Virginia Tech. CS 4604 – Introduction to DBMS

Computer Science Fall 2018, Prakash

Homework 2: E/R Models and More SQL [SOLUTION]

(due September 25th, 9:30am, in class—hard-copy please)

Reminders:

a. Out of 100 points. Contains 4 pages.
b. Rough time-estimates: 4~6 hours.
c. Please type your answers. Illegible handwriting may get no points, at the discretion of the grader. Only drawings may be hand-drawn, as long as they are neat and legible.
d. There could be more than one correct answer. We shall accept them all.
e. Whenever you are making an assumption, please state it clearly.
f. Unless otherwise mentioned, you may use any SQL/RA operator seen in class/in textbook.
g. Unless otherwise specified, assume set-semantics for RA and bag-semantics for SQL.
h. Feel free to use the linear notation for RA and create intermediate views for SQL.
i. **Important:**
    a. For E/R diagrams, use only the style/notation/material given in the lecture slides.
    b. A useful tool for creating E/R diagrams: http://logicnet.dk/DiagramDesigner/. You may have to manually draw-in some things though (like adding proper constraints etc.). There are other such good programs/sites too (like http://creately.com/, http://visio.microsoft.com/).
    Note: Microsoft Visio is a commercial tool.

Q1. Life without HAVING [20 points]

You are given the following relations:

Take(StudentID, CourseID)
RequiredForGraduation(CourseID)

The Take relation lists IDs of students and IDs of courses taken by the students. (Note that in this problem, we are assuming that each course has a unique ID, as opposed to the scheme we used in class.) The RequiredForGraduation relation lists the courses every student must take to graduate. The following query finds the students who have satisfied all the requirements for graduation.

```
SELECT StudentId
FROM Take AS T, RequiredForGraduation AS R
```

```
WHERE T.CourseID = R.CourseID
GROUP BY T.StudentID
HAVING COUNT(T.CourseId) =
        (SELECT COUNT(CourseId) FROM RequiredForGraduation);
```

You hate the HAVING clause and do not see the point of views. Rewrite this query without using views or the HAVING clause.

Ans:

```
SELECT distinct x.sid FROM ( SELECT e1.sid, e1.cno, (SELECT COUNT(*) FROM enroll e, required r
WHERE e.cno = r.cno and e.sid =e1.sid) AS NumCourses FROM enroll e1) as x WHERE NumCourses =
(SELECT COUNT(cno) FROM required);
```

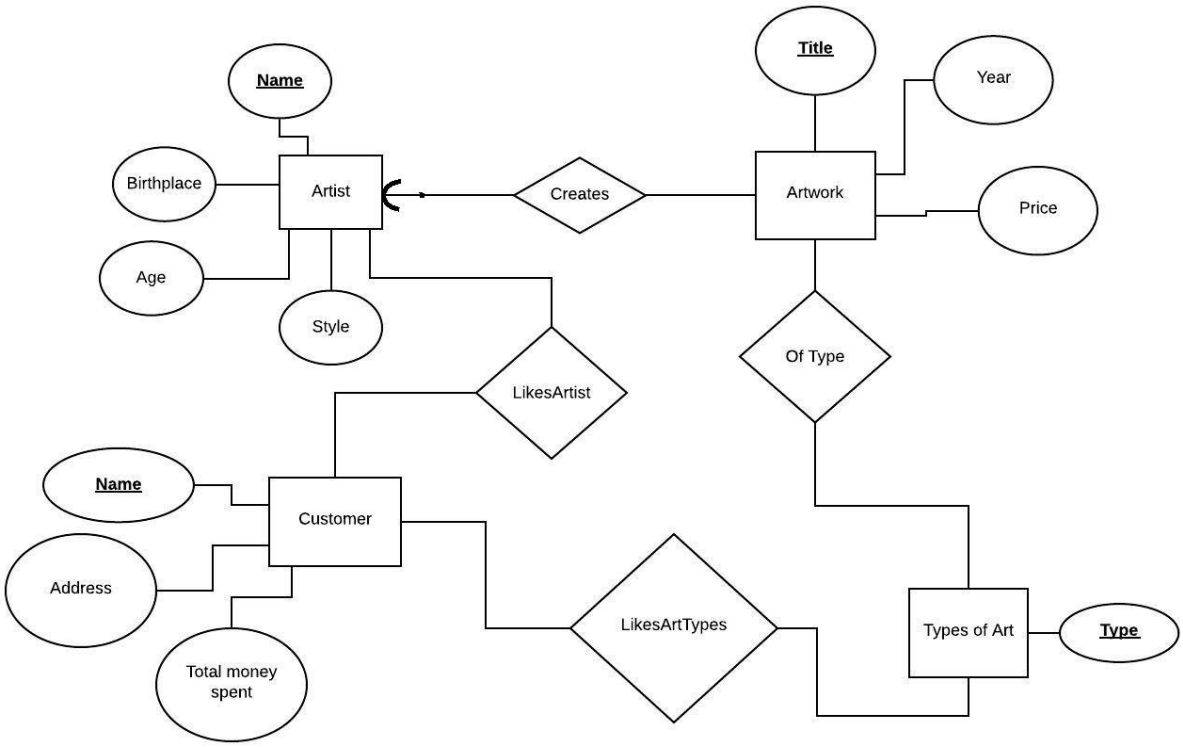
Q2. Where Art thou? [15 points]

