Scalable Reconstruction of RDF-archived Relational Databases

Silvia Stefanova
Department of Information Technology
Uppsala University
Silvia.Stefanova@it.uu.se

Tore Risch
Department of Information Technology
Uppsala University
Tore.Risch@it.uu.se

ABSTRACT
We have investigated approaches for scalable reconstruction of relational databases (RDBs) archived as RDF files. An archived RDB is reconstructed from a data archive file and a schema archive file, both in N-Triples formats. The archives contain RDF triples representing the archived relational data content and the relational schema describing the content, respectively. When an archived RDB is to be reconstructed, the schema archive is first read to automatically create the RDB schema using a schema reconstruction algorithm which identifies RDB elements by queries to the schema archive. The RDB thus created is then populated by reading the data archive. To populate the RDB we have developed two approaches, the naive Insert Attribute Value (IAV) and Triple Bulk Load (TBL). With the IAV approach the data is populated by stored procedures that execute SQL INSERT or UPDATE statements to insert attribute values in the RDB tables. In the more complex TBL approach the database is populated by bulk loading CSV files generated by sorting the data archive triples joined with schema information. Our experiments show that the TBL approach is substantially faster than the IAV approach.

Categories and Subject Descriptors
H.2.4 [Database Management]: Systems, H.3.4 [Systems and Software]: Performance evaluation

General Terms
Algorithms, Measurement, Performance

Keywords
Preservation of relational databases, reconstruct relational database from RDF, RDF-archive, schema archive, data archive

1. INTRODUCTION
The importance of digital preservation of scientific data and databases has been increased for the past ten-fifteen years [4][5]**Errore. L'origine riferimento non è stata trovata.**[12][13]. By preserving selected subsets of both data and publications within one digital object, future reuse, verification, and heritage [3] of the published scientific results can be guaranteed.

Semantic Web technologies and in particular RDF seems promising for long-term preservation of databases since it provides a neutral format for representation of data and information. For selective long-term preservation and reconstruction of relational databases (RDBs) in terms of RDF we have developed the SAQ (Semantic Archive and Query) system [9]. The approach is suitable for archiving scientific data used in scientific publications where it is desirable to preserve only selected parts of an RDB, e.g. only data about a specific set of artifacts in the database related to some publications [5]. To select the parts of an RDB to archive as RDF, a SAQ user defines an *archival query* to an RDF view of the RDB called the *RD-view*. An archival query is defined in an extended SPARQL query language, A-SPARQL [9] where the selected parts of the RDB to archive are specified using a general SPARQL-like query to the RD-view. An archival query produces a *data archive* file in N-Triples format representing the relational data content to be preserved. It also generates a *schema archive* file in N-Triples format, where sufficient meta-data is saved to be able to fully reconstruct the archived database.

The focus of this paper is the *reloader* module of SAQ which reconstructs an RDF-archived RDB in order to make it live again. When the contents of an archived RDB is to be reconstructed, the reloader first reads the schema archive and executes a *schema recreation algorithm* to automatically construct the minimal RDB schema required for the archived data. Since only selected parts of the RDB are archived, a corresponding partial RDB, *a reconstructed RDB* is created containing only the relevant parts of the schema. The RDB thus created is then populated by reading the data archive and converting the read data into relational attribute values according to the schema.

To populate the reconstructed RDB we have developed two approaches called *Insert Attribute Value* (IAV) and *Triple Bulk Load* (TBL), respectively. With the naive IAV approach the relational data is populated by the generated stored procedures that execute SQL INSERT or UPDATE statements to incrementally insert attribute values reading and converting the triples in the data archive file. With the TBL approach the data is instead populated by using the bulk load facility of the RDBMS which requires one CSV file of rows for each reconstructed table. Since RDF does not prescribe any specific triple order the reloader needs to sort the triples in the data archive per table row so that a CSV file per table can be generated in one pass with limited memory. We compare the performance for the two approaches to reconstruct an RDB and show that the TBL approach is substantially faster than IAV despite the added sorting and post-processing.

