# On Query-Update Independence for SPARQL

Nicola Guido, Pierre Genevès, Nabil Layaïda, Cécile Roisin

# On Query-Update Independence for SPARQL

Nicola Guido
nicola.guido@inria.fr

Pierre Genevès
pierre.geneves@cnrs.fr

Nabil Layaïda
nabil.layaida@inria.fr

Cécile Roisin
cecile.roisin@inria.fr

Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
CNRS, LIG, F-38000 Grenoble, France
Inria

## ABSTRACT

This paper investigates techniques for detecting independence of SPARQL queries from updates. A query is independent of an update when the execution of the update does not affect the result of the query. Determining independence is especially useful in the context of huge RDF repositories, where it permits to avoid expensive yet useless re-evaluation of queries. While this problem has been intensively studied for fragments of relational calculus, very few works exist for the standard query language for the semantic web. We report on our investigations on how a notion of independence can be defined in the SPARQL context.

## 1. INTRODUCTION

The Resource Description Framework (RDF) is a graph data format for the representation of information in the Web. An RDF statement is a subject-predicate-object structure, called RDF triple, intended to describe resources and properties of those resources. Due to their homogeneous structure, RDF databases can be considered as labeled directed graphs, where each triple defines an edge from the subject to the object node under label predicate [6].

With SPARQL [7], the W3C has recommended a query language for RDF. SPARQL comes with a powerful graph matching facility, whose basic constructs are called triple patterns. During query evaluation, variables inside these patterns are matched against the RDF input graph. The solution of the evaluation process is then described by a set of mappings, where each mapping associates a set of variables with graph components.

With the introduction of a standard update language for RDF in SPARQL 1.1. Update [7], the query-update independence problem has been introduced.

A query and an update are independent when the query result is not affected by update execution, on any possible input database. Detecting query-update independence is of crucial importance in many contexts, for example to minimize view re-materialization or to ensure isolation, when queries and updates are executed concurrently. Benefits are amplified when query-update independence can be checked by analyzing only the structure of the query against the structure of the update, hence avoiding the costs of (even partial) reevaluation of queries on large datasets. This paper formulates and investigates the problem of SPARQL query-update independence. To the best of our knowledge, it is the first work in this direction.


*CIKM'15,* October 19–23, 2015, Melbourne, Australia.
 ISBN 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806586.

### *Outline and Contributions*

We present some necessary background on RDF and SPARQL in Section 2. We formalize a notion of query-update independence for SPARQL in Section 3. This is the first formalization of the SPARQL query-update independence problem. We discuss characteristics desired for algorithms that effectively check for independence. We introduce a simple condition for detecting independence, which admits an efficient implementation. We discuss the difficulties introduced by SPARQL's open world assumption and draw perspectives for further research in Section 4. Finally we conclude in Section 6.

## 2. PRELIMINARIES

**RDF**. RDF is a graph data format for the representation of information in the Web. An RDF statement is a subject-predicate-object structure, called RDF triple, intended to describe resources and properties of those resources. Let $U, B, L$ be three disjoint infinite sets denoting the set of URIs (identifying a resource), blank nodes (denoting an unidentified resource) and literals (a character string or some other type of data) respectively. We abbreviate any union of these sets as for instance, $UBL = U \cup B \cup L$. More formally, an RDF triple $t$ is a tuple $(s, p, o) \in UB \times U \times UBL$, where $s$ is the subject, $p$ the predicate and $o$ the object. A set of RDF triples is often referred to as an RDF graph.

Example 1 (Rdf Graph). *Consider 10 triples about employers and their informations (all identifiers correspond to* URI*s):*

{*(A1, name, Joe), (A1, email, Joe@y.com), (A2, name, Martin), (A2, fax, 111111), (A3, name, Louisa), (A3, phone, 222222), (A4, name, John), (A4, web, www.John.com), (A4, email, John@y.com), (A4, phone, 333333)* }

The authors are supported in part by ANR TYPEX ANR-11-BS02-00.