Consider the situation in Exercise 2.8 in your textbook. We repeat it here for your convenience. Although you always wanted to be an artist, you ended up being an expert on databases because you love to cook data and you somehow confused "database" with "data baste". Your old love is still there, however, so you set up a database company, ArtBase that builds a product for art galleries. The core of this product is a database with a schema that captures all the information that galleries need to maintain. Galleries keep information about artists, their names (which are unique), birthplaces, age, and style of art. For each piece of artwork, the artist, the year it was made, its unique title, its type of art (e.g., painting, lithograph, sculpture, photograph), and its price must be stored. Pieces of artwork are also classified into groups of various kinds, for example, portraits, still-lifes, works by Picasso, or works of the 19th century; a given piece may belong to more than one group. Each group is identified by a name (like those just given) that describes the group. Finally, galleries keep information about customers. For each customer, galleries keep that person's unique name, address, total amount of dollars spent in gallery (very important!), and the artists and groups of art that the customer tends to like.

The ER model your DB engineer designed was lost, and he has resigned. So you need to step in:

Q2.1. (10 points) Draw an ER diagram for this database. Make sure to indicate primary keys, cardinality constraints, weak entities (if any), and participation constraints. List any assumptions you make in the process.

Ans:



The most important constraint placed in this diagram is that no Artist should ever be deleted from the database while their Artwork is still in the gallery.

Q2.2. (5 points) Translate the ER diagram in Q2.1 into relational database tables (i.e. give the SQL DDL statements). Make sure that the translation captures key constraints (primary keys and foreign keys if applicable) and participation constraints in the ER diagram. Identify constraints, if any, that you are not able to capture.

Ans:

create table Artist (name VARCHAR(30) PRIMARY KEY, birthplace VARCHAR(30), age INT, style VARCHAR(30))

create table Artwork(title VARCHAR(30) PRIMARY KEY, year INT)

create table TypeOfArtwork(type VARCHAR(30))

create table Customers(name VARCHAR(30) PRIMARY KEY, address VARCHAR(200), totalmoneyspent FLOAT(12,3))

create table Creates(artistname VARCHAR(30), artworktitle VARCHAR(30), FOREIGN KEY (artistname) REFERENCES Artist(name), FOREIGN KEY (artworktitle) REFERENCES Artwork(title))

