CSE 718 - SEMINAR REPORT

SCALABLE SEMANTIC WEB DATA MANAGEMENT USING VERTICAL PARTITIONING

Sneha Godbole
godbole@buffalo.edu

OVERVIEW

The Semantic Web is an extension of the World Wide Web. It represents data in such a way that it can be accessible in different forms for a variety of different applications. The Semantic Web makes data globally available and it can be thought of as a globally linked database. Semantic Web also expresses data as future possibilities that might be implemented later. Some of the components of Semantic Web are XML, Resource Description Framework (RDF) and Web Ontology Language (OWL).

This paper first explores the poor scalability limitations of current data management solutions for RDF data. It then proposes a solution over these limitations – the property tables. After discussing the limitations of property tables, the author's propose an alternative solution – vertically partitioning the RDF data. The paper concludes by comparing the performance of vertically partitioning with other approaches by considering queries generated by a Web-based RDF browser over a large-scale catalog of library data.

The Semantic Web data model is called the RDF. It represents data as statements about resources. A graph can be drawn representing the nodes as resources and labeled arcs between them representing properties. However, this graph can be parsed into a set of triples where each triple represents a statement. A triple can be of the form <*subject, property, object*>. This method, although flexible, has serious performance issues because having all the triples stored in one single RDF table requires several self-joins. Thus, as queries become more complex the execution time increases.

Property tables denormalize RDF tables by physically storing them in a wider and flattened representation. There are two types of property tables – Clustered Property Table and Property-Class table. Properties that tend to be defined together are clubbed into clusters and put into a wider property table. The Property-Class table collects a similar set of subjects together in the same table. In both cases, left-over triples are stored in another triples table. The difference between these two approaches is that, the Clustered Property Tables can have one particular property present only in one single table whereas in Property-Class Tables, a property can be present in several different tables.

But making the tables wider introduces NULL values for properties that are not defined for certain subjects. More wider the table, more the number of NULLs which has space overhead. Moreover, multi-valued attributes are difficult to be represented in flattened tables. Further, if a query does not restrict on property value or if the property value will be bound when

the query is processed then all flattened tables will have to be queried and the results will have to be combined by using complex union clauses or through joins.

The Vertically Partitioned approach is stated in this paper at this stage which can be used speed up queries over a triple store. The concept used here is to store triples into *n* two column tables. Here *n* is the number of unique properties in the data. The first column in each table is the *subject* and the second column is the *object*. This approach supports multi-valued attributes; does not introduce unnecessary NULLs by eliminating subjects that do not define a particular property; does not require clustering algorithms and has fewer unions and joins since all data for a particular property is located in the same table. The author's have extended a Column-Oriented DBMS to implement this approach.

The paper then describes the RDF benchmark developed to evaluate the performance of the three RDF databases. The paper concludes by comparing the results of the schemas on the execution of the seven benchmark queries.

DETAILED COMMENTS

The paper discusses the results of all the queries executed on the four different architectures discussed earlier in this paper. The performance of each of the architectures is plotted as an average of three runs of the queries. The comparison plots show that performances of property table and vertically partitioned table are similar. Thus, the author's strongly argue that vertically partitioning a database gives a significant performance improvement over the triple-store schema and performs similarly to property tables. Also vertical partitioning is easier to implement. Similar arguments are put forward after showing the performance of the architectures when the number of triples are scaled from one million to fifty million.

The author's also state that using the concept of materialized path expressions by adding one extra pre-computed path expression column in the property table, increases the performance in the column store by two orders of magnitude as compared to the triple store.

The paper also strongly states the results of widening a property table. The results clearly argue that though the property tables sometimes have better performance as compared to vertical partitioning on a row-oriented store, a poor choice of property table can result in poorer query performance.

The paper does not discuss the practical applications of RDF data. The RDF graph and RDF triples table has been implemented in the latest social networking application named Facebook. Many Semantic Web tools and business solutions have been built by organizations like Oracle, IBM, Adobe. The Sun's white paper collection site uses Semantic Web technologies in the background.

The vertical partitioning solution discussed in the paper has been proved to be the best solution among the rest, but the paper doesn't discuss how it can be used to perform reasoning inside the database itself. Also, SPARQL is the dominant RDF query language, but it is easier to translate to SQL queries over RDF triples table as compared to vertical schema. Thus, future work can be related to translating SPARQL to queries over vertical schema.