**SPARQL**. SPARQL is a W3C recommended query language for RDF. We restrict ourselves to the core fragment of SPARQL over simple RDF graph, which is sufficient for our purposes.

SYNTAX. SPARQL is based on the notion of query patterns defined inductively from triple patterns. A triple pattern is a tuple $tp \in UBV \times UV \times UBLV$, with $V$ a set of variables disjoint from $UBL$. Triple patterns grouped together using SPARQL operators (like AND, OPT, UNION) form query patterns (or graph patterns).

DEFINITION 1 (QUERY PATTERN). *The syntax of a query pattern $q$ is inductively defined as follows:*

$$q ::= tp \mid q \text{ AND } q \mid q \text{ OPT } q \mid q \text{ UNION } q$$

SPARQL provide several kind of queries. We focus on `SELECT` queries which are the core of SPARQL queries.

DEFINITION 2 (SELECT QUERY). *A `SELECT` QUERY is a query of the form $q(\overrightarrow{w})$ where $\overrightarrow{w}$ is a tuple of variables in $V$ which are called distinguished variables, and $q$ is a query pattern.*

EXAMPLE 2 (SELECT QUERY). . *Consider the query:*

`SELECT` ?A ?E ?W
`WHERE` {(?A *email* ?E) OPT(?A *web* ?W)}

*that queries an* RDF *graph for an email* ?E *of a person* ?A *and, if available, for a web page* ?P *of* ?A*, where* ?A,?E,?W *are distingueshed variables and email and web are* URI*s.*

SEMANTICS. The standard semantics of SPARQL queries is given by a partial mapping function $\rho : V \rightarrow UBL$, that assigns RDF terms of an RDF graph to variables of a SPARQL query. For a triple pattern $t$, we denote by $\rho(t)$ the triple obtained by replacing the variables in $t$ according to $\rho$. For a query pattern $q$, we denote by $\rho(q)$ the set of ground triples (triples not containing any variables) obtained by replacing the variables in $q$ according to $\rho$. The set of all possible mapping-sets, each of which represents a SPARQL query solution, is denoted by $\Omega$.

EXAMPLE 3 (QUERY SOLUTION). *Consider the query pattern in Example 2:*

{(?A *email* ?E) OPT (?A *web* ?W}

*that queries the* RDF *graph in Example 1, the query pattern solution $\Omega$ is represented as the set of $\{\rho_1, \rho_2\}$ where:*

- *$\rho_1$ is the result of successful match of the triple pattern* $(?A, email, ?E)$ *against triple* $(A_1, email, Joe@y.com)$
- *$\rho_2$ is the result of successful match of the triple patterns* $(?A, email, ?E)\text{OPT}(?A, web, ?W)$ *against the two triples* $(A_4, email, John@y.com)$ *and* $(A_4, web, www.John.com)$

| | | |
|---|---|---|
| $?A \rightarrow A_2$ | $?E \rightarrow Joe@y.com$ | |
| $?A \rightarrow A_4$ | $?E \rightarrow John@y.com$ | $?W \rightarrow www.John.com$ |

Two mappings $\rho_1$ and $\rho_2$ are said to be compatible (written $\rho_1 \sim \rho_2$) when $\forall ?X \in dom(\rho_1) \cap dom(\rho_2)$ we have $\rho_1(?X) = \rho_2(?X)$. Mappings with disjoint domains are always compatible. Let $\Omega_1$ and $\Omega_2$ be sets of mappings, the following operators are defined:

$$\Omega_1 \bowtie \Omega_2 = \{\rho_1 \cup \rho_2 |\ \rho_1 \in \Omega_1, \rho_2 \in \Omega_2, \rho_1 \sim \rho_2\}$$
$$\Omega_1 \cup \Omega_2 = \{\rho |\ \rho \in \Omega_1 \text{ or } \rho \in \Omega_2\}$$
$$\Omega_1 \setminus \Omega_2 = \{\rho \in \Omega_1 |\ \forall \rho' \in \Omega_2 . \rho \not\sim \rho'\}$$
$$\Omega_1 :\bowtie \Omega_2 = \{(\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2)\}$$

