Name

CWID

Homework Assignment 3

April 2015

CS520

Please leave this empty!

Sum

Instructions

- Try to answer all the questions using what you have learned in class
- The assignment is not graded
- There is a theoretical and practical part
- **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!**

Lab Part

- This part of the assignment helps you to practice the techniques we have introduced in class
- In this assignment we will work with some existing tools to get experience in using the matching and mapping techniques we have discussed in class.
- We first give an overview of how to get these tools working on your machine and then discuss what tasks you should perform with these tools.

Coma 3.0

- *Coma 3.0* is a tool for schema matching that is freely available at `http://dbs.uni-leipzig.de/Research/coma.html`. It is implemented in Java
- The system requires a running MySQL server on your machine and will not start unless it is able to connect to this MySQL server
- Configuration settings are in `coma.properties`
  - You can set the JDBC connection URL as well as user and password for your MySQL server instance there
- The program comes with a `coma.bat` to start it on windows. On unix or Mac OS you need to write a shell script or call `java` directly to run it. See below for an example script.

```
export CLASSPATH="lib/coma-gui.jar:lib/coma-engine.jar\
:lib/additional/*:lib/maven/*:lib/additional/*"
java -cp ${CLASSPATH} -Xmx500M de.wdilab.coma.gui.Main
```

++Spicy

- *++Spicy* is a data exchange tool implemented in Java.
- You can download it for free from `http://www.db.unibas.it/projects/spicy/`
- It can use a backend Postgres or DB2 database or also works without a database
- The website also has several examples, e.g., `http://www.db.unibas.it/projects/spicy/software/examples.zip`

Part 2.1 Coma 3.0: (Total: 0 Points)

- Coma supports several file formats, e.g., XML schema. For instance, you can use some of the XML schemas from the ++spicy examples
- Load such files as source and target and run the matchers

Part 2.2 ++spicy: Use Examples to Understand the Tool (Total: 0 Points)

- Install ++spicy
- Download the examples mentioned above
- Startup the ++spicy GUI
- You can load existing scenarios from the examples files
- ++spicy supports all tasks needed for data exchange
  - schema matching
  - mapping generation
  - creating transformations
  - executing transformations
- test mapping generation
  - load the `personCarCity` example
  - run `generate transformations` from the `map` menu item. This creates mappings
  - inspect some of the mappings that were generated, observe how they use schema matches
- test creating transformations
  - run `generate SQL` from the `map` menu item.
  - inspect the code

Theory Part

- This part of the assignment helps you to practice the techniques we have introduced in class.

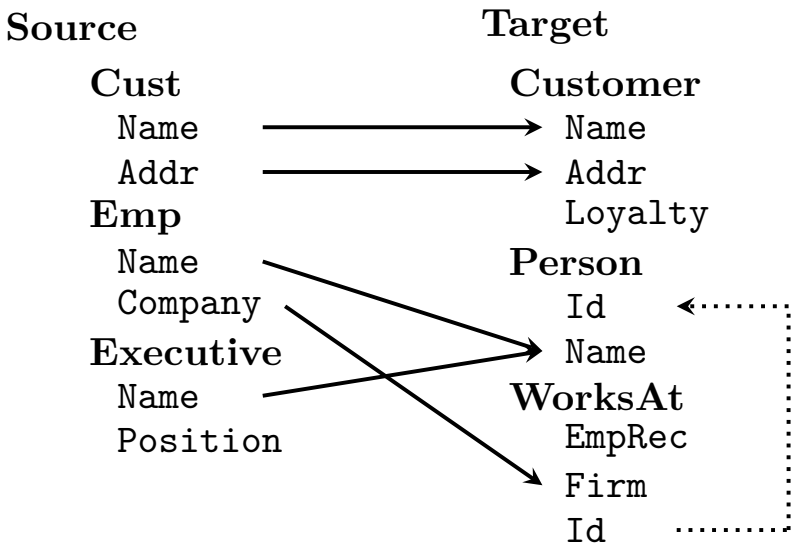
Part 2.3 Schema Matching (Total: 0 Points)

Source	Target
Company	**Brand**
CompName	BrandName
StockPrice	Headquarter
NumExpl	StockValue
Model	**Car**
ModelNumber	Color
Color	ModelSerial

Question 2.3.1 Element Name Matching (0 Points)

Consider the two schemas above and compute similarity of schema elements using a matcher that uses edit distance between element names (attributes) as similarity measure. Recall that a similarity-based matcher has to compute all pairwise similarities and uses a threshold to determine matches. Compute all pairwise similarities and consider all pairs with edit distance less than < 7 as matches

Part 2.4 Schema Mappings (Total: 0 Points)



Question 2.4.1 (0 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). Recall that 1) is solved by chasing the inclusion constraints (foreign keys) and then removing associations that are subsumed by other associations. Write down the associations first and then the mappings as source-to-target tuple-generating dependencies (st-tgds). For the second step, recall that mappings are generated by combining a source with a target association and all correspondences that are covered by these two associations. Note that Clio would remove subsumed mappings. For sake of this homework do not remove such mappings, but mark them in the result. A pair of associations is subsumed by another pair of associations if both cover the same correspondences and the relations used by the subsumed pair are a superset of the relations covered by the other pair.

Part 2.5 Virtual Data Integration (Total: 0 Points)

Question 2.5.1 (0 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.

Find maximally contained rewritings for the queries shown below. Remember that queries are rewritten using GLAV mappings by first finding a rewriting using the views over the global (target) schema using any of the algorithms for maximally contained rewritings introduced in class, then replace the global schema views with the source schema views and unfold them.

Use the bucket algorithm for the first step. Recall that the bucket algorithm groups views based on which query predicates they cover, builds all combinations of picking one view per bucket, and does a query containment check to determine whether the generated rewriting is contained in the query. In case a candidate is not contained then the algorithm will try whether it is possible to add predicates to the candidate to make it contained (if we do not consider queries and views with contained predicates then this is limited to equating variables). All contained rewritings are then combined using union to create the maximally contained rewriting for the query.

$$Q_1(X) :- Person(Y, X)$$
$$Q_2(X, Y) :- Person(Z, X), WorksAt(A, Y, Z)$$

Part 2.6 Data Exchange (Total: 0 Points)

Consider the following transportation database schema and example instance:

Cust

Name	Addr
Peter	Chicago
Bob	South Park
Alice	Chicago

Emp

Name	Company
Gert	IBM
Pferd	Oracle

Executive

Name	Position
Pferdegert	Senior VP
Heinzgert	CPO

Question 2.6.1 (0 Points)

Consider the source instance for the schema from the previous questions shown above. Use the data exchange query generation technique discussed in class to generate SQL queries implementing the mappings designed in the previous questions. Show the result of running these queries over the example source instance.