Name

CWID

Final Exam

May 4th, 2014
8:00-10:00

CS520 - Database Organization

Please leave this empty!

1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 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:

song

title	length	writtenBy	writtenYear
De Charybde en Scylla	5:35	Louis Sclavis	2008
Un Vent Noir	3:40	Louis Sclavis	2008
Standing Ovation	4:26	Louis Sclavis	1995
moto [ein weltbewusstsein.]	6:00	Michael Wertmueller	2000

songInAlbum

song	album	trackNumber
De Charybde en Scylla	Lost on the Way	1
Un Vent Noir	Lost on the Way	9
Standing Ovation	Carnet De Routes	1
moto [ein weltbewusstsein.]	Werthmueller	4

album

aTitle	releaseYear	producedBy	playedBy
Lost on the Way	2008	ECM	Louis Sclavis
Carnet de Routes	1995	Label Blue	Louis Sclavis
Werthmueller	2005	GROB	Steamboat Switzerland

artist

name	nationality
Louis Sclavis	french
Steamboat Switzerland	swiss
Michael Wertmueller	swiss

label

name	revenue
Sony	50,000,000
ECM	3,000,000
Label Bleu	150,000
GROB	10,000

Hints:

- Attributes with black background form the primary key of a relation (.e.g, *title* for relation *song*)
- The attribute *writtenBy* of relation *song* is a foreign key to relation *artist*. Note that the artist writing a song is not necessarily the artist playing the song.
- The attribute *album* of relation *songInAlbum* is a foreign key to relation *album*.
- The attribute *song* of relation *songInAlbum* is a foreign key to relation *song*.

- The attribute *producedBy* of relation *album* is a foreign key to relation *label*.
- The attribute *playedBy* of relation *album* is a foreign key to relation *artist*.
- All foreign keys have been created with the **CASCADE** option.

Part 1.1 Datalog (Total: 24 Points)

Recall that datalog applies set semantics.

Question 1.1.1 (3 Points)

Write a **datalog program** that returns the titles and artists (attribute *playedBy*) of all albums in the database.

Question 1.1.2 (3 Points)

Write a **datalog program** that returns the titles of albums played by swiss artists (*nationality* attribute)

Question 1.1.3 (4 Points)

Write a **datalog program** that returns the length of all songs written in 2008 (attribute *writtenYear*) or written by Louis Sclavis (attribute *writtenBy*).

Question 1.1.4 (4 Points)

Write a **datalog program** that returns names of artists that have released at least one album produced by label ECM (attribute *producedBy*) and at least one album released by label GROB.

Question 1.1.5 (5 Points)

Write a **datalog program** that returns songs (their title) played by (songs is on an album played by) french artists or written by french artists (attribute *writtenBy*).

Question 1.1.6 (5 Points)

Consider two artists to be connected if they have released albums with at least one common label. For instance, if Peter has released an album with ECM and Bob has also released an album with ECM then they are considered connected. Write a **datalog program** that computes all pairs of artists that are directly or indirectly connected. For example, if Bob is connected to Alice and Alice is connected to Peter, then Bob is also connected to Peter.

Part 1.2 Constraints (Total: 20 Points)

Question 1.2.1 Expressing Constraints in First-Order Logic (10 Points)

Recall the logical representation of constraints introduced in class. Write down the logical definiton for the following constraints for the example schema:

- The primary key of relation *song*
- The foreign key from attribute *playedBy* of relation *album* to relation *artist*.
- The foreign key from attribute *album* of relation *songInAlbum* to relation *album*.
- The foreign key from attribute *song* of relation *songInAlbum* to relation *song*.
- The following functional dependency for relation *album*: $releaseYear, playedBy \rightarrow producedBy$

Question 1.2.2 Creating Denial Constraints (10 Points)

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

- Albums released before 2000 cannot have more than 20 tracks. You can assume that tracks are numbered consecutively starting from 1. For instance, the track numbers for an album with three tracks would be numbered 1,2, and 3.
- No label has a revenue higher than 50,000,000.
- All songs are between 2:00 and 20:00 minutes long.
- The first track of an album is never longer than 5:00 minutes.

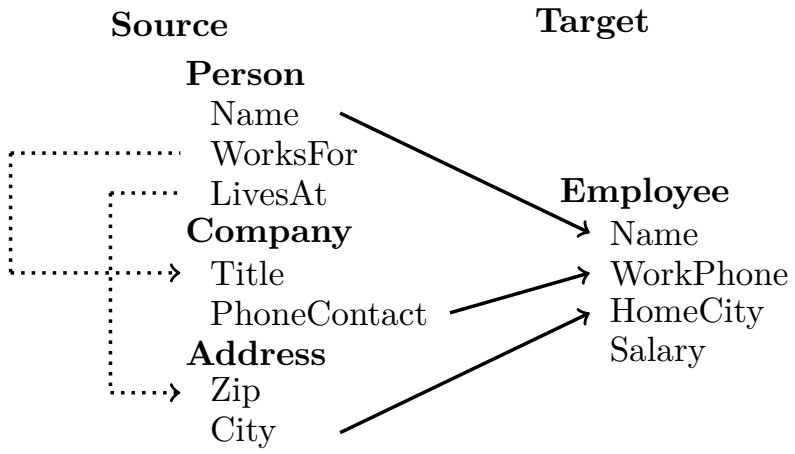
Part 1.3 Query Containment And Equivalence (Total: 18 Points)

Question 1.3.1 (18 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.

$$Q_1(X, Y) :- R(X, Y), R(Z, Y).$$
$$Q_2(X, Y) :- R(X, Y), R(Y, X).$$
$$Q_3(X, Y) :- R(X, X), R(Y, Y).$$

Part 1.4 Schema Mappings (Total: 16 Points)



Question 1.4.1 (16 Points)

Consider the source and target schemas shown above. Arrows form source attributes to target attributes represent schema matches. The dotted arrows represent foreign key constraints. Use the Clio algorithm presented in the data exchange part of class to 1) determine all source and target associations and 2) construct schema mappings using these associations and the schema matches (attribute correspondences). Do not generate mappings that are subsumed by other mappings.

Part 1.5 Virtual Data Integration (Total: 22 Points)

Question 1.5.1 (24 Points)

Use the mappings from the previous part (recall the st-tgds are equivalent to GLAV mappings under the open world assumption). As a first step write down the GLAV expressions equivalent to these st-tgs. Then find a maximally contained rewriting for the query shown below using the bucket algorithm.

$$Q(W) : -Employee(U, V, W, X)$$