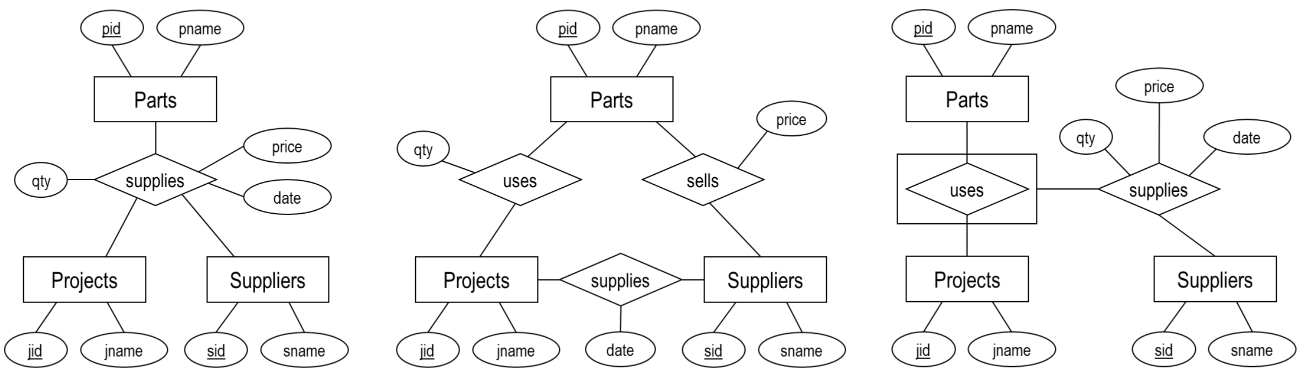
Discussions

1. **(ER Analysis)** Consider an application about Parts (with identifier `pid` and name `pname`) supplied by Suppliers (with identifier `sid` and name `sname`) to Projects (with identifier `jid` and name `jname`) where `price` represents the unit price of the part supplied, `qty` represents the quantity of the the part supplied, and `date` represents the date of transaction.

   We have three different ER designs for the application:



Design A: Ternary Design B: Binary Design C: Aggregate

(a) Discuss whether the designs are *equivalent* in the sense that they are capturing the same constraints of the application. For simplicity, focus on the following set of constraints:

> 1. A Suppliers S may sell a Parts P without being used by a Projects J.
> 2. A Suppliers S may sell a Parts P to Projects J_1 for a `price` \$10 each but the same Suppliers S could be selling the same Parts P to Projects J_2 for a `price` \$20 each.
> 3. The information of which Suppliers S selling Parts P to be used by Projects J can be obtained (*i.e.*, we may be able to obtain the triple (S, P, J)).

(b) If none of the design satisfies all the constraints above, draw a possible design that satisfies all of the constraints above.

Suggested Guide:

(a)
- **Design A: Ternary**
  1. (✗) This constraint is not satisfied as a relationship in the supplies relationship set must at least contain the triple (S, P, J) so a Suppliers S cannot sell a Parts P without any value of Projects J.
  2. (✓) This constraint is satisfied as the triple (S, P, J) can uniquely identify `price`. Therefore, we may have the following two relationships in supplies $(S, P, J_1, \$10, \_, \_)$ and $(S, P, J_2, \$20, \_, \_)$. A potential table is shown below.

supplies Relationship Set

sid	pid	jid	price	qty	date
S	P	J_1	\$10	_	_
S	P	J_2	\$20	_	_

3. (✓) This constraint is satisfied as we are recording $(S, P, J, \_, \_, \_)$ in supplies that contains the information (S, P, J).

- **Design B: Binary**
  1. (✓) This constraint is satisfied by the relationship set sells. The relationship in sells contains the information $(S, P, \_)$. Additionally, the relationship set sells is independent from the relationship set supplies. As such, the relationship set supplies may be empty.
  2. (✗) This constraint is not satisfied as the information $(S, P, \_)$ in sells disallows the pair (S, P) to have different values of `price`.
  3. (✗) This constraint is not satisfied. Consider the following two information:
     - Suppliers S_1 sells Parts P_1 to be used by Projects J.
     - Suppliers S_2 sells Parts P_2 to be used by Projects J.