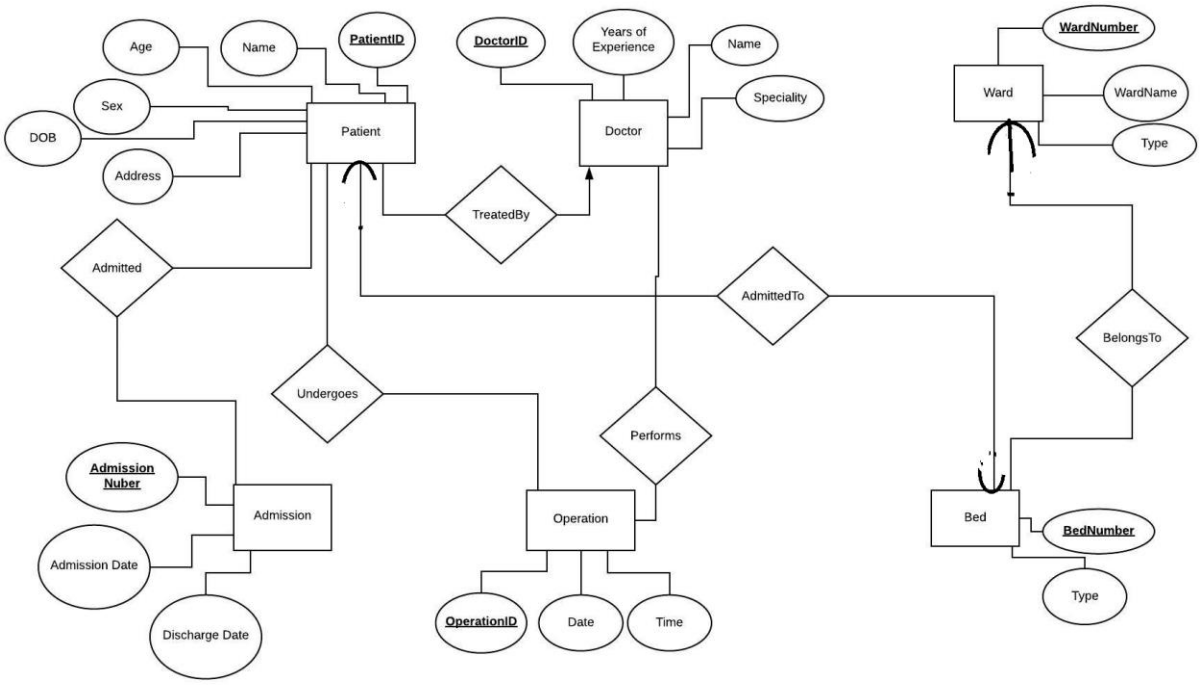
create table OfType(artworktitle VARCHAR(30), year INT, price FLOAT(12,3), artworktype VARCHAR(30), FOREIGN KEY (artworktype) REFERENCES TypesOfArt(type))

create LikesArtTypes(customername VARCHAR(30), favouriteart VARCHAR(30), FOREIGN KEY (customername) REFERENCES customer(name), FOREIGN KEY (favouriteart) REFERENCES TypesOfArt(type))

create LikesArtist(customername VARCHAR(30), artistname VARCHAR(30), FOREIGN KEY (customername) REFERENCES Customer(name), FOREIGN KEY (artistname) REFERENCES Artist(name))

Q3. Hospital DB [20 points]

We want to design a database schema for a hospital. The patient is admitted to a hospital with a medical condition. The hospital maintains patients' information. Name, age, sex, DOB, and address are recorded. Hospital identifies each patient by a unique id and creates a patient admission record. Each admission record has an admission number, admission date, and discharge date information. Hospital assigns a doctor to treat a patient. Hospital stores doctor's name, id, specialty and year of experience. Each doctor has multiple patients. A patient is admitted to a ward. Ward is identified by ward number, name, and type (e.g., medical/surgical). Each ward contains multiple beds. Bed has number and type (e.g., side room bed/ open ward bed) as well. If patient needs a surgery than hospital schedule an operation. Operation number, date, time, patient id and doctors' id are required to schedule an operation. We assume multiple doctors participated in an operation. Draw an ER diagram based on above scenario. List all of your assumptions you make for your drawing.



Q4. Insurance Database [25 points]

