

CS 246 Exercise Set #1

In this assignment, you will explore different data formats and learn the basics of Infomaster. Your group will present your work in class on Thursday, April 17. Please also email your answers to the written assignments to mkassoff@cs and genesereth@cs by classtime (no hardcopy is required). If your group does not have a laptop, please tell the TA (mkassoff@cs) and we will figure something out. For the written questions, please be concise. You only need 1 copy of the written questions per group, but everyone should collaborate on the questions and understand the answers.

Again, brevity is a virtue for the written problems. Points will be taken off for overly wordy answers.

Exercise #1

In this exercise, you will become familiar with basic Infomaster operations.

- 1) Start Infomaster. **Load** up the University example. Note that the Macintosh loader files have the suffix `.mcl`, while Windows loader files have the suffix `.franz`. On Windows, you can use the path local to the Infomaster executable- for example `examples\university\load.franz`. On Macs, use the full path, i.e. `gullible:Infomaster:examples:university:load.mcl` if your hard drive name is `gullible`.
- 2) Go to the frontpage of the standard interface. **Seek** for Senior Computers majors who have an advisor who is a full professor. Go to the **View** page for that student by clicking on his ID in the right-hand panel. Click on the course in which that student got a C. **Displaytable** for the Gradesheet relation by clicking on its link.
- 3) Go to the **Seek** page for the manager. Find the Standard agent by browsing for agent interfaces. **View** the metadata for Standard as an interface by clicking on its ID.
- 4) Remove Thing as a rootclass. Make instead Book and Computer as rootclasses. Commit your changes by clicking on the Change button.
- 5) Note where the header HTML is stored. Open the header up in an editor (notepad, emacs, etc.) and make some personal changes to it.
- 6) Go back to the Standard **Seek** page and note your changes. **View** one of the books, and then **Displayclass** for all of the books by clicking on the Book link. Note what century the books were all published in.
- 7) *Written question 1.1:* In what way do tables and classes differ? You might want to go back to the book **View** page and click on the different attributes to invoke

the **Displaytable** operation. You also might want to look at the metadata for Class and Relation in the manager.

- 8) Using the **Transform** operation, convert the book publication years to 2 digit form (i.e. 1977 would be 77 in 2-digit form). Note that you will need to also remove the old facts from the system by concluding their negation. The **Transform** interface is an example of QBE (query by example). Checkboxes represent negation. Variables are represented by strings that start with a question mark, (i.e., ?X, ?BOOK). Constants are entered as is (i.e. sammy.squirrel, 17). Note that for arithmetic predicates, the first two arguments must be given, and the computed answer is returned in the third argument. Think of **Transform** as implementing IF *condition* THEN *conclusion*, where the conclusion is actually stored in the database.
- 9) Go back and view your changed book years.
- 10) Use the **Examine** command to view the data in your system. Enter a question mark (?) as the argument to see all of the data in your system, or type a string (i.e. standard) to see all of the data that contains that string. Note the amount of metadata in the system.
- 11) Save your work by **Dumping** it to a file. Call it exercise1.franz or exercise1.mcl.

Exercise #2

In this exercise, you will create an Infomaster instance from scratch. Revert the system back to its initial state by using the **ResetSystem** command.

Make an instance of Infomaster with the following classes and attributes:

Classes	Predicate	Attributes
Person	Person.Instance	Person.Name
Advisor	Advisor.Instance	Advisor.Student, Advisor.Institution
Student	Student.Instance	Student.Advisor
Institution	Institution.Instance	Institution.Name

Create the proper class structure with superclasses. Use the Repository as your data store. Add the proper metadata to Standard to allow for **Seeking** for instances of those classes. Edit the header, footer, and frontpage appropriately. Create a KIF file called exercise2.kif containing some instances of each class, and relate them using the attributes. Load your KIF file using **Loadfile**. Make some changes to your data using the Web interface you have made. Using **DumpAgent**, write your changed data to disk as TDC. Look at your dumped data in an editor. If you have one, a good editor would be a spreadsheet like Excel.

Save your work by **Dumping** it to a file. Call it exercise2.franz or exercise2.mcl.

Using **DumpAgent**, save your data to a file as SemanticXML. Look at the result in an editor. Also look at the file <http://logic.stanford.edu/~mkassoff/jdutree.xml>, which contains similar data for Jeff Ullman's students. *Written question 2.1:* how does the SemanticXML differ fundamentally from the Ullman XML data? What is "semantic" about it? (Hint: Try loading the jdutree.xml file into the system. Use the **Examine** command to see what was actually loaded.)

Exercise #3

Create an instance of Infomaster of a schema of your choosing. Make a front end for it. Populate it with data. Save your work as exercise3.franz or exercise3.mcl. Your instance should be on the scale of the examples provided.

Exercise #4

A few written questions:

- 4.1) Can you convert any KIF file to TDC? Explain. What about TDC to XML? XML to KIF? Explain your answers.
- 4.2) What distribution of data would stress the Infomaster system and make it run slow? Conversely, what distribution of data could Infomaster handle large amounts of well (i.e. speedily)?
- 4.3) Dump your data from Exercise 3 as KIF and as OOKIF. Open up the files in an editor and compare them. In what way are the files different? In what way are they the same? Which do you prefer? As a side note, OOKIF does not support representation of rules and n-ary relations (which your example doesn't contain anyway), while KIF does.
- 4.4) A *compound index* is an index on more than one dimension. Describe the issues that would arise when implementing compound indexing in a dataweb management system like Infomaster. Think about both updating and searching using the index. A few paragraphs should suffice here.