RDF For the Data Driven Age

Databases and knowledge representation both have decades of history but to date exchange of ideas and techniques between these disciplines has been limited. The intuition that there would be value in greater cooperation has not failed to occur to researchers on either side, after all, both sides deal with data. From this, we have seen deductive databases emerge, as well as more recently ' database friendly' profiles of OWL.

In this position paper we will examine what, in the most concrete terms, is needed in order to bring leading edge database technology together with expressive querying and reasoning. This draws on our experience in building Virtuoso, one of today's leading RDF stores. Following this, we argue for the creation of benchmarks and challenges that in fact do reflect reality and facilitate open and fair comparison of products and technologies.

Data integration is often mentioned as the motivating use case for RDF. Database research has over the past few years produced great advances for business intelligence, i.e. complex queries and read-mostly workloads. These advances are typified by compressed columnar storage and architecture-conscious execution models, mostly based on the idea of always processing multiple sets of values in each operation (vectoring). With these techniques, raw performance with relatively simple schemas and regular data, e.g. TPC H is no longer a barrier to extracting value from data.

A similar breakthrough has not been seen on the semantics side. Data integration still requires manual labor. Publishing datasets in RDF is a good and necessary intermediate stage but producing these datasets from diverse sources Is not fundamentally different from doing the same work without RDF. Even so, RDF serves as a catalyst for a culture of publishing datasets.

RDF, as a base model for integration, offers the following benefits over a purely relational result format: All entities have globally unique identifiers, any statements may be associated ad hoc to any entities and these statements can be scoped into graphs according to their provenance, time, validity etc.

Obtaining this flexibility on a relational basis would simply require moving to an RDF-like representation with essentially one row per attribute. Indeed, we see key-value stores being used in online applications with high volatility of schema (e.g. social networks, search) and we also see relational applications making provisions for post-hoc addition of per-entity attributes, i.e. associating a bag of mixed non-first normal form data to entities. The benefits of a schema-last approach are recognized in many places.

RDF seems a priori a fit for all these requirements, thus how will it claim its place as a solution?

The first part of the answer lies in learning all the relevant database lessons. The second part lies in eliminating the impedance mismatch between querying and reasoning. The third and most important part consists of substantiating these claims in a manner that is understandable to the relevant publics, finally leading to the creation of a semantics-aware segment of the database industry. We will address each of these aspects in turn.

RDF and Database

The problem is divided into storage format, execution and query optimization. For the first two, Daniel Abadi' s renowned Ph.D. thesis holds most of the keys. Space efficiency is specially important for RDF, since data is often voluminous and many datasets have to be brought together for integration. Access patterns are also unpredictable, with indexed random access predominating, as opposed to RDB BI workloads where sequential scans and hash joins represent the bulk of the work. However, we find that a sorted column-wise compressed representation of RDF with a single quad table for all statements gives excellent space efficiency and good random access as well as random insert speed. The space efficiency is close to par with the equivalent column-wise relational format since three of the four columns of the quad table compress to almost nothing. As many sorted orders may be maintained as is necessary but we find that two are enough, with some extra data structures for dealing with queries where the predicate is unspecified. The details are found in [ref to VLDB2010 Semdata workshop]. Since RDF is a model typed at run time, the engine must support an ' any' data type for columns and query variables, where values on successive rows may be of different types. This is a straightforward enhancement.

Vectored execution is traditionally associated with column stores because the per-row access cost is relatively high, thus needing to access many nearby rows at a time in order to amortize the overhead. Aside this, vectored execution provides many opportunities for parallelism, from the instruction level all the way to threading and distributed execution on clusters, thus some form of execution on large numbers of concurrent query states is needed for RDF stores, just as it is needed for RDBMS"s.

Query optimization for RDF is similar to that for RDBMS' s, except that the statistics can no longer be collected by column and table but must rather apply to individual entities and ranges of a single quad table. This can be provided through run time sampling of the database based on constants in the query being optimized. Beyond this, interleaving execution and optimization as in [ROX] seems to offer limitless possibilities, specially when inference is introduced, making optimizer statistics less predictive.

In summary, starting with a RDBMS and going to RDF entails changes to all parts of the engine but these changes are not fundamental. One does need to own the engine, however, otherwise the expertise for efficiently implementing these changes will not exist. Essentially any DBMS technique may be translated to an RDF use case if its application can be decided at run time. RDF may be schema-less, yet most datasets have fairly regular structure, the question is simply to reconstruct the needed statistics and schema information from the data on an as you go basis. Techniques with high up-front cost, like constructing specially orderdd materializations for optimizing specific queries are harder to deploy but still conceivable for RDF also.

Database and Inference

Compared to the straightforwardly performance oriented world of database engines, the contours of the landscape become less defined when moving to inference. Databases, whether relational or schema-less all perform roughly the same functions but inference is more diverse. We include here also techniques like machine learning and meta-reasoning for guiding reasoning, although these might not strictly fit the definition.

As we posit that data integration is the motivating use case for RDF as opposed to RDM (Relational Data Model), we must ask which modes of inference are actually required for data integration. Further, we need to ask whether these inferences ought to be applied as a preprocessing step (ETL or forward chaining) or as needed (backward chaining). Some low-hanging fruit can be collected by simply constructing class or property hierarchies, e.g. in the data at hand, the following properties have the meaning of company name and the following classes have the meaning of company. We have found that such techniques can be efficiently supported at run time, without materialization if the support is simply built into the engine, which is in itself straightforward as long as one controls the engine. The same applies to trivial identity resolution, such as owl:sameAs or resolution of identity based on sharing an inverse functional property value. These things take longer at run time but if one caches and reuses the result, one can get around materializing.