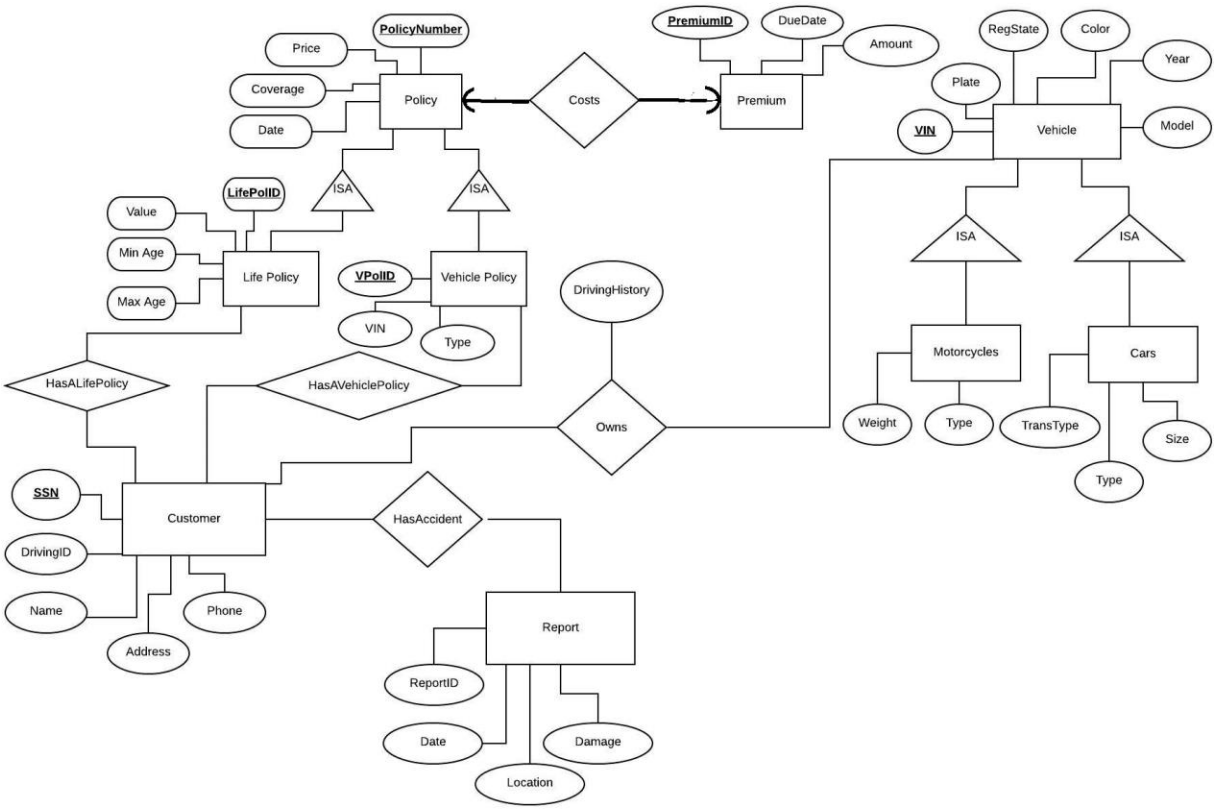
We are asked to design a database management system for all information related to VT-Insurance, a life and vehicle insurance company at Virginia Tech. The first step is to organize the information given about insurance. We have collected the following data:

- Every insurance company has its own policies. A Policy has policy_number, term_price, coverage and date.
- There are two types of policies: vehicle_policy, life_policy.
- For life_policy, we have its ID, value, minimum age and maximum age.
- For vehicle_policy, we have its ID, vehicle VIN number, type (new/old), and the driver's driving history,
- Every insurance policy covers one or more vehicles/life and premium payments associated with it. Each payment has its id, due date, and payment amount.
- Vehicle supported by VT-Insurance are only cars or motorcycles.
- Every vehicle has its VIN number, plate, registered state, color, year and model.
- For motorcycles, we should keep track of its weight, and type (standard, cruiser, sport bike, sport touring, dual sport, scooters, etc.)
- For cars, we need to have car type (sedan, coupe, etc.), transmission type (automatic, manual), and size (subcompact, compact, midsize, large).
- For each customer, we have his/her SSN, driving id, name, telephone and address.

- When a vehicle accident happens, a report should be prepared including report number, date, location, and damage cost. Accidents are also associated with a customer; so, driving license should be mentioned in the report.

Please answer the following questions:

Q4.1. (15 points) Draw an ER diagram for this database. Make sure to indicate primary keys, cardinality constraints, weak entities (if any), and participation constraints. List any assumptions you make in the process.
Hint: You may need an ISA hierarchy somewhere.



Q4.2. (10 points) Translate the ER diagram in Q4.1 into relational database tables (i.e. give the SQL DDL statements). Make sure that the translation captures key constraints (primary keys and foreign keys if applicable) and participation constraints in the ER diagram. Identify constraints, if any, that you are not able to capture.
Ans:

Create table Policy(PolicyNumber INT PRIMARY KEY, Price FLOAT(12,3), Coverage FLOAT(12,3), PolicyDate DATE)

Create table LifePolicy(PolicyNumber INT, LifePolID INT PRIMARY KEY, Value FLOAT(12,3), MinAge INT, MaxAge INT)

Create table VehiclePolicy(PolicyNumber INT, VPolID INT PRIMARY KEY, VIN INT, Type VARCHAR(30), FOREIGN KEY (VIN) REFERENCES Motorcycle(VIN))

