symmetry

MDPI

Review

RDF 1.1: Knowledge Representation and Data Integration Language for the Web

Dominik Tomaszuk [1,*] and David Hyland-Wood [2]

[1] Institute of Informatics, University of Białystok, Konstantego Ciołkowskiego 1M, 15-245 Białystok, Poland
[2] School of Information Technology and Electrical Engineering, The University of Queensland, 4067 Brisbane, Australia; d.hylandwood@uq.edu.au
* Correspondence: d.tomaszuk@uwb.edu.pl

Received: 9 November 2019; Accepted: 23 December 2019; Published: 2 January 2020

Abstract: Resource Description Framework (RDF) can seen as a solution in today's landscape of knowledge representation research. An RDF language has symmetrical features because subjects and objects in triples can be interchangeably used. Moreover, the regularity and symmetry of the RDF language allow knowledge representation that is easily processed by machines, and because its structure is similar to natural languages, it is reasonably readable for people. RDF provides some useful features for generalized knowledge representation. Its distributed nature, due to its identifier grounding in IRIs, naturally scales to the size of the Web. However, its use is often hidden from view and is, therefore, one of the less well-known of the knowledge representation frameworks. Therefore, we summarise RDF v1.0 and v1.1 to broaden its audience within the knowledge representation community. This article reviews current approaches, tools, and applications for mapping from relational databases to RDF and from XML to RDF. We discuss RDF serializations, including formats with support for multiple graphs and we analyze RDF compression proposals. Finally, we present a summarized formal definition of RDF 1.1 that provides additional insights into the modeling of reification, blank nodes, and entailments.

Keywords: RDF; RDFS; Semantic Web; Linked Data; data model; blank node; reification; complexity; entailment; compression; binary RDF; data integration

1. Introduction

The Resource Description Framework (RDF) version 1.1 is a modern and complete knowledge representation framework that is seemingly under-represented within the traditional knowledge representation research community. We seek to clarify some differences between the way RDF 1.1 was defined in World Wide Web Consortium (W3C) specifications and the ways in which it is reinterpreted during implementation. Firstly, we need to discuss how RDF relates to the broader field of knowledge representation.

Knowledge representation can be seen as the way in which knowledge is presented in a language. More precisely it was clarified by Sowa [1], who presents five characteristics of knowledge representation:

1. It is most fundamentally a surrogate.
2. It is a collection of ontological commitments.
3. It is a fragmental intelligent reasoning theory.
4. It is a pragmatically efficient computation medium.
5. It is a human expression medium.

Natural language can be defined as one of the methods of knowledge representation. The fundamental unit of knowledge in such languages is often a sentence that consists of a set

of words arranged according to grammatical rules. In spite of the existence of grammatical rules that encode expectations of word order, irregularities and exceptions to the rules make it difficult for machines to process natural languages.

The RDF data model was a response to this problem for knowledge representation on the World Wide Web. This language and the concepts from which it originates have enabled free data exchange, formalization, and unification of stored knowledge. RDF was developed iteratively over nearly two decades to address knowledge representation problems at Apple Computer, Netscape Communications Corporation, and the Semantic Web and Linked Data projects at the World Wide Web Consortium.

An assumption in RDF [2] is to define resources by means of the statement consisting of three elements (the so-called RDF *triple*): *subject*, *predicate*, and *object*. RDF borrows strongly from natural languages. An RDF *triple* may then be seen as an expression with *subject* corresponding to the subject of a sentence, *predicate* corresponding to its verb and *object* corresponding to its object [3]. So the RDF language may be categorized into the same syntactic criteria as natural languages. According to these premises, RDF belongs to the group of Subject Verb Object languages (SVO) [4]. The consistency and symmetry of the RDF language allows knowledge representation that is easily processed by machines, and because its structure is similar to that of natural languages, it is reasonably readable for people.

On the other hand, following Lenzerini [5], data integration is the issue of joining data stored at disparate sources, and providing the user with an integrated perspective of these data. Much of the data on the Web is stored in relational databases. A similar amount of data exists in hierarchical files such as XML documents. Integration of all of this data would provide huge profits to the organizations, enterprises, and governments that own the data.

Interoperability is the capability of two or more (different) software systems or their components to interchange information and to use the information that has been shared [6]. In the context of the Web, interoperability is concerned with the support of applications that exchange and share information across the boundaries of existing data sources. The RDF world is a satisfying method for organizing information from different information sources. In particular, RDF can be seen as a general proposition language for the Web, which consolidates data from heterogeneous sources. It can provide interoperability between applications that exchange the data.