     Such information cannot be stored in the design without the data being split into multiple relationship sets. Consider the following two additional information:
     - Suppliers S_1 sells Parts P_1 and Parts P_2.
     - Suppliers S_2 sells Parts P_1 and Parts P_2.

     The relationship sets sells, uses, and supplies that contains this information must at least be the following:

     sells

sid	pid	price
S_1	P_1	_
S_1	P_2	_
S_2	P_1	_
S_2	P_2	_

     uses

pid	jid	price
P_1	J	_
P_2	J	_

     supplies

sid	jid	price
S_1	J	_
S_2	J	_

     We then have no way of retrieving the two information we started with.

- **Design C: Aggregate**
  1. (✗) This constraint is not satisfied as a relationship in the supplies relationship set must at least contain the triple (S, P, J) so a Suppliers S cannot sell a Parts P without any value of Projects J.
  2. (✓) This constraint is satisfied as the triple (S, P, J) can uniquely identify `price`. This is similar to design A.
  3. (✓) This constraint is satisfied as we are recording $(S, P, J, \_, \_, \_)$ in supplies that contains the information (S, P, J).

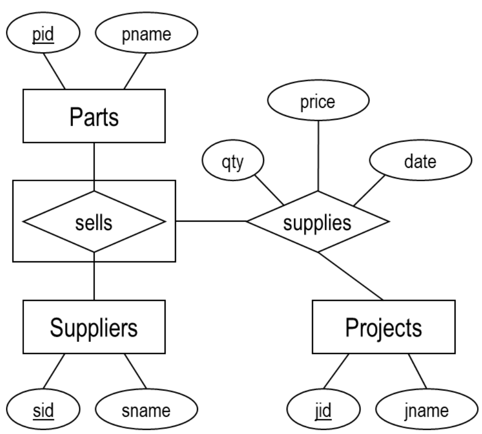
(b) As we have seen above, none of the design satisfies all the constraints.

Constraints	**Design A**	**Design B**	**Design C**
Constraint #1	✗	✓	✗
Constraint #2	✓	✗	✓
Constraint #3	✓	✗	✓

We look at each constraints above and try to satisfy each of the constraints.

1. We must have a relationship set sells that only relates Suppliers and Parts without Projects.
2. The attribute `price` cannot be uniquely identified by the pair (S, P) but must be uniquely identified by the triple (S, P, J) instead. This means that the attribute `price` cannot be in the relationship set sells assuming sells only relates Suppliers and Parts.
3. We must have a relationship set supplies such that the triple (S, P, J) appear. We can use a ternary relationship set supplies similar to design A, but we can do better to avoid duplication of data. Since we already have a relationship set sells to satisfy constraint 1 above, we can have a relationship set supplies relating sells and Projects. This means that sells must be an aggregate.

Combining all the ideas above, we arrive at the following possible design.

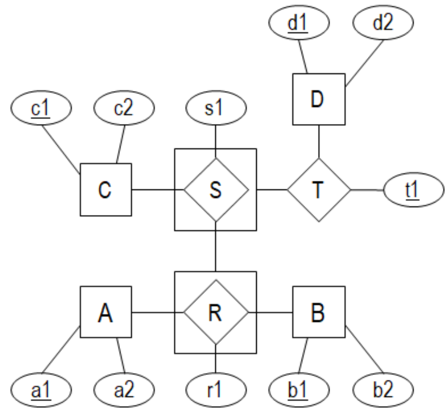


2. (**ER to Schema**)

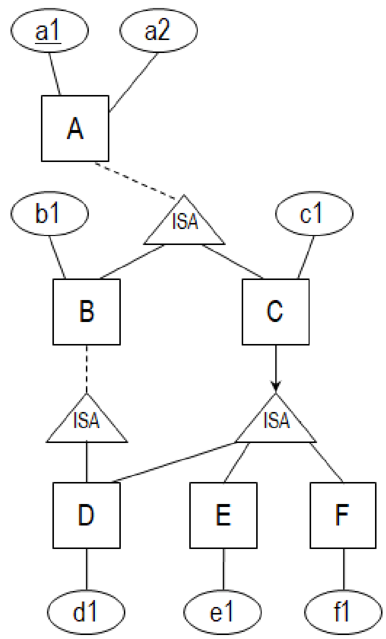
Translate each ER diagram on the right into a relational schema. In your translation,

