Poster Paper

Double Chain-Star: an RDF indexing scheme for fast processing of SPARQL joins

Marios Meimaris
[1]University of Thessaly,
[2]ATHENA Research Center,
Greece
m.meimaris@imis.athena-innovation.gr

George Papastefanatos
ATHENA Research Center,
Greece
gpapas@imis.athena-innovation.gr

ABSTRACT

State of the art RDF stores often rely on exhaustive indexing and sequential (self-)joins for SPARQL query processing. However, query execution is dependent on, and often limited by the underlying storage and indexing schemes. Even though RDF can give birth to datasets with loosely defined schemas, it is common for an emerging structure to be present in the data. In this paper we introduce a novel indexing scheme, called Double Chain Star (DCS), that takes advantage of the inherent structure that is often found in RDF datasets by extending the notion of Characteristic Sets to cater for chain-star joins. DCS essentially reduces pairs of chain-star patterns that typically involve multiple self-joins, to mere index scans. We perform preliminary experiments and show promising results in comparison with Jena TDB and RDF-3X.

Categories and Subject Descriptors

H.2.8 [**Information Systems Applications**]: Database Management—*Database Applications*

Keywords

RDF, SPARQL, Query Processing, Query Optimization, Performance

1. INTRODUCTION

RDF and SPARQL are W3C recommendations for representing and querying graph data in the Data Web. There exists a rich body of literature for storing and querying RDF, however, many of these do not take advantage of the data's inherent structure in order to accelerate query processing. SPARQL optimizers depend on traditional methods for providing good query plans, including the use of data statistics and cardinality estimation for ordering triple patterns. The assumption of data independence imposes a risk of propagating errors in the planning process, especially in joins that reside deeper in the query plan. This can result in the creation of plans with large intermediate results between joins.

In this paper, we discuss a novel indexing scheme that aims to decrease the effects of bad estimates by quickly filtering triples that collectively participate in multiple joins, in one single scan. We extend the notion of Characteristic Sets, that typically represents the inherent structure of nodes in an RDF dataset, in order to characterize subject-object joins instead of single subject nodes. We call this *Extended Characteristic Sets* (ECS), and we discuss how ECS can be used for the construction of an inverted index, named *Double Chain-star*, that maps ECS's to collections of triples.

2. RELATED WORK

RDF stores often rely on the mapping of triples to relational settings, such as a single table with three columns representing subjects, predicates and objects (SPO) [2], or sets of *property tables* that are used for grouping instances of the same classes [7]. Other approaches include indexing of various SPO permutations. For example, RDF-3X [5] and Hexastore [6] make use of exhaustive indexing which includes all six permutations of subject-predicate-object, while exclusively relying on the indexes for the actual storage. Virtuoso [3] uses a large table for triples (quads), and a combination of full and partial indexes.

Characteristic Sets have been introduced as a way to provide better estimations for join cardinalities [4], and implemented in the RDF-3X high performance triple store. Brodt et al [1] discuss how an SPO index can be used to identify Characteristic Sets (CS) and use them for fast retrieval of star-shaped queries. We propose an extension of CS as a means to store and index triples, and enable scan-based answering of chain-star queries.

3. EXTENDED CHARACTERISTIC SETS

By definition, a Characteristic Set of a subject node s contains all properties p_i that appear in triples with s as subject. More formally, given a collection of triples D, and a node s, the Characteristic Set $S_c(s)$ of s, as given by [4], is:

$$S_c(s) = \{p \mid \exists o : (s, p, o) \in D\}$$

and the set of all S_c for a dataset D is:

$$S_c(D) = \{S_c(s) \mid \exists p, o : (s, p, o) \in D\}$$

The upper bound for $|S_c(D)|$ is $|D|$, but the existence of an inherent structure in RDF data makes the distinct set of Characteristic Sets that appear in real-world data small [4]. We introduce the *Extended Characteristic Set* ECS, as the

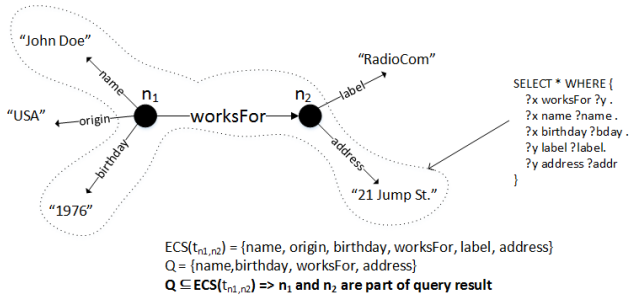

Figure 1: Double chain-star graph consisting of a star-shaped graph around n_1 and a star-shaped graph around n_2 .