Knowledge representation and data integration in the context of RDF is relevant for several reasons, including: Promotes data exchange and interoperability; facilitates the reuse of available systems and tools; enables a fair comparison of Web systems by using benchmarks. In particular, this article shows how the RDF data model can be related to other models.

The RDF language enables large portions of existing data to be processed and analyzed. This produces the need to develop the foundations of this language. This article addresses this challenge by developing an abstract model that is suitable to formalize and explain properties about the RDF data. We study the RDF data model, minimal and maximal representations, and show complexity bounds for the main problems.

1.1. Contributions

When we examine the state of the RDF data model, we see evidence of trade-offs that occurred as various constituencies took part in the design process. Many of these trade-offs were never completely summarized in the RDF standards. Our article reviews a final state of RDF, and to identify areas where this data model is poorly understood. Our contributions are:

1. To compare the RDF reification approaches.
2. To analyze the RDF 1.1 interpretations, entailments and their complexity.
3. To study the RDF blank nodes and their complexity.
4. To compare the various RDF data integration approaches.
5. To compare the RDF 1.1 serialization formats, including multiple graph syntaxes.
6. To compare the RDF binary and compression formats.

In addition, one of our main aims is to isolate a core fragment of RDF, which is the point of a formal analysis. We discuss a formalization of a RDF syntax and semantics, which is beneficial, for example, to help identify relationships among the constructors, to assist in optimization and validations of software implementations, and to identify redundant notions.

1.2. Review Organization

The remainder of this article is as follows: Section 2 presents related work and formalized concepts for RDF. In Section 3 we discuss RDF blank nodes and their complexity. Section 4 analyzes semantics for the RDF and outlines a set of different entailment regimes. Section 5 overviews and compares various proposals for RDF data integration. Section 6 briefly introduces and compares various serialization formats for RDF 1.1. In Section 7 we overview and compare various proposals for RDF compression. Finally, Section 8 provides concluding remarks.

2. Literature Review

In this Section, we present chronologically related works, and show a formalized syntax and concept for RDF.

The pre-version of RDF was published in 1999 [7]. In this document RDF did not have many features known from current versions, e.g., there were no explicit blank nodes. One of the first papers was published in 2000 [8], which presented RDF and RDFS. In 2001 Champin [9] focused on the RDF model and XML syntax of RDF. In [10] Carroll provided a formal analysis of comparing RDF graphs. The author proved that isomorphism of a RDF graph could be reduced to known graph isomorphism problems. In [11], the authors focused on delineating RDFS(FA) semantics for RDF Schema (RDFS), which could interoperate with common first-order languages. Grau [12] continued the RDF(FA) approach and proposed a possible simplification of the Semantic Web architecture. Yang et al. [13] proposed semantics for anonymous resources and statements in F-logic [14]. Another overview was presented in [15] by Berners-Lee and in [16] by Marin.

The RDF 1.0 recommendation [17] has been reviewed by several analyses. In 2004 Gutierrez et al. [18] formalized RDF and investigated the computational aspects of testing entailment and redundancy. In 2005, in [19,20], the authors proposed a logical reconstruction of the RDF family languages. Feigenbaum [21] briefly described RDF with an emphasis on the Semantic Web. In [22,23], the authors discussed fragments of RDF and systematized them. These papers also outlined the complexity bound for ground entailment in the proposed fragment. Yet another approach was provided in [24], which presented the domain-restricted RDF (dRDF). In 2011 and 2012 more descriptions were shown in [25,26], which presented Semantic Web technologies. In 2014 Curé et al. [27] briefly introduced RDF 1.0 and RDFS 1.0. There were also a lot of papers that extended to RDF annotations [28–30] e.g., for fuzzy metrics [31], temporal metrics [32], spatial metrics [33] and trust metrics [34]. There is a separate group of publications on data integration in the context of RDF [35–38]. Furthermore, the term relational database to RDF mapping has been used in [36]. In [35,38] the authors proposed direct mappings, and in [37] indirect mappings are presented.

In order for machines to exchange machine-readable data, they need to agree upon a universal data model under which to structure, represent and store content. This data model should be general enough to provide representation for arbitrary data content regardless of its structure. The data model should also enable the processing of this content. The core data model selected, for use on the Semantic Web and Web of Data digital ecosystems is RDF.

RDF constitutes a common method of the conceptual description or information modeling accessible in Web resources. It provides the crucial foundation and framework to support the description and management of data. Especially, RDF is a general data model for resources and relationship descriptions between them.

