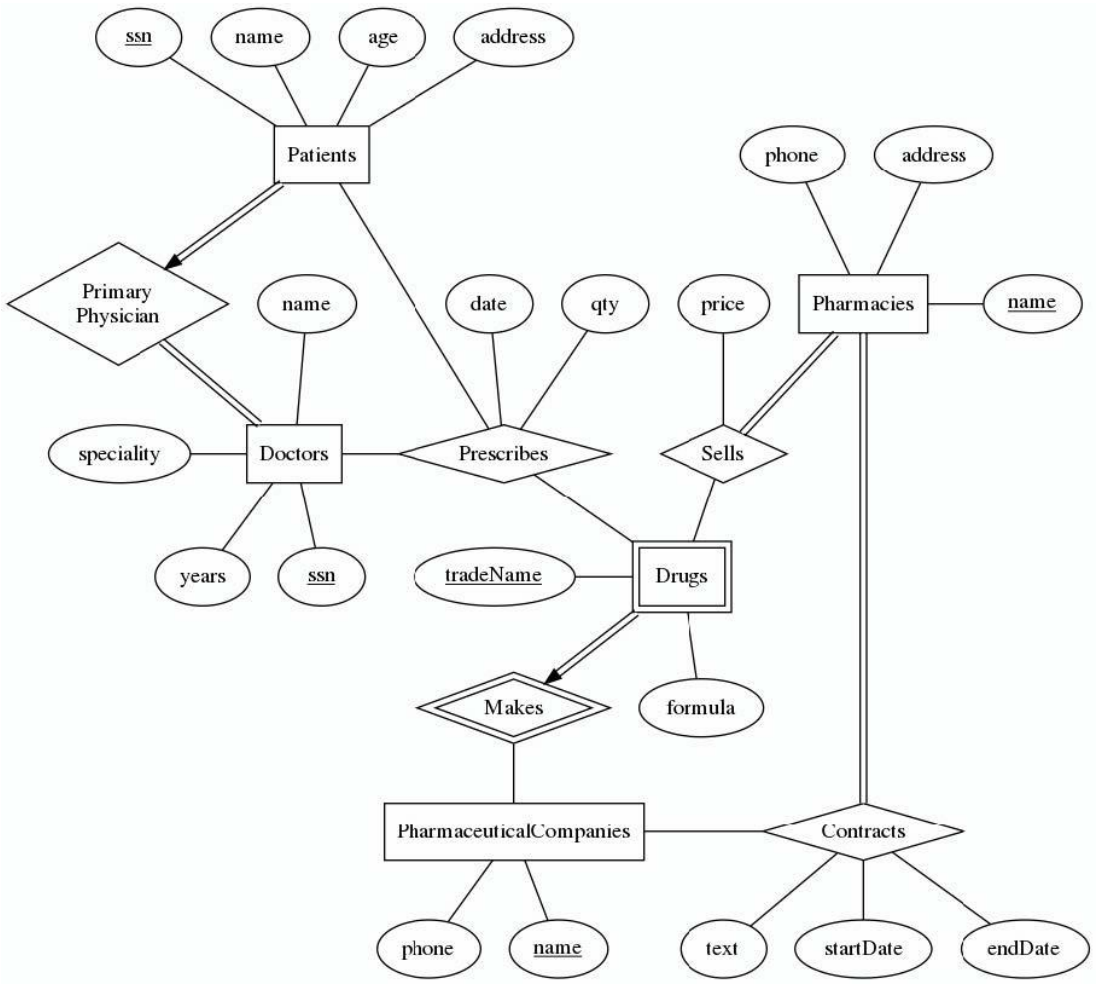


Solution:

a) The following constraints are not captured by the ER design:

- **C1.** `Drugs` is a weak entity set that is dependent on the owner entity set `PharmaceuticalCompanies`
- **C2.** The key and total participation constraints of `Patients` w.r.t. `PrimaryPhysician` relationship
- **C3.** The total participation constraint of `Doctors` w.r.t. `PrimaryPhysician` relationship
- **C4.** The constraint that each pharmacy sells more than one drug
- **C5.** The constraint that there is exactly one contract between a pharmaceutical company and a pharmacy if and only if that pharmacy sells some drug that is made by that pharmaceutical company

