Incremental SPARQL Evaluation for Query Answering on Linked Data

Florian Schmedding

Department of Computer Science
Albert Ludwig University of Freiburg, Germany
schmeddi@informatik.uni-freiburg.de

Abstract. SPARQL is the standard query language for RDF data. However, its application to Linked Data is challenging because the assumption that all necessary data is present at the beginning of the evaluation does not apply. Some relevant data sources may only be discovered by processing available data. Existing approaches provide implementations that compute results for basic graph patterns incrementally while retrieving the data. We contribute to this area by a formal analysis of the SPARQL algebra to provide incremental adaptions of the operations. This enables us to evaluate the costs of the incremental evaluation for the design of optimizers that choose the presumably best computation depending on the number of insertions and deletions. In addition, we propose a construction of the SPARQL dataset from Linked Data resources that enables the usage of the Graph-operator in query answering for Linked Data.

1 Introduction

On the Semantic Web, data publishing according to the Linked Data [4] principles gained significant importance. The numerous projects mentioned in the Linking Open Data cloud diagram[1] and recent ones in librarianship [8,17] show its broad adaption by private, public, and governmental initiatives. Apart from its appealing simplicity in data publication, its inherent distribution can forward the demanded decentrality of the Web [2] by allocating data at many sites rather than in centralized stores. However, Linked Data raises new challenges for query answering. In our research, we investigate these problems and contribute novel approaches for SPARQL [19] evaluation over Linked Data.

By interweaving name and address (cf. [5]), resources become dereferenceable in Linked Data and return an RDF description [14] on request. To put it simply, each resource can be perceived as data source. This has three implications:

1. The number of data sources is proportional to the number of resources.
2. Creating and deleting resources changes the number of data sources.
3. Data sources cannot be classified without retrieving their content because resource names are not related to the content.

[1] See http://richard.cyganiak.de/2007/10/lod/

Accordingly, query answering on Linked Data is quite different from traditional distributed query answering. On the one hand, in general it is not possible to specify all relevant data sources for a query in advance. Even in the case that all relevant sources have been identified for some query, another query may require different sources, and any new resource in the data may be an additional relevant source. On the other hand, subqueries cannot be delegated to the remote sources because Linked Data does not require the presence of query processors. Thus concepts like the SERVICE-operator from the federation extension[2] [6] of SPARQL 1.1 cannot be used to distribute parts of the query, for instance.

We illustrate our scenario in a small example with the query 'SELECT $?x$, $?n$ WHERE $\{$alice knows $?x$. $?x$ name $?n\}$' to find out the names of Alice's friends. Applying the "follow your nose"-approach from Hartig et al. [11] to discover relevant data sources, we start by dereferencing alice. We may receive the triples $\{$(alice, knows, bob), (alice, knows, charlie), (bob, name, "Bobby")$\}$, and compute the result $\{\{?x \mapsto$ bob, $?n \mapsto$ "Bobby"$\}\}$. Next, we request data about charlie receiving $\{$(charlie, name, "Charlie")$\}$, and extend the result with $\{?x \mapsto$ charlie, $?n \mapsto$ "Charlie"$\}$. Searching for more results, we request also bob and get $\{$(bob, name, "Bob")$\}$. Having traversed all links[3], the final result is $\{\{?x \mapsto$ bob, $?n \mapsto$ "Bobby"$\}$, $\{?x \mapsto$ charlie, $?n \mapsto$ "Charlie"$\}$, $\{?x \mapsto$ bob, $?n \mapsto$ "Bob"$\}\}$. In the present work, we investigate the suggested incremental computation of the results formally.

1.1 Related Work

Recent work has presented different strategies and implementations to evaluate queries over Linked Data. In terms of Ladwig et al. [15], Hartig et al. [11] follow a *bottom-up* approach, i. e., a query is evaluated without any prior information. Dereferencing the query constants and then following some links in the data, the result is computed incrementally, similar to our above example. However, their implementation with so-called non-blocking iterators may miss some results depending on the evaluation order. Hartig alleviates this problem in [10] with heuristic adjustments. In contrast, Harth et al. [9] employ a *top-down* (cf. [15]) strategy. Acquired knowledge about sources is organized in a special index before queries are processed. The index is used to select the most promising sources for a given query. For our example their index would recommend alice, bob, and charlie at best letting the result be generated in a single run. Corresponding to our arguments, Harth et al. also mention that the query completeness can increase when data sources which are encountered in the evaluation but are not indexed are considered during the evaluation (e. g., retrieve charlie if only alice and bob are indexed). Ladwig et al. elaborate this idea and propose a *mixed* resp. *exploration-based* approach. They propose strategies for query-specific source selection, and use a symmetric hash join for the incremental computation that, unlike the non-blocking iterators, generates all results. In [16] they refine their join method with

[2] Working Draft at
`http://www.w3.org/TR/2010/WD-sparql11-federated-query-20100601/`

[3] We do not consider the predicates here.

the intention to consider a local storage beside the remote sources. However, the implementations are not based on an analysis of the SPARQL semantics. They consider only basic graph patterns, a subset of SPARQL and do not deal with decrements potentially induced by optional graph patterns.

1.2 Contribution

We are interested in the incremental SPARQL evaluation over increasing datasets generated by bottom-up Linked Data query answering. In accordance to [10] and [15], our assumption is that the solutions should be generated after each addition—we speak of an *immediate processing*—because an exhaustive link traversal prior to returning the first results seems unfeasible. The contrary, *deferred processing*, would delay the result computation until all data has been loaded. However, we need to investigate whether an *incremental computation*, i. e. modifying current results when the data changes, is superior to a *direct computation*, i. e. deriving all results from scratch in this case. At a first glance, this seems true for monotonically increasing results, and questionable for optional graph patterns that can introduce negation by failure. Therefore we study each SPARQL operator in detail to provide adaptions that enable incremental computation. By an estimation of the processing costs we get evidences for criteria that render one computation method preferable over the other, and make a step towards optimized immediate query processing for Linked Data.