- enforce as many constraints captured in the ER diagram as possible
- do not enforce additional constraints that are not captured in the ER diagram
- assume that all the attributes have INT domain

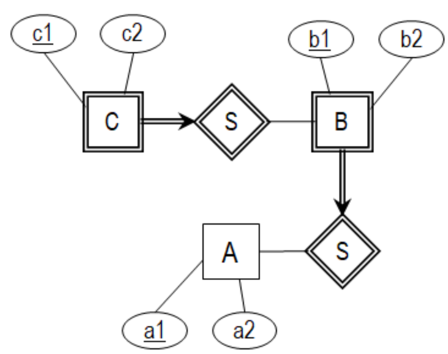
(a) **ER A**



(b) **ER B**



(c) **ER C**



Suggested Guide:

The idea is to start translating the ER diagram from the elements (*entity set, relationship set, or aggregate*) with the *least* number of foreign key. Once this is done, we select from the remaining elements, the next smallest number of foreign key.

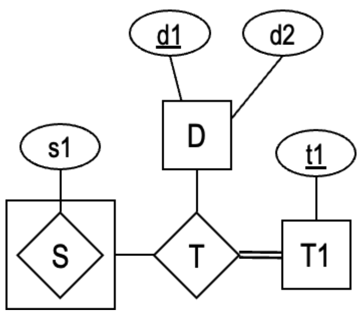
The process can be repeated until all elements are translated. If there are *circular foreign key*, break the dependency and use `ALTER TABLE` after all relations are translated.

(a) **ER A ("`T02Q2A.sql`")**

```sql
CREATE TABLE A (
  a1  INT PRIMARY KEY,
  a2  INT
);

CREATE TABLE B (
  b1  INT PRIMARY KEY,
  b2  INT
);

CREATE TABLE C (
  c1  INT PRIMARY KEY,
  c2  INT
);

CREATE TABLE D (
  d1  INT PRIMARY KEY,
  d2  INT
);

CREATE TABLE R (
  a1  INT REFERENCES A,
  b1  INT REFERENCES B,
  r1  INT,
  PRIMARY KEY (a1, b1)
);

CREATE TABLE S (
  a1  INT,
  b1  INT,
  c1  INT REFERENCES C,
  s1  INT,
  PRIMARY KEY (a1, b1, c1),
  FOREIGN KEY (a1, b1) REFERENCES R
);
```

```sql
CREATE TABLE T (
  a1  INT,
  b1  INT,
  c1  INT,
  d1  INT REFERENCES D,
  t1  INT,
  PRIMARY KEY (a1, b1, c1, d1, t1),
  FOREIGN KEY (a1, b1, c1) REFERENCES S
);
```

Note that the attribute `t1` in `T` is only a part of the key attribute and not the only primary key. The equivalent ER diagram for `T` is shown in the image below.



The use of key attributes on relationship set is non-standard and should not be used on anything other than simplifying the given design above. Any other meaning is not accepted.

(b) **ER B (“`T02Q2B.sql`”)**

```sql
CREATE TABLE A (
  a1  INT PRIMARY KEY,
  a2  INT
);

CREATE TABLE B (
  a1  INT PRIMARY KEY REFERENCES A
        ON DELETE CASCADE,
  b1  INT
);

CREATE TABLE C (
  a1  INT PRIMARY KEY REFERENCES A
        ON DELETE CASCADE,
  c1  INT
);
```

```sql
CREATE TABLE D (
  a1  INT PRIMARY KEY
        REFERENCES B ON DELETE CASCADE
        REFERENCES C ON DELETE CASCADE,
  d1  INT
);

CREATE TABLE E (
  a1  INT PRIMARY KEY REFERENCES C
        ON DELETE CASCADE,
  e1  INT
);

CREATE TABLE F (
  a1  INT PRIMARY KEY REFERENCES C
        ON DELETE CASCADE,
  f1  INT
);
```

Note the use of multiple foreign key constraint (*i.e.*, `REFERENCES`) on table `D`. The use of `ON DELETE CASCADE` is attached to each foreign key constraints. This is consistent with the meaning of "inheritance" for which ISA hierarchy is based on. For an entity in `D` to exists, the entity must first be an entity in both `B` and `C`.

