Mv-Index: An Efficient Index for Graph-Query Containment

Theofilos Mailis[1,2], Yannis Kotidis[3], Vaggelis Nikolopoulos[1,2], Evgeny Kharlamov[4,5], Ian Horrocks[6], and Yannis Ioannidis[1,2]

[1] Athena Research Centre, Greece
[2] National and Kapodistrian University of Athens, Greece
[3] Athens University of Economics and Business, Greece
[4] Bosch Center for Artificial Intelligence, Germany
[5] University of Oslo, Norway
[6] University of Oxford, United Kingdom

Abstract. Query containment is a fundamental operation used to expedite query processing in view materialization and query caching techniques. Since query containment is NP-complete for arbitrary conjunctive queries on RDF graphs, we have introduced a simpler form of conjunctive queries that we name f-graph queries. During the demo, we will show why containment checking for f-graph queries can be solved in polynomial time. We will present the mv-index, a novel indexing structure that allows for fast containment checking between a single RDF-conjunctive query and an arbitrary number of stored queries. The mv-index structure takes advantage of the interesting properties of f-graph queries. With the mv-index usage, the containment test against hundreds of thousands of queries that are indexed within our structure is performed in microseconds or less.

1 Introduction

The growing popularity of graph-structured data in many real-world applications such as Oil and Gas [7], Energy sector [9], E-commerce [4], industrial monitoring [10], factory automation [6], industrial analytics [8], and intelligent querying and search [17,1] has led to a renaissance of research on graph data management. RDF [2] and SPARQL [15] are promising examples of a graph data model and the corresponding query language that have gained a lot of attraction. In order to handle the burst of RDF data that is available on the Web, much research has been devoted on scalable techniques for RDF processing such as indexing, caching, and view materialization in order to accelerate the execution time of SPARQL queries.

View materialization is directly related to the problem of query containment [11] that has been proved to be NP-complete for arbitrary conjunctive queries [3] and unions of conjunctive queries [16] over relational databases. The same results also apply for conjunctive queries on RDF graphs and their SPARQL counterparts [14,5].

In this demo, we will present f-graph queries, a restricted form of conjunctive queries that allows to solve the containment checking problem in PTime. Additionally we will present an efficient indexing structure, the mv-index, for checking the containment relation between a single f-graph query Qf and a set of indexed queries W in PTime w.r.t. the combined size of the query and the index. We will further show how

we have extended mv-indices to evaluate containment for arbitrary conjunctive queries on RDF graphs. This translates to microseconds or less for the containment test against hundreds of thousands of queries that are indexed within the structure. The latter makes mv-indices the perfect candidate to be combined with existing and novel materialization and techniques. The corresponding demo is based on the work presented in [12].

2 The Mv-Index Structure

We will now describe the fundamentals of our indexing structure that is build upon the f-graph class of queries.

F-Graph Queries & Query Containment. To solve the containment problem and build the corresponding indexing structure, we initially focus on its variation $Q_f \sqsubseteq W$ where Q_f belongs to a special class of conjunctive queries that we name f-graph queries and W belongs to the class of conjunctive queries that have only IRIs as predicates. What motivates the choice of f-graph queries in the left-hand side of a query containment is that: *(i)* containment for f-graph queries can be solved in PTime; *(ii)* f-graph queries appear in real- world query workloads; *(iii)* f-graph queries can be employed as representatives of arbitrary conjunctive queries.

An f-graph query Q_f is a conjunctive query for which: *(i)* For every pair of terms o_1, o_2 such that $o_1 \neq o_2$, the triple patterns (s, p, o_1), (s, p, o_2) cannot both appear in Q_f; *(ii)* For every pair of terms $s_1, s_2 \in I \cup X$ such that $s_1 \neq s_2$, the triple patterns (s_1, p, o), (s_2, p, o) cannot both appear in Q_f. We name these queries f-graphs because of the functional and inverse functional characteristics of their predicates. Note that containment of such queries is polynomial due to the strong requirements for the f-graph structure. Once a variable v in W has been mapped to a term v' in Q, there is a single deterministic choice for mapping the remaining variables appearing in W.

MV-Indices. In the case that we want to check for containment between a single f-graph query Q_f and a set of conjunctive queries $\mathcal{W}$, it would be inefficient to make each and every comparison. For that reason, we have introduced the "Materialized-View Index" structure, denoted with *mv-index*, that allows to store a set of queries $\mathcal{W}$ and use it to check for containment. Our structure is based on Radix trees, ordered tree data structures that are used in string matching [13].