Outline. Next, we introduce RDF, SPARQL, Linked Data, and some algebraic equivalences that are necessary for our approach. In Sec. 3 we elaborate on our approach and show the incremental adaptions of the SPARQL operators. We compare our approach to the direct computation in Sec. 4. In Sec. 5, we conclude our work and sketch a possible further development, the application of the HTTP cache mechanism in query answering for Linked Data.

2 Preliminaries

We introduce the RDF data model and the SPARQL query language under special attention to the construction of the dataset by link traversal and to the relation between the Graph-operator and Linked Data resources.

2.1 RDF

The RDF data format [12] describes essentially graphs with directed labeled edges. We consider RDF terms T comprising three pairwise disjoints sets I (IRIs), B (blank nodes), and L (literals). An RDF *triple* $(s, p, o) \in I \cup B \times I \times T$ connects node s (subject) through the directed labeled edge p (predicate) with node o (object). A finite set of triples is called RDF *graph*. The blank nodes in an RDF graph G are denoted by blank(G). We distinguish blank nodes with the prefix '_:' and literals with double quotes (e. g., _:bn01, "Rain") from IRIs.

A *named graph* $\mathcal{G}$ [7] is an entity $\langle u, G\rangle$ with a name $u \in I$ and an RDF graph G. It holds $\mathrm{name}(\mathcal{G}) = u$ and $\mathrm{gr}(\mathcal{G}) = G$. Distinct named graphs do not share any blank nodes.

An RDF *dataset* D (cf. [1]) is a set containing a possibly empty default graph $\mathrm{dg}(D) = G_0$ and zero or more named graphs, $\mathrm{ngs}(D) = \varnothing$ or $\mathrm{ngs}(D) = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$ with $\mathcal{G}_i = \langle u_i, G_i\rangle$, where for $i \neq j$ (i) $\mathrm{name}(\mathcal{G}_i) \neq \mathrm{name}(\mathcal{G}_j)$, and (ii) $\mathrm{blank}(G_i) \cap \mathrm{blank}(G_j) = \varnothing$. We write $\mathrm{names}(D)$ to denote $\{\mathrm{name}(\mathcal{G}_i) \mid \mathcal{G}_i \in \mathrm{ngs}(D)\}$, and $\mathrm{gr}(u)_D = \mathrm{gr}(\mathcal{G}_i)$ if $\mathrm{name}(\mathcal{G}_i) = u$ and $\mathcal{G}_i \in \mathrm{ngs}(D)$, otherwise it is the empty RDF graph.

We use the operator '$\sqcup$' to merge two RDF graphs and define the operator '$+$' as follows. For a dataset D and a named graph $\mathcal{G}$, $D + \mathcal{G} = D \cup \mathcal{G}$ with $G_0 = \mathrm{dg}(D) \sqcup \mathrm{gr}(\mathcal{G})$ if $\mathrm{name}(\mathcal{G}) \notin \mathrm{names}(D)$, else $D + \mathcal{G} = D$.

2.2 SPARQL

SPARQL is a W3C-recommended [19] query language for RDF data. We follow the compositional semantics in [18,20] and include the Graph-operator as in [1]. Like there, our syntax differs from the W3C syntax in the previous example.

Syntax. Let V be a set of variables, $V \cap T = \varnothing$. We indicate variables with a leading question mark (e. g., $?x$). A *triple pattern* $(s, p, o) \in IV \times IV \times ILV$ is a *SPARQL expression.*[4] The variables of a triple pattern t are denoted by $\mathrm{vars}(t)$. If P_1 and P_2 are SPARQL expressions, so are P_1 Filter R, P_1 Union P_2, P_1 And P_2, P_1 Opt P_2, u Graph P_1 ($u \in I$), and $?x$ Graph P_1. R means a *filter condition*: $?x = ?y$, $?x = c$ ($c \in L \cup I$), and $\mathrm{bnd}(?x)$ are filter conditions; if R_1 and R_2 are filter conditions then $\neg R_1$, $R_1 \wedge R_2$, and $R_1 \vee R_2$ are, too. Finally, a *query* has the form $\textsc{Select}_{S,F_1,F_2}(P)$ where S is a finite subset of V, P is a SPARQL expression, and the *dataset specifications* F_1 and F_2 are possibly empty finite subsets of I (cf. From and From Named in Sec. 8 of [19]).

Semantics. A *mapping* μ is a partial function $\mu\colon V \to T$; the domain of μ is denoted by $\mathrm{dom}(\mu)$. There is an empty mapping μ_0 with $\mathrm{dom}(\mu_0) = \varnothing$. Two mappings μ_1, μ_2 are *compatible* if for all $?x \in \mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)$ holds $\mu_1(?x) = \mu_2(?x)$, written as $\mu_1 \sim \mu_2$. A mapping μ_1 is *subsumed* by a mapping μ_2, denoted $\mu_1 \sqsubseteq \mu_2$, if $\mu_1 \sim \mu_2$ and $\mathrm{dom}(\mu_1) \subseteq \mathrm{dom}(\mu_2)$. Mappings can be applied to triple patterns, written as $\mu(t)$, and replace all $?x \in \mathrm{dom}(\mu) \cap \mathrm{vars}(t)$ in t by $\mu(?x)$.

A mapping μ *satisfies* the filter condition R, denoted $\mu \vDash R$, iff one of the following six conditions holds: (i) R is $\mathrm{bnd}(?x)$ and $?x \in \mathrm{dom}(\mu)$, or (ii) R is $?x = ?y$ and $\mathrm{bnd}(?x) \wedge \mathrm{bnd}(?y) \wedge \mu(?x) = \mu(?y)$, or (iii) R is $?x = c$ and $\mathrm{bnd}(?x) \wedge \mu(?x) = c$, or (iv) R is $\neg R_1$ and $\neg(\mu \vDash R_1)$, or (v) R is $R_1 \wedge R_2$, $\mu \vDash R_1$, and $\mu \vDash R_2$, or (vi) R is $R_1 \vee R_2$, and $\mu \vDash R_1$ or $\mu \vDash R_2$.[5]

[4] Blank nodes are not considered because they act as anonymous variables and can be replaced w. l. o. g. by unselected variables in a query.

[5] The distinction between *false* and *error* in case of $\mu \nvDash R$ can be safely ignored in our context.

