Disentangling the Notion of Dataset in SPARQL

Daniel Hernández and Claudio Gutierrez

Center for Semantic Web Research,
Department of Computer Science, Universidad de Chile

Abstract. The notion of dataset in SPARQL seems to be neither a simple nor a well defined notion. In this paper we first review the literature, current documentation and SPARQL engines to show the subleties behind this apparently simple notion and some of the ambiguities of its specification. Then we present formal specifications and algorithms to deal with them in practice.

Introduction

The concept of *dataset* in SPARQL is introduced in several different parts of the W3C documentation (for example in [2, 3, 4, 7, 9, 10, 12]). The specification is spread around, leaves open issues, contain subtleties that result in manifold interpretations, and even in some corner cases, contradict each other.

A first source of misunderstandings is triggered by the two related, but different notions of *default dataset* and *dataset description*. A dataset is defined by the standards of SPARQL and RDF as a set $\{G_0, (u_1, G_1), \ldots, (u_n, G_n)\}$ where each G_i is an RDF graph and each u_i is an IRI. G_0 is called the default graph and each pair (u_i, G_i) is called a named graph. The default dataset is the one a SPARQL endpoint uses when no explicit dataset description is provided in a query request. In the mentioned documentation there is no unique model proposed for dataset descriptions. Indeed, the standard defines two formats for dataset descriptions: (i) In the SPARQL grammar as a sequence of '`FROM` u' and '`FROM NAMED` u' clauses where u is an IRI, and (ii) in the HTTP request with the query string parameters '`default-graph-iri`' and '`named-graph-iri`'. If a dataset description is provided, a dataset must be generated from it. The IRI u in the dataset description clause '`FROM NAMED` u' is assumed to be a reference to a resource that serializes a graph G_u. Thus, the reference is indirect. On the contrary, in the resulting dataset, u will name directly the graph G_u.

The second source of misunderstandings comes from the use of blank nodes. In our experience the following concepts related to blank nodes are difficult to understand or have subtleties when applied to datasets: the *scoping graph*, the operation *merge* when composing the *default graph* and the scope limited to files that are specified for *blank node labels* in RDF and SPARQL. Finally, there are extensions that allow blank nodes as names for named graphs and literals as subjects of RDF triples.

Structure of this paper. The paper is structured in two main sections. In the section 1 we thoroughly analize current misunderstantings in the documentation regarding the definition and use of datasets. In section 2 we propose a formal model for dataset descriptions and give algorithms to build and use the query dataset.

1 Datasets in the literature and engines

Can blank nodes be used as names of named graphs? According to the SPARQL specification *"An RDF dataset is a set* $\{G, (u_1, G_1), (u_2, G_2), \ldots, (u_n, G_n)\}$ *where* G *and each* G_i *are graphs, and each* u_i *is an IRI. Each* u_i *is distinct.* G *is called the default graph.* (u_i, G_i)*, are called named graphs."* [4, §18.1.3]. Thus, blank nodes as names of graphs are not allowed in SPARQL. Restricting names to IRIs is consistent with the SPARQL need to use names to retrieve graphs from the Web. How we interpret the clause '`FROM _:b`'?. We cannot use a blank node to identify a resource or in the Web or a graph in the default dataset. The SPARQL grammar does not allow the clause '`GRAPH _:b {` P `}`'; if it where allowed, then '`_:b`' would not reference a specific graph in the dataset because blank nodes in the query and in the data are in different scopes.

Despite the problems that blank nodes introduce in SPARQL, the RDF specification allows blank nodes as names of graphs[1]. *"An RDF dataset is a collection of RDF graphs, and comprises: (i) Exactly one default graph, being an RDF graph. The default graph does not have a name and MAY be empty. (ii) Zero or more named graphs. Each named graph is a pair consisting of an IRI or a blank node (the graph name), and an RDF graph. Graph names are unique within an RDF dataset."* [7, §4]. There is another specification, TriG, that assumes the possibility of blank nodes as names of graphs: *"In a TriG document a graph IRI or blank node may be used as label for more than one graph statements. The graph label of a graph statement may be omitted. In this case the graph is considered the default graph of the RDF Dataset."* [2, §2.2].

Let s be and endpoint and u be the name of the graph G_u in the default dataset of s. Then, according the The SPARQL 1.1 Graph Store Protocol [9] the IRI s`?graph=`u allows to retrieve the graph G_u. What happens if u is a blank node? According to RDF a blank node must never be used as a name to access a resource, because it *"has no intrinsic name."* [7, §9]. Thus, providing an IRI for G_u based on a blank node u contradicts the RDF semantics.