An mv-index $\mathfrak{M}$ is a tree structure (V, E, L) where: *(i)* V is a set of vertices; *(ii)* $E \sqsubseteq V^2$ is a finite set of edges; *(iii)* L is a labelling function that maps each edge to a non-empty ordered list of distinct elements (IRIs, literals, and parenthesis symbols) and each vertex to the serialized form of an f-graph query.

The intuition for this form of representation is that queries are represented by their serialized form in the mv-index structure, either as intermediate or leaf vertices, using the labelling function L. For a vertex α in the mv-index structure, $L(\alpha)$ is its corresponding query in serialized form. The serialized form of the query represented by a vertex can be also obtained by following the path from the root of the mv-index to the specific vertex and concatenating the corresponding edge labels. Therefore, in our actual implementation we only store edge labels.

During the insertion phase, mv-indices are treated as regular Radix trees that instead of strings or numbers are used to represent queries in their serialized form. Therefore, instead of characters within a string, or digits within a number, mv- indices use IRIs,

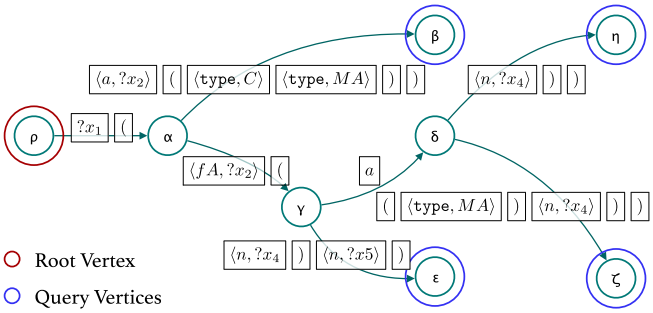

Fig. 1. A simple mv-index. The initial letters fA, n, a, MA, C depict the IRIs *fromAlbum*, *name*, *artist*, *MusicalArtist*, and *Composer* respectively.

literals, variables and separators such as parenthesis symbols in order to represent serialized queries. More information on how insertion works in Radix trees can be found in the literature [12]. To check for query containment using mv-indices, we have devised an algorithm that takes advantage of the properties of an f-graph. The intuition underlying the algorithm is that each time an edge of the mv-index is examined, a corresponding containment mapping from the corresponding queries in the mv-index to the examined query is created.

F-Graph Witnesses for Conjunctive Queries. For our algorithm to represent more expressive conjunctive queries, we have introduced f-graph witnesses. The intuition is that each conjunctive query can be represented in the form of an f-graph when checking for query containment.

For a conjunctive query Q, its corresponding f-graph witness can be obtained by merging terms that violate conditions (i), (ii) in the definition of f-graph queries. To perform the aforementioned task, we initially define the equivalence relation $\sim$ on variables, IRIs, and literals in Q such that $o_1 \sim o_2$ when there exists a term s for which either the triple patterns (s, p, o_1) and (s, p, o_2) both appear in Q, or the triple patterns (o_1, p, s) and (o_2, p, s) both appear in Q. For a term s in Q, $[s]$ denotes its equivalence class on the $\sim$ relation that contains all the terms that are merged with s. The f-graph witness Q_w of the query Q is obtained by replacing each triple pattern (s, p, o) in the body of Q with a triple pattern $([s], p, [o])$ where s, o are terms, $[s]$, $[o]$ their corresponding equivalence classes, and p is a predicate.

For a conjunctive query Q, its corresponding f-graph witness Q_w and a conjunctive query W, the following implication applies: $Q \sqsubseteq W \Rightarrow Q_w \sqsubseteq Q$. The previous proposition conveys that we need to check for containment $Q \sqsubseteq W$ only when the containment relation for the witness of Q is satisfied, i.e., $Q_w \sqsubseteq W$. Checking $Q_w \sqsubseteq W$ can be performed in PTime, while checking $Q \sqsubseteq W$ is in the worst case a NP-complete problem. Therefore, we pay a PTime budget to solve specific instances of a NP-complete problem by "postponing" non-deterministic checks in favor of a proof, computed in PTime, that $Q \sqsubseteq W$ does not apply.

3 Demo

In the demo we will show how we have implemented our novel structures and algorithms and test their efficiency in a combined query workload consisting of DBPedia, WatDiv, BSBM, LUBM, and LDBC queries. We will evaluate insertion and containment performance with respect to different query and mv-index properties. The average

time for query containment against an mv-index containing 397,507 distinct queries from all 5 workloads was between 0.0093 msec and 0.041 msec. A presentation of our work is available online.