The *solution* of a SPARQL expression or query is a set of mappings. Let R be a filter condition and S a finite set of variables. For mapping sets Ω_1, Ω_2, the SPARQL *set algebra* defines the operations *join* ($\bowtie$), *union* ($\cup$), *minus* ($\setminus$), *left join* ($\mathbin{\text{⟕}}$), *selection* (σ), and *projection* (π):

$$\begin{aligned}
\Omega_1 \bowtie \Omega_2 &\coloneqq \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2 \colon \mu_1 \sim \mu_2\} \\
\Omega_1 \cup \Omega_2 &\coloneqq \{\mu \mid \mu \in \Omega_1 \vee \mu \in \Omega_2\} \\
\Omega_1 \setminus \Omega_2 &\coloneqq \{\mu_1 \in \Omega_1 \mid \forall \mu_2 \in \Omega_2 \colon \mu_1 \not\sim \mu_2\} \\
\Omega_1 \mathbin{\text{⟕}} \Omega_2 &\coloneqq (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2) \\
\sigma_R(\Omega_1) &\coloneqq \{\mu \in \Omega_1 \mid \mu \vDash R\} \\
\pi_S(\Omega_1) &\coloneqq \{\mu \mid \mathrm{dom}(\mu) \subseteq S \wedge \exists \mu' \colon \mathrm{dom}(\mu') \cap S = \varnothing \wedge \mu \cup \mu' \in \Omega_1\}
\end{aligned}$$

The *evaluation* semantics is defined by the help of a function $[[.]]$ that transfers a query Q into set algebra, written as $[[Q]]$. For expressions P, the same function is used with two additional arguments to indicate the dataset D and an *active* graph G for the evaluation, written $[[P]]_G^D$. Let t be a triple pattern, P_1, P_2 SPARQL expressions, $u \in I$, and R, S as before.

$$\begin{aligned}
[[t]]_G^D &\coloneqq \{\mu \mid \mathrm{dom}(\mu) = \mathrm{vars}(t) \wedge \mu(t) \in G\} \\
[[P_1 \text{ F}\textsc{ilter } R]]_G^D &\coloneqq \sigma_R([[P_1]]_G^D) \\
[[P_1 \text{ U}\textsc{nion } P_2]]_G^D &\coloneqq [[P_1]]_G^D \cup [[P_2]]_G^D \\
[[P_1 \text{ A}\textsc{nd } P_2]]_G^D &\coloneqq [[P_1]]_G^D \bowtie [[P_2]]_G^D \\
[[P_1 \text{ O}\textsc{pt } P_2]]_G^D &\coloneqq [[P_1]]_G^D \mathbin{\text{⟕}} [[P_2]]_G^D \\
[[u \text{ G}\textsc{raph } P_1]]_G^D &\coloneqq [[P_1]]_{\mathrm{gr}(u)_D}^D \\
[[?x \text{ G}\textsc{raph } P_1]]_G^D &\coloneqq \bigcup\nolimits_{u_i \in \mathrm{names}(D)} \big([[u_i \text{ G}\textsc{raph } P_1]]_G^D \bowtie \{\{?x \mapsto u_i\}\}\big) \\
[[\text{S}\textsc{elect}_{S,F_1,F_2}(P_1)]] &\coloneqq \begin{cases} \pi_S([[P_1]]_{G_0}^{D^*}) & \text{if } F_1 = F_2 = \varnothing \\ \pi_S([[P_1]]_{G_0}^{D'}) & \text{else} \end{cases}
\end{aligned}$$

According to [19], queries without dataset specification are evaluated over a default dataset D^* available to the query processor. Otherwise the dataset is constructed as $D' \coloneqq \big\{\bigsqcup_{u_i \in F_1} \mathrm{deref}(u_i)\big\} \cup \big(\bigcup_{v_i \in F_2} \{\langle v_i, \mathrm{deref}(v_i)\rangle\}\big)$. The function deref maps an IRI to its corresponding graph.

2.3 Linked Data

The term *Linked Data* [3] refers to several conventions that integrate data publication into the Web's HTTP stack as well as to the published data itself. We focus on a key aspect, the identification of *non-information* (e. g., a person, cf. [13]) resources with URLs though their essence is not a transmittable message. A request for such a resource $u \in I$ is thus redirected to an *information* resource $\mathrm{f}(u)$ which serves a description (e. g., RDF graph or HTML page) for u. So the references to other resources in descriptions are traversable and carry over the “follow your nose”-principle from the Web of Documents to the Web of Data.

Graphs. Among others, the SPARQL Graph-operator is useful for restricting mappings to authoritative information (the information provided by the URI owner of a resource, cf. Sec. 2.2.2.1 in [13] and Sec. 5.1 in [3]). Unfortunately, the rather intuitive adaption of our introductory example, $\textsc{Select}_{\{?x,?n\},\varnothing,\varnothing}$ $((\mathsf{alice},\ \mathsf{knows},\ ?x)\ \textsc{And}\ (?x\ \textsc{Graph}\ (?x,\ \mathsf{name},\ ?n)))$, is inconsistent because non-information resources (friends of Alice here) are not RDF graphs. Therefore,

- we define the dataset $D' := \{\bigsqcup_{u_i \in F_1} \mathrm{deref}(u_i)\} \cup (\bigcup_{v_i \in F_2} \{\langle \mathrm{f}(v_i), \mathrm{deref}(v_i)\rangle\})$ to beware the equation of non-information resources with named graphs. Consequently, however, graph names in D' are unpredictable before evaluating a query, so
- we add $(u, \mathsf{f}, \mathrm{f}(u))$ to G_0 for each dereferenced resource u to make the relation between u and its description $\mathrm{f}(u)$ explicitly available.

Of course, triples t with $t.p = \mathsf{f}$ got from external sources are not inserted into D' to prevent tampering. Hence the previous query can be expressed as $\textsc{Select}_{\{?x,?n\},\varnothing,\varnothing}(((\mathsf{alice},\ \mathsf{knows},\ ?x)\ \textsc{And}\ (?x,\ \mathsf{f},\ ?y))\ \textsc{And}\ (?y\ \textsc{Graph}\ (?x,$ $\mathsf{name},\ ?n)))$ and does not return Alice's nickname for Bob, Bobby.