The rest of this paper is organized as follows. Section 2 presents a usage scenario, and Section 3 presents the reloader and the two approaches. Section 4 evaluates the performance of the population approaches. Section 5 describes related work, and Section 6 provides a summary.

2. USAGE SCENARIO

Fig. 1 shows a small RDB called *bdb,* which contains parts of the relational Berlin benchmark dataset [2]. The database has four tables *product*, *productfeature*, *productfeatureproduct,* and *producer*, populated with some data. The columns *pno*, *pfno,* and *prodno* are primary keys in the tables *product*, *productfeature*, and *producer*. The column *producer* in the table *product* references the column *prodno* in the table *producer* as foreign key. The table *productfeatureproduct* is a many-to-many link table between the tables *product* and *productfeature*.

A SAQ user has preserved data about products having *pNum1 > 300* by executing the following A-SPARQL query to the RDF

Table ***product***

pno	pNum1	producer
1	100	2
2	450	3

Table ***productfeature***

pfno	publishDate
3	2000-06-22
4	2000-07-08

Table ***productfeatureproduct***

product	productFeature
2	3
2	4

Table ***producer***

prodno	country	label
2	DE	NULL
3	SE	Prod3

Figure 1. RDB *bdb*

view named <*bdb*> representing the RDB *bdb*:

```
ARCHIVE AS 'data.nt', 'schema.nt'
FROM <bdb>
TRIPLES
{?product    ?property           ?value      }
WHERE
{?product    rdf:type            bdb:product .
 ?product    bdb:product_pNum1   ?pNum1 .
 FILTER      (pNum1  > 300 )                  }
```

The execution of the above archival query produces two NT-triples files:

1. The *data archive* file, *'data.nt',* shown in Fig. 2 containing the archived products' date.
2. The schema archive file, *'schema.nt'*, containing the schema information required for recreating the parts of the RDB schema describing the archived data.

When the reloader reconstructs the archived RDB the schema archive *'schema.nt'* is first read and the RDB schema of the reconstructed RDB, named *r_bdb* in Fig. 3 is created. It contains only the parts of the tables of the RDB in Fig. 1 archived by the archival query. Then the RDB is populated by reading the data archive *'data.nt'*.

3. THE RELOADER

Fig. 4 illustrates the overall architecture of the reloader. When an RDB is restored from an *archive repository*, the *schema restorer* first imports the *schema archive* triples and stores them in a triple table called the *imported schema* inside the reloader. The

```
<bdb:product/2>   <bdb:product_pNum1>   "450"^^<xsd:int>
<bdb:product/2>   <bdb:product_pno>     "2"^^<xsd:int> .
<bdb:product/2>   <bdb:producer_of_product>  <bdb:producer/3> .
<bdb:product/2>   <bdb:product_Feature>  <bdb:productfeature/_3>
<bdb:product/2>   <bdb:product_Feature>  <bdb:productfeature/_4>
<bdb:productfeature/_3><bdb:productfeature_pfno> "3"^^<xsd:int> .
<bdb:productfeature/_4><bdb:productfeature_pfno> "4"^^<xsd:int> .
```

Figure 2. Data archive *'data.nt'*

Table product

pno	pNum1	producer
2	450	3

Table productfeature

pfno
3
4

Table ***productfeatureproduct***

product	productFeature
2	3
2	4

Figure 3. The reconstructed RDB *r_bdb*

imported schema represents relational concepts (tables, attributes, foreign and primary keys) for the reconstructed RDB as RDF/RDFS concepts (classes, properties, domains, ranges, and sequences) into. The schema restorer executes a *schema reconstruction algorithm* to convert the imported schema into a relational schema of the *reconstructed RDB* in the *destination RDBMS* by sending SQL CREATE TABLE statements through the *JDBC interface*. Once the schema is created, the reconstructed RDB is populated with the data in the data archive by the *data restorer* using either IAV or TBL approaches. This involves converting data archive triples into relational attribute values. The conversion is based on the W3C recommendations [1] for direct mapping of relational data to RDF.