union of the properties that appear in the subject t_s and the object t_o of a triple t, i.e. the union of the Characteristic Sets of t_s and t_o:

$$E_c(t) = \{p_1 \cup p_2 \mid \exists o_1 : (t_s, p_1, o_1) \in D \text{ and } \exists o_2 : (t_o, p_2, o_2) \in D\}$$

or simply:

$$E_c(t) = \{S_c(t_s) \cup S_c(t_o)\}$$

The set of all ECS in D is given by:

$$E_c(D) = \{E_c(t) \mid \exists p : (t_s, p, t_o) \in D\}$$

Essentially, $E_c(t)$ helps to quickly identify the largest superset of graph patterns that contain *double chain-stars* consisting of a star pattern around t_s, a star pattern around t_o, and a subject-object join between t_s and t_o, with t_s being the subject and t_o the object in a common triple. An example double chain-star graph pattern is shown in Figure 1, where nodes n_1 and n_2 are present in the same triple t_{n_1,n_2} as subject and object respectively, and descriptive star patterns are present for each of the two nodes. A Characteristic Set $S_c(s)$ is also an ECS, meaning that *leaf* star patterns are also parts of the ECS space. Preliminary experiments have shown $|E_c(D)|$ to be low, specifically for LUBM100 with ∼15m triples, this number is 109, for 27 distinct CSs.

4. THE DOUBLE CHAIN-STAR INDEX

The Extended Characteristic Sets of a given dataset can act as filters for collections of triples that fulfil a certain query pattern, reducing costly subject-object joins to mere index scans. For this to be feasible, we propose the Double Chain-Star (DCS) index, which is essentially an inverted index that maps collections of triples to ECSs. A triple t is mapped to an ECS if the ECS contains properties that appear in triples of either the subject, or the object of t, and t cannot belong to more than one ECS. Because $ECS(t)$ contains all properties (outgoing edges) from nodes t_s and t_o, it also contains all subsets of properties for these two nodes. Therefore, we can check if an incoming query pattern q is a subset of $ECS(t)$, in which case the nodes of $ECS(t)$ are potential candidates for evaluating q. Physically, each ECS can be represented as a bit vector, where each bit represents a property in an (ordered) set of properties P appearing in D. Assuming a dictionary of properties and their position in P, an incoming query q can be split to a set of chain-star sub-queries $q_1, q_2, \ldots q_n$, and each q_i in q will be represented as a bit vector that instantiates the bits corresponding to the properties in q_i. Therefore, if $q_i \subseteq ECS(t)$, the triples mapped to $ECS(t)$ effectively contribute to the evaluation of q_i. An example can be seen in Figure 1. The properties need to maintain their interesting order throughout sequential updates, which is left as future work. Patterns where a subject is joined with more than one objects with their own star-shaped graphs, are subject to cost estimation, in order to determine the order in which the DCS index should be accessed.

Table 1: Query execution runtime in seconds.

	Jena	RDF-3X	DCS
LUBM10	248.66	8.24	0.56
LUBM100	timeout	timeout	29.58

We implemented a naive version on top of Jena TDB, with the DCS index kept in-memory, and compared with Jena TDB (default stats-based optimizer) and RDF-3X, measuring wall-clock time (time out after 6 hours), for LUBM10 and LUBM100. For a DISTINCT * query with three double chain-star patterns, DCS outperforms the rest by orders of magnitude. For LUBM100 the other two approaches failed to answer completely, while ours needed a few seconds. The results can be seen in Table 1.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we introduce the notion of *Extended Characteristic Sets* and propose the DCS indexing scheme, an inverted index for fast retrieval of double chain-star patterns that are present in many types of queries. Through these structures, we intend to reduce the time spent in sequential (self-)joins of stars forming around subject-object joins into mere index scans, and provide an implementation for storing and querying RDF data that provides faster query answering even for complex types of queries.

Acknowledgements. This work is supported by the EU-funded ICT project "DIACHRON" (agreement no 601043).

6. REFERENCES

[1] A. Brodt, O. Schiller, and B. Mitschang. Efficient resource attribute retrieval in rdf triple stores. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1445–1454. ACM, 2011.

[2] E. I. Chong, S. Das, G. Eadon, and J. Srinivasan. An efficient sql-based rdf querying scheme. In *Proceedings of the 31st international conference on Very large data bases*, pages 1216–1227. VLDB Endowment, 2005.

[3] O. Erling and I. Mikhailov. *Virtuoso: RDF support in a native RDBMS*. Springer, 2010.

[4] T. Neumann and G. Moerkotte. Characteristic sets: Accurate cardinality estimation for rdf queries with multiple joins. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 984–994. IEEE, 2011.

[5] T. Neumann and G. Weikum. The rdf-3x engine for scalable management of rdf data. *The VLDB Journal*, 19(1):91–113, 2010.

[6] C. Weiss, P. Karras, and A. Bernstein. Hexastore: sextuple indexing for semantic web data management. *VLDB Endowment*, 1(1):1008–1019, 2008.

[7] K. Wilkinson, C. Sayers, H. A. Kuno, D. Reynolds, et al. Efficient rdf storage and retrieval in jena2. In *SWDB*, volume 3, pages 131–150. Citeseer, 2003.