Query Answering. We follow the bottom-up approach from [11] to illustrate our approach. First, for all dereferenceable constants $c \in I$ in a query we add $\mathrm{f}(c)$ to F_1 and F_2 and compute the mapping sets for this dataset. Second, we consider each dereferenceable c occurring in a mapping as a relevant source and insert $\mathrm{f}(c)$ into F_1 and F_2. The results over the extended dataset are computed incrementally based on the present mapping sets. We repeat the second step until F_1 and F_2 remain unchanged.

2.4 Algebraic Equivalences

Our approach is based on transformations of SPARQL algebra expressions. We define *difference* ($-$) and *intersection* ($\cap$) for mapping sets as usual (cf. *union* above) and introduce two new equivalence rules additional to those from the synoptical table in [20] shown in Tab. 1. With '$P_1 \equiv P_2$' we denote the equivalence between the algebra expressions P_1 and P_2. Note that *minus* is indeed distinct from *difference*: Consider $\Omega_1 = \{\{?x \mapsto \mathsf{a}, ?y \mapsto \mathsf{b}\}, \{?x \mapsto \mathsf{a}\}\}$ and $\Omega_2 = \{\{?x \mapsto \mathsf{a}, ?y \mapsto \mathsf{b}\}\}$, then $\Omega_1 \setminus \Omega_2 = \varnothing$ as against $\Omega_1 - \Omega_2 = \{\{?x \mapsto \mathsf{a}\}\}$.

Lemma 1 (FDPush). *Let Ω_1, Ω_2 be mapping sets and R a filter condition, then $\sigma_R(\Omega_1 - \Omega_2) \equiv \sigma_R(\Omega_1) - \sigma_R(\Omega_2)$.*

Proof. We fix a mapping μ and show that it is contained in left hand side iff it is contained in the right hand side. "$\Rightarrow$": Suppose $\mu \in \sigma_R(\Omega_1 - \Omega_2)$. It holds that $\mu \in \Omega_1$, thus $\mu \in \Omega_1 - \Omega_2$ because selection does not add mappings and $\mu \notin \Omega_2$. It follows immediately that $\mu \in \sigma_R(\Omega_1)$ but $\mu \notin \sigma_R(\Omega_2)$. "$\Leftarrow$": Suppose $\mu \in \sigma_R(\Omega_1) - \sigma_R(\Omega_2)$. Then it holds that $\mu \in \Omega_1$ and $\mu \vDash R$. We distinguish two cases. Case (1): We assume $\mu \notin \Omega_2$ and are done. Case (2): We assume $\mu \in \Omega_2$. Then $\mu \in \sigma_R(\Omega_2)$ because $\mu \vDash R$ and so $\mu \notin \sigma_R(\Omega_1) - \sigma_R(\Omega_2)$. This contradicts the first presumption. □

$(A \cup B) \cup C$	$\equiv A \cup (B \cup C)$	(UAss)
$(A \bowtie B) \bowtie C$	$\equiv A \bowtie (B \bowtie C)$	(JAss)
$A \cup B$	$\equiv B \cup A$	(UComm)
$A \bowtie B$	$\equiv B \bowtie A$	(JComm)
$(A \cup B) \bowtie C$	$\equiv (A \bowtie C) \cup (B \bowtie C)$	(JUDistR)
$A \bowtie (B \cup C)$	$\equiv (A \bowtie B) \cup (A \bowtie C)$	(JUDistL)
$(A \cup B) \setminus C$	$\equiv (A \setminus C) \cup (B \setminus C)$	(MUDistR)
$(A \setminus B) \setminus C$	$\equiv A \setminus (B \cup C)$	(MMUCorr)

Table 1. Algebraic Equivalences, where A, B, C denote mapping sets.

Lemma 2 (MDReord). *Let Ω_1, Ω_2, Ω_3 be mapping sets, then $(\Omega_1 - \Omega_2) \setminus \Omega_3 \equiv (\Omega_1 \setminus \Omega_3) - \Omega_2$.*

Proof. We proceed like above. "⇒": Suppose $\mu \in (\Omega_1 - \Omega_2) \setminus \Omega_3$. Then for all $\mu' \in \Omega_3$ holds $\mu \not\sim \mu'$, and $\mu \notin \Omega_2$ but $\mu \in \Omega_1$. It follows that $\mu \in \Omega_1 \setminus \Omega_3$, and thus also $\mu \in (\Omega_1 \setminus \Omega_3) - \Omega_2$. "⇐": Suppose $\mu \in (\Omega_1 \setminus \Omega_3) - \Omega_2$. Then $\mu \notin \Omega_2$, and for all $\mu' \in \Omega_3$ holds $\mu \not\sim \mu'$. So $\mu \in \Omega_1 - \Omega_2$ and finally $\mu \in (\Omega_1 - \Omega_2) \setminus \Omega_3$. □

3 Incremental SPARQL Evaluation

We want to evaluate SPARQL over an increasing dataset while we are interested in the result of a query after each addition to the data as outlined in the introduction. This can certainly be achieved by a complete evaluation over the whole data. However, we think that an approach that takes previously computed results into account might perform better. Therefore we provide an incremental adaption for each algebra operation to compute the result based on the changes of the operands between the dataset D and the increased dataset $D + \Delta_D$. We consider also the mechanism to select the active graph (Graph) and the evaluation of triple patterns.

Definition 1 (Insertions and Deletions). *For a SPARQL expression P, an RDF dataset D, and a named graph Δ_D, let $A = [[P]]_G^D$ and $A' = [[P]]_G^{D+\Delta_D}$. We define* insertions $\Delta_A^+ := A' - A$ *and* deletions $\Delta_A^- := A - A'$.

It follows that (i) $\Delta_A^+ \cap \Delta_A^- = \varnothing$, (ii) $\Delta_A^+ \cap A = \varnothing$, (iii) $\Delta_A^- \cap A' = \varnothing$, (iv) $\Delta_A^- \subseteq A$, and (v) $A' = (A - \Delta_A^-) \cup \Delta_A^+$. This must hold in the following transformations.

