IMPLEMENTATION SO FAR

ANDRIANA GKANIATSOU

1. Overview

The system, until now, provides the following functionalities: (i) Take as input a predicate and translate it into an SQL or SPARQL query. (ii) Query the appropriate dataset. (iii) Check whether the query retuned results. (iv) Extract the dataset schema. Due to restricted access to PostgreSQL (databases), we have only tested SPARQL queries.

The system, presented in Figure 1, is decomposed to the translation system, and to the query system

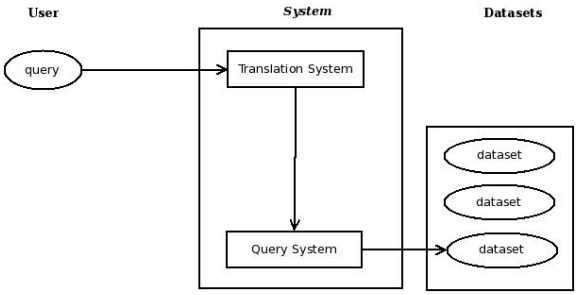

Figure 1. Overview of the System

2. Translation System

`translation.sh` which is a bash script, and `translation.pl` which a Prolog program, constitute the translation system. In more detail:

(1) `Translation.sh` is a bash script, which takes as input a predicate. It converts to lower case the appropriate characters, and then converts the query into a list. The outcome of this procedure is a prolog file that contains a list of the form [H | T] where H is the Predicate Name, and T are the arguments of the predicate [1].

(2) `Translation.pl` translates the list into an SQL or SPARQL query (on demand). This system takes as input (i) the list which contains the name of the predicate and the arguments, (ii) the types of the arguments, (iii) The appropriate prefixes (SPARQL). The outcome of the query is written into an external file.

[1]We are unaware of the incoming query *e.g.* we do not know the number of arguments the query has. So, we decided to convert it into a list, because it is easier to handle such cases

An overview of the translation system is presented in Figure 2.

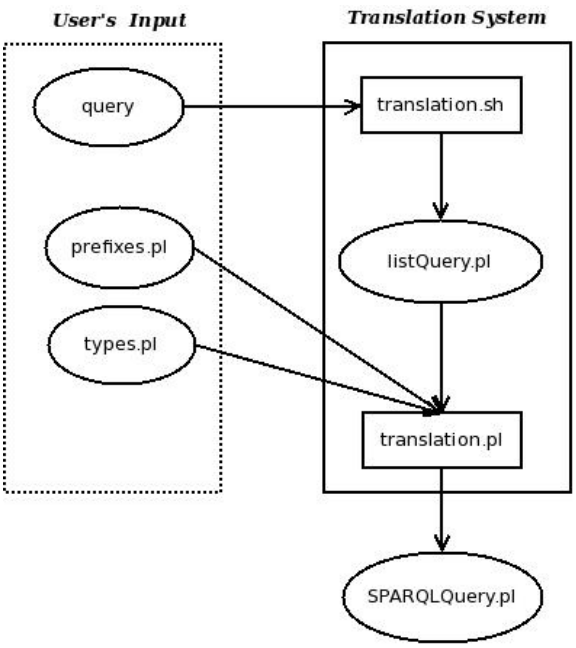

FIGURE 2. Overview of the Translation System

Translation Process SQL translation is similar to SPARQL. First step is to find the name of the table (SQL) or the name of the dataset (SPARQL), which is the head of the incoming list. Next step is to distinguish the variables from the constants.

A variable is considered to have the same name as its type *i.e.* the class that this variable belongs to. In this way, a variable denotes a SQL column or a SPARQL predicate. Constants denote the constraints of the query.

2.1. **SELECT Statement.** SQL: We place all the variables in the `SELECT` statement. These variables denote the columns (types) that we are looking for.

SPARQL: `SELECT` statement consists of variables whose names are independent from their types (or predicates in RDF).

2.2. **FROM Statement.** SQL/SPARQL: The name of the table (SQL) or the dataset (SPARQL) is the predicate name.

2.3. **WHERE Statement.** SQL: `WHERE` statement consists of constants (if we have any) and their types. For instance if the incoming query is

bathingWaters(highland, WaterBodyName)

and we know that:

class(highland, location)

then we have:
`WHERE` location='highland'

SPARQL: `WHERE` statement consists of the subject of the query, the predicates, the variables and the constants (if we have any). Variables have the same name as the ones in `SELECT`. The predicate names are the types of that variables. In addition, we add the appropriate prefixes. For instance consider the incoming query:
bathingWaters(highland, WaterBodyName, sepaidw:200168)
and the types:
class(highland, location)
class(WaterBodyName, waterBodyName)
prefix(sepaw, location)
prefix(sepaw, waterBodyName)
then we have:
`WHERE` {
sepaidw:200168 sepaw:waterBodyName ?waterBodyName ;
sepaw:location 'highland'.}

We have implemented two ways for finding the subject of the predicates: (i) We assume that the subject is given by the user, (ii) We do not have any information of the subject, so, we randomly choose a constant. In case of no constants, we randomly choose a variable.

3. Query System

Query system is responsible for sending the query to the appropriate dataset. Then, it checks the results of the query: if the query did not return any results, then it extracts the schema of the dataset. The functionalities of the query system are presented in Figure 3.

4. Discussion

4.1. **Implementation and Assumptions.** This implementation works under certain assumptions. Firstly, we assume that we know the appropriate prefixes for a SPARQL query and the types of the arguments (these might be wrong). We also assume, that a variable will have the same name as its type. Finally, we assume that we have permission to extract the dataset schema.

4.2. **Repair.** Repair is failure driven: if the query fails to return results, then we repair it until we get some. However, a query might not return results not only because it has errors, but also because there is not any data that satisfies the query. SO, how do we distinguish these two cases?

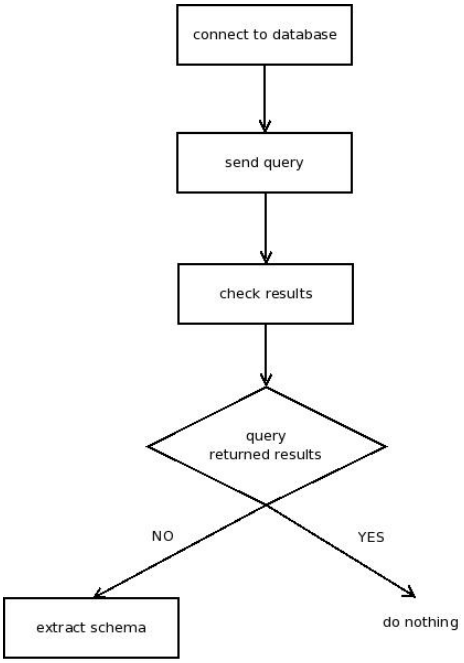

FIGURE 3. Functionalities of the Query System System