International Journal of Computer Applications Technology and Research
Volume 6–Issue 3, 130-132, 2017, ISSN:-2319–8656

An Extension of RETRO Framework: Translating SQL Insert, Update and Delete Queries to SPARQL UPDATE

Abatal Ahmed
University Hassan,
Faculty of Science
and Technologies,
Settat, Morocco

Alaoui Khadija
University Hassan,
Faculty of Science
and Technologies,
Settat, Morocco

Larbi Alaoui
International
University of Rabat,
Sala Al Jadida,
Morocco

Bahaj Mohamed
University Hassan,
Faculty of Science
and Technologies,
Settat, Morocco

Abstract—in this paper we propose a set of algorithms to improve the functioning of RETRO framework, these algorithms will convert SQL queries like INSERT, UPDATE and DELETE to the equivalent queries SPARQL Update.

Keywords—*RDB, RDF, SQL, SPARQL, Query Translation*

1. INTRODUCTION

RDF (Resource Description Framework) which was standardized by the W3C is a language for describing the semantics of data that allows sharing of its meaning between different applications. RDF provides a powerful data model based on representing data in RDF graphs that can be queried using SPARQL. SPARQL (SPARQL Protocol and RDF Query Language) was proposed and standardized by W3C as a query language for RDF.

2. RETRO FRAMEWORK

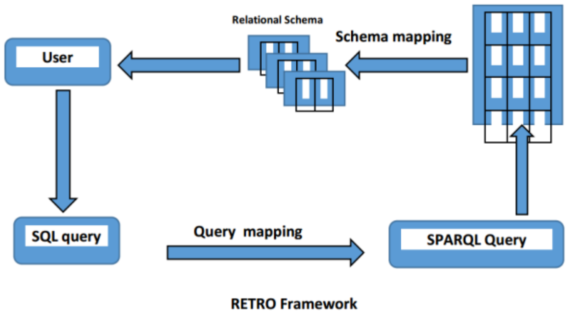
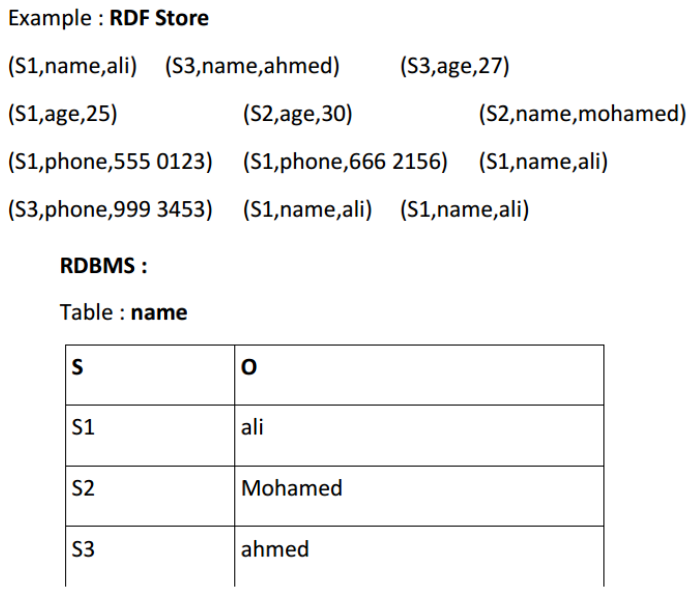


Figure 1. RETRO Schema and Query Mapping

RETRO Framework is to conserve and translate SQL queries into SPARQL. This framework is composed of several algorithms.

The first *schema-mapping ()* algorithm is to convert an RDF store RDB, this algorithm uses the property table method (for every single predicate it creates a table with two columns (subject and object). This algorithm returns a map P representing the relational schema for users, this relational schema is generated by extracting each predicate name from the map P, S and O are added to each predicate.



Example : **RDF Store**

(S1,name,ali) (S3,name,ahmed) (S3,age,27)

(S1,age,25) (S2,age,30) (S2,name,mohamed)

(S1,phone,555 0123) (S1,phone,666 2156) (S1,name,ali)

(S3,phone,999 3453) (S1,name,ali) (S1,name,ali)

RDBMS :

Table : **name**

S	O
S1	ali
S2	Mohamed
S3	ahmed

Figure. 2 Example of an RDB Table

Before starting to translate queries, RETRO separates SQL clauses, so that an algorithm as specified must convert each clause.

The *TransSqlFromClause ()* algorithm that will return triple patterns in the SQL FROM clause. Then *TransSqlWhereClause ()* algorithm that will translate into SPARQL WHERE clause WHERE clause, using pattern returned by *TransSqlFromClause ()*. After the algorithm Trans SQL Select clause () also translate the SQL select clause in select SPARQL.

Then a main algorithm *query-mapping ()* that brings together the outputs of these sub-programs, to combine and give a query SPARQL Equivalent to the SQL.

3. RETRO UPDATE QUERIES

In this section we give an extension to the Framework retro, our proposal is to convert SQL queries type INSERT / DELETE / UPDATE to equivalent SPARQL UPDATE queries.

Our algorithms are based on the *query-mapping ()* algorithm [1] *query-mapping ()* is to convert an RDF STORE RDB, this algorithm property table method, for every single predicate it creates a table with 2 columns (subject and object).