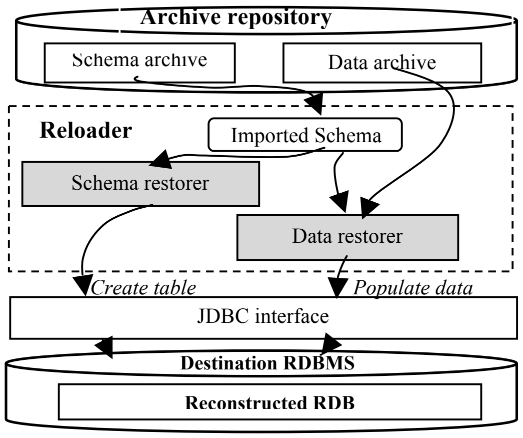

Figure 4. Reloader architecture

3.1 Data restorer using IAV approach

With the naive IAV approach the data is populated by stored procedures automatically generated for each reconstructed table attribute. A system table maps an URI representing a table attribute to its corresponding stored procedure. During the data restoring the data archive is read and the reconstructed RDB is populated triple-by-triple by looking up the system table and calling the stored procedure.

3.2 Data restorer using TBL approach

The reconstructed RDB is populated by using the bulk load facility of the destination RDBMS, which requires one CSV file of rows per table to restore. Since RDF does not prescribe any specific triple order, the triples in the data archive are sorted so that the CSV files can be generated in one pass.

To do the sorting all the unsorted data archive triples are first bulk loaded into an RDB table *Triples(s, p, o)*. Another RDB table *Attributes(file, attributeURI, pos)* is populated with meta-data about the archived tables' attributes and their positions, which is needed to produce the *sorted data archive*. The table *Attributes* has columns: *file* storing CSV file names for each reconstructed table; *attributeURI* storing the URIs representing each reconstructed tables' attribute which is the primary key; and *pos* storing the row position of an attribute represented by *attributeURI,* enumerated 0 and up. Given these two tables, the sorting is done by running the *sorting query*:

```
SELECT a.file, t.s, a.pos, t.o
FROM Triples t, Attributes a
WHERE t.p = a.attributeURI
ORDER BY a.file, t.s, a.pos   .
```

The query sorts the data archive triples per CSV file for each table to populate, per triple subject, and per attribute order in the table. Since the subject URI is prefixed by the table name, ordering data triples by their subjects implies grouping them by table rows. This is because the URIs encode both the table name and primary key value(s) of the rows in the archived RDB as a concatenation of the name of the table with a string of key values, as Fig. 2 exemplifies.

The result of the query is converted to one CSV file per table by running a *CSV generation algorithm* applied on each result tuple *(file, s, pos, o)* from the result scan of the sorting query. The algorithm produces the CSV files in one pass with limited memory.

The details of the two approaches can be found in http://www.it.uu.se/research/group/udbl/software/sard/TReport_Reloader_Stefanova_Risch.pdf.

4. PERFORMANCE

The performance of the reloader to populate a reconstructed RDB was compared for the two approaches IAV and TBL.

The experiment configuration was the following:

1) The measurements were made on a PC with Intel(R) Core(TM), i5 CPU with 2.67 GHz and 8 GB RAM running 64-bits Windows 7 Professional.
2) MS SQL server 2008 R2 configured with 6 GB buffer pool was used for the reconstruction.
3) The schema and data file archives were produced by archiving as RDF the entire relational databases generated by the Berlin benchmark data generator. The size of the archived database was scaled from 94 MB to 10 GB. The number of RDF triples in the data archived varied from 2 009 722 to 195 528 165.

The following notation is used in the performance diagrams. ***TBL***: the complete time spent for the TBL approach; ***IAV***: the complete time spent for the IAV approach; ***Load data***: time spent to load data by reading the data archive and executing the stored procedures with TBL; ***Create FK constraints***: time spent to create foreign key constraints for both IAV and TBL; ***Triple bulk load***: time spent to bulk load data archive triples before sorting in the TBL approach; ***Sort***: time spent in TBL to sort the date archive triples by the ORDER BY query; ***Create CSV***: time spent on CSV file generation with TBL; ***Table bulk load***: time spent to bulk load the generated CSV files with TBL.