Although it may seem surprising that we can use multiple foreign key on a single attribute, its usage is consistent with the use of multiple *constraints* on a single attribute. We have seen multiple usage of this such as having a combination of `PRIMARY KEY`, `UNIQUE`, `NOT NULL`, `CHECK`, and `REFERENCES` on a single attribute. This is simply a logical extension of that.

Also note that for table `D`, `E`, and `F`, we cannot reference directly to `A.a1` as it would allow for an entity `A` to exist without it being an entity in `C` yet it is an entity in `E`. This violates the meaning of inheritance.

Finally, note that if `D` has multiple inheritance but without "common" key attribute (*e.g.*, in this case it is due to both `B` and `C` share a superclass), then we simply choose one of the key attribute as the primary key for `D` and we let the other key attribute to be candidate key (*i.e.*, `UNIQUE` and `NOT NULL`).

(c) **ER C ("`T02Q2C.sql`")**

```sql
CREATE TABLE A (
  a1  INT PRIMARY KEY,
  a2  INT
);
```

```sql
CREATE TABLE B (
  a1  INT REFERENCES A
        ON DELETE CASCADE,
  b1  INT,
  b2  INT,
  PRIMARY KEY (a1, b1)
);

CREATE TABLE C (
  a1  INT,
  b1  INT,
  c1  INT,
  c2  INT,
  PRIMARY KEY (a1, b1, c1),
  FOREIGN KEY (a1, b1) REFERENCES B
        ON DELETE CASCADE
);
```

Challenge

The answers to the following questions is given without explanation. Please discuss them on Canvas.

1. The Prescription-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:

   1. Patients are identified by an SSN. Their names, addresses and ages must also be recorded.
   2. Doctors are identified by an SSN. Their name, specialty and years of experience must also be recorded.
   3. Each pharmaceutical company is identified by name and has a phone number.
   4. For each drug, the *trade name* and formula must be recorded.
   5. 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.
   6. Each pharmacy has a name, address and phone number.
   7. Every patient has a primary physician.
   8. Every doctor is the primary physician of at least one patient.
   9. 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.
   10. 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 is stored.
   11. 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 relational 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.