3.1 Algebra operations

For the transformations of *union*, *join*, and *minus* assume that $A = [[P_1]]_G^D$, $B = [[P_2]]_G^D$, $A' = [[P_1]]_G^{D+\Delta_D}$, and $B' = [[P_2]]_G^{D+\Delta_D}$ have already been computed.

Projection. Given $C_\pi = [[\textsc{Select}_{S,F_1,F_2}(P)]]$, $A = [[P]]_{G_0}^D$, $A' = [[P]]_{G_0}^{D+\Delta_D}$, and $\Delta_D = \langle \mathrm{f}(u), G \rangle$, we are interested in $C'_\pi = [[\textsc{Select}_{S,F_1 \cup \{u\},F_2 \cup \{u\}}(P)]]$. Δ_π^-, Δ_π^+ can be used when projections are pushed down in query optimizations.

$$[[\textsc{Select}_{S,F_1 \cup \{u\},F_2 \cup \{u\}}(P)]] = \pi_S([[P]]_{G_0}^{D+\Delta_D}) = \pi_S(A')$$

$$\begin{aligned}
&= \{\mu \mid \mathrm{dom}(\mu) \subseteq S \wedge \exists \mu'\colon\ \mathrm{dom}(\mu') \cap S = \varnothing \wedge \mu \cup \mu' \in A'\} \\
&= \{\mu \mid \mathrm{dom}(\mu) \subseteq S \wedge \exists \mu'\colon\ \mathrm{dom}(\mu') \cap S = \varnothing \wedge \mu \cup \mu' \in (A - \Delta_A^-)\} \\
&\quad \cup \{\mu \mid \mathrm{dom}(\mu) \subseteq S \wedge \exists \mu'\colon\ \mathrm{dom}(\mu') \cap S = \varnothing \wedge \mu \cup \mu' \in \Delta_A^+\} \\
&= (\pi_S(A) - \{\mu \in \pi_S(\Delta_A^-) \mid \neg\exists \mu' \in A'\colon \mu \sqsubseteq \mu'\}) \cup \pi_S(\Delta_A^+) \\
\Delta_\pi^- &= \{\mu \in \pi_S(\Delta_A^-) \mid \neg\exists \mu' \in A'\colon \mu \sqsubseteq \mu'\} \\
\Delta_\pi^+ &= \{\mu \in \pi_S(\Delta_A^+) \mid \mu \notin \pi_S(A)\}
\end{aligned}$$

Selection. Given $C_\sigma = [[P \text{ FILTER } R]]_G^D$, we are interested in $C'_\sigma = [[P \text{ FILTER } R]]_G^{D+\Delta_D}$. Assume $A = [[P]]_G^D$ and $A' = [[P]]_G^{D+\Delta_D}$ have been computed yet.

$$\begin{aligned}
[[P \text{ FILTER } R]]_G^{D+\Delta_D} &= \sigma_R((A - \Delta_A^-) \cup \Delta_A^+) \\
&= \sigma_R(A - \Delta_A^-) \cup \sigma_R(\Delta_A^+) && \text{(FUPush)} \\
&= (\sigma_R(A) - \sigma_R(\Delta_A^-)) \cup \sigma_R(\Delta_A^+) && \text{(FDPush)} \\
\Delta_\sigma^- &= \sigma_R(\Delta_A^-) \\
\Delta_\sigma^+ &= \sigma_R(\Delta_A^+)
\end{aligned}$$

Union. Given $C_\cup = [[P_1 \text{ UNION } P_2]]_G^D$, find $C'_\cup = [[P_1 \text{ UNION } P_2]]_G^{D+\Delta_D}$.

$$\begin{aligned}
&[[P_1 \text{ UNION } P_2]]_G^{D+\Delta_D} = A' \cup B' \\
&\quad = ((A - \Delta_A^-) \cup \Delta_A^+) \cup ((B - \Delta_B^-) \cup \Delta_B^+) \\
&\quad = ((A - \Delta_A^-) \cup (B - \Delta_B^-)) \cup (\Delta_A^+ \cup \Delta_B^+) \qquad \text{(UAss, UComm)} \\
&\quad = \{\mu \in A \cup B \mid (\mu \in A \wedge \mu \notin \Delta_A^-) \vee (\mu \in B \wedge \mu \notin \Delta_B^-)\} \cup (\Delta_A^+ \cup \Delta_B^+) \\
&\Delta_\cup^- = \{\mu \in A \cup B \mid (\mu \notin A \cup \Delta_A^+ \vee \mu \in \Delta_A^-) \wedge (\mu \notin B \cup \Delta_B^+ \vee \mu \in \Delta_B^-)\} \\
&\Delta_\cup^+ = \{\mu \in \Delta_A^+ \cup \Delta_B^+ \mid \mu \notin A \cup B\}
\end{aligned}$$

Join. Given $C_\bowtie = [[P_1 \text{ AND } P_2]]_G^D$, find $C'_\bowtie = [[P_1 \text{ AND } P_2]]_G^{D+\Delta_D}$.

$$\begin{aligned}
&[[P_1 \text{ AND } P_2]]_G^{D+\Delta_D} = A' \bowtie B' \\
&\quad = ((A - \Delta_A^-) \cup \Delta_A^+) \bowtie ((B - \Delta_B^-) \cup \Delta_B^+) \\
&\quad = ((A - \Delta_A^-) \bowtie (B - \Delta_B^-)) \\
&\qquad \cup ((A - \Delta_A^-) \bowtie \Delta_B^+) \cup (\Delta_A^+ \bowtie B') \qquad \text{(JUDistR, JUDistL)} \\
&\quad = \{\mu \in A \bowtie B \mid \exists \mu_1 \in A, \mu_2 \in B\colon \mu = \mu_1 \cup \mu_2 \wedge \mu_1 \notin \Delta_A^- \wedge \mu_2 \notin \Delta_B^-\} \\
&\qquad \cup ((A - \Delta_A^-) \bowtie \Delta_B^+) \cup (\Delta_A^+ \bowtie B') \\
&\Delta_\bowtie^- = \{\mu \in A \bowtie B \mid \forall \mu_1 \in A \cup \Delta_A^+, \forall \mu_2 \in B \cup \Delta_B^+\colon \\
&\qquad\qquad (\mu_1 \cup \mu_2 = \mu) \to (\mu_1 \in \Delta_A^- \vee \mu_2 \in \Delta_B^-)\} \\
&\Delta_\bowtie^+ = \{\mu \in ((A - \Delta_A^-) \bowtie \Delta_B^+) \cup (\Delta_A^+ \bowtie B') \mid \mu \notin A \bowtie B\}
\end{aligned}$$

