Integrating Open Sources and Relational Data with SPARQL

Orri Erling and Ivan Mikhailov

OpenLink Software, 10 Burlington Mall Road Suite 265 Burlington, MA 01803 U.S.A
{oerling,imikhailov}@openlinksw.com,
http://www.openlinksw.com

Abstract. We believe that the possibility to use SPARQL as a front end to heterogeneous data without significant cost in performance or expressive power is key to RDF taking its rightful place as the lingua franca of data integration. To this effect, we demonstrate how RDF and SPARQL can tackle a mix of standard relational workload and data mining in public data sources.

We discuss extending SPARQL for business intelligence (BI) workloads and relate experiences on running SPARQL against relational and native RDF databases. We use the well known TPC H benchmark as our reference schema and workload. We define a mapping of the TPC H schema to RDF and restate the queries as BI extended SPARQL. To this effect, we define aggregation and nested queries for SPARQL.

We demonstrate that it is possible to perform the TPC H workload restated in SPARQL against an existing RDBMS without loss of performance or expressivity and without changes to the RDBMS.

Finally, we demonstrate how to combine TPC-H or XBRL financial reports with RDF data from CIA factbook and DBpedia.

1 Introduction and Motivation

RDF promises to be a top level representation for data extracted or accessed or demand from any conceivable source. Thus, chief promise of RDF is in the field of information integration, analysis and discovery. Yet it is difficult to imagine any business reporting, let alone more complex information integration task that would not involve aggregating and grouping.

As a data access and data integration vendor, OpenLink has a natural interest in seeing SPARQL succeed as a top level language for answering business questions on data mapped from any present day data warehouse or other repository.

For SPARQL to deliver on this potential, several extension and scalability issues have to be addressed.

These include:

- Expressive power of SPARQL must be at least on par with SQL. As of the present, the SPARQL recommendation is lacking aggregation, grouping, expressions in result sets, nested subqueries and full text support, to name

S. Bechhofer et al.(Eds.): ESWC 2008, LNCS 5021, pp. 838–842, 2008.

a few. All these are either part of SQL or universally available in RDBMS, as in the case of full text. The baseline business intelligence benchmark, TPC H, relies on these all, except for full text.
- Efficient mapping of SPARQL to relational queries against one or more relational databases. SPARQL's promise is greatest in combining data from diverse sources. Still, in cases where a straightforward translation of SPARQL to SQL is possible, the performance should not be much less than that of the relational back-end when accessed through SQL.
- Scalability of RDF storage. Parallelization and clustering are needed for scaling into the tens of billions of triples and beyond.

We intend to demonstrate how we address all these questions with our Virtuoso product.

2 The Data and Queries

We draw on a combination of real-world and synthetic data sets for the demonstration. In specific, we use the following:

- DBpedia;
- US Census;
- real world XBRL financial data mapped into RDF;
- various Linking Open Data sets, such as Geonames and the CIA Factbook;
- TPC H benchmark data, a scalable industry standard benchmark data set.

The TPC H data is stored in relational form as well as as RDF triples. We demonstrate queries combining these data in novel ways. For example:

- comparing sales figures from the TPC H data with population and GDP figures from the CIA Factbook;
- combining XBRL financial results with geography and DBpedia information on the same companies;
- comparing two TPC H data sets, one as a relational database and one in RDF form.

In addition to aggregate queries such as the above, we show navigation by following data links between these sets.

We also present loading and query times for data sets such as the LUBM benchmark data and the Uniprot data set.

The complete source code of the queries and data definitions and mappings is published at the OpenLink web site at the time of the demonstration (hboxhttp://demo.openlinksw.com/tpc-h/). The data itself is either linked open data or synthetic data that can be generated with generally available tools. Thus the things demonstrated are readily reproducible.

2.1 SPARQL Extensions

We show how we have extended SPARQL with the following:

- Subqueries and derived tables.
- Aggregates, grouping and expressions in results.
- Syntax sugar for following chains of references, as in region of country of customer of order X.

The below is the SPARQL version of Q2 from the TPC H queries.

```
prefix tpcd: <http://www.openlinksw.com/schemas/tpcd#>
select
  ?supp+>tpcd:acctbal  ?supp+>tpcd:name
  ?supp+>tpcd:has_nation+>tpcd:name as ?nation_name
  ?part+>tpcd:partkey  ?part+>tpcd:mfgr
  ?supp+>tpcd:address  ?supp+>tpcd:phone  ?supp+>tpcd:comment
from <http://example.com/tpcd>
where {
  ?ps a tpcd:partsupp ; tpcd:has_supplier ?supp ; tpcd:has_part ?part .
  ?supp+>tpcd:has_nation+>tpcd:has_region tpcd:name 'EUROPE' .
  ?part tpcd:size 15 .
  ?ps tpcd:supplycost ?minsc .
  { select ?part min(?ps+>tpcd:supplycost) as ?minsc
    where {
        ?ps a tpcd:partsupp ;
          tpcd:has_part ?part ; tpcd:has_supplier ?ms .
        ?ms+>tpcd:has_nation+>tpcd:has_region tpcd:name 'EUROPE' .
      } }
    filter (?part+>tpcd:type like '%BRASS') }
order by
  desc (?supp+>tpcd:acctbal)
  ?supp+>tpcd:has_nation+>tpcd:name
  ?supp+>tpcd:name
  ?part+>tpcd:partkey
```