We do not believe in weak statements of identity, as in X is similar to Y, since the meaning of similarity is entirely contextual. X and Y may or may not be interchangeable depending on the application, thus the statement on identity needs to be strong but it must be easy to modify the grounds on which such a statement is made. This is a further argument on why one should not automatically materialize consequences of identity, particularly if dealing with web data where identity is specially problematic.

Real-world problems are however harder than just bundling properties, classes or instances into sets of interchangeable equivalents, which is all we have mentioned thus far. There are differences of modeling (address as many columns in customer table vs. address normalized away under a contact entity) , normalization (first name and last name as one or more properties, national conventions on person names, tags as comma separated in a string or as a one to many) , incomplete data (one customer table has family income bracket, the other does not) , diversity in units of measurement (Imperial vs metric), variability in the definition of units (seven different things all called blood pressure), variability in unit conversions (currency exchange rates), to name a few. What a world!

If data exists, the conversion questions are often answerable but their answer depends on context, e.g. date of transaction concerned for currency exchange, source of data for the definition of blood pressure.

Alongside these, there remain issues of identity, e.g. depending on the perspective, a national subsidiary is or is not the same entity as the parent company, companies with the same name can be entirely unrelated in different jurisdictions.

It appears that we may need a multi-level approach, combining different techniques for different phases of the integration process. We do not a priori believe that using SQL vviews for unit and modeling conversion and then OWL for unifying terminology on top of this were the whole solution. Even if this were the solution, the pipeline from the relational sources to SPARQL and OWL needs to be optimized for real-world BI information volumes and the query language needs to be able to express the business questions and needs to interface with the reporting tools the analyst has come to expect.

Our answer so far consists of a SPARQL extension with non-recursive rules, roughly equivalent to SQL views in expressive power, tightly integrated to the query engine. There is also limited support for recursion through transitive subqueries, thus one can compactly express things like ' all parts of all assemblies and subassemblies must satisfy applicable safety requirements, where the requirements depend on the type of the part in question.'

This is only an intermediate step. We believe that a database-scale generic inference engine with at least Datalog power with second order extensions like computed predicates is needed, executing inside the DBMS, benefiting from the whole array of optimizations database-science expects of execution engines is part of the answer.

This will not relieve the analyst of having to consider that the currency rates in effect at the time of conversion must be taken into account when calculating profits but this will at least make expressing this and similar pieces of context more compact.

We note that time to answer has historically won over raw performance. This was also the case for RDBMS" s when these were the fresh challenger to the CODASYL incumbants, just as was the case with the adoption of high level languages. The key is that the raw performance must be sufficient for the real world task. With the adoption of the database lessons outlined in the previous section, we believe this to be the case for RDF.

Substantiating the Claims

Benchmarks have a stellar record for improving any metric they measure. The question is, how can be make a metric that measures RDF' s ability to deliver on its claim to fame, time to answer for big data, with all the integration and other complexities this entails?

So far, RDF benchmarks have consisted of workloads where RDBMS' s are clearly better, e.g. LUBM or the Berlin SPARQL Benchmark. This does not remove their usefulness for RDF but does not constitute an RDF selling point either.

We suggest a dual approach. The first part is demonstrating that RDF is scalable for BI: We take the industry standard decision support benchmark TPC H, which is very favorable to relational and quite unfavorable to RDF and show that we can tackle the workload at reasonable cost. If TPC H is all one wants, an RDBMS will stay a better fit but then this benchmark does not capture any of the heterogeneity, schema evolution or other such requirements real world data warehouses face. This is still a qualification test, not the selling point.

The issue of benchmark is inextricably tied to the issue of messaging. This must be a compelling story, with which the IT community can identify. Further, the benchmark must capture real world challenges in the area of interest. With all this, the benchmark should not be too expensive to run. Here too, a multistage approach suggests itself.

Our tentative answer to this question is the Social Intellligence Benchmark (SIB), developed together with CWI and other partners in the LOD2 consortium. This simulates a social network and combines an online workload with complex analytics. This benchmark should cover all of the target areas of the LOD2 project, so that the project itself generates its own metric of success. The project has clear data integration targets, specially as applies to web and linked data. Questions of integration with enterprise sources need to be further developed, for example comparing CRM data with extractions from the online conversation space for market research.

Data integration will invariably involve human effort and the the area cannot be satisfactorily covered with metrics of scale and throughput alone. Development time, accuracy of results and cost of

maintenance are all factors. Furthermore, the task being modeled must correspond to reality, still without being too domain specific or prohibitively time consuming to implement.

Conclusions

The data driven world will increase rewards for efficiency in data integration. We believe that such efficiency crucially depends on semantics. Real world requirements just might throw the database and AI communities together with enough heat and pressure for fusion to ignite, allegorically speaking. Without a clear and present need, the geek world analog of electrostatic repulsion will keep the communities separate, as has been the case thus far, and no qualitatively different, new element will arise.

Efforts such as this present STI summit and the LOD2 project are needed for setting directions and communicating the requirement to the research world. In our fusion analogy, this is the field which directs the nuclei to collide.

Once there is an actual reaction that produces more than it consumes by a sufficient margin, regular business dynamics will take over and we will have an industry with several products of comparable capability, as well as a set of metrics, all to the benefit of the end user.

References

TPC H results page

Daniel Abadi's thesis

Our VLDB 2010 Semdata workshop paper

ROX from CWI

LOD2 web site