EVALUATION. Now, we can formalize the evaluation of query patterns over an RDF graph $G$ as a function $[\![.]\!]_G$, inductively defined as follows:

$$[\![t]\!]_G = \{\rho \mid dom(\rho) = var(t) \text{ and } \rho(t) \in G\}$$
$$[\![q_1 \text{ AND } q_2]\!]_G = [\![q_1]\!]_G \bowtie [\![q_2]\!]_G$$
$$[\![q_1 \text{ OPT } q_2]\!]_G = [\![q_1]\!]_G :\bowtie [\![q_2]\!]_G$$
$$[\![q_1 \text{ UNION } q_2]\!]_G = [\![q_1]\!]_G \cup [\![q_2]\!]_G$$
$$[\![q\{(\overrightarrow{\omega})\}]\!]_G = \pi_{\overrightarrow{\omega}}([\![q]\!]_G)$$

where $var(t)$ is the set of variables occurring in $t$, $dom(\rho)$ is the domain of $\rho$, and the projection operator $\pi_{\overrightarrow{\omega}}$ selects only those part of the mapping relevant to variables in $\overrightarrow{\omega}$.

**SPARQL Updates**. We now define the syntax and semantics of the SPARQL Update language [7] whose purpose is to modify a RDF database.

SYNTAX. SPARQL Update uses a syntax derived from the SPARQL query language introduced previously. Update operations are performed on a collection of graphs. Possible operations include updating, creating and removing RDF graphs in a collection. In this paper, we consider the operations performed on a single graph such as:

- INSERT DATA - this operation adds some triples, given inlined in the request, into a graph. For example: `INSERT DATA` $\{A_1, phone, 444444\}$
- DELETE DATA - this operation removes some triples, given inlined in the request, if the respective graph contains these triples. For example: `DELETE DATA` $\{A_1, email, Joe@y.com\}$
- INSERT/DELETE - these actions consist of groups of triples to be deleted and groups of triples to be added.

We focus mostly on INSERT/DELETE operations that constitute the fundamental pattern-based actions for graph updates. The specifiation of triples is based on quad patterns.

DEFINITION 3 (QUAD PATTERN). *A quad pattern $q_u$ is inductively defined as follows:*

$$q_u ::= tp \mid q_u \text{ AND } q_u \mid q_u \text{ UNION } q_u \mid q_u \text{ OPT } q_u$$

*where $tp \in UV \times UV \times ULV$. A quad pattern is a query pattern that doesn't allow blank nodes.*

The INSERT/DELETE operation can be used to remove or add triples from/to a graph based on bindings for a quad pattern specified in a WHERE clause.

DEFINITION 4 (SPARQL UPDATE OPERATION [1]). *Let $q_d$, $q_i$, and $q_w$ be quad Patterns, then $u(q_d,q_i,q_w)$, an update operation, has the form*

$$u(q_d, q_i, q_w) = \texttt{DELETE}\ q_d\ \texttt{INSERT}\ q_i\ \texttt{WHERE} q_w$$

SEMANTICS. Intuitively, the semantics of the execution of $u(q_d, q_i, q_w)$ on $G$, denoted as $G_{u(q_d,q_i,q_w)}$ or simply $u(G)$ is defined interpreting both $q_d$ and $q_i$ as "templates" to be instantiated with the solution $\Omega = [\![q_w]\!]_G$