SQL Query	**SPARQL Query**

delete from name where subject='S1'	DELETE DATA {
	's1' name o
	}

delete from name where object='Mohammed'	DELETE DATA {
	s name 'mohammed'
	}

3.1 Insert Query

To insert the table name in the query is executed

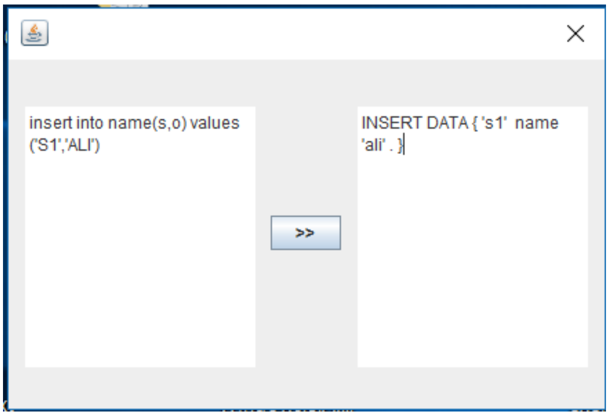
Insert into name (subject, object) values ('S1', 'Ali').

Our algorithm *transSqlInsert ()*, receives the SQL query string form, and it will extract the triple patterns to generate an equivalent SPARQL INSERT

Therefore, the algorithm takes the table 'name' as a predicate for the new triple pattern and the values corresponding to the two columns (subject, object) subject and object to have a triple subject predicate object.

INSERT DATA {'S1' name 'Ali' }

- The algorithm uses the method *splitQuery ()* to cut the SQL query and extract the name of the table matches the predicate, the values of subject and object.
- Then do call the *generate ()* method that will deduct the triple pattern from the result *splitQuery ()* and creating an equivalent SPARQL INSERT query.



insert into name(s,o) values ('S1','ALI')

INSERT DATA { 's1' name 'ali' . }

Figure. 3 Example of an SQL Insert converted

3.2 Delete Query

To delete from the table name it executes the query

Delete from name Where subject = 'S1';

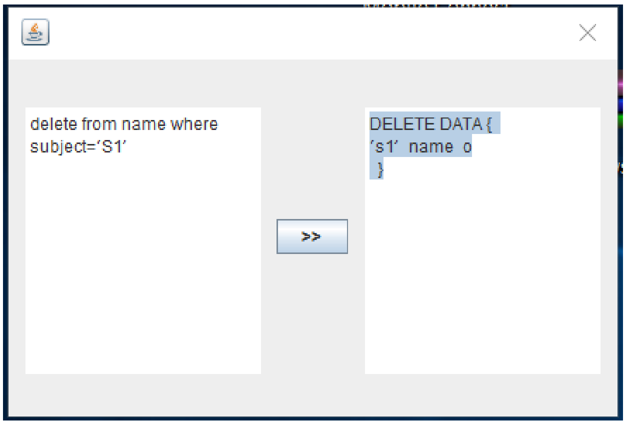
TransSqlDelete () receives the SQL query string form, and it will extract the triple pattern to generate an equivalent SPARQL DELETE.

ALGORITHM: ***TRANSSQLDELETE ()***

Input	Q an SQL Insert Query
Output	Z a SPARQL Delete Query

1
2
3
4
5
6
7
8
	p<= ''
	tps<=null
	o<= ''
	q=splitQuery(Q) ;
	p=q.getPredicatValue() ;
	tps=q.getTriples() ;
	Z =generateQuery(p,tps) ;
	Return Z ;

This algorithm begins by extracting the name of the table representing the predicate of GDI drawing propose, and extract the Boolean condition in the WHERE clause to build a triple pattern is using *spliteQuery ()* method, which receives as a parameter the SQL DELETE as a string. After extracting the predicate for the request, the method *generateQuery (p, tps)* reference SPARQL query Delete, from the extracted variable (p predicate and list of pattern triples).



```
delete from name where
subject='S1'
```

```
DELETE DATA {
's1'  name  o
 }
```

Figure. 4 Example of an SQL Delete converted

3.3 Update Query

In the SPARQL language there is no equivalence of the update query, so to execute an update query, you must combine the two DELETE queries then INSERT, but you must keep the delete values to re-insert them in the RDF without loss of data

4. CONCLUSION

In this paper, we described an extension of RETRO framework, this extension is in the form of n set of algorithms to convert SQL queries INSERT, DELETE and UPDATE to SPARQL Update, and these algorithms are implemented and tested by java language. Another promising avenue for future work is to implement a framework for translating queries for other storing methods.

4. REFERENCES

[1] Jyothsna Rachapalli, Vaibhav Khadilkar, Murat Kantarcioglu and Bhavani Thuraisingham, "RETRO: A Framework for Semantics Preserving SQL-to-SPARQL Translation", EvoDyn Workshop, 2011.

[2] Bajda-Pawlikowski, K.: Querying RDF data stored in DBMS: SPARQL to SQL Conversion. Technical Report TR-1409, Yale Computer Science Department, USA

[3] A. Seaborne, G. Manjunath: SPARQL/Update: A language for updating RDF graphs. External HPL-2007-102, HP Labs Bristol (2007).

[4] W3C, Resource description framework (RDF): concepts and abstract syntax, in: G. Klyne, J.J. Carroll, B. McBride (Eds.), and W3C Recommendation [S], 10 February2004. http://www.w3.org/TR/2004/REC-rdf-concepts- 20040210