Despite these these issues, in Jena 2.0 and Virtuoso 6.1 blank nodes as names of dataset graphs are supported.

Can an RDF triple have a literal as subject? According to the RDF specification *"An RDF triple consists of three components: the subject, which is an IRI or a blank node; the predicate, which is an IRI; and the object, which is an IRI, a literal or a blank node."* [7, §3.1]. Thus, an RDF triple must not have a literal as

[1] Note that the concept of dataset was not defined in the earlier RDF 1.0 [6] but was included in RDF 1.1 [7], after the inclusion of RDF datasets in SPARQL.

subject. However, the definition of *triple pattern* suggest that RDF triples may include literals as subject, as triple patterns do: *"A triple pattern is member of the set:* $(T \cup V) \times (I \cup V) \times (T \cup V)$*."* [4, §18.1.5]. Note that T denotes the set $I \cup B \cup L$, called the set of RDF terms. Indeed, the inclusion of literals as triple subjects has been accepted by the RDF core Working Group:

> *[The RDF core Working Group] noted that it is aware of no reason why literals should not be subjects and a future WG with a less restrictive charter may extend the syntaxes to allow literals as the subjects of statements.*
>
> —Should the subjects of RDF statements be allowed to be literals?
> `http://www.w3.org/2000/03/rdf-tracking/#rdfms-literalsubjects`

In our experience SPARQL implementations do not support literals as triple subjects. Jena 2.0 and Virtuoso 6.1 raise an error when uploading files with literals in the subject position.

The default dataset. The SPARQL specification states: *"If a query provides such a dataset description, then it is used in place of any dataset that the query service would use if no dataset description is provided in a query."* [4, §13.2]. Thus, every SPARQL endpoint may provide a *default dataset* to be used in the absence of a dataset description. Note that the dataset description could be defined not only in the query but also in parameters of the request: *"The RDF dataset may also be specified in a SPARQL protocol request, in which case the protocol description overrides any description in the query itself."* [4, §13.2]. The description can be included in three forms: A parameter in the IRI of the HTTP request, a parameter in the body of the HTTP request or as dataset clauses in the query [3, §2.1].

Angles and Gutierrez [1] have interpreted the specification in a different way. They assumed that the default dataset has no named graphs and an empty default graph, i.e., the default dataset is always $\{\emptyset\}$. This interpretation follows the principle of running queries against the Web so that the evaluation of a query does not depend on the particular SPARQL endpoint that evaluates it. On the contrary, in the specification, a query may be evaluated against a default dataset of the SPARQL endpoint where the query is submitted.

SPARQL federation allows using more than one dataset in the same query. When a query includes a '`SERVICE` s `{` P `}`' clause, then the SPARQL endpoint identified as s may evaluate the graph pattern P against the default dataset of s. Indeed, the specification [10, §3.2] states that the '`SERVICE`' generate a request to the endpoint identified as s with the query $Q =$ '`SELECT * WHERE {` P `}`'. Thus, as the query Q has no dataset description, the dataset used to evaluate P is the default dataset of the endpoint identified by s.

Note that as subselects cannot include dataset descriptions, all dereferencing of graphs from the Web must be done by the endpoint that receives the whole query. Thus, endpoints that are used to delegate the evaluation of graph patterns can only use their own default datasets.

Jena 2.0 and Virtuoso 1.6 follow the SPARQL specification, i.e., they use the default dataset in the absence of a dataset description and in federated queries.

Can graphs in RDF datasets share blank nodes? The RDF specification is clear in allowing blank nodes to be shared across graphs: *"Blank nodes can be shared between graphs in an RDF dataset."* [7, §4]. The TriG language for serializing datasets supports sharing blank nodes across graphs: *"BlankNodes sharing the same label in differently labeled graph statements are considered to be the same BlankNode."* [2, §2.3.1]. Datasets engines such as Jena and Virtuoso preserve the identity of blank nodes when loading dataset serializations that share blank nodes across named graphs.

Despite the clarity in the standards and the assumptions made by engine developers, this question has been a source of misunderstandings. Mallea, Arenas, Hogan and Polleres [8] assumed that blank nodes cannot be shared across graphs. In a later work [5] they recognize their mistake: *"This clarification may serve as a corrigendum for our previous paper in which we stated that blank nodes cannot be shared across graphs in SPARQL [48]. This statement is misleading in that although blank nodes cannot be shared across scoping graphs, they can be shared across named graphs"*. Perez, Arenas and Gutierrez [11] assumed *"for the sake of the simplicity"* that blank nodes where not shared by graphs in a RDF dataset.