The revised ER design shown below captures constraints C1, C2, and C3. Constraint C4 is captured only partially (total participation of `Pharmacies` w.r.t. `Sells` relationship) but it does not require that each pharmacy sells more than one drug. Constraint C5 is not fully captured as the ER design merely specifies that every pharmacy `P` must have at least one contract with some pharmaceutical company `PC`; note that `P` does not necessarily sell some drug that is made by `PC`.



b) SQL Code

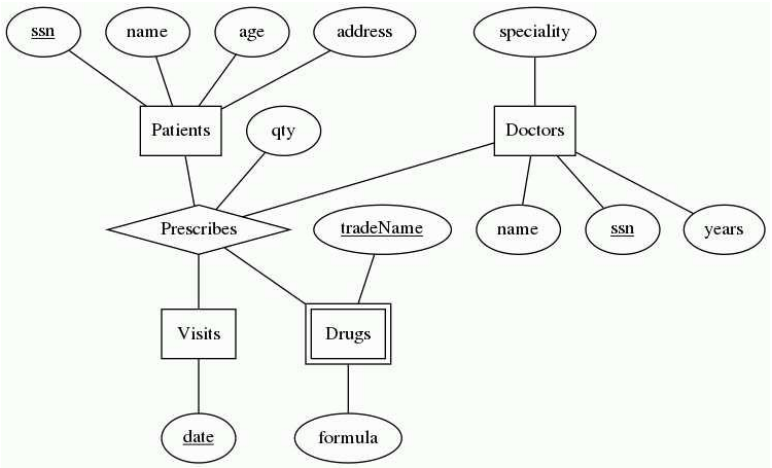
```
CREATE TABLE Doctors (
  ssn     char(10) PRIMARY KEY,
  name    char(30),
  spec    char(20),
  years   integer  );
```

```
CREATE TABLE Patients (
  ssn     char(10) PRIMARY KEY,
  priPhy  char(10) NOT NULL REFERENCES Doctors (ssn),
  name    char(30),
  addr    char(30),
  age     integer  );
CREATE TABLE Pharmacies (
  name    char(30) PRIMARY KEY,
  phone   char(10),
  addr    char(30)  );
CREATE TABLE PharmaceuticalCompanies (
  name    char(30) PRIMARY KEY,
  phone   char(10)  );
CREATE TABLE Drugs (
  pcname  char(30) REFERENCES PharmaceuticalCompanies (name)
                   ON DELETE CASCADE,
  tname   char(30),
  formula char(20),
  PRIMARY KEY (pcname, tname)  );
CREATE TABLE Prescribes (
  dssn    char(10) REFERENCES Doctors  (ssn),
  pssn    char(10) REFERENCES Patients (ssn),
  pcname  char(30),
  tname   char(30),
  date    date,
  qty     integer,
  PRIMARY KEY (dssn, pssn, pcname, tname),
  FOREIGN KEY (pcname, tname) REFERENCES Drugs  );
CREATE TABLE Contracts (
  pcname  char(30) REFERENCES PharmaceuticalCompanies (name),
  pname   char(30) REFERENCES Pharmacies (name),
  start   date,
  end     date,
  text    char(60),
  PRIMARY KEY (pname, pcname)  );
CREATE TABLE Sells (
  pname   char(30),
  pcname  char(30),
  tname   char(30),
  price   numeric,
  PRIMARY KEY (pname, pcname, tname),
  FOREIGN KEY (pname , pcname) REFERENCES Contracts (pname, pcname),
  FOREIGN KEY (pcname, tname ) REFERENCES Drugs  );
```