The RDF data model rests on the concept of creating web-resource statements in the form of subject-predicate-object expressions, which in the RDF terminology, are referred to as *triples*.

An RDF triple consists of a subject, a predicate, and an object. In [39], the meaning of subject, predicate and object is clarified. The *subject* expresses a resource, the *object* fills the value of the relation, the *predicate* refers to the features or aspects of resource and expresses a subject-object relationship. The predicate indicates a binary relation, also known as a property.

Following [39], we provide definitions of RDF triples below.

Definition 1 (RDF triple)**.** *Assume that $\mathcal{I}$ is the set of all Internationalized Resource Identifiers (IRIs), $\mathcal{B}$ (an infinite) set of blank nodes, and $\mathcal{L}$ the set of literals. An RDF triple is defined as a triple $t = \langle s, p, o \rangle$ where $s \in \mathcal{I} \cup \mathcal{B}$ is called the subject, $p \in \mathcal{I}$ is called the predicate and $o \in \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$ is called the object.*

Example 1. *The example presents an RDF triple consisting of subject, predicate and object.*

```
<http://example.com/me#john>   foaf:firstName   "John" .
```

The primitive constituents of the RDF data model are terms that can be utilized in reference to resources: anything with identity. The set of the terms is divided into three disjoint subsets:

- IRIs;
- literals;
- blank nodes.

Definition 2 (IRIs)**.** *IRIs are a set of Unicode names in registered namespaces and addresses referring to registered protocols or namespaces used to identify a resource. For example, <`http://dbpedia.org/resource/House`> is used to identify the house in DBpedia [40].*

Note that in RDF 1.0 identifiers were RDF URI (Uniform Resource Identifier) References. Identifiers in RDF 1.1 are now IRIs, which are URIs generalization, which allows a wider range of Unicode codes. Please notice that each absolute URL (and URI) is an Internationalized Resource Identifier, but not every IRI is a URI. When one is implemented in operations that are exclusively specified for URI, it should first be transformed.

IRIs can be shortened. RDF syntaxes use two similar ways: CURIE (compact URI expressions) [41] and QNames (qualified names) [42]. Both are comprised of two components: An optional prefix and a reference. The prefix is separated from the reference by a colon. The syntax of QNames is restrictive and does not allow all possible IRIs to be represented, i.e., `issn:15700844` is a valid CURIE but an invalid QName. Syntactically, QNames are a subset of CURIEs.

Definition 3 (Literals)**.** Literals *are a set of lexical forms and datatype IRIs. A lexical form is a Unicode string, and a datatype IRI identifies an attribute of data that defines how the user intends to use the data. RDF borrows many of the datatypes defined in XML Schema 1.1 [43]. For example, `"1"^^http://www.w3.org/2001/XMLSchema#unsignedInt`, where `1` is a lexical form and should be treated as unsigned integer number.*

Note that in RDF 1.0 literals with a language tag do not support a datatype URI. In RDF 1.1 literals with language tags have the datatype IRI `rdf:langString`. In current version of RDF all literals have datatypes. Implementations might choose to support syntax for literals that have lexical form only, but it should be treated as synonyms for `xsd:string` literals. Moreover, RDF 1.1 may support the new datatype `rdf:HTML`. Both `rdf:HTML` and `rdf:XMLLiteral` depend on DOM4 (Document Object Model level 4) (DOM4 is an interface that treats an XML or HTML elements as objects, see http://www.w3.org/TR/dom/).

Definition 4 (Blank nodes)**.** *Blank nodes are defined as elements of an infinite set disjoint from IRIs and literals.*

In RDF 1.1 blank node identifiers are local identifiers adapted in particular RDF serializations or implementations of an RDF store.

A set of RDF triples represents a labeled directed multigraph. The nodes are the subjects and objects of their triples. RDF is also related to as being *graph structured data* where each $\langle s, p, o \rangle$ triple can be interpreted as an edge $s \xrightarrow{p} o$.