This misunderstanding comes from the principle that blank nodes are scoped to files (that is, according to the specification) and the assumption that each graph must be contained in its own file (which is wrong). *"Blank node identifiers are local identifiers that are used in some concrete RDF syntaxes or RDF store implementations. They are always locally scoped to the file or RDF store, and are not persistent or portable identifiers for blank nodes. Blank node identifiers are not part of the RDF abstract syntax, but are entirely dependent on the concrete syntax or implementation. The syntactic restrictions on blank node identifiers, if any, therefore also depend on the concrete RDF syntax or implementation. Implementations that handle blank node identifiers in concrete syntaxes need to be careful not to create the same blank node from multiple occurrences of the same blank node identifier except in situations where this is supported by the syntax."* [7, §3.2]. *"A blank node is a node that is not a URI reference or a literal. In the RDF abstract syntax, a blank node is just a unique node that can be used in one or more RDF statements, but has no intrinsic name."* [7, §3.2].

The SPARQL specification states that to evaluate a query, files referenced in the dataset description must be retrieved to build the dataset. Nothing is said about graphs that are included in the default dataset. *"If a query provides more than one* `FROM` *clause, providing more than one IRI to indicate the default graph, then the default graph is the RDF merge of the graphs obtained from representations of the resources identified by the given IRIs."* [4, §13.2.1]. *"The* `FROM NAMED` *syntax suggests that the IRI identifies the corresponding graph, but the relationship between an IRI and a graph in an RDF dataset is indirect. The IRI identifies a resource, and the resource is represented by a graph (or, more precisely: by a document that serializes a graph)."* [4, §13.2.2].

The use of the merge operation to combine graphs into the default graph suggests that blank nodes can be shared by the RDF files dereferenced when interpreting the dataset description. Moreover, merge is not applied on named graphs, a fact that suggest that blank nodes coming from different files can be shared across named graphs of the dataset resulting of the evaluation of a dataset description.

What does occur if a dataset description references only one remote file as part of the default graph? The merge operation must be applied over a single graph? How is evaluated the dataset description '`FROM` u `FROM` u'? Must the graph resulting of dereferencing u be merged with itself?

The way in which the merge operation is applied is another source of misunderstandings. *"In an RDF merge, blank nodes in the merged graph are not shared with blank nodes from the graphs being merged."* [4, §13.1]. A different definition of merge was provided by Angles and Gutierrez [1]: *"The merge of graphs, denoted $G_1 + G_2$, is the graph $G_1 \cup G'_2$ where G'_2 is the graph obtained from G_2 by renaming its blank nodes to avoid clashes with those in G_1."* According the SPARQL specification $G_1 + G_2$ does not share blank nodes with G_1 nor G_2 as in the definition provided by Angles and Gutierrez.

We identify three strategies to avoid blank node clashes: (i) Rename blank nodes when interpreting files, ensuring that no blank nodes are shared. (ii) Use the merge operator to combine the graphs that compose the default graph. (iii) Use both strategies, (i) and (ii).

The strategy (i) is not mentioned in the SPARQL specification, but in our experience many people think that it must be used as a direct consequence of the fact that blank nodes are scoped to files. The strategy (i) is sufficient to ensure that blank nodes are scoped to files. On the contrary, the strategy (ii) is not sufficient, as blank nodes in named graphs are not renamed, hence can be shared. The use of merge in strategies (ii) and (iii) may introduce spurious identities for blank nodes. For example, let u be an IRI that references a file that serializes the graph G_u. Let us consider a dataset description that contains both clauses: '`FROM` u' and '`FROM NAMED` u'. Then, a blank node that occurs in G_u may be renamed with a fresh blank node in the default graph when applying the merge operation, losing its link with its occurrence in the named graph (u, G_u).

The documentation about merge is not clear. The SPARQL 1.1 Service Description specification [12] introduces the property UnionDefaultGraph to indicate that a service uses the union of the named graphs as the default graph. A property to describe default graphs as a merge of named graphs is not introduced for service descriptions. This seems preferable as in the specifications of SPARQL and RDF 1.1 named graphs are allowed to share blank nodes. However, it seems contrary to the use of merge to construct the default graph described by several '`FROM`' clauses in the query as the SPARQL specification stated.