Acknowledgements. This work was partially funded by the SIRIUS Centre, Norwegian Research Council project number 237898. T. Mailis was financed by EU Horizon2020, "DARE" project, Grant Agreement nr. 777413. Y. Kotidis was financed by the Research Centre of Athens University of Economics and Business, in the framework of the project entitled Original Scientific Publications.

References

1. Arenas, M., Grau, B.C., Kharlamov, E., Marciuska, S., Zheleznyakov, D.: Faceted search over rdf-based knowledge graphs. J. Web Semant. **37-38**, 55–74 (2016)
2. Brickley, D., Guha, R.V.: Rdf vocabulary description language 1.0: Rdf schema (2004)
3. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: STOC. ACM (1977)
4. Charron, B., Hirate, Y., Purcell, D., Rezk, M.: Extracting semantic information for e-commerce. In: ISWC. pp. 273–290 (2016)
5. Gutierrez, C., Hurtado, C.A., Mendelzon, A.O., Pérez, J.: Foundations of semantic web databases. JCSS **77**(3), 520–541 (2011)
6. Ho, V.T., Stepanova, D., Gad-Elrab, M.H., Kharlamov, E., Weikum, G.: Rule Learning from Knowledge Graphs Guided by Embedding Models. In: ISWC. pp. 72–90 (2018)
7. Kharlamov, E., Hovland, D., Skjæveland, M.G., Bilidas, D., Jiménez-Ruiz, E., Xiao, G., Soylu, A., Lanti, D., Rezk, M., Zheleznyakov, D., Giese, M., Lie, H., Ioannidis, Y.E., Kotidis, Y., Koubarakis, M., Waaler, A.: Ontology Based Data Access in Statoil. J. Web Semant. **44**, 3–36 (2017)
8. Kharlamov, E., Kotidis, Y., Mailis, T., Neuenstadt, C., Nikolaou, C., Özçep, Ö.L., Svingos, C., Zheleznyakov, D., Ioannidis, Y.E., Lamparter, S., Möller, R., Waaler, A.: An ontology-mediated analytics-aware approach to support monitoring and diagnostics of static and streaming data. J. Web Semant. **56**, 30–55 (2019)
9. Kharlamov, E., Mailis, T., Mehdi, G., Neuenstadt, C., Özçep, Ö.L., Roshchin, M., Solomakhina, N., Soylu, A., Svingos, C., Brandt, S., Giese, M., Ioannidis, Y.E., Lamparter, S., Möller, R., Kotidis, Y., Waaler, A.: Semantic access to streaming and static data at Siemens. J. Web Semant. **44**, 54–74 (2017)
10. Kharlamov, E., Mehdi, G., Savkovic, O., Xiao, G., Kalayci, E.G., Roshchin, M.: Semantically-enhanced rule-based diagnostics for industrial internet of things: The SDRL language and case study for siemens trains and turbines. J. Web Semant. **56**, 11–29 (2019)
11. Levy, A.Y., Mendelzon, A.O., Sagiv, Y.: Answering queries using views. In: PODS. pp. 95–104. ACM (1995)
12. Mailis, T., Kotidis, Y., Nikolopoulos, V., Kharlamov, E., Horrocks, I., Ioannidis, Y.E.: An efficient index for RDF query containment. In: SIGMOD. pp. 1499–1516 (2019)
13. Morrison, D.R.: Patricia—practical algorithm to retrieve information coded in alphanumeric. JACM **15**(4), 514–534 (1968)
14. Polleres, A.: From sparql to rules (and back). In: WWW. pp. 787–796. ACM (2007)
15. Prud, E., Seaborne, A., et al.: Sparql query language for rdf. `https://www.w3.org/TR/rdf-sparql-query/` (2006)
16. Sagiv, Y., Yannakakis, M.: Equivalences among relational expressions with the union and difference operators. JACM **27**(4), 633–655 (1980)
17. Soylu, A., Kharlamov, E., Zheleznyakov, D., Jiménez-Ruiz, E., Giese, M., Skjæveland, M.G., Hovland, D., Schlatte, R., Brandt, S., Lie, H., Horrocks, I.: Optiquevqs: A visual query system over ontologies for industry. Semantic Web **9**(5), 627–660 (2018)