We notice a subquery used for determining the lowest supply cost for a part. We also notice the pattern

```
{ ?ms+>tpcd:has_nation+>tpcd:has_region tpcd:name 'EUROPE' }
```

which is a shorthand for

```
{ ?ms tpcd:has_nation ?t1 . ?t1 tpcd:has_region ?t2 .
  ?t2 tpcd:name "EUROPE" }
```

The notation with `+>` differs from a join path expressed with `[]` in that these are allowed in expressions and that common subpaths are guaranteed to be included only once in the evaluation. Thus

```
sum ( ?c+>has_order+>has_line+>l_extendedprice *
  (1 - ?c+>has_order+>has_line->l_discount) )
```

evaluates to the sum of each line's extendedprice multiplied by the line's discount whereas

```
sum ( ?extprice * (1 - ?discount))
...
?c has_order [ has_line [ l_extendedprice ?extprice ] ] .
?c has_order [ has_line [ l_discount ?discount ] ]
```

would mean the sum of every price times every discount.

For brevity we have omitted the declarations for mapping the TPC H schema to its RDF equivalent. The mapping is straightforward, with each column mapping to a predicate and each table to a class.

2.2 Linked Data

Virtuoso has an integrated HTTP server used for providing web services end points and web app hosting. For presenting the TPC H data as linked data, we have added a virtual collection which presents the data as dereferenceable URI's, redirecting the dereference to a describe query against the SPARQL end point.

2.3 Performance of Mapping

As a baseline, we take the performance of Virtuoso executing TPC H queries in SQL against Oracle. There Virtuoso parses the SQL query, makes a distributed execution plan, finds out the whole query can go to Oracle and finally rewrites the query as a single Oracle SQL query. This takes an average of 7 ms per query, including time to send and retrieve results. The rest of the real time is spent by Oracle.

Adding the SPARQL to SQL layer on top of this adds another 9 ms to each query. The cost of SPARQL is negligible in the cases where the resulting SQL query passes as a single unit to Oracle.

We note that the single most important factor in any distributed query performance as opposed to local query performance is the number of synchronous round trips between the processes involved.

Some SPARQL queries make a suboptimal SQL that does not pass as a unit to Oracle (even if it should), so the execution is divided between Virtuoso and Oracle and there is significant cost from message latency. Fixing this is a current work in progress.

3 System Demonstrated

The demonstration databases run on a cluster of X86-64 servers either at our offices or Amazon's EC2. Smaller scale local demonstration can be run on laptops with the same software but less data.

The software demonstrated includes:

- Virtuoso 6.0 RDBMS and triple store.
- Oracle 10G RDBMS accessed both directly and through Virtuoso's RDF to relational mapping.
- Diverse RDF browsers (Tabulator, OpenLink RDF Browser and Zitgist).

4 Conclusions

Mapping of relational data to RDF has existed for a long time [6][7]. The work shown here represents its coming of age. We can tackle a standard SQL workload without loss of performance or added complexity. Basically, we can bring any data warehouse to the world of linked data, giving dereferenceable URI's and SPARQL while retaining the performace of SQL.

We would point out that bringing SPARQL on par with SQL for decision support queries is not aimed at replacing SQL but at making SPARQL capable of fulfilling its role as a language for integration.

Indeed, we retain all of SPARQL's and RDF's flexibility for uniquely identifying entities, for abstracting away different naming conventions, layouts and types of primary and foreign keys and so forth.

In the context of mapping relational data to RDF, we can map several instances of comparable but different schemes to the common terminology and couch all our queries within this terminology. Further, we can join from this world of mapped data to native RDF data, such as the data in the Linking Open Data project.

Once we have demonstrated that performance or expressivity barriers do not cripple SPARQL when performing traditional SQL tasks, we have removed a significant barrier from enterprise adoption of RDF and open data.

References

1. W3C RDF Data Access Working Group: SPARQL Query Language for RDF, `http://www.w3.org/TR/rdf-sparql-query/`
2. Transaction Processing Performance Council: TPC-H – a Decision Support Benchmark, `http://www.tpc.org/tpch/`
3. Linking Open Data Project, `http://linkeddata.org/`
4. DBpedia – A Community Effort to Extract Structured Information From Wikipedia, `http://dbpedia.org/`
5. XBRL - Extensible Business Reporting Language, `http://www.xbrl.org/Home/`
6. Seaborn, A.: Counting and GROUP BY in ARQ, `http://seaborne.blogspot.com/2007/09/counting-and-group-by.html`
7. Weiske, C., Auer, S.: Implementing SPARQL Support for Relational Databases and Possible Enhancements. In: Proceedings of the 1st Conference on Social Semantic Web. Leipzig (CSSW 2007), SABRE. LNI 113 GI 2007, Bonner Kollen Verlag (2007), `http://www.informatik.uni-leipzig.de/~auer/publication/sparql-enhancements.pdf`, ISBN 978-3-88579-207-9
8. Erling, O., Mikhailov, I.: Adapting an ORDBMS for RDF Storage and Mapping. In: Proceedings of the 1st Conference on Social Semantic Web. Leipzig (CSSW 2007), SABRE. LNI 113 GI 2007, Bonner Kollen Verlag (2007) ISBN 978-3-88579-207-9