Minus. Given $C_{\setminus} = A \setminus B$, we are interested in $C'_{\setminus} = A' \setminus B'$.

$$\begin{aligned}
C'_{\setminus} &= ((A - \Delta_A^-) \cup \Delta_A^+) \setminus B' \\
&= ((A - \Delta_A^-) \setminus ((B - \Delta_B^-) \cup \Delta_B^+)) \cup (\Delta_A^+ \setminus B') \quad \text{(MUDistR)} \\
&= ((A - \Delta_A^-) \setminus ((B \cup \Delta_B^+) - \Delta_B^-)) \cup (\Delta_A^+ \setminus B') \quad (\Delta_B^- \cap \Delta_B^+ = \varnothing) \\
&= ((A - \Delta_A^-) \setminus (B \cup \Delta_B^+)) \\
&\quad \cup \{\mu \in A - \Delta_A^- \mid \forall \mu' \in (B \cup \Delta_B^+) \colon \mu \sim \mu' \to \mu' \in \Delta_B^-\} \cup (\Delta_A^+ \setminus B') \\
&= (((A \setminus B) - \Delta_A^-) \setminus \Delta_B^+) \qquad\qquad\qquad\qquad \text{(MMUCorr, MDReord)} \\
&\quad \cup \{\mu \in A - \Delta_A^- \mid \forall \mu' \in (B \cup \Delta_B^+) \colon \mu \sim \mu' \to \mu' \in \Delta_B^-\} \cup (\Delta_A^+ \setminus B') \\
\Delta_{\setminus}^- &= \{\mu \in A \setminus B \mid \mu \in \Delta_A^- \vee \exists \mu' \in \Delta_B^+ \colon \mu \sim \mu'\} \\
\Delta_{\setminus}^+ &= \{\mu \in A - \Delta_A^- \mid \forall \mu' \in (B \cup \Delta_B^+) \colon \mu \sim \mu' \to \mu' \in \Delta_B^-\} \cup (\Delta_A^+ \setminus B')
\end{aligned}$$

Left Join. $C'_{\sqsupset\!\!\bowtie} = [[P_1 \text{ OPT } P_2]]_G^{D+\Delta_D} = A' \sqsupset\!\!\bowtie B'$ can be expressed by join and minus, thus $C'_{\sqsupset\!\!\bowtie} = C'_{\bowtie} \cup C'_{\setminus}$, $\Delta^-_{\sqsupset\!\!\bowtie} = \Delta^-_{\bowtie} \cup \Delta^-_{\setminus}$, and $\Delta^+_{\sqsupset\!\!\bowtie} = \Delta^+_{\bowtie} \cup \Delta^+_{\setminus}$.

3.2 Active Graph Selection and Triple Patterns

The selection of the active graph propagates to the subexpressions and finally takes effect in the evaluation of triple patterns. The active graph can also be changed inside the scope of a GRAPH-operator, yet it is not possible to reactivate the default graph. For example, $[[u_1 \text{ GRAPH } (P_1 \text{ AND } (u_2 \text{ GRAPH } P_2))]]_G^D$ is equivalent to $[[P_1]]^D_{\mathrm{gr}(u_1)_D} \bowtie [[P_2]]^D_{\mathrm{gr}(u_2)_D}$, $u_i \in I$.

Default Graph. Let t be a triple pattern and $\Delta_D = \langle u, G \rangle$. Given $C_{\mathrm{DG}} = [[t]]^D_{\mathrm{dg}(D)}$, we are interested in $C'_{\mathrm{DG}} = [[t]]^{D+\Delta_D}_{\mathrm{dg}(D+\Delta_D)}$.

$$\begin{aligned}
[[t]]^{D+\Delta_D}_{\mathrm{dg}(D+\Delta_D)} &= [[t]]^{D+\langle u,G\rangle}_{\mathrm{dg}(D+\langle u,G\rangle)} = [[t]]^{D+\langle u,G\rangle}_{\mathrm{dg}(D)\sqcup G} \\
&= \{\mu \mid \mathrm{dom}(\mu) = \mathrm{vars}(t) \wedge t \in \mathrm{dg}(D) \sqcup G\} \\
&= \{\mu \mid \mathrm{dom}(\mu) = \mathrm{vars}(t) \wedge t \in \mathrm{dg}(D)\} \\
&\quad \cup \{\mu \mid \mathrm{dom}(\mu) = \mathrm{vars}(t) \wedge t \in G\} \\
&= [[t]]^D_{\mathrm{dg}(D)} \cup [[t]]^{\langle u,G\rangle}_G \\
\Delta^+_{\mathrm{DG}} &= \{\mu \in [[t]]^{\langle u,G\rangle}_G \mid \mu \notin [[t]]^D_{\mathrm{dg}(D)}\}
\end{aligned}$$

Fixed Graph. The expression u GRAPH P with fixed graph name u is evaluated as $[[P]]^D_{\mathrm{gr}(u)_D}$. Let t be a triple pattern and $C_{\mathrm{FG}} = [[t]]^D_{\mathrm{gr}(u)_D}$. We are interested in $C'_{\mathrm{FG}} = [[t]]^{D+\Delta_D}_{\mathrm{gr}(u)_{D+\Delta_D}}$ with $\Delta_D = \langle u', G \rangle$. The expression is rewritten like above.

$$[[t]]^{D+\Delta_D}_{\mathrm{gr}(u)_{D+\Delta_D}} = \begin{cases} [[t]]^D_{\mathrm{gr}(u)_D} & \text{if } u \in \mathrm{names}(D) \\ [[t]]^{\{\langle u',G\rangle\}}_{\mathrm{gr}(u)_{\Delta_D}} & \text{if } u = u' \end{cases}$$

$$\Delta^{+}_{\mathrm{FG}} = \begin{cases} [[t]]^{\{\langle u',G\rangle\}}_{\mathrm{gr}(u)_{\Delta_D}} & \text{if } u = u' \\ \varnothing & \text{else} \end{cases}$$