Create table Customer(SSN INT PRIMARY KEY, DrivingID INT, Name VARCHAR(30), Address VARCHAR(250), Phone INT)

Create table Report(ReportID INT PRIMARY KEY, Date DATE, Location VARCHAR(30), Damage FLOAT(12,3))

Create table Premium(PremiumID INT PRIMARY KEY, DueDate DATE, Amount FLOAT(12,3))

Create table Vehicle(VIN INT PRIMARY KEY, Plate VARCHAR(20), RegState VARCHAR(30), Color VARCHAR(30), Year INT, Model VARCHAR(30), Weight FLOAT(12,3), Type VARCHAR(30))

Create table Motorcycle(VIN INT, Weight FLOAT(12,3), Type VARCHAR(30))

Create table Cars(VIN INT, TransType VARCHAR(30), Type VARCHAR(30), Size INT)

Create table HasALifePolicy(LifePolID INT, SSN INT, FOREIGN KEY (LifePolID) REFERENCES LifePolicy(PolicyNumber), FOREIGN KEY (SSN) REFERENCES Customer(SSN))

Create table HasAVehiclePolicy(PolicyNumber INT, SSN INT, FOREIGN KEY (VPolID) REFERENCES VehiclePolicy(PolicyNumber), FOREIGN KEY (SSN) REFERENCES Customer(SSN))

Create table Owns(SSN INT, VIN INT, DrivingHistory VARCHAR(500), FOREIGN KEY (SSN) REFERENCES Customer(SSN), FOREIGN KEY (VIN) REFERENCES Vehicle(VIN))

Create table Cost(PolicyNumber INT, PremiumID INT, FOREIGN KEY (PolicyNumber) REFERENCES Policy(PolicyNumber), FOREIGN KEY (PremiumID) REFERENCES Premium(PremiumID))

Create table HasAccident(SSN INT, ReportID INT, FOREIGN KEY (SSN) REFERENCES CUSTOMER(SSN), FOREIGN KEY (ReportID) REFERENCES Report(ReportID))

Q5. Declarative Permutations [20 points]

This is a tricky problem designed to help you think out of the box on the use of declarative database programming for solving problems.

SQL is pretty good about letting you do cross products to get all possible pairs (x, y) from two sets of elements with a simple query, for example:
SELECT x, y
FROM BigX, BigY;

But sometimes you would like to do this sort of thing horizontally instead of vertically. A permutation is an ordered arrangement of elements of a set. For example, if we have the set {1, 2, 3}, the permutations of those elements are (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), and (3, 2, 1). The rule is that for (n) elements, you have a factorial number (n!) of permutations.

What we would like is a SQL query that returns one permutation per row from a set of the first seven integers (that will give us 5,040 rows).

Assume you are given the following table Elements:
CREATE TABLE Elements
(i INTEGER PRIMARY KEY);

INSERT INTO Elements

VALUES (1), (2), (3), (4), (5), (6), (7);

Q5.1. (15 points) Write down your SQL query.
Hint: Maybe you can use some NOT INs.
Ans:

```
select *
from elements a JOIN elements b ON a.i <> b.i
JOIN elements c ON c.i NOT IN (a.i,b.i)
JOIN elements d ON d.i NOT IN (a.i,b.i,c.i)
JOIN elements e ON e.i NOT IN (a.i,b.i,c.i,d.i)
JOIN elements f ON f.i NOT IN (a.i,b.i,c.i,d.i,e.i)
JOIN elements g ON g.i NOT IN (a.i,b.i,c.i,d.i,e.i,f.i);
```

Q5.2. (5 points) The solution you get for the problem when you use an SQL interpreter and RDBMS to solve this problem (e.g. you can use SQLite) i.e. run the create table/insert statements and then your SQL query from Q5.1 above. Copy-paste only the *first 10 rows* you get in the output.

i	i	i	i	i	i	i
2	1	3	4	5	6	7
2	1	3	4	5	7	6
2	1	3	4	6	5	7
2	1	3	4	6	7	5
2	1	3	4	7	5	6
2	1	3	4	7	6	5
2	1	3	5	4	6	7
2	1	3	5	4	7	6
2	1	3	5	6	4	7
2	1	3	5	6	7	4

Note: The SQL query may be quite long so you may find it useful to create the query in a text file and use the source command (or equivalent) in your SQL interpreter to read in and execute the query.