DEFINITION 5 (NAÏVE UPDATE SEMANTICS [1]). *Let $G$ be a triple store, and $u(q_d, q_i, q_w)$ an update operation, then, a naïve update of $G$ with $u(q_d, q_i, q_w)$, denoted $G_{u(q_d,q_i,q_w)}$ is defined as $(G \setminus A_d) \cup A_i$, where $A_d = \bigcup_{\rho \in [\![q_w]\!]_G} \rho(q_d)$ and $A_i = \bigcup_{\rho \in [\![q_w]\!]_G} \rho(q_i)$*

# 3. NOTION OF INDEPENDENCE

In this section we introduce and formalize the query-update independence problem for SPARQL. Intuitively, a query $q$ is independent of an update $u$, if evaluating $q$ after or before $u$ returns the same result. In other terms, independence holds whenever solutions of query patterns remain unchanged during the addition or deletion of triples from the RDF graph. We first recall the closest notion that we have found in the literature:

PROPOSITION 1 (QUERY SOLUTION INVARIANCE[11]). *If $\Omega$ is the set of all solutions for the query pattern $q$, with respect to an* RDF *graph $G$ and for a triple $t$, there exists no mapping $\rho$ from query variables to* RDF *terms such that $t \in \rho(q)$, then $\Omega$ is also the set of all solutions for $G_+ = G \cup \{t\}$ and $G_- = G \setminus \{t\}$*

We generalize this proposition below, taking into consideration an update $u$ that defines a set of triples to add or to delete from a graph G.

## 3.1 Definition

PROPOSITION 2. *Let $q$ be a select query and $u$ be an update. We say that $q$ and $u$ are independent iff*

$$\forall G.\ [\![q]\!]_G = [\![q]\!]_{u(G)}$$

Independence between queries and updates is thus expressed as the equivalence of two evaluations: one evaluation that computes the answer to the query before the update, and a second evaluation that computes the answer after the update.

Detecting independence is important for several reasons. It can be used in view maintenance to identify that some views are independent of certain updates. We can provide greater flexibility by identifying that one query is independent of updates made by another program or person. Finally. We can use independence in query optimization by ignoring parts of the RDF dataset for which updates do not affect a specific query (to ensure isolation).

## 3.2 Requirements for Independence Analyses

**Costs**. Obviously, we can verify at runtime whether an update impacts a query: we simply run the update after a first query evaluation, then re-run the query, and finally compare the results. The overall cost $c$ of such a check is dominated by the cost of evaluating the query (twice) on the whole RDF dataset and the update (once). Thus, a method for testing independence makes sense only if its cost is lower than $c$. A first class of interesting methods regroups all *purely static* analysis methods, whose cost depends only on the size of the query and on the size of the update, and not on the size of the RDF dataset. A second class of interesting methods regroups hybrid static/dynamic methods that might involve evaluation or partial evaluation of the query on a fraction of the RDF dataset, and whose cost lower than $c$ nevertheless depends on the RDF dataset size.

**Dealing with the Open-World Assumption**. While similar query-update independence problems have been intensively studied for other query languages (in particular for fragments of the relational calculus), very few works exist for SPARQL. We believe that one reason for this is due to SPARQL's underlying open world assumption (a.k.a OWA). We recall that OWA applies to a system that has incomplete information. The web, for instance, is traditionally considered as a system with incomplete information. The absence of some information on the web does not mean that this information is false, but simply that this information has not been made explicit: it is unknown. SPARQL, as a query language for RDF, inherits from this assumption. It thus constitutes an open-world framework that allows anyone to make statements about any resource. Any independence analysis method should thus consider this assumption at its core.

In the next section, we propose a simple condition for testing independence, in the presence of OWA.