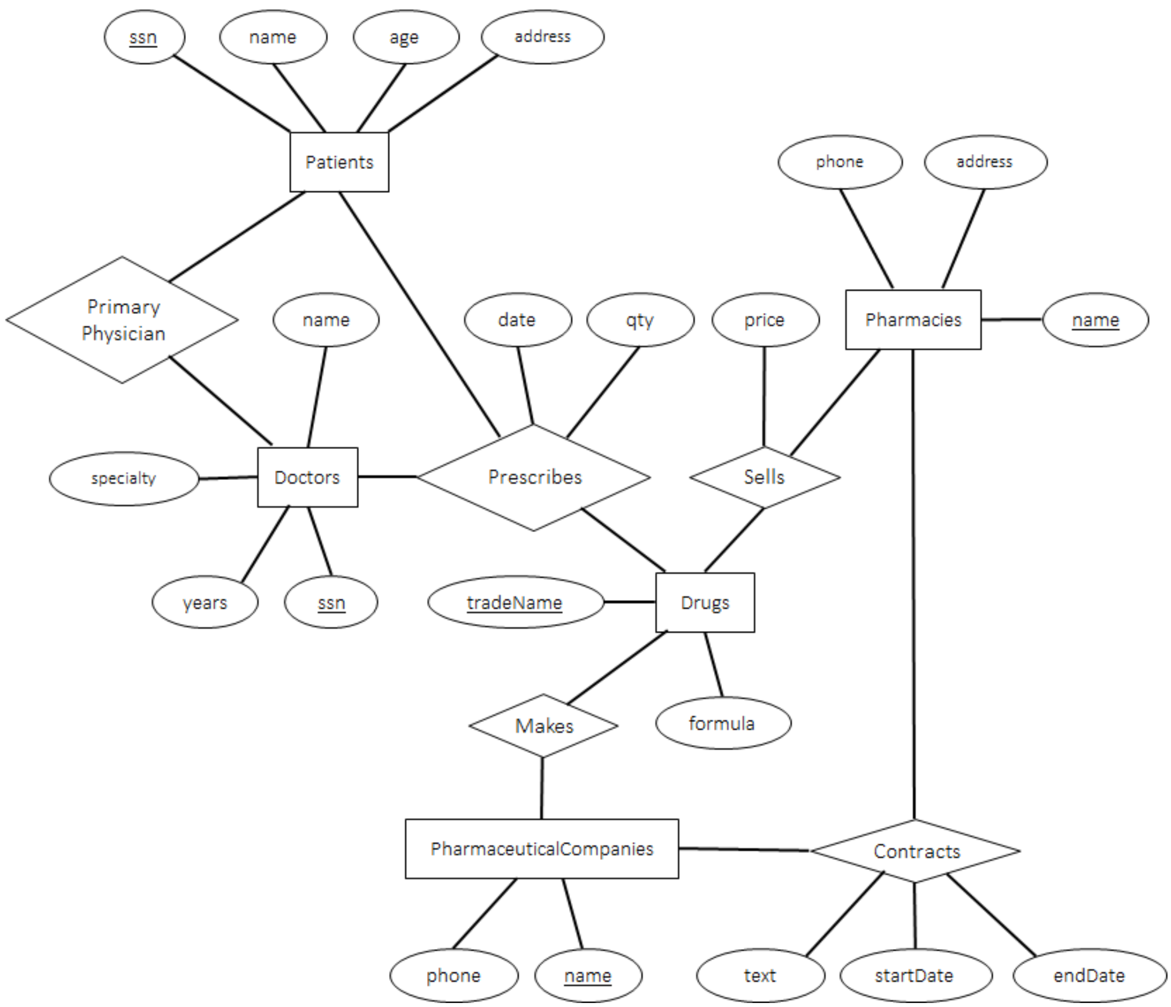
   (c) How would your design in part a change if each drug must be sold at a fixed price by all pharmacies?

   (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 e into a relational schema. Identify any constraints that are not captured by your relational schema.



Suggested Guide:

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

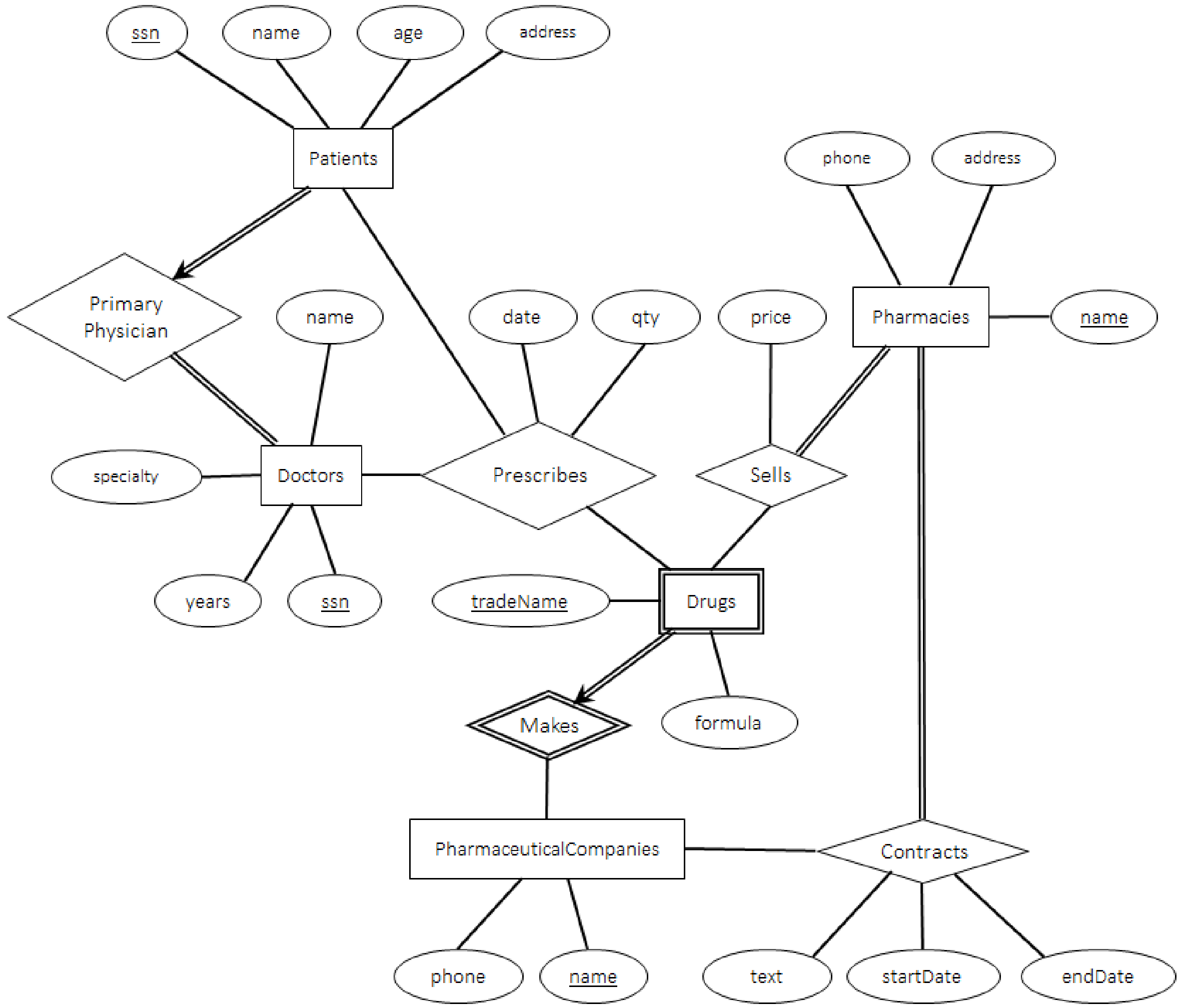
C1. Drugs is a weak entity set that is dependent on PharmaceuticalCompanies (*i.e., the owning entity set is PharmaceuticalCompanies*).