Neither Jena nor Virtuoso allow referencing remote graphs in the dataset description. Thus, they never dereference an IRI. Moreover, the merge operation is never performed. Indeed, if (u_1, G_1) and (u_2, G_2) are two named graph in a default graph of an endpoint identified as s, then the default graph specified in the dataset description '`FROM` u_1 `FROM` u_2' is $G_1 \cup G_2$. Thus, no blank node renaming is done by Jena and Virtuoso at the moment of evaluating queries.

The task of renaming blank nodes to ensure the file scope of them is performed by Jena and Virtuoso at the moment of loading RDF files. Both engines rename all blank nodes of the form '`_:bn`' with fresh blank nodes and set fresh blank nodes for nodes that have no label. However, Virtuoso 6.1 use identifiers as `<nodeID://b01>` as blank node identifiers. This kind of identifiers responds true when evaluating the function '`isBlank(`b`)`' but (as IRIs) are not scoped. This use of blank node identifiers in Virtuoso does not follow the standard. Indeed, a blank node identifier can be used in the query to refer a blank node in the data. On the contrary, the standard limits the scope of blank nodes to basic graph patterns and disallows the sharing of blank nodes across basic graph patterns in queries. Jena follows the standard raising the error *"blank node reuse is not allowed in this point"* if a blank occurs in more that one basic graph pattern.

How must be interpreted an IRI that occurs several times in the dataset description of a query? The SPARQL specification leaves this question open in the case where an IRI that occurs in more that one '`FROM`' or '`FROM NAMED`' clauses must be dereferenced one or more times. *"The actions required to construct the dataset are not determined by the dataset description alone. If an IRI is given twice in a dataset description, either by using two* `FROM` *clauses, or a* `FROM` *clause and a* `FROM NAMED` *clause, then it does not assume that exactly one or exactly two attempts are made to obtain an RDF graph associated with the IRI. Therefore, no assumptions can be made about blank node identity in triples obtained from the two occurrences in the dataset description. In general, no assumptions can be made about the equivalence of the graphs."* [4, §12.2.3]. Note that if an RDF file is dereferenced twice the file may change during the attempts to get it, resulting in different files. Thus, the results depend on the policy of the service to handle different versions of an RDF file.

Let us consider the dataset description '`FROM NAMED` u `FROM NAMED` u'. The specification states that names must not be repeated in the dataset. *"Each* u_i *[the IRI that names a graph] is distinct."* [4, §18.1.3]. *"Graph names are unique within an RDF dataset."* [7, §4]. *"Using the same IRI in two or more* `FROM NAMED` *clauses results in one named graph with that IRI appearing in the dataset."* [4, 13.2.2]. Thus the question arises: What is the graph that should be used if u is dereferenced twice (and these copies are not equal)?

Let us consider the dataset description '`FROM` u `FROM` u' and let be G_u the graph obtained of the dereferencing of u (assuming that only one request for u was issued). If the default graph is the union of G_u with itself then the default graph of the dataset will be G_u. On the contrary, if merge is applied, the default graph G_0 will be a graph that duplicate every triple containing a blank node in G_u. Thus G_u will entail G_0 but $G_u \not\subseteq G_0$.

Note that neither Jena nor Virtuoso handle this issue because they do not support dereferencing IRIs that are not names of named graphs in the default datasets.

2 The notion of dataset

The Web is constituted by a finite set of HTTP servers. A server can send HTTP requests and responses to another server. Servers can be disconnected or added to the Web. The internal configuration of a server may change, so the same request can be answered with different responses if the server configuration changes between the requests.

We will assume that servers follow the REST design principles, so a GET request does not change the internal configuration of the server that receives it. As, an SPARQL client that sent only GET requests have no control over changes in the data, then the order and repetition of requests cannot be used by the client to ensure a result.

In the Web, there is a finite set of files serializing RDF graphs. Every file is accessible sending an HTTP request GET u to an HTTP server. The request GET u may result in a response with the file referenced by u if the server s associate u to a file.

In the Web, there is a finite set of particular types of HTTP servers, called SPARQL endpoints, that provide access to datasets via requests that contain SPARQL queries. A SPARQL endpoint is identified by an IRI that can be used to send queries. Let Q be a SPARQL query, then the request GET s`?query=`Q is the SPARQL GET request that sends the query Q to the endpoint identified as s. A successful response with a file serializing the result of Q is sent back if the query is accepted and if no error occurs during the query execution. The result is interpreted as a Boolean value, a sequence of mappings or an RDF graph.