The results from the experiments are presented in Fig. 5, 6, and 7. It can be seen on Fig. 5 that the TBL approach substantially outperforms the IAV approach. Using TBL is 18-25 times faster than the IAV approach.

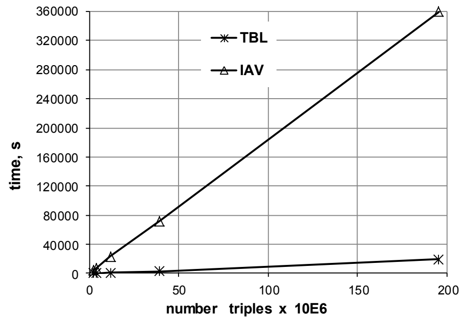

Figure 5. IAV and TBL approach

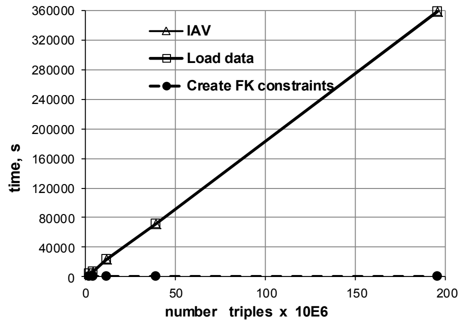

Figure 6. IAV approach

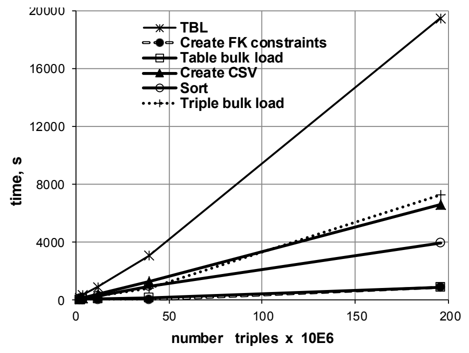

Figure 7. TBL approach

Fig. 6 and 7 show how much time is spent on the different phases of the reloading. For IAV (Fig. 6) 99.9 % of the time is spent on inserting attribute values into the RDB and 0.1 % on constructing foreign keys. Fig. 7 shows that for TBL around 25-37 % of the time is spent on bulk loading triples to be sorted, and the sorting itself takes 20-30 % of the time. Converting the sorted triples to CSV takes 33-43 % of the time. Finally, only 1-4 % is spent on constructing foreign keys, and 3-4% on the final bulk loading of the reconstructed RDB.

5. RELATED WORK

To the best of our knowledge there is no other paper investigating the performance of algorithms for restoring relational databases archived in RDF.

RDF2RDB [11] also constructs and populates relational databases from RDF data files, but do not present details on how the RDB is populated and there are no performance investigations. In contrast, we have investigated two different approaches to populate an RDB with relational data converted from RDF triples. Our experiments show that the TBL approach is substantially faster than IAV.

The R2D system [6][7] generates a relational view over an RDF store. The purpose is to process SQL over RDF data. R2D does not reconstruct archived relational databases.

[14][15] convert XML Schema to relational tables, which are then populated by reading XML documents. XML is different from RDF in that it is based on tree structures while RDF represents meta-data graphs as unordered triples, which requires the special handling presented here.

6. SUMMARY

We have investigated approaches for scalable recreation of relational databases archived as RDF files. An RDF-archived relational database is reconstructed from a schema archive file and a data archive file, both in N-Triples formats. The archives contain RDF triples representing the relational schema for the archived content, and relational data content, respectively. When an archived RDB is to be reconstructed, the schema archive is read to automatically reconstruct the RDB schema in another RDBMS. The schema reconstruction algorithm is based on identifying relational database schema elements by queries to the schema archive. The reconstructed RDB is then populated by reading the data archive triples and converting them into relational attribute values according to the schema. We have investigated two approaches to populate data into the reconstructed RDB: the IAV approach and the TBL approach.