## 3.3 A Condition for Independence

We define a condition for checking query-update independence, as follows:

THEOREM 1 (CONDITION FOR INDEPENDENCE). *Let $\mathcal{Q}$ be the set of all triple patterns that appear in the query pattern of the query $q$ and $\mathcal{U}$ be the set of all quad patterns that appear in the DeleteClause and InsertClause of an update $u$:*

$$\forall G.\ [\![q]\!]_G = [\![q]\!]_{u(G)} \Leftrightarrow\ \ \mathcal{Q} \cap \mathcal{U} = \emptyset$$

*where $\mathcal{Q} \cap \mathcal{U} = \emptyset$ is defined as follows:*

$$\forall t_q \in \mathcal{Q},\ \forall t_u \in \mathcal{U}, t_q \neq t_u$$

*and* $\quad t_q \neq t_u \Leftrightarrow \forall \varrho_q \in [\![t_q]\!]_G, \forall \varrho_u \in [\![t_u]\!], \varrho_q(t_q) \neq \varrho_u(t_u).$

PROOF. (Sketch) ($\Rightarrow$) By contradiction.
Suppose $\exists G'.[\![q]\!]_{G'} \neq [\![q]\!]_{u(G')}$, it means that there exists at least a triple $t$ that is added or deleted from $G'$ such that $t$ is a triple in the path or in the solution of the query $q$. We can easily check that $t \in \mathcal{Q} \cap \mathcal{U} \neq \emptyset$. In fact, $t \in \mathcal{U}$ holds because $t$ is a triple added to or deleted from $G'$, so $\exists t_u \in \mathcal{U}.t = t_u$ and $t \in \mathcal{Q}$ holds because the result of applying $q$ after $u$ changes, so $\exists t_q \in \mathcal{Q}.t = t_q$.
($\Leftarrow$) By contradiction.
Suppose $\mathcal{Q} \cap \mathcal{U} \neq \emptyset$, by definition, $\exists t_u \in \mathcal{U}, t_q \in \mathcal{Q}.t_u = t_q$. Let $t_u = t_q$, there always exists a ground triple $t$ that matches both $t_q$ and $t_u$ (i.e. replacing variables by blank nodes). Starting from $t$ we can easily construct a graph $G$, such that $[\![q]\!]_G \neq \emptyset$ and $[\![q]\!] \neq [\![q]\!]_{G \setminus \{t\}}$ or $[\![q]\!]_G \neq [\![q]\!]_{G \cup \{t\}}$. □

Our condition above can be checked statically with the query and the update, independently from any particular RDF dataset. Such a test fits in the category of purely static analysis methods. It can easily be implemented in linear time with respect to the size of the query and the update. When independence is detected, it allows to avoid the cost of re-evaluation of queries on large datasets.

# 4. DISCUSSION

We identify two main research directions for the further development of methods aimed at detecting query-update independence:

**Hybrid static/dynamic methods.** The condition that we presented above involves reasoning over all graphs, and is thus purely static. It would be very interesting to develop hybrid (static and dynamic) methods that would also take into account a specific dataset as input. Such methods should satisfy our cost requirements in order to remain relevant (c.f. Section 3.2) but would unlock the potential for many more detections of independence cases.

**Methods supporting schema constraints.** One perspective is the development of methods that analyse the structures of the query and of the update in the presence of constraints on the dataset, when such constraints are available. Typical constraints on RDF datasets include OWL2 constraints that can express, for example, the fact that two classes are disjoint. Such constraints – and more generally constraints expressed with any description logic supporting negation – are game-changers for the independence problem. The potential contradictions that they introduce can be leveraged for detecting more cases of independence.

In order to understand the impact of schema constraints, let us consider the following example.