A SPARQL GET request may have the following query string parameters: '`query`' (exactly 1), '`default-graph-uri`' (0 or more) and '`named-graph-uri`' (0 or more). The SPARQL protocol includes also the possibility to send queries via URL-encoded POST or via POST directly. In this paper we will focus in SPARQL queries sent using the HTTP method GET.

The description above roughly indicates how the protocols necessary to evaluate SPARQL queries work. In what follows, we present a formal model that simplifies and make explicit such protocols.

2.1 A data model for SPARQL

Let I, B, L and V be infinite disjoint sets containing the IRIs, the blank nodes, the literals and the variables. Let IB and IBL be the sets $I \cup B$ and $I \cup B \cup L$, respectively. A triple is a tuple in $IB \times I \times IBL$. A graph is a set of triples. A dataset is set $\{G_0, (u_1, G_1), \ldots, (u_n, G_n))\}$ where $G_0, \ldots, G_n$ are graphs, $u_1, \ldots, u_n$ are different IRIs and $n \geq 0$. G_0 is called the default graph and the rest are called the named graphs. A dataset description is a pair (A, B) where A and B are sets of IRIs. A mapping is a partial function with a finite domain in V with range in IBL. There is a function δ that takes a pair (u, t) where u is an IRI and t is a positive real number representing time, and returns either null, or a graph or a dataset. Intuitively, $\delta(u, t)$ is the data that u makes accessible at the instant t. We call endpoints the IRIs that make datasets accessible during a period.

Note that most definitions presented above where taken from the standards and the literature, except the model of the dataset description. The definition of a dataset description as a pair of sets (instead of a list of dataset clauses) gives an unequivocal semantics to the order of dataset clauses and to the repetition of dataset clauses in the syntax of the dataset description. We argued previously that assuming order or considering repetitions makes no sense if the client has no control over how the data changes over time.

2.2 Algorithms for the dataset

A request is a pair of IRIs (a, b) at time t. A response is a tuple (a, b, r) where a and b are IRIs and r is null (interpreted as a not-found error message[2]), a graph, a sequence of mappings or a Boolean value. A query IRI is an IRI u that has the format $s?p$ where prefix s is called the IRI endpoint and the query string p must contain the parameter '`query`' and zero or more parameters '`default-graph-uri`' and '`named-graph-uri`'. The procedure to generate the response to a request is described by the Algorithm 1. Note that this algorithm allows retrieving the whole dataset through the endpoint IRI. This is not supported by the current standard. However, it is possible to download it with queries or with the graph store protocol [9].

Algorithm 1: Response(a, b)

```
Data: A request $(a, b)$ sent at an instant $t$.
Result: The response of the request $(a, b)$.
if $b$ is a query IRI $s?p$ and $\delta(s, t)$ is a dataset then
    return $(s, a, r)$ where $r$ is the result of evaluating the query and parameters
    indicated in the query string $p$ in the endpoint $s$, that is, using the dataset
    $\delta(s, t)$ as the default dataset.
else
    return $(b, a, \delta(b, t))$
end
```

The procedure to generate the dataset description is presented in Algorithm 2. The endpoint must first check the parameters in the request. If no dataset description is provided in such parameters it finds the dataset description in the query.

Each IRI u in a dataset description must be associated with a graph G_u. If the default daset contains a graph named u, then G_u is the graph associated with u in the dataset. Else, the graph G_u is the result of renaming the blank

[2] Every request must result in a response that occurs after the request. In the HTTP protocol, a request may result in manifold unsuccessful results (a time out, an internal error, etc.). For the sake of the simplicity, in our model we consider an unsuccessful result as a null result.

Algorithm 2: DatasetDescription$(a, s?p)$

```
Data: A request $(a, s?p)$ where $s?p$ is a query IRI.
Result: The dataset description of $(a, s?p)$.
Let $Q$ be the value that occurs in $p$ as the property 'query'.
Let $A$ and $B$ be the set of IRIs that occur in $p$ associated to the properties
'default-graph-iri' and 'named-graph-iri', respectively.
if $A \cup B \neq \emptyset$ then
    return $(A, B)$
else if there is at least a 'FROM' or a 'FROM NAMED' clause in $Q$ then
    Let $A$ and $B$ be the set of IRIs that occur in $Q$ in the clauses 'FROM' and
    'FROM NAMED', respectively.
    return $(A, B)$
else
    the request has no dataset description.
end
```

nodes with fresh labels in the response of the request (s, u). This procedure is formally described in Algorithm 3. Finally, Algorithm 4 describes the algorithm to be used when answering the query.