C2. The key and total participation constraints on Patients w.r.t. PrimaryPhysician relationship.

C3. The total participation constraint on 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.



(b) Relational schema

```sql
CREATE TABLE Doctors (
  ssn          TEXT PRIMARY KEY,
  name         TEXT,
  specialty    TEXT,
  years        INT
);
```

```sql
CREATE TABLE Patients (
  ssn          TEXT PRIMARY KEY,
  physician    TEXT NOT NULL REFERENCES Doctors,
  name         TEXT,
  address      TEXT,
  age          INT
);
```

```sql
CREATE TABLE Pharmacies (
  name        TEXT PRIMARY KEY,
  phone       TEXT,
  address     TEXT,
);
```

```sql
CREATE TABLE PharmaceuticalCompanies (
  name        TEXT PRIMARY KEY,
  phone       TEXT
);
```

```sql
CREATE TABLE Drugs (
  pcname      TEXT REFERENCES PharmaceuticalCompanies
                ON DELETE CASCADE,
  tradename   TEXT,
  formula     TEXT,
  PRIMARY KEY (pcname, tradename)
);
```

```sql
CREATE TABLE Prescribes (
  dssn        TEXT REFERENCES Doctors,
  pssn        TEXT REFERENCES Patients,
  pcname      TEXT REFERENCES PharmaceuticalCompanies
                ON DELETE CASCADE,
  tradename   TEXT,
  pdate       DATE,
  qty         INT,
  PRIMARY KEY (dssn, pssn, pcname, tradename),
  FOREIGN KEY (pcname, tradename) REFERENCES Drugs
);
```

```sql
CREATE TABLE Contracts (
  pname       TEXT REFERENCES Pharmacies,
  pcname      TEXT REFERENCES PharmaceuticalCompanies
                ON DELETE CASCADE,
  start_date  DATE,
  end_date    DATE,
  comments    TEXT,
  PRIMARY KEY (pname, pcname)
);
```