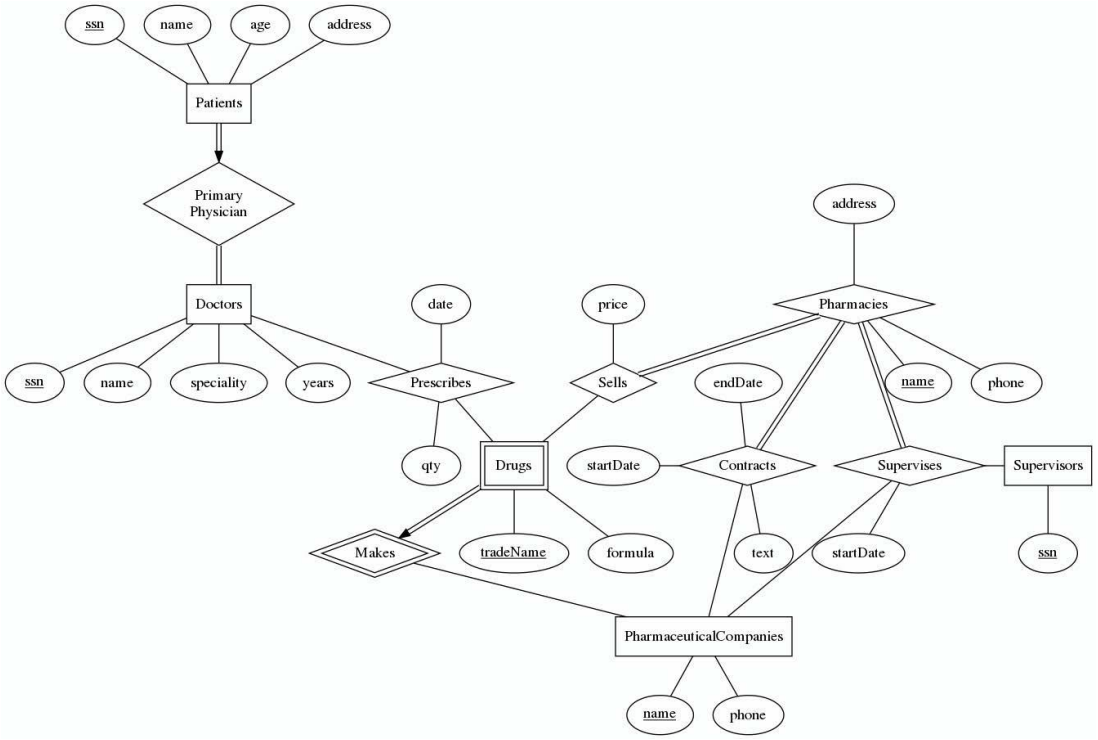
The above relational schema does not enforce constraints C3 and C4. Constraint C5 is enforced only partially: the foreign key constraint from Sells to Contracts guarantees that if a pharmacy sells some drug made by a pharmaceutical company, then there is a contract between that pharmacy and pharmaceutical company. However, it is possible for a contract to exist in the database between a pharmacy and a pharmaceutical company where that pharmacy does not sell any drug made by that pharmaceutical company.

c) Instead of modeling `price` as an attribute of `Sells` relationship set, we model `price` as an attribute of `Drugs` entity set.

d) `Prescribes` is modelled as a 4-ary relationship set that involves an additional entity set `Visits` with primary key `date`.

