CMPUT 391 DATABASE MANAGEMENT SYSTEMS (LEC A1 Fa16)

Assignment 3

Here is a link to a more readable version of this assignment.

This assignment was designed to be done in pairs, but individual submissions are OK. If you work in a pair, write both names in the readme.txt file and have either you or your partner (*but not both*) submit through eClass.

You must disclose any discussion (within what is allowed by the course collaboration policies) you have with anyone except your partner, the instructor and the TAs.

NOTE: the deadline for the instructor and the TAs to answer questions about this assignment is three days prior to the submission date.

Learning Objectives

1. Get an introduction to, and practice with, RDF and SPARQL
2. Learn how to represent graph-based data (in RDF) in a relational database
3. Learn how to translate graph-based queries (in SPARQL) into SQL over a relational representation of the data

Tasks

The Semantic Web is a visionary development aimed at having web sites provide structured data with associated machine-readable semantics. There is quite a lot of activity around this initiative, including the Linked Open Data movement. In this paradigm, data is modeled as a graph encoded with RDF (Resource Description Framework) and queries are written in the SPARQL language. In Part I of the assignment, you will write SPARQL queries over one publicly available RDF store on the Web.

Every time new data models (such as RDF) and query languages (such as SPARQL) are introduced, before developing full data management systems, one good approach is to write programs to *map* the new data into a relational database and *translate* the queries in the new language into SQL over the mapped data. This is what you will do in parts II and III of the assignment.

Notes about coding: because there will be a lot of string manipulation in this assignment, you are allowed to use any of these languages in your answers: C, C++, Python, Java, awk or sed, as well as bash scripts. However, you are **not allowed** to use any third-party or non-standard library except with prior consent in writing (email) from the instructor.

Required reading:

1. RDF 1.1 Turtle: real all
2. SPARQL 1.1 overview: read all
3. SPARQL 1.1 Query Language: Sections 1, 2, 3, 4, 5, 8, 11, 17.

Further suggested reading:

- Learn RDF and Learn SPARQL from Cambridge Semantics
- RDF Primer - Turtle version from the W3C

- RDF Primer, Turtle version, from the W3C
- Querying Semantic Data, by LinkedDataTools.com

Part I: SPARQL (30 marks)

The purpose of these queries are for you to practice SPARQL and get acquainted with DBPedia, a very useful and freely available online RDF graph derived from Wikipedia.

Q0 (5 marks)

In a plain text ASCII file called q0.txt write a query to find all lakes located in the Jasper National Park.

Q1 (5 marks)

In a plain text ASCII file called q1.txt write a query to find which are the stadiums used by Italian soccer teams. Have the team and the stadium in the output. If available, output the capacity of the stadium as well.

Q2 (5 marks)

In a plain text ASCII file called q2.txt write a query to find all international airports in Canada, and the cities where they are located in.

Q3 (5 marks)

In a plain text ASCII file called q3.txt write a query to find the number of South American soccer players who (appear in Wikipedia and) have played for a club in the Spanish "La Liga" that has never been relegated, grouping by country.

Q4 (5 marks)

In a plain text ASCII file called q4.txt write a query to find the number of World Cup Final matches played by every national soccer team who has played in more than 3 finals.

Q5 (5 marks)

In a plain text ASCII file called q5.txt write a query to find, for every city in Alberta that has a hospital, the name of the city, and the ratio of population over the number of hospitals, sorted in decreasing order of this ratio.

How Your Queries Will be Graded

We will copy and paste your query into a fully working DBPedia SPARQL endpoint, such as YASGUI. Queries that have errors or produce no results when one is expected will get no marks. Full marks will be given to queries that express all the conditions in the statement of the question and produce reasonable results. Part marks will be given based on how well a query answers the question.

Part II: Storing RDF data into SQLite (30 marks)

This part of the assignment is concerned with the mapping between the graph data model of RDF into the relational model.

Your program **must check for errors** in the input RDF data. If an error is found, the program must print a message and stop. None of the triples inserted prior to the error being detected should persist in the database.

Your program must be able to properly load a valid RDF graph in the Turtle format, such as this sample: Edmonton.txt, without any errors.

Simplifying assumptions: for the sake of this assignment, you are only required to

Simplifying assumptions: for the sake of this assignment, you are only required to handle text in English (identified by the @en tag), and literals of type string, integer, float and decimal. You may, without any penalties, discard all triples with a language tag that is not @en and store literals of other types as plain strings. You do not need to store the language tag either.

Q6 (10 marks)

Design a relational schema to represent an RDF data graph. Write in a plain text ASCII file called q6.txt a short report with:

- Your relational schema and the SQLite statements to create an empty database with that schema
- A short (max 2 paragraphs) explanation of your strategy to convert the RDF data into a database according to that schema
- An example of a short (say 10-20 triples) input RDF graph and the corresponding relational database; your example should include prefix declarations.