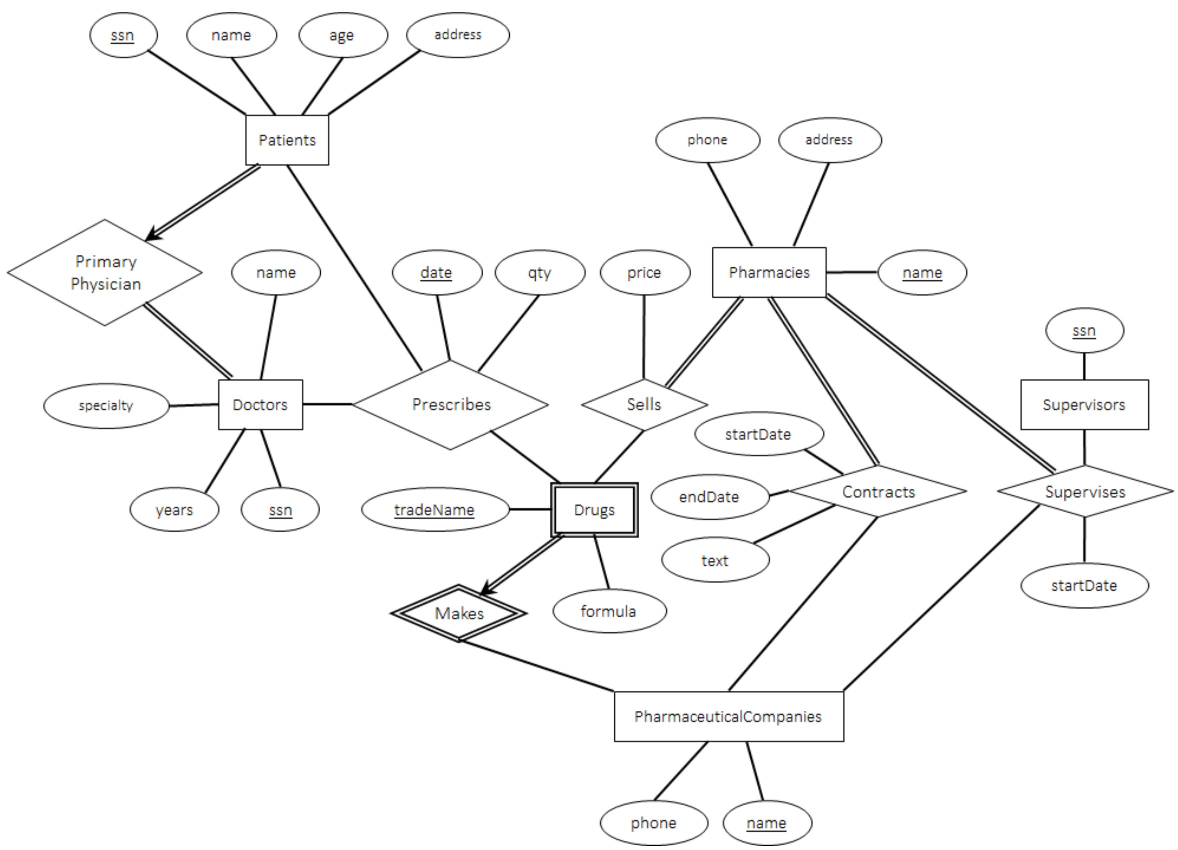
```sql
CREATE TABLE Sells (
  pname       TEXT REFERENCES Pharmacies,
  pcname      TEXT REFERENCES PharmaceuticalCompanies
                ON DELETE CASCADE,
  tradename   TEXT,
  price       NUMERIC,
  PRIMARY KEY (pname, pcname, tradename),
  FOREIGN KEY (pname, pcname) REFERENCES Contracts,
  FOREIGN KEY (pcname, tradename) REFERENCES Drugs
);
```

Constraints not enforced or partially enforced: C3, C4, and C5.

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

(d) Let Key(`R`) denote the primary key of the relation `R`. The `date` attribute of `Prescribes` is changed to a key attribute. Thus, Key(`Prescribes`) consists of the following attributes: Key(`Patients`), Key(`Drugs`) and `date`, where Key(`Patients`) = `{ssn}` and Key(`Drugs`) = `{name, tradename}`.