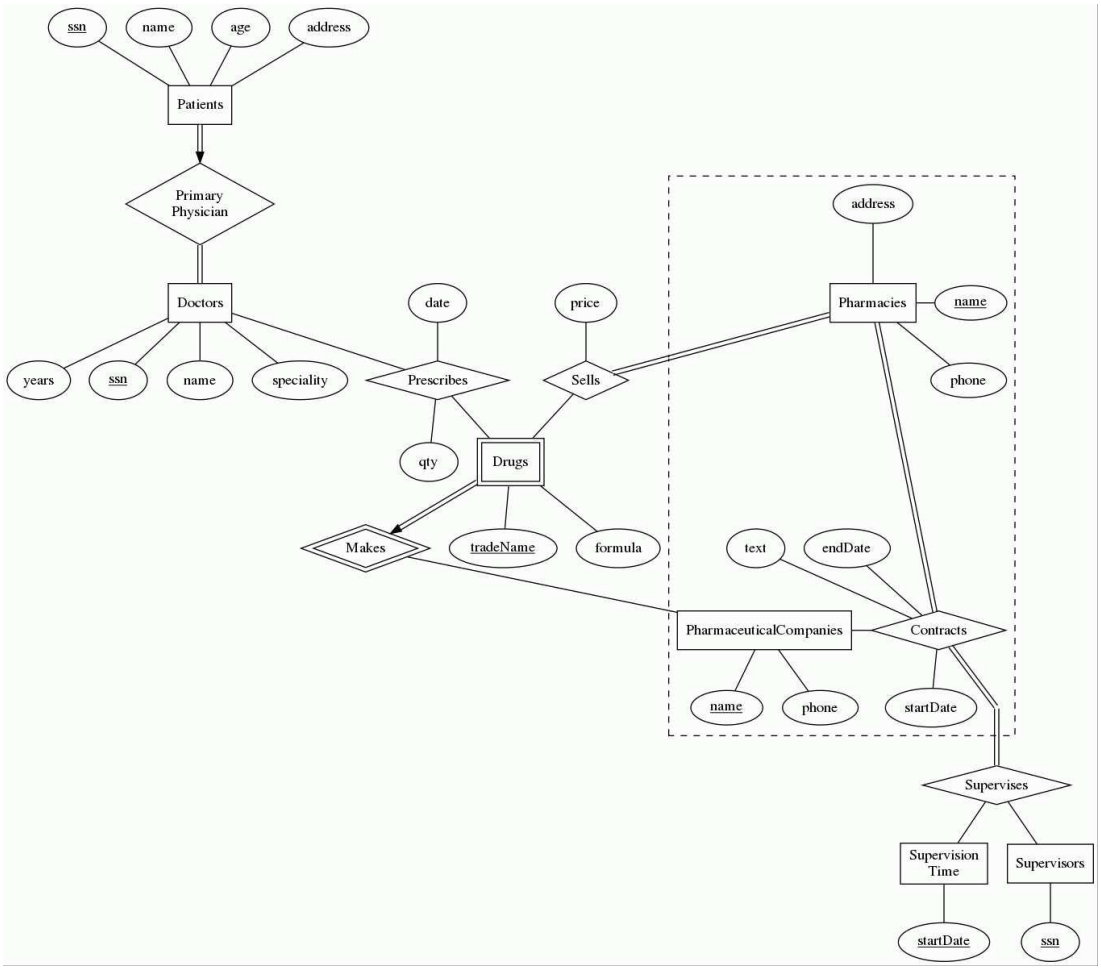


e) Consider the following ER design which introduces a new ternary relationship set "`Supervises`" to relate `Pharmacies`, `PharmaceuticalCompanies`, and a new entity set "`Supervisors`". Since each pharmacy has at least one contract, a total participation constraint on `Pharmacies` w.r.t. `Supervises` is required.



The revised ER design doesn't completely capture the new requirement: the existence of `(P,PC,S)` in a Supervises relationship does not necessarily mean that there exists a contract between pharmacy `P` and pharmaceutical company `PC`. Furthermore, this ER design also does not allow supervisor to supervise the same contract multiple times (over different time periods).

A better ER design (shown below) is to introduce an additional entity set `SupervisionTime` (with key attribute `start`) and to model `Supervises` using an aggregation to relate `Contracts`, `Supervisors`, and `SupervisionTime`. This ensures that each contract will be supervised by some supervisor, and a supervisor could supervise the same contract multiple times.



Finally, we discuss one last design. Instead of using aggregation with the `Supervises` relationship set (as shown in the last ER diagram), this design simply do away with using aggregation and the `Supervises` relationship set by connecting the two supervision-related entity sets (i.e., `Supervisors` and `SupervisionTime`) directly to the `Contracts` relationship set thereby turning it into a 4-ary relationship set. A disadvantage of this final ER design is that it would lead to *data redundancy* in the relational table representing the `Contracts` relationship set. This is because the information related to a contract (i.e., `start`, `end`, and `text`) is actually dependent only on the pair of pharmaceutical company and pharmacy entities that define the contract, and is independent of the supervision-related pair of entities. However, by storing the contract-related information together with the supervision-related information in the same table, the contract-related information will be unnecessarily replicated for every supervision for that contract. We will be examining this data redundancy issue in greater detail when we discuss the topic on schema refinement.

f) The following shows the additional relational tables derived from the last aggregation-based ER diagram.

```
CREATE TABLE Supervisors (
  ssn    char(10) PRIMARY KEY  );
CREATE TABLE SupervisionTime (
  start  date PRIMARY KEY  );
CREATE TABLE Supervises (
  pcname char(30),
  pname  char(30),
  spssn  char(10) NOT NULL REFERENCES Supervisors (ssn),
  start  date             REFERENCES SupervisionTime (start),
  PRIMARY KEY (pname, pcname, start),
  FOREIGN KEY (pname, pcname) REFERENCES Contracts  );
```

The primary key ensures that there can't be more than one supervisor supervising a contract at the same time. The foreign key constraint from `Supervises` to `Contracts` ensures that every supervised pair of pharmacy and pharmaceutical company indeed has a contract between them. However, the above schema does not enforce the total participation constraint of `Contracts` w.r.t. `Supervises`.