Algorithm 3: DatasetFromDescription(A, B)

```
Data: A dataset description $(A, B)$.
Result: The dataset built from the description $(A, B)$.
Let $G \leftarrow \bigcup G_i$ for all graph $G_i$ in the default dataset (the scoping graph); let
$G_0 \leftarrow \emptyset$ (the default graph); let $D \leftarrow \{G_0\}$ (the resulting dataset); let $D_0$ be the
default dataset.
for $u \in A \cup B$ do
    if exists $(u, G') \in D_0$ then
        Let $G_u \leftarrow G'$
    else
        Send the request $(s, u)$ and let $(u, s, r)$ be its response.
        if $r$ is a graph then
            Let $G'$ be the result of replacing all blank nodes in $r$ with blank
            nodes that does not occurs in $G$;
            Let $G_u \leftarrow G'$
        end
    end
    if $u \in A$ then
        Let $G_0 \leftarrow G_u$
    end
    if $u \in B$ then
        Let $D \leftarrow D \cup \{(u, G_u)\}$
    end
end
return $D$
```

Algorithm 4: Dataset$(a, s?p)$

```
Data: A request $(a, s?p)$ where $s?p$ is a query IRI and $\delta(s,t)$ is a dataset.
Result: The dataset of the request at $t$.
if DatasetDescription$(a, s?p)$ is defined then
    $(A, B) \leftarrow$ DatasetDescription$(a, s?p)$;
    return DatasetFromDescription$(A, B)$
else
    return $\delta(s,t)$.
end
```

3 Conclusions

We have proposed a formal model that defines the dataset that is used to evaluate every graph pattern in a query. The model was designed to be as close as possible to the RDF and SPARQL specifications; to formalize current natural-language specification; and to clarify some ambiguities it has.

Acknowledgements: The authors thank funding to Millennium Nucleus Center for Semantic Web Research under Grant NC120004.

Bibliography

[1] R. Angles and C. Gutierrez. SQL Nested Queries in SPARQL. In *Procedings on the Alberto Meldenzon Workshop (AMW)*, 2010.

[2] C. Bizer and R. Cyganiak. RDF 1.1 Trig: RDF Dataset Language. Recommendation, World Wide Web Consortium, Febrary 2014.

[3] L. Feigenbaum, G. T. Willians, K. G. Clark, and E. Torres. SPARQL 1.1 Protocol. Recommendation, World Wide Web Consortium, 2013.

[4] S. Harris and A. Seaborne. SPARQL 1.1 Query Language. Recommendation, World Wide Web Consortium, 2013.

[5] A. Hogan, M. Arenas, A. Mallea, and A. Polleres. Everything you always wanted to know about blank nodes. *Web Semantics: Science, Services and Agents on the World Wide Web*, 27–28(0):42 – 69, 2014. ISSN 1570-8268. doi: http://dx.doi.org/10.1016/j.websem.2014.06.004. URL `http://www.sciencedirect.com/science/article/pii/S1570826814000481`. Semantic Web Challenge 2013.

[6] G. Klyne and J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. Recommendation, World Wide Web Consortium, February 2004.

[7] G. Klyne, J. Carroll, and B. McBride. RDF 1.1 Concepts and Abstract Syntax. Recommendation, World Wide Web Consortium, February 2014.

[8] A. Mallea, M. Arenas, A. Hogan, and A. Polleres. On blank nodes. In L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy, and E. Blomqvist, editors, *The Semantic Web – ISWC 2011*, volume 7031 of *Lecture Notes in Computer Science*, pages 421–437. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25072-9. doi: 10.1007/978-3-642-25073-6_27. URL `http://dx.doi.org/10.1007/978-3-642-25073-6_27`.

[9] C. Ogbuji. SPARQL 1.1 Graph Store Protocol. Recommendation, World Wide Web Consortium, Mar. 2013.

[10] E. Prud'hommeaux and C. Buil-Aranda. SPARQL 1.1 Federated Query. Recommendation, World Wide Web Consortium, 2013.

[11] J. Pérez, M. Arenas, and C. Gutierrez. Semantics of SPARQL. Technical Report, TR/DCC-2006-17, Universidad de Chile, Octover 2006. URL `http://users.dcc.uchile.cl/~jperez/papers/sparql_semantics.pdf`.

[12] G. T. Willians. SPARQL 1.1 Service Description. Recommendation, World Wide Web Consortium, Mar. 2013.