Name

CWID

Midterm Exam

March 19th, 2018
1:50-3:05

CS520 - Data Integration, Warehousing, and Provenance

Results

Please leave this empty!

1.1 1.2 1.3 1.4 Sum

Instructions

- Try to answer all the questions using what you have learned in class. Keep hard questions until the end.
- **When writing a query, write the query in a way that it would work over all possible database instances and not just for the given example instance!**
- The exam is closed book and closed notes! No calculator, smartphones, or similar allowed!

Consider the following database schema and example instance about music albums:

user

nickname	name	postVisibility	country
BobAwesome	Bob	FOF	USA
Ali12	Alice	friends	France
Peter	Peter	friends	India
Pokegert	Gert	public	China

friends

person	friend
BobAwesome	Ali12
Ali12	BobAwesome
BobAwesome	Peter
Peter	Pokegert

posts

pid	user	text	time
1	BobAwesome	Hello just brought ...	2018-01-10
2	BobAwesome	meet @Ali12 at ...	2018-01-11
3	Peter	... is great, would recommend.	2018-01-15

Hints:

- Attributes with black background form the primary key of a relation (e.g., *nickname* for relation *user*)
- The attributes *person* and *friend* of relation *friends* are a foreign key to relation *user*.
- The attribute *user* of relation *posts* is a foreign key to relation *user*.

Part 1.1 Datalog (Total: 25 Points)

Recall that Datalog applies set semantics.

Question 1.1.1 (4 Points)

Write a **Datalog program** that returns the *nicknames* of users which have posted at least one post.

Solution

```
Q(X) :- posts(_,X,_,_).
```

Question 1.1.2 (6 Points)

Write a **Datalog program** that returns the nicknames of users that are friends of a user with nickname BobAwesome or are friends of friends of BobAwesome. For instance, in the example database Pokegert is a friend of one of BobAwesome's friends.

Solution

```
Q(X) :- friends(BobAwesome,X).
Q(X) :- friends(BobAwesome,Y), friends(Y,X).
```

Question 1.1.3 (6 Points)

Write a **Datalog program** that returns the names of users that have not written any posts. Note that we are asking for names and not nicknames.

Solution

```
hasPost(X) :- posts(_,X,_,_).
Q(Y) :- users(X,Y,_,_), not hasPost(X).
```

Question 1.1.4 (9 Points)

Write a **Datalog program** that returns all pairs of users (X, Y) such that user X can see the posts of user Y. Represent users using their nicknames. Whether the posts of a user are visible to another user is determined based on the postVisibility attribute for the user. The possible values are

- "friends": only friends of the user can see the users posts
- "FOF": in addition to friends, also friends of friends of the user can see the users posts
- "public": everybody can see the users posts

For example, one result returned for the example database would be (Pokegert,BobAwesome), because user BobAwesome has set his post visibility to FOF and Pokegert is a friend of Peter who is a friend of BobAwesome.

Solution

```
fof(X,Y) :- friends(X,Y).
fof(X,Y) :- friends(X,Z), friends(Z,Y).
nicks(X) :- user(X,_,_,_).
Q(Y,X) :- user(X,_,_,public), nicks(Y).
Q(Y,X) :- user(X,_,_,friends), friends(X,Y).
Q(Y,X) :- user(X,_,_,FOF), fof(X,Y).
```

Part 1.2 Constraints (Total: 30 Points)

Question 1.2.1 Expressing Constraints in First-Order Logic (15 Points)

Recall the representation of constraints as universally quantified formulas in first-order logic introduced in class. Write down the logical formals encoding the following constraints over the example schema:

- The foreign key from attribute person of relation friends to relation user.
- The foreign key from attribute friend of relation friends to relation user.
- The primary key of relation user
- The following functional dependency for relation posts: $user, time \rightarrow text$

Solution

$$FK_1 : \forall p, f : friends(p, f) \rightarrow \exists x_1, x_2, x_3 : user(p, x_1, x_2, x_3)$$
$$FK_2 : \forall p, f : friends(p, f) \rightarrow \exists x_1, x_2, x_3 : user(f, x_1, x_2, x_3)$$
$$PK : \forall u, x_2, x_3, x_4, y_2, y_3, y_4 : user(u, x_2, x_3, x_4) \wedge user(u, y_2, y_3, y_4) \rightarrow x_2 = y_2 \wedge x_3 = y_3 \wedge x_4 = y_4$$
$$FD : \forall u, t, x_1, x_2, y_1, y_2 : posts(x_1, u, x_2, t) \wedge posts(y_1, u, y_2, t) \rightarrow x_2 = y_2$$

Question 1.2.2 Creating Denial Constraints (15 Points)

Create denial constraints over the example schema based on the following descriptions. Some of the constraints may require you to write more than one denial constraint!

