Name	CWID

Midterm Exam

March 10th, 2016
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:

product

pid	version	title	category
1	8.3c	VCleaner	antivirus
1	6.0	VCleaner	antivirus
2	0.3	EncM	music
3	0.4	EncM	musc

supporter

name	salary	location
Bob	40,000	Chicago
Alice	54,000	Austin

bug

bugNumber	product	version	description	status
1	1	8.3c	Does not start on windows	resolved
2	1	6.0	Crashes after scan	open
3	2	0.4	Does not play mp3	open

bugAssignment

name	bug
Bob	1
Bob	2
Alice	2

Hints:

- Attributes with black background form the primary key of a relation (e.g., *name* for relation *supporter*)
- The attributes *product* and *version* of relation *bug* are a foreign key to relation *product*.
- The attribute *name* of relation *bugAssignment* is a foreign key to relation *supporter*.
- The attribute *bug* of relation *bugAssignment* is a foreign key to relation *bug*.

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 product titles (attribute `title` of relation `product`).

Solution

$$Q(X) :- product(Y_1, Y_2, X, Y_3).$$

Question 1.1.2 (6 Points)

Write a **Datalog program** that returns the description and status of bugs for product "VCleaner".

Solution

$$Q(X_1, X_2) :- bug(Y_1, Y_2, Y_3, X_1, X_2), product(Y_2, Y_3, Y_4, Y_5), Y_4 = \texttt{VCleaner}.$$

Question 1.1.3 (7 Points)

Write a **Datalog program** that returns all products (attribute `title`) that belong to category *antivirus* or *office* (attribute `category`).

Solution

$$Q(X) : -product(Y_1, Y_2, X, Y_3), Y_3 = \texttt{antivirus}.$$
$$Q(X) : -product(Y_1, Y_2, X, Y_3), Y_3 = \texttt{office}.$$

Question 1.1.4 (8 Points)

Write a **Datalog program** that returns the names of supportes that are not assigned to any open bugs (attribute `status`).

Solution

$$Q_{openAssign}(X) : -bug(Y_1, Y_2, Y_3, Y_4, Y_5), bugAssignment(X, Y_1), Y_5 = \texttt{open}.$$
$$Q(X) : -supporter(X, Y_1, Y_2), \neg Q_{openAssign}(X)$$

Part 1.2 Constraints (Total: 30 Points)

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

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

- The foreign key from attributes *product* and *version* of relation *bug* to relation *product*.
- The primary key of relation *product*
- The following functional dependency for relation *supporter*: *location* → *salary*

Solution

$$FK: \forall b, p, v, d, s : bug(b, p, v, d, s) \rightarrow \exists t, a : product(p, v, t, a)$$
$$PK: \forall p, v, t, c, t', c' : product(p, v, t, c), product(p, v, t', c') \rightarrow t = t' \wedge c = c'$$
$$FD: \forall n, s, l, n', s' : supporter(n, s, l) \wedge supporter(n', s', l) \rightarrow s = s'$$

Question 1.2.2 Creating Denial Constraints (15 Points)

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

- All supporters earn less than $20,000.
- Resolved bugs (attribute `status`) should not be assigned to any supporter
- Each bug is assigned to at most one supporter

Solution

$$d_1 : \forall x, y, z : \neg(supporter(x, y, z) \wedge s \geq 20000)$$
$$d_2 : \forall x, y, z, a, b, c : \neg(bug(x, y, z, a, b) \wedge bugAssignment(c, x) \wedge b = resolved)$$
$$d_3 : \forall x, y, z : \neg(bugAssignment(x, y) \wedge bugAssignment(z, y) \wedge x \neq z)$$

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.

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

Solution

$Q_1 \rightarrow Q_2$:
no containment mapping exists

$Q_1 \rightarrow Q_3$:
no containment mapping exists

$Q_2 \rightarrow Q_1$:
no containment mapping exists

$Q_2 \rightarrow Q_3$:
no containment mapping exists

$Q_3 \rightarrow Q_1$:

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

$Q_3 \rightarrow Q_2$:

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

Part 1.4 Virtual Data Integration (Total: 18 Points)

Question 1.4.1 (9 Points)

Check all correct statements below. You have to answer the question (incorrect blanks are considered errors)

- ■ GLAV mappings can be expressed as tuple-generating dependencies.
- ■ Both the inverse rule algorithm and the Minicon algorithm compute maximally contained rewritings.
- ❑ Maximally contained rewritings are independent of the query language used for expressing rewritings.
- ❑ If there exists a maximally contained rewriting for Q given a set of views then there has to exist an equivalent rewriting for a query Q using the same set of views.
- ❑ The open world assumption is the same as the closed world assumption.
- ❑ $Q_G(X) : -Person(X, Y) \supseteq Q_L(X) : -P(X, Y, Z)$ is a GAV mapping.

Question 1.4.2 (9 Points)

Rewrite the following query using the inverse rules algorithm.

$$Q(X, A, Y, B) : -G(X, A, Y, B)$$

The available views are:

$$\begin{aligned} V_1(X, Y) &: -G(X, A, Y, B) \\ V_2(X, A) &: -G(X, A, Y, B) \\ V_3(Y, B) &: -G(X, A, Y, B) \end{aligned}$$

Solution

$$\begin{aligned} Q(X, A, Y, B) &: -G(X, A, Y, B) \\ G(X, f_{A_1}(X, Y), Y, f_{B_1}(X, Y)) &: -V_1(X, Y) \\ G(X, A, f_Y(X, A), f_{B_2}(X, A)) &: -V_2(X, A) \\ G(f_X(Y, B), f_{A_2}(Y, B), Y, B) &: -V_3(Y, B) \end{aligned}$$