RDF Data Browser

Project Members:

Angelique E Moscicki
Sergio Herrero
Oshani Seneviratne

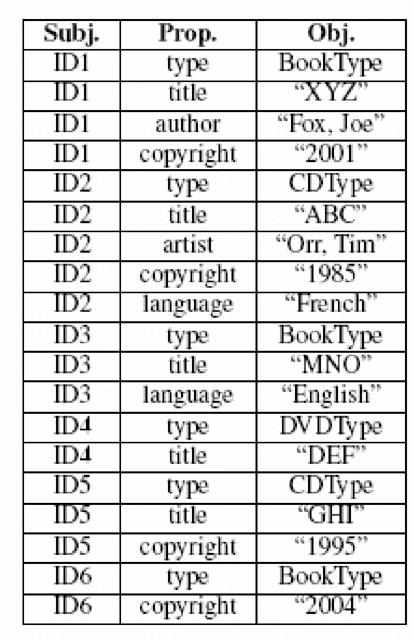
Problem Statement

The problem that we are seeking to address in our final project is that of finding a way to translate an RDF store into a relational database in such a way that supports both fast queries and efficient storage. It is not efficient to represent RDF data directly because the triples are structured as a graph and it can be difficult to construct queries that extract useful information from them. Translating an RDF store into a relational database imposes an organization more amenable to queries.

Some solutions that have been proposed include transforming the RDF store into property tables, and vertically partitioning the store into a set of two-column tables. We would like to explore the first solution in more detail. The problem with property tables is that usually the data in the RDF store is not structured; there are very few integrity constraints, so it is difficult to determine which set of properties are associated with a given object. The result is an all-inclusive approach which can give rise to an unacceptable number of NULLs in the monolithic property table. This disadvantage must be balanced against the consideration that if too many properties are treated as outliers, and stored in other auxiliary property tables, the entire system loses its ability to quickly answer queries because many joins are required to execute interesting queries. It is our task to find an optimal wide representation of the RDF store, as well as present to the user a browser for navigating the relational schema we have generated.

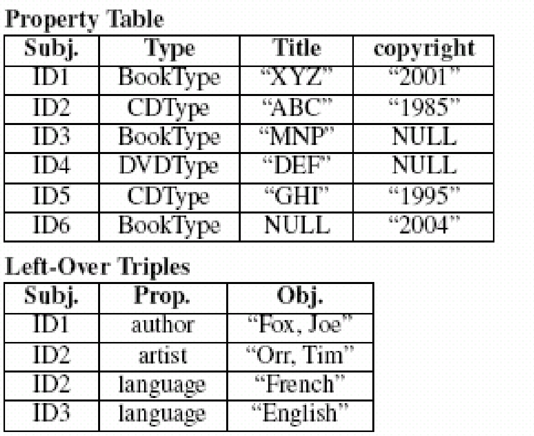
Proposed Solution:

The goal is to design and implement a clustering algorithm that converts RDF data of the general form <subject, property, object> into a relational model. The process denormalizes RDF tables by physically storing them into one or more flattened relational schemas. Table 1 is the general form of representation of RDF data organized in triples. Table 2 shows a possible flattened representation of the RDF data in a relational schema. It can be seen that some NULL values are generated. Left-Over Triples are kept aside since their addition to the property table would have generated several more NULLs.



Subj.	Prop.	Obj.
ID1	type	BookType
ID1	title	"XYZ"
ID1	author	"Fox, Joe"
ID1	copyright	"2001"
ID2	type	CDType
ID2	title	"ABC"
ID2	artist	"Orr, Tim"
ID2	copyright	"1985"
ID2	language	"French"
ID3	type	BookType
ID3	title	"MNO"
ID3	language	"English"
ID4	type	DVDType
ID4	title	"DEF"
ID5	type	CDType
ID5	title	"GHI"
ID5	copyright	"1995"
ID6	type	BookType
ID6	copyright	"2004"

Table 1.- RDF triple set [1]



Property Table

Subj.	Type	Title	copyright
ID1	BookType	"XYZ"	"2001"
ID2	CDType	"ABC"	"1985"
ID3	BookType	"MNP"	NULL
ID4	DVDType	"DEF"	NULL
ID5	CDType	"GHI"	"1995"
ID6	BookType	NULL	"2004"

Left-Over Triples

Subj.	Prop.	Obj.
ID1	author	"Fox, Joe"
ID2	artist	"Orr, Tim"
ID2	language	"French"
ID3	language	"English"

Table 2.-Wide representation of the RDF data [1]

Our algorithm addresses the NULL generation problem stated above. It will partition a set of triples into groups such that triples within a group are similar to each other. The output would be 'k' groups such that some criterion that evaluates the clustering quality is optimized. The selected criterion for our algorithm is to find the wide representation of RDF data that minimizes the number of NULLs.

The algorithm also has to make sure that is scalable if the running time grows linearly in proportion to the size of the triple set and the available system resources.