Definition 5 (RDF graph). *Let $\mathcal{O} = \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$ and $\mathcal{S} = \mathcal{I} \cup \mathcal{B}$. $G \subset \mathcal{S} \times \mathcal{I} \times \mathcal{O}$ is a finite subset of RDF triples and called a RDF graph.*

Example 2. *The example in Figure 1 presents a RDF graph of a FOAF [44] profile. This graph includes four RDF triples:*

```
<#js>   rdf:type    foaf:Person  .
<#js>   foaf:name   "John Smith" .
<#js>   foaf:workplaceHomepage <http://univ.com/> .
<http://univ.com/>  rdfs:label "University"       .
```

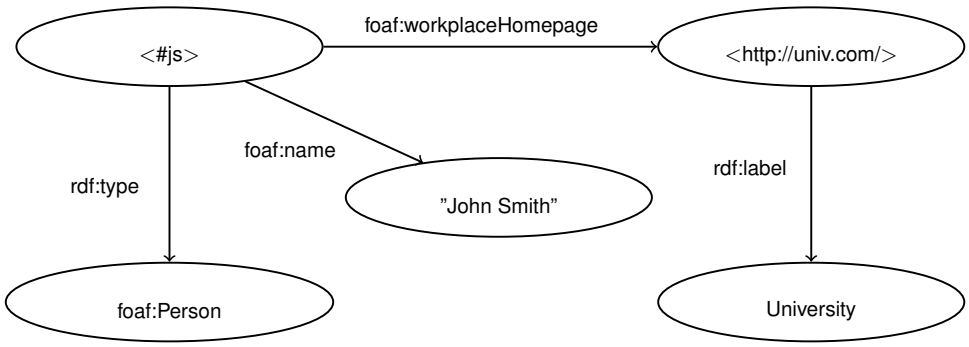

Figure 1. A Resource Description Framework (RDF) graph with four triples.

The RDF syntax and semantics can be widened to named graphs [45]. The named graph data model is a variation of the RDF data model. The basic concept of the model consists of proposing a graph naming mechanism.

Definition 6 (Named graph). *A named graph NG is a pair $\langle u, G \rangle$, where $u \in \mathcal{I} \cup \mathcal{B}$ is a graph name and G is a RDF graph.*

Example 3. *The example in Figure 2 presents a named graph of a FOAF profile. This graph has the name* `http://example.com/#people` *and includes three RDF triples:*

```
<#people> {
<#js>  rdf:type   foaf:Person  .
<#js>  foaf:name  "John Smith" .
<#js>  foaf:workplaceHomepage <http://univ.com/> .
}
```

RDF 1.1 describes the idea of RDF datasets, a collection of a distinguished RDF graph and zero or more graphs with context.Whereas RDF graphs have a formal semantics that establishes what arrangements of the universe make an RDF graph true, no agreed model-theoretic semantics exists for RDF datasets. For more about the above characteristics, we refer the interested reader to the *RDF 1.1: On Semantics of RDF Datasets* [46] which specify several semantics in terms of model theory.

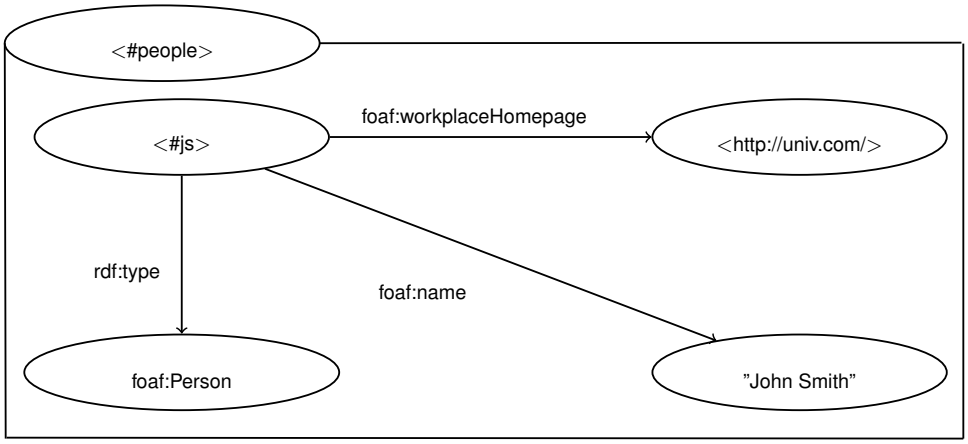

Figure 2. A named graph identified by `<#people>` with three triples.

Definition 7 (RDF dataset)**.** *A RDF dataset DS includes one nameless RDF graph, called the default graph and zero or more named graphs, where each is identified by IRI or blank node,* $DS = \{G, \langle u_1, G_1 \rangle, \langle u_2, G_2 \rangle, \ldots, \langle u_i, G_i \rangle\}$.