Variable Graph. With variable graph name, $?x$ $\textsc{Graph}$ P is evaluated as $\bigcup_{u_i \in \mathrm{names}(D)} \big([[u_i \ \textsc{Graph} \ P]]^{D}_{G} \bowtie \{\{x \mapsto u_i\}\}\big)$. Unlike before, it cannot be completely pushed down to the subexpressions. Let be $\Delta_D = \langle u, G\rangle$ as above. Given $C_{\mathrm{VG}} = \big([[P]]^{D}_{\mathrm{gr}(u_1)_D} \bowtie \{\{x \mapsto u_1\}\}\big) \cup \ldots \cup \big([[P]]^{D}_{\mathrm{gr}(u_n)_D} \bowtie \{\{x \mapsto u_n\}\}\big)$ and $A_i = [[P]]^{D}_{\mathrm{gr}(u_i)_D}$, $A'_i = [[P]]^{D+\Delta_D}_{\mathrm{gr}(u_i)_D}$ for $u_i \in \mathrm{names}(D)$ we are interested in $C'_{\mathrm{VG}} = \bigcup_{u_i \in \mathrm{names}(D+\Delta_D)} \big([[u_i \ \textsc{Graph} \ P]]^{D+\Delta_D}_{G} \bowtie \{\{x \mapsto u_i\}\}\big)$.

$$\begin{aligned} &\bigcup_{u_i \in \mathrm{names}(D+\Delta_D)} \big([[P]]^{D+\Delta_D}_{\mathrm{gr}(u_i)_{D+\Delta_D}} \bowtie \{\{x \mapsto u_i\}\}\big) \\ &\qquad = \bigcup_{u_i \in \mathrm{names}(D)} \big(\underbrace{A'_i \bowtie \{\{x \mapsto u_i\}\}}_{=B'_i}\big) \cup \big([[P]]^{D+\Delta_D}_{G} \bowtie \{\{x \mapsto u\}\}\big) \end{aligned}$$

$$\begin{aligned} \Delta^{-}_{\mathrm{VG}} &= \bigcup_i \Delta^{-}_{B_i} \\ \Delta^{+}_{\mathrm{VG}} &= \bigcup_i \Delta^{+}_{B_i} \cup \big([[P]]^{D+\Delta_D}_{G} \bowtie \{\{x \mapsto u\}\}\big) \end{aligned}$$

$\Delta^{-}_{B_i}$ and $\Delta^{+}_{B_i}$ are composed as in Sec. 3.1 and thus disjoint. It holds $\Delta^{+}_{\mathrm{VG}} \cap \Delta^{-}_{\mathrm{VG}} = \varnothing$ because $\mu_1(?x) \neq \mu_2(?x)$ for $\mu_1 \in \Delta^{+}_{B_i}$, $\mu_2 \in \Delta^{-}_{B_j}$ where $i \neq j$.

4 Comparing Incremental and Direct Computation

We evaluate our approach by comparing the incremental computation to the direct computation. We exemplify these considerations for *projection* and leave out the other operations due to space limitations. *Union* behaves similarly, *join* and *minus* are slightly more complicated, and *selection* is easier.

Definition 2 (Projection of mappings). *Let μ be a mapping and S a finite set of variables. The mapping $\mu[S]$ is the* projection *of μ onto S where (i) $\mathrm{dom}(\mu[S]) \coloneqq \mathrm{dom}(\mu) \cap S$ and (ii) $\mu[S](?x) \coloneqq \mu(?x)$.*

Costs of Projection. Let sets with the operations $\textsc{insert}$, $\textsc{delete}$, and $\textsc{fsub}$ to check for a subsuming mapping be given. We do not assume a specific order for the sets. The costs of evaluating an operation *op* are denoted by $\|op\|$.

The direct computation of the projection simply performs '**for** $\mu \in A'$ **do** $\mu' \longleftarrow \mu[S]$; C'_π $\textsc{insert}$ μ' **end**', so its costs can be estimated at $|A'| \cdot (\|\mu[S]\| + \|C'_\pi \ \textsc{insert} \ \mu'\|)$. An achievable proceeding for the incremental case is shown in Alg. 1. Its approximated costs are given in the following sum where α is the number of successful subsumption checks and β the number of changes to C_π.

$$\begin{aligned} |\Delta^{-}_{A}| \cdot \big(\|\mu[S]\| + \|A' \ \textsc{fsub} \ \mu'\|\big) + (|\Delta^{-}_{A}| - \alpha) \big(\|C'_\pi \ \textsc{delete} \ \mu'\| + \|\Delta^{-}_{\pi} \ \textsc{insert} \ \mu'\|\big) \\ + |\Delta^{+}_{A}| \cdot \big(\|\mu[S]\| + \|C'_\pi \ \textsc{insert} \ \mu'\|\big) + \beta \, \|\Delta^{+}_{\pi} \ \textsc{insert} \ \mu'\| \end{aligned}$$

Overestimating the insertions into Δ^{+}_{π} we can conclude that the incremental adaption has fewer costs if $\Delta^{-}_{A} = \varnothing$ and $|A'| > 2 \cdot |\Delta^{+}_{A}|$. Otherwise it depends on the size of Δ^{-}_{A} and opens the way for optimizations.

```
Algorithm 1: Compute $C'_\pi = \pi_S(A')$ incrementally
Data: Mapping sets $C_\pi = \pi_S(A)$, $A'$, $\Delta^-_A$, $\Delta^+_A$, variable set $S$
Result: $C'_\pi = \pi_S(A')$
begin
    $\Delta^-_\pi \longleftarrow \varnothing$; $\Delta^+_\pi \longleftarrow \varnothing$
    for $\mu$ in $\Delta^-_A$ do
        $\mu' \longleftarrow \mu[S]$; if not $A'$ FSUB $\mu'$ then $C_\pi$ DELETE $\mu'$; $\Delta^-_\pi$ INSERT $\mu'$
    for $\mu$ in $\Delta^+_A$ do
        $\mu' \longleftarrow \mu[S]$; if $C_\pi$ INSERT $\mu'$ changes $C_\pi$ then $\Delta^+_\pi$ INSERT $\mu'$
```

Towards an optimizer. A useful cost estimation is subject to the data and especially to the chosen implementation. If the direct evaluation is presumably cheaper, an optimizer may choose to compute C'_π only from A'. However, to support the incremental computation for operations that use C'_π as input, the costs to compute $\Delta^-_\pi := C_\pi - C'_\pi$ and $\Delta^+_\pi := C'_\pi - C_\pi$ must be considered, too.