- Users from France cannot have a postVisibility of *"public"*.
- Users from USA cannot be friends with users from France and vice versa.
- Implement the functional dependency user, time $\rightarrow$ text over relation posts

Solution

$d_1 : \forall x, y, z, a : \neg(user(x, y, z, a) \wedge a = France \wedge z = public)$

$d_2 : \forall x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4 : \neg(user(x_1, x_2, x_3, x_4) \wedge users(y_1, y_2, y_3, y_4) \wedge friends(x_1, y_1) \wedge x_4 = USA \wedge y_4 = France)$
$\forall x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4 : \neg(user(x_1, x_2, x_3, x_4) \wedge users(y_1, y_2, y_3, y_4) \wedge friends(x_1, y_1) \wedge x_4 = France \wedge y_4 = USA)$

$d_3 : \forall x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4 : \neg(posts(x_1, x_2, x_3, x_4) \wedge posts(y_1, y_2, y_3, y_4) \wedge x_2 = y_2 \wedge x_4 = y_4 \wedge x_3 \neq y_3)$

Part 1.3 Query Containment And Equivalence (Total: 27 Points)

Question 1.3.1 (27 Points)

Consider the 3 queries shown below. Check all possible containment relationships. If there exists a containment mapping from Q_i to Q_j then write down the mapping.

```
Q1(X) :- R(X,Z), R(X,X), R(Z,A).
Q2(X) :- R(X,Y), R(Y,Z).
Q3(X) :- R(X,Z), R(Z,A), R(A,Y).
```

Solution

$Q_1 \rightarrow Q_2$:
no containment mapping exists

$Q_1 \rightarrow Q_3$:
no containment mapping exists

$Q_2 \rightarrow Q_1$:

$$X \rightarrow X$$
$$Y \rightarrow Z$$
$$Z \rightarrow A$$

$Q_2 \rightarrow Q_3$:

$$X \rightarrow X$$
$$Y \rightarrow Z$$
$$Z \rightarrow A$$

$Q_3 \rightarrow Q_1$:

$$X \rightarrow X$$
$$Y \rightarrow A$$
$$Z \rightarrow X$$
$$A \rightarrow Z$$

$Q_3 \rightarrow Q_2$:
no containment mapping exists

Part 1.4 Virtual Data Integration (Total: 18 Points)

Question 1.4.1 (18 Points)

Given are the views and schema shown below. Find a maximally contained rewriting of the following query using these views. The query returns titles of books, their authors, price, and year where the author is from IIT. You are allowed to use any of the algorithms discussed in class such as the Bucket algorithm. However, only the result counts, you are not required to use any of these algorithms.

```
Q(Title,Author,Price,Genre) :- Books(Title,X,Y,Genre,Z,Price),
                               Author(Title,Author),
                               Person(Author,A,'IIT').
```

Views

```
V1(Title, Pub, Genre, Price, Year) :- Books(Title,Pub, X, Genre, Year, Price),
                                      Genre = 'CS'.

V2(Title, Pub, Price, Ed) :- Books(Title, Pub, Ed, X, Y, Price).

V3(Title, Author, Affl) :- Author(Title,Author),
                           Person(Author, 'Dr.', Affl).

V4(Author) :- Author(X, Author).

V5(Title,Author) :- Books(Title, X, Y, Z, Year, A),
                    Author(Title, Author),
                    Person(Author, B, 'IIT'),
                    Year = 2017.
```

Schema

```
Books(title, publisher, edition, genre, year, price)
Author(bookTitle, authorName)
Person(name, title, affiliation)
Publisher(name, country)
```

Solution

```
Q(Title,Author,Price,Genre) :- V1(Title,X,Genre,Price,Y),
                               V3(Title,Author,'IIT').
Q(Title,Author,Price,Genre) :- V1(Title,X,Genre,Price,Y),
                               V5(Title,Author).
```

Additional conjunctive rewritings are possible but are contained in the ones shown above

```
Q(Title,Author,Price,Genre) :- V1(Title,X,Genre,Price,Y),
                               V3(Title,Author,'IIT'),
                               V5(Title,Author).
```