EXAMPLE 4 (SCHEMA EXAMPLE). *Let $q$ be a query and $u$ be an update defined as follows:*

$$q(\{?\text{X}\}) = \{(?\text{X}, phone, '222222')$$
$$u\Big(\{(?\text{X}, ?\text{Y}, ?\text{Z})\}, \{\}, \{(A_4, phone, ?\text{X})\}\Big)$$

*$q$ and $u$ are dependent under OWA. For example, if we add the triple $t = (\_a, phone, A_3)$ in the graph of Example1, the resulting graph $G$ is such that $[\![q]\!]_G \neq [\![q]\!]_{u(G)}$.*

*However, the situation changes completely if we add schema constraints stating that phone has a domain "Person" and a range "Number" and in addition that "Person" is disjoint from "Number" (which can be easily formulated in* OWL*2 for instance). In this case, we can see that the variable ?*X *of $q$ belongs to "Person', while the ?*X *of $u$ belongs to "Number", so the update does not affect the query result. In the presence of such constraints $q$ and $u$ are independent.*

# 5. RELATED WORK

To the best of our knowledge, our work is the first to formalize and investigate the query-update independence problem for the SPARQL language. Several techniques have been developed for the static analysis of SPARQL, mainly focusing on containment and equivalence of queries [12, 4, 9], possibly in the presence of schemas [4, 5]. Such techniques aim at detecting relations (typically inclusions) between two queries, both evaluated using the same semantics. This common semantics is a key ingredient that these techniques exploit. These techniques hardly extend to the present context where the semantics of an update is significantly different from the semantics of the query. In the presence of schema constraints, as pointed out in [1] several semantics can be provided for SPARQL Update and, there is no "one-size-fits-all" update semantics.

This work is partly inspired by the works on SPARQL query caching [11], where a first notion of query solution invariance for caching is defined. We formalized the notion of query-update independence and made this notion more precise and more general. Outside the SPARQL context, the query-update independence problem has been intensively studied, in particular in the relational context [3, 10] and in the setting of XML structures [2, 8]. However, due to the SPARQL/RDF's open world assumption, these results do not transfer to the SPARQL context.

# 6. CONCLUSION

We present a first formalization of the SPARQL query-update independence problem. We motivate the interest of searching for methods that solve this problem. We also explain difficulties and discuss characteristics desired for algorithms that effectively check for independence. We introduce a simple condition for detecting independence, which admits an efficient implementation. We identify more general classes of interesting methods and draw perspectives for further research.

# 7. REFERENCES


[1] A. Ahmeti and A. Polleres. SPARQL update under RDFS entailment in fully materialized and redundancy-free triple stores. In *OrdRing'13, Co-located with ISWC '14*, pages 21–32, 2013.

[2] M. Benedikt and J. Cheney. Destabilizers and independence of XML updates. *Proc. VLDB Endow.*, 3(1-2):906–917, Sep. 2010.

[3] J. A. Blakeley, N. Coburn, and P.-A. Larson. Updating derived relations: Detecting irrelevant and autonomously computable updates. In *VLDB'89*, number 3, pages 369–400.

[4] M. W. Chekol, J. Euzenat, P. Genevès, and N. Layaïda. PSPARQL query containment. In *DBPL'11*.

[5] M. W. Chekol, J. Euzenat, P. Genevès, and N. Layaïda. SPARQL query containment under SHI axioms. In *AAAI'12*.

[6] C. Gutierrez, C. A. Hurtado, and A. O. Mendelzon. Foundations of semantic web databases. In *PODS'04*, pages 95–106.

[7] S. Harris and A. Seaborne. SPARQL 1.1 query language. W3C Recommendation, March 2013.

[8] M. Junedi, P. Genevès, and N. Layaïda. XML query-update independence analysis revisited. In *DocEng'12*, pages 95–98.

[9] A. Letelier, J. Pèrez, R. Pichler, and S. Skritek. Static analysis and optimization of semantic web queries. In *PODS'12*, pages 89–100.

[10] A. Y. Levy and Y. Sagiv. Queries independent of updates. In *PODS'93*, pages 171–181.

[11] M. Martin, J. Unbehauen, and S. Auer. Improving the performance of semantic web applications with SPARQL query caching. In *ESWC'10*, pages 304–318.

[12] G. Serfiotis, I. Koffina, V. Christophides, and V. Tannen. Containment and minimization of RDF/S query patterns. In *ISWC'05*, volume 3729, pages 607–623.