A different improvement can be achieved by using mapping multi-sets (cf.[20]) that combine a mapping μ with a multiplicity $\mathrm{m}(\mu)$. In the computation of Δ^-_π, it must be checked for each mapping $\mu \in \Delta^-_A$ whether there are still justifications for $\mu[S]$ in A'. By contrast, the multiplicities in a mapping multi-set are evidences for the number of justifications. By defining $-A$ as A with each multiplicity multiplied by -1, we can compute $C'_\pi = \{\mu \in \pi_S(A) \cup (-\pi_S(\Delta^-_A) \cup \pi_S(\Delta^+_A)) \mid \mathrm{m}(\mu) > 0\}$ (assuming that the operations cover multiplicities). Δ^-_π and Δ^+_π can be assigned during the computation of the leftmost union for deletions (μ with $m(\mu) < 0$) and insertions (μ with $m(\mu) > 0$).

5 Conclusion and Future Work

We have presented a novel analysis of incremental SPARQL evaluation. Our results show means to design an optimizer that is able to choose the presumably best computation in the described Linked Data query answering scenario where the immediate processing of new data is desired. We have also proposed an integration of Linked Data resources and the GRAPH-operator based on specific construction of the SPARQL dataset. We think that our findings provide a sound formal basis for further research in this area.

Future Work. Next, we want to transfer the presented analysis to the SPARQL bag semantics and implement our approach with the described possibilities for optimization, and generalize Δ_D in order to let it contain more than one named graph. Our approach may also be useful in combination with local caches. As Linked Data is built on top of HTTP, the cache-control mechanism[6] could be used to detect and update outdated data on the fly in order to integrate the latest information during the query processing.

[6] RFC #2616 Draft Standard at `http://tools.ietf.org/html/rfc2616`

References

1. Angles, R., Gutierrez, C.: The Expressive Power of SPARQL. In: Proceedings of the 7th Int'l Semantic Web Conference (ISWC). Karlsruhe, Germany (2008)
2. Berners-Lee, T.: Long Live the Web: A Call for Continued Open Standards and Neutrality. Scientific American Magazine 12 (2010)
3. Bizer, C., Cyganiak, R., Heath, T.: How to Publish Linked Data on the Web. Tech. Rep., Freie Universität Berlin, The Open University (2007), `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`, Accessed Aug 15, 2011
4. Bizer, C., Heath, T., Berner-Lee, T.: Linked Data – The Story So Far. International Journal on Semantic Web and Information Systems (IJSWIS) 5(3) (2009)
5. Booth, D.: Four uses of a url: Name, Concept, Web Location and Document Instance (Jan 2003), `http://www.w3.org/2002/11/dbooth-names/dbooth-names_clean.htm`, Accessed Aug 15, 2011
6. Buil-Aranda, C., Arenas, M., Corcho, O.: Semantics and Optimization of the SPARQL 1.1 Federation Extension. In: Proceedings of the 8th Extended Semantic Web Conference (ESWC). Heraklion, Greece (2011)
7. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named Graphs. Web Semantics: Science, Services and Agents on the World Wide Web 3(4) (2005)
8. Hannemann, J., Kett, J.: Linked Data for Libraries. In: World Library and Information Congress: 76th IFLA Gen. Conf. and Assy. Gothenburg, Sweden (2010)
9. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.U., Umbrich, J.: Data Summaries for On-Demand Queries over Linked Data. In: Proceedings of the 19th Int'l Conference on World Wide Web (WWW). Raleigh, NC, USA (2010)
10. Hartig, O.: Zero-Knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution. In: Proceedings of the 8th Extended Semantic Web Conference (ESWC). Heraklion, Greece (2011)
11. Hartig, O., Bizer, C., Freytag, J.C.: Executing SPARQL Queries over the Web of Linked Data. In: Proc. of the 8th Int'l Sem. Web Conf. Chantilly, VA, USA (2009)
12. Hayes, P., McBride, B.: RDF Semantics. W3C Recommendation (Feb 2004), `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`, Accessed Aug 15, 2011
13. Jacobs, I., Walsh, N.: Architecture of the World Wide Web. W3C Rec. (Dec 2004), `http://www.w3.org/TR/2004/REC-webarch-20041215/`, Accessed Aug 15, 2011
14. Klyne, G., Caroll, J.J., McBride, B.: Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation (Feb 2004), `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`, Accessed Aug 15, 2011
15. Ladwig, G., Tran, T.: Linked Data Query Processing Strategies. In: Proceedings of the 9th Int'l Semantic Web Conference (ISWC). Shanghai, China (2010)
16. Ladwig, G., Tran, T.: SIHJoin: Querying Remote and Local Linked Data. In: Proc. of the 8th Extended Semantic Web Conference (ESWC). Heraklion, Greece (2011)
17. Nandzik, J., Heß, A., Hannemann, J., Flores-Herr, N., Bossert, K.: Contentus – Towards Semantic Multi-Media Libraries. In: World Library and Information Congress: 76th IFLA General Conf. and Assembly. Gothenburg, Sweden (2010)
18. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. ACM Transactions on Database Systems (TODS) 34(3) (2009)
19. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation (Jan 2008), `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`, Accessed Aug 15, 2011
20. Schmidt, M., Meier, M., Lausen, G.: Foundations of SPARQL Query Optimization. In: Proceedings of the 13th International Conference on Database Theory (ICDT). Lausanne, Switzerland (2010)