Q7 (5 marks)

Provide in a plain text ASCII file called q7.txt the SQL commands to create any indexes you find appropriate for your RDF store. Justify your choices. If you don't think indexes are necessary, explain why.

Q8 (15 marks)

Write a program, in a file called q8.XYZ, where XYZ is the appropriate extension of your chosen programming language, that:

- reads in two parameters from the command line: (1) the name of a SQLite database to store the RDF graph and (2) a path to a file RDF triples in Turtle format
- loads the RDF triples into the database

Part III: Querying RDF data in SQLite (40 marks)

This part of the assignment is concerned with rewriting queries in a very restricted subset of SPARQL into equivalent SQL queries that will compute the answers from your mapped database.

Q9 (40 marks)

Write a program, in a file called q9.XYZ, where XYZ is the appropriate extension of your chosen programming language, that:

- reads in two parameters from the command line: (1) the name of a SQLite database file containing an RDF graph and (2) a path to a file with a SPARQL query
- prints the output of that query against that graph

For the sake of this assignment, you will deal with conjunctive SPARQL queries, without negation and without aggregations. At a minimum, your program must handle a **fully specified** query such as the one below, where the output is fully specified by variables that are used among the graph patterns.

PREFIX p:<URI>

SELECT ?v1 ?v2 ... ?vn

WHERE {

subj1 pred1 obj1 .

...

subjm predm objm

}

You can assume that:

- There can be blank lines in the query file, consisting of zero or more space characters (e.g., \s or \t)
- There can be zero or many PREFIX declarations, and that there will always be one SELECT ... WHERE statement in each query
- Every PREFIX declaration comes in a separate line and before the SELECT statement
- Every non-blank line inside the WHERE clause contains a single subj pred obj pattern or a single FILTER constraint
- Every sub, pred, obj will be one of: a variable, a fully prefixed resource (predicate or entity), or a literal.
- All literals in any query will be enclosed with double quotes, and only strings, integers, and decimal literals will be used.

If it helps, you may require that every variable in the SELECT appears in at least one pattern in the WHERE clause, and throw an error otherwise. Also, you can assume that every regex filter will be of the form: FILTER (regex(?v, "<text>")), without any other operators.

How to test your code

You can test your code with Apache Jena, which is an open source system that includes a SPARQL query processor that operates on local files. The same query file should work identically in Jena and in your system.

Create a small test RDF graph and a separate query test file for each of the kinds of SPARQL queries in the assignment. You have to submit your test graph, and you may submit your test queries if you want.

If you want to test with the sample Edmonton.txt, the query test_a3.txt returns one result: dbr:Edmonton.

Marking breakdown

- 25 marks for handling a fully specified conjunctive query in SPARQL
- 9 marks for handling * instead of a variable list in the SELECT clause of the SPARQL query
- 3 marks for handling FILTER clause with numeric constraints
- 3 marks for handling FILTER clause with regex constraints

How Your Code Will be Graded

In order for your submission to any given question get *any* marks:

1. Your code must *compute* the answer from the database created by your solution to part II (i.e., it cannot just *read* the answer from somewhere);
2. Your TA must be able to compile and execute your code without errors on a Linux workstation such as the ones in the CS instructional labs, by following only the instructions submitted with your assignment.

In order for that submission to get *full* marks:

1. Your code must produce the correct answer;
2. Your code must produce readable output (e.g., comparable to that of YASGUI);
3. Your TA must be able to *understand* your source code

Part marks will be given based on how well your submission meets the criteria above.

What and how to submit

NOTES: (1) submissions that do not comply with these instructions will not be accepted; (2) late submission penalties will apply for any re-submissions.

Write a plain text ASCII file called readme.txt in which you provide your name and that of your partner (if you worked in a pair), as well as compilation and execution instructions for your program, clearly listing all libraries that are needed to compile your code *and* any command line arguments that are needed for your programs to run. In this file, also disclose any collaborations you had while working on this assignment.

TEST that your compilation and execution instructions actually work, without any errors/warnings, on the Linux workstations in the lab.

Create an empty folder called c391f16a3_LNAME1_LNAME2 where LNAME{1,2} are your last names.

Copy your readme file, all files answering the questions above, all compilation and building files, the RDF graph you used to test your code for parts II and III, a SQLite database containing that graph into the folder.

Create a **ZIP archive** of the folder with the zip command (available on the teaching CS labs and OSX and most Linux distributions).

TEST your archive, by uncompressing it in another empty folder to make sure it produces a single folder with the requested files and nothing else.

Submission status

Attempt number	This is attempt 1 (1 attempts allowed).
Submission status	No attempt
Grading status	Not marked
Due date	Wednesday, 30 November 2016, 11:55 AM
Time remaining	28 days 18 hours
Last modified	Tuesday, 1 November 2016, 6:11 PM
Submission comments	