Example 4. *This example presents a RDF dataset consisting of one unnamed (default) graph and two named graphs.*

```
{ }  # empty default graph
ex:g1 { ex:s  ex:p  ex:o }
ex:g2 { ex:x  ex:y  ex:z }
```

In addition, the RDF Schema Recommendation [47] provides a collection of built-in vocabulary terms under a core RDF namespace that unifies popular RDF patterns, such as RDF collections, containers, and RDF reification.

RDF is a provider of vocabularies for container description. Each container has a type, and their members can be itemized with the use of a fixed set of container membership properties. In order to provide a way to separate the members from one another, the properties must be indexed by integers; however, these indexes can not be regarded as specifying an ordering of the RDF container itself. The RDF containers are RDF graph entities that use the vocabulary in order to provide basic information about the entities and give a description of the container members. Following [47], RDF gives vocabularies for specifying three container classes:

- `rdf:Bag` is an unordered container and allows duplicates;
- `rdf:Seq` is an ordered container;
- `rdf:Alt` is considered to define a group of alternatives.

Example 5. *This example presents an* `rdf:Bag` *container representing the group of resources.*

```
<http://example.com/p> ex:teachers _:a .
_:a rdf:type rdf:Bag .
_:a rdf:_1 <http://example.com/p/js> .
_:a rdf:_2 <http://example.com/p/ak> .
```

Another feature of RDF is its vocabulary for describing RDF collections. Since the RDF data model has no inherent ordering, collections can be used to determine an ordered and linear collection using a linked list pattern. RDF collections are in the form of a linked list structure such that it comprises of elements with a member and a pointer to the next element. Moreover, collections are closed lists in contrast to containers, which allow the set of items in the group to be precisely determined by applications. However cyclic or unterminated lists in RDF are possible.

Example 6. *This example presents a collection representing the group of resources. In the graphs, each member of the collection is the object of the* `rdf:first` *predicate whose subject is a blank node representing a list, that links by the* `rdf:rest` *predicate. The* `rdf:rest` *predicate, with the* `rdf:nil` *resource as its object, indicates the end of the list.*

```
<http://example.com/p> ex:teachers _:x .
_:x rdf:first <http://example.com/p/js> .
_:x rdf:rest _:y .
_:y rdf:first <http://example.com/p/ak> .
_:y rdf:rest rdf:nil .
```

Another RDF feature is reification (denoted sr in Table 1), which provides an approach to talk about individual RDF triples themselves within RDF. The method allows for constructing a new resource that refers to a triple, and then for adding supplementary information about that RDF statement.

Example 7. *This example presents a standard RDF reification.*

```
ex:t rdf:type      rdf:Statement .
ex:t rdf:subject   <#js> .
ex:t rdf:predicate foaf:name .
ex:t rdf:object    "John Smith" .
ex:t ex2:certainty 0.5 .
```

An extension of the previous method is N-ary Relations [48] (denoted nr in Table 1). This approach is not strictly designed for reification, but focuses on additional arguments in the relation to provide extra information about the relation instance itself.

Example 8. *This example presents an N-ary Relations.*

```
<#js> foaf:name "John Smith" .
<#js> ex2:certainty _:r .
_:r ex2:certainty-value 0.5 .
```

There are other proposals [49,50] of RDF reification. The first proposal called RDF*/RDR (Also known as Reification Done Right (RDR)) [49] (denoted rdr in Table 1) is an alternative approach to represent statement-level metadata. It is based on the idea of using a triple in the subject or object positions of other triples that represent metadata about the embedded statement. To reified RDF data an additional file format based on Turtle has been introduced. Unfortunately, in RDF*/RDR several reification statements of the same triple are translated into one standard reification part, so it is not possible to distinguish grouped annotations.

Example 9. *This example presents a RDF*/RDR.*

```
<<<#js> foaf:name "John Smith">> ex2:certainty 0.5 .
```

The second proposal is called Singleton Property [50] (denoted sp in Table 1). It is for representing statements about statements. It uses a unique predicate for every triple with associated metadata to the statement, which can be linked to the high-level predicate. Authors propose the special predicate `singletonPropertyOf` to link to the original predicate. Since the predicate resource uses the predicate `singletonPropertyOf`, it is possible to use the RDFS entailment rules to infer the original statements.

Example 10. *This example presents a Singleton Property.*

```
foaf:name#1 rdf:singletonPropertyOf foaf:name .
<#js> foaf:name#1 "John Smith" .
foaf:name#1 ex2:certainty 0.5 .
```

It is also possible to use named graphs directly (denoted ng in Table 1). A named graph concept allows assigning an IRI for one or more triples as a graