COMP2212 - Programming Language Concepts

STQL Language Documentation

Louis Barton (lb2g20)
Christos Mousoulides (cm10g19)

September 2, 2022

1 Language Overview

STQL is a strong statically-typed Language created for the purpose of querying simple RDF documents and serves as coursework for the module COMP2212: Programming Language Concepts of the University of Southampton.

1.1 Syntax

The Syntax of STQL was created using the lexical analyser generator library of Haskell "Alex". A very big influence for the syntax of this language was SQL, with some functions retaining the same names as their SQL counterpart like "SELECT","FROM" and "INSERT". All of the functions are written in uppercase letters, but variables can take any form. Moreover, this language does not support the creation of new functions from the developer using it, so the use of the already defined ones is mandatory to achieve the desired results. A typical STQL program will begin with the keywords "SELECT FROM", followed by the filename that contains the triples in turtle format to be filtered, then the keyword "AS" with a string specified by the user serve as a variable statement that relates to the contents of that file. After that the code varies depending on the implementation needed and it usually end in the keyword "PUT IN" followed by a String containing a filename that denotes the output file.

1.2 Grammar

The Grammar of STQL was created using the Parser generator library of Haskell "Happy".The rest of this section is dedicated to explaining the Grammar rules of STQL, but for further understanding please look at the example problems section below.

- Every program must start with the "SELECT FROM" keyword followed by a file which is a String denoting the contents of the file that is going to be used for querying.
- Every program must end with the "PUT IN" keyword followed by a file which is a String containing the name of the output file.
- Functions are evaluated using white-spaces so there's no need of using semicolons.
- Variables can be used to represent a file as a single object or an object representing operations between two or more files, operations included are "UNION", "INTERSECTION" and "DISJOINT".
- Variables followed by a "." can be used to access the predicate,subject or object contained in triples that reside in the file or file Operation described above and follow this format: "variableName"."tripleItem", where triple item can be "SUBJECT","PREDICATE" or "OBJECT" and their plural if multiple items wish to be selected.
- The "WHERE" and "INSERT" keywords followed by the rule explained above are used to perform filtering and arithmetic operations on the triples of the input files. Moreover, these functions can be used 0 or more times and they must be located right after the declaration of variables and before the contents of the second rule.
- Variables must always be declared after the contents of the first rule with the keyword "AS" followed by the desired variable name.
- Subjects can only take string values.
- Likewise Predicates can only take String values as well.
- Objects on the other hand can take numerical values, Strings and Booleans.
- With Objects you can also equate the object to an Integer, a String or a URI and this selects all objects of that certain type for the specified file. You type this out as: "fileVariable".Objects = URL, where URL can also be replaced with String or Int.
- The scope of STQL's variables is single layered, thus all variables are global.
- Triples can be inserted directly into the program using the following format:
  <subjectString> <predicateString> <objectString>
  where the objectString with its angle brackets can be substituted with just a String or just an Integer.
- Comments must always begin and end with a pair of backslashes.

2 Execution Model

The first thing that the execution model does is read the input file given that contains the STQL code and store its filename. After that the file contents are read and the tokens are scanned by the lexer. With the tokens having been scanned they are passed to the grammar tool to be parsed. Now the parsed input is passed to the interpreter. The interpreter spots patterns in the parsed input and when it matches one it moves the system state to the next one with a new configuration, otherwise it throws a run-time error. The program stops processing expressions when the "PUT IN" keyword with the output file is evaluated. Then the program checks for the formatting of the triples in both the input and the output to ensure the correctness of the input and output respectively. After that, the output containing triples is printed into stdout (Standard out) and can also be found in the output file specified by the programmer. Moreover, it's notable that comments are completely ignored by the interpreter, so they can be used whenever in the code given that they have the correct structure.

3 Error Messages

The Error Messages of this language are printed in stderr (Standard Error). There are overall three types of Error Messages that can be found in this Language that are Compiler Errors, Run-time Errors and Validation Errors regarding the layout of the RDF formatted input and output files, with the latter being a subset of Run-time Errors.

3.1 Compiler Errors

The following types of Errors are displayed during compile time of an STQL program:

- The first type of compile-error is thrown by the lexer and its thrown whenever the user is trying to input an unknown token that the lexer cannot tokenize.
- The second type of compile-error is thrown by the parser and its thrown whenever the user is trying to input the tokens of STQL in the correct order, or if the user is trying to input an item of a different type unsupported by the Grammar of this language.

3.2 Run-time Errors

The Run-time Errors that are thrown in STQL are the following:

- Referencing a Variable that doesn't exist.
- Referencing a Subject or a Predicate or an Object that do not exist.

- Trying to perform an Arithmetic operation without using an Integer.
- Trying to use a file that doesn't exist as input.
- Trying to pass an RDF file with incorrect format into the program.(This one is the validation Error Mentioned Above).

4 Additional Features

Several Additional Features have been developed to improve the quality of life of the programmer using this language. The following is a list that explains each one of them in detail:

- Users can input Comments whenever they want in their program with the use of double backslashes at the beginning and end of the string they wish to leave as a comment.
- RDF File Validation as mentioned in the section above as well has been developed to help the user distinguish if his input and output files are in correct Turtle Format, so that the document is correct and can be used in other Parsing Languages without any issues.
- Parenthesis can be used when dealing with Boolean and Arithmetic Operations to make the visibility of the code better.

5 STQL Semantics with Examples

In this section i will go through a few examples that will encapsulate everything the at the programmer can do with this Language.

5.1 Conjunction

Given two files merge their triples into one file:

```
1 SELECT FROM "foo.ttl" UNION "bar.ttl" AS foobar
2 PUT IN "file.txt"
```

Here the first line of the code selects the first file and performs the "UNION" operation with the second file to create a variable out of the union of both files. After that the contents of these two files is output in "file.txt" and stdout. Also, if the user wants to perform another operation the "INTERSECTION" and "DISJOINT" functions can replace "UNION" if they are needed.

5.2 Pattern Matching and Commenting

Given one input file output all of its triples with subject: "http://www.cw.org/example" or predicate: "http://www.cw.org/problem3/examplePred" and object equal to "exampleString":

```
1  //This is a Comment//
2  SELECT FROM "foo.ttl" AS foo
3  //Pattern Matching is Fun! :) //
4
5  WHERE foo.SUBJECT = "http://www.cw.org/#exampleSub"
6     OR foo.PREDICATE = "http://www.cw.org/example/#exPred"
7     OR foo.OBJECT = "exampleString" //object matching //
8
9  PUT IN "output.txt"
```

As you can see here comments can be used anywhere in the code without any negative impact. Moreover, you can perform pattern matching on strings or alternatively you can pattern match with numbers.

5.3 Inserting items and adding Objects

Given two input files the output should contain all of the triples of the first file where Obj appears as a subject of a triple in the second file. Additionally, whenever the object of the first file is between 0 and 99 insert the full triple in the output file but with its object being added by one and another triple confirming that the number is in range below it:

```
1  SELECT FROM "first.ttl" AS fst, "second.ttl" AS snd
2  INSERT
3       fst
4  WHERE
5       snd.SUBJECTS CONTAINS fst.OBJECT
6  INSERT
7        <fst.SUBJECT> <fst.PREDICATE> fst.OBJECT + 1,
8        <fst.SUBJECT> <"http://www.cw.org/ex/#inRange"> true
9  WHERE
10      fst.OBJECT > -1 AND fst.OBJECT < 100
11
12 PUT IN "output.txt"
```

In this example the keyword "CONTAINS" is used as a condition to denote that the triples to be selected from the first file should have the same object as the subject of the second file, and the keyword "SUBJECTS" is used as all of the subjects of that file need to be in the output. In other cases the keywords "OBJECTS" and "PREDICATES" can be used with the same effect but for different items. Moreover, after the first insert a comma is used to denote that more than one line of triples are gonna be put in one place, so that the user can insert as many triples as he wishes at any given line of the output.