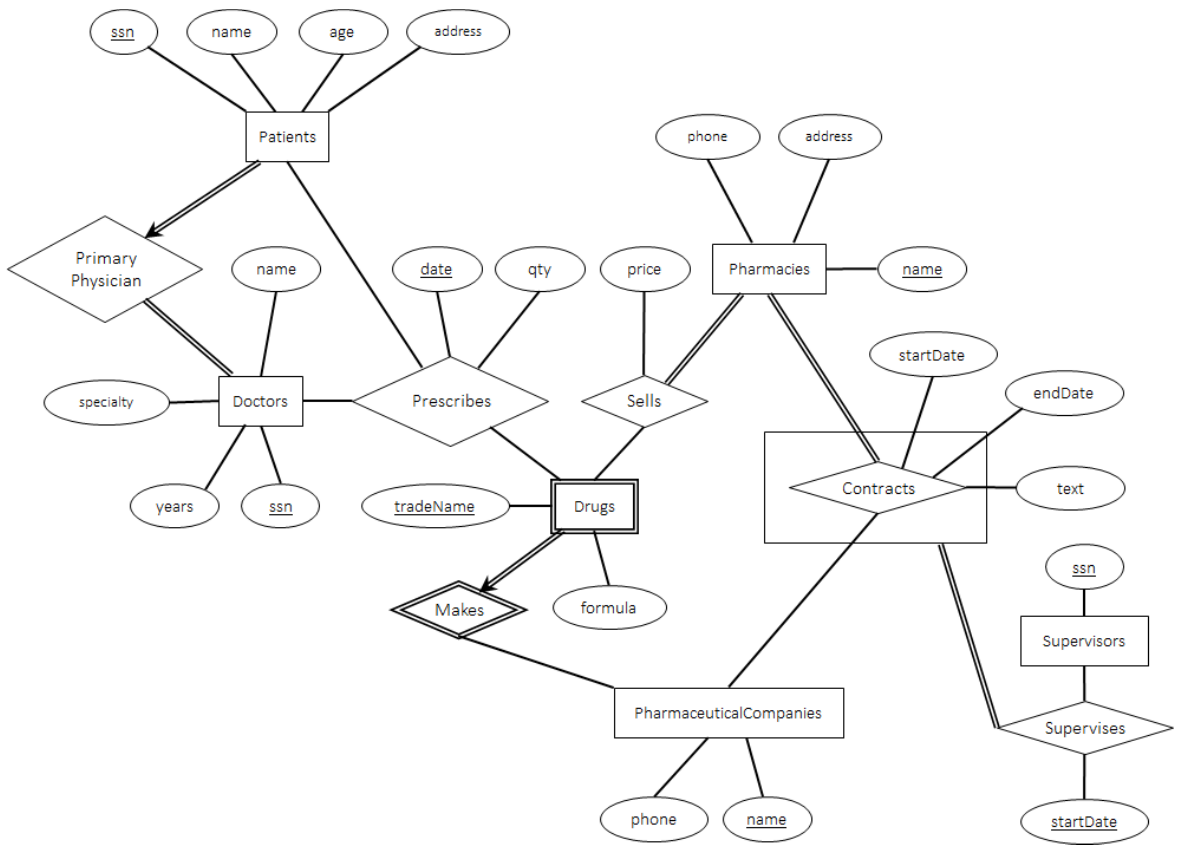
(e) **Possible Design A:** partial constraint.



Constraints not satisfied:

- The existence of (P, PC, S) in `Supervises` relationship does not necessarily mean that there exist a contract between pharmacy P and a pharmaceutical company PC.
- Does not capture the constraint that each contract is supervised by only one contractor at any time.
- Does not allow a supervisor to supervise the same contract multiple times (*over a different time periods*).

Possible Desing B: partial constraint.



Constraint not satisfied:

- Does not capture the constraint that each contract should be supervised by only one supervisor at any time.

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

```sql
CREATE TABLE Supervisors (
  ssn         TEXT PRIMARY KEY
);
```

```sql
CREATE TABLE Supervises (
  pcname     TEXT,
  pname      TEXT,
  ssn        TEXT NOT NULL REFERENCES Supervisors,
  start_date DATE,
  PRIMARY KEY (pname, pcname, start_date),
  FOREIGN KEY (pname, pcname) REFERENCES Contracts
);
```

Constraint not satisfied:

- Does not enforce the total participation constraint of `Contracts` w.r.t. `Supervises`.