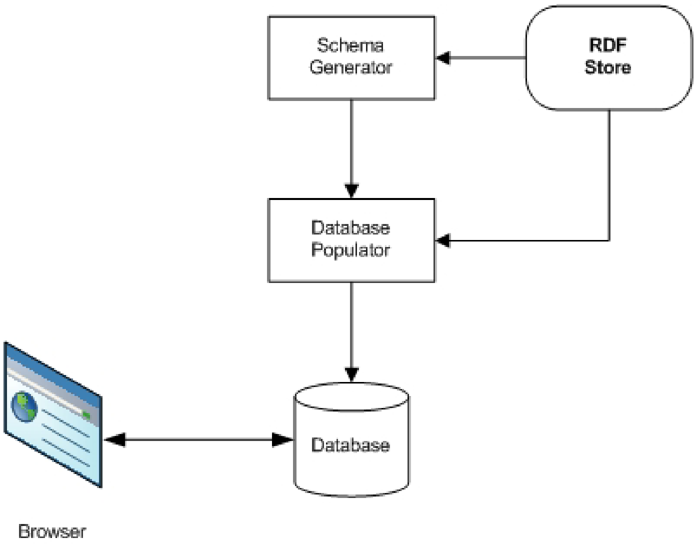
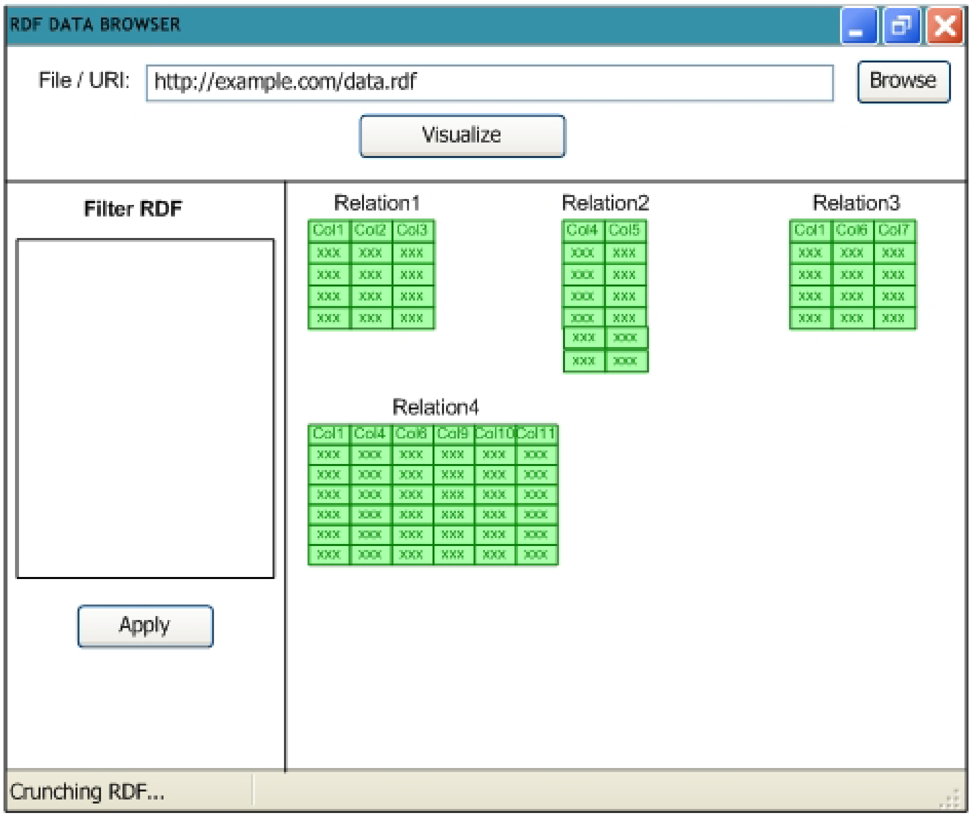


Figure 1: Architecture of the proposed solution

Given an RDF store, our system will use the Schema Generator to determine the optimal way of representing the data, and pass the result to a module which will read the RDF store and insert the triples into their places in the resulting property tables. The Browser module will then be able to execute queries on the improved RDF representation.

Deliverables:

- An efficient clustering algorithm which will transform the RDF data which is in the general form of <subject, predicate, object> into a more useful wide relational schema.
- A Graphical User Interface which allows visualizing of the RDF data in the best relational representation as determined by the clustering algorithm (*Figure-1*): The input would be either an RDF file found in the local file system or a Uniform Resource Identifier (URI). In addition to visualizing the raw RDF data, there would also be an option to run an SQL query on the relational data which is generated from the RDF.
- Documentation: This will include a comprehensive User Manual and Developer Notes.



File / URI: http://example.com/data.rdf

Relation1

Col1	Col2	Col3
xxx	xxx	xxx
xxx	xxx	xxx
xxx	xxx	xxx
xxx	xxx	xxx

Relation2

Col4	Col5
xxx	xxx
xxx	xxx
xxx	xxx
xxx	xxx
xxx	xxx
xxx	xxx

Relation3

Col1	Col6	Col7
xxx	xxx	xxx
xxx	xxx	xxx
xxx	xxx	xxx
xxx	xxx	xxx

Relation4

Col1	Col4	Col8	Col9	Col10	Col11
xxx	xxx	xxx	xxx	xxx	xxx
xxx	xxx	xxx	xxx	xxx	xxx
xxx	xxx	xxx	xxx	xxx	xxx
xxx	xxx	xxx	xxx	xxx	xxx
xxx	xxx	xxx	xxx	xxx	xxx
xxx	xxx	xxx	xxx	xxx	xxx

Crunching RDF...

Figure 2: GUI of the RDF Data Browser

References:

[1] D.J. Abadi, A.Marcus, S.R. Madden, K.Hollenbach. Scalable Sematic Web Data Management Using Vertical Partitioning. VLDB 07.