With the naive IAV approach the database is populated by automatically generated stored procedures which execute an SQL INSERT or UPDATE statement for each attribute value read from the data archive file.

With the TBL approach the database is populated by bulk loading CSV files generated from the data archive. The bulk loader requires one CSV file of rows for each reconstructed table and therefore the data archive needs to be regrouped per table and row in order to create the CSV files. The regrouping is made by first bulk-loading the archived triples into an RDB and then issuing an ORDER BY query in the RDBMS to group the data archive triples per CSV file to generate. Finally, a CSV generation algorithm is applied on each query result tuple to produce the CSV files in one pass with limited memory.

Our experiments show that the TBL approach is substantially faster than the IAV approach, despite the added sorting and post-processing.

REFERENCES

[1] Arenas, M., Bertails, A., Prud'hommeaux E. and J. Sequeda. 2012. A Direct Mapping of Relational Data to RDF, W3C, Recommendation 27 September 2012, http://www.w3.org/TR/rdb-direct-mapping/

[2] Bizer, C. and Schultz, A, Berlin SPARQL Benchmark (BSBM) Specification, Data Generator and Test Driver, http://www4.wiwiss.fuberlin.de/bizer/BerlinSPARQLBenchmark/spec/20080912/index.html#datagenerator

[3] Borghoff, U. et al. 2010. Long-Term Preservation of Digital Documents, Springer

[4] Buneman, P., Khanna, S., Tajima, K. and Tan, W. 2004. Archiving Scientific Data, ACM Transactions on Database Systems, Vol. 29, No. 1, pp. 2-42, 2004

[5] Hunter, J. 2006. Scientific Publication Packages – A Selective Approach to the Communication and Archival of Scientific Output, The International Journal of Digital Curation, iss. 1 vol. 1, pp. 33-52, 2006

[6] Ramanujam, S., Gupta, A., Khan, L., Seida, S., and Thuraisingham, B., 2009. R2D: Extracting Relational Structure from RDF Stores, 2009 IEEE/WIC/ACM Int. Conference on Web Intelligent Technology-Workshop, 2009

[7] Ramanujam, S., Gupta, A., Khan, L., Seida, S. and Thuraisingham, B. 2009. R2D: A Bridge between the Semantic Web and Relational Visualization Tools, 2009 IEE Int. Conference on Semantic Computing

[8] Sequeda, J. F. and Miranker, D. Ultrawrap: SPARQL Execution on Relational Data, Technical Report http://apps.cs.utexas.edu/tech_reports/reports/tr/TR-2078.pdf

[9] Stefanova S. and Risch, T. 2012. Scalable Long-term Preservation of Relational Data through SPARQL queries, Semantic Web journal, Submitted for publication, October, 2012

[10] Stuckenschmidt H. and Harmelen, F. 2005. Information Sharing on the Semantic Web,Springer,ISBN 3-540-20594-2

[11] Teswanich, W. and Chittayasothorn S. 2007. A Transformation from RDF Documents and Schemas to Relational Databases, IEEE PacificRim Conferences on Communications, Computers and Signal Processing, 2007, pp. 38-41

[12] Ad hoc Strategic Committee on Information and Data, Final Report to the ICSU Committee on Scientific Planning and Review, http://www.icsu.org/publications/reports-and-reviews/scid-report/scid-report.pdf, 2008

[13] National Science Board: Long-Lived Digital Data Collections: Enabling Research and Education in the 21st Century, NSB 05-40, http://www.nsf.gov/pubs/2005/nsb0540/nsb0540.pdf, 2005

[14] Oracle® XML DB Developer's Guide, 11g Release 2 (11.2), http://docs.oracle.com/cd/E11882_01/appdev.112/e23094/toc.htm

[15] XML Data (SQL Server), http://msdn.microsoft.com/en-us/library/bb522446.aspx

ACKNOWLEDGEMENTS

This project is supported by the eSSENCE Collaboration Project.