Midterm Exam

March 12th, 2020
10:00-11:15

CS520 - Data Integration, Warehousing, and Provenance

Please leave this empty!

1.1  1.2  1.3  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**

<table>
<thead>
<tr>
<th>nickname</th>
<th>name</th>
<th>postVisibility</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>BobAwesome</td>
<td>Bob</td>
<td>FOF</td>
<td>USA</td>
</tr>
<tr>
<td>Ali12</td>
<td>Alice</td>
<td>friends</td>
<td>France</td>
</tr>
<tr>
<td>Peter</td>
<td>Peter</td>
<td>friends</td>
<td>India</td>
</tr>
<tr>
<td>Pokegert</td>
<td>Gert</td>
<td>public</td>
<td>China</td>
</tr>
</tbody>
</table>

**friends**

<table>
<thead>
<tr>
<th>person</th>
<th>friend</th>
</tr>
</thead>
<tbody>
<tr>
<td>BobAwesome</td>
<td>Ali12</td>
</tr>
<tr>
<td>Ali12</td>
<td>BobAwesome</td>
</tr>
<tr>
<td>BobAwesome</td>
<td>Peter</td>
</tr>
<tr>
<td>Peter</td>
<td>Pokegert</td>
</tr>
</tbody>
</table>

**posts**

<table>
<thead>
<tr>
<th>pid</th>
<th>user</th>
<th>text</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BobAwesome</td>
<td>Hello just brought ...</td>
<td>2018-01-10</td>
</tr>
<tr>
<td>2</td>
<td>BobAwesome</td>
<td>meet @Ali12 at ...</td>
<td>2018-01-11</td>
</tr>
<tr>
<td>3</td>
<td>Peter</td>
<td>...is great, would recommend.</td>
<td>2018-01-15</td>
</tr>
</tbody>
</table>

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: 38 Points)

Recall that Datalog applies set semantics.

**Question 1.1.1  (5 Points)**

Write a Datalog program that returns the *name* and *nickname* of users from USA.

**Question 1.1.2  (7 Points)**

Write a Datalog program that returns the *names* of users which are living in France and have posted before 2018-01-10 or live in USA and whose posts are visible to the public (*post visibility = public*).
Question 1.1.3  (8 Points)

Write a Datalog program that returns the names of users that are not friends of user BobAwesome nor are they friends of a friend of BobAwesome. For example, in the example EDB instance there is no such person.

Question 1.1.4  (9 Points)

Write a Datalog program that returns users who have friends or friends of friends in every country.
Question 1.1.5  (9 Points)

Write a Datalog program that returns pairs of countries \((C_1,C_2)\) such that there exists at least one path in the friendship graph that connects a user from country \(C_1\) with a user from country \(C_2\). Here we do not care about the direction of edges, e.g., there is a path from Pokegert to Ali12.
Part 1.2  Constraints (Total: 26 Points)

Question 1.2.1  Expressing Constraints in First-Order Logic (13 Points)

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

- The foreign key from attribute friend of relation friends to relation user.
- Friendship has to be reciprocal, i.e., if $X$ is a friend of $Y$, then also $Y$ has to be a friend of $X$.
- The primary key of relation posts
- The following functional dependency for relation users: $country \rightarrow postVisibility$
Question 1.2.2  Creating Denial Constraints (13 Points)

Create denial constraints over the example schema based on the following descriptions.

- The friendship graph is not allowed to contain any triangles, i.e., this constraint is violated if there exists users $X$, $Y$, and $Z$ such that $X$ is a friend of $Y$, $Y$ is a friend of $Z$, and $Z$ is a friend of $X$ (Note that the direction of edges matters!).
- Users of country USA are not allowed to post after 2020-03-11.
- Implement the primary key of relation users.
Part 1.3  Query Containment And Equivalence (Total: 36 Points)

Question 1.3.1  (36 Points)

Consider the 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.

$Q_1(X,Y) :- R(X,Z), S(Z,A), R(A,Y)$.
$Q_2(Y,X) :- R(Z,X), R(Y,B), R(C,B), R(D,B)$.
$Q_3(X,Y) :- R(X,Z), R(A,Y)$.
$Q_4(A,B) :- T(X,Z), R(A,X), R(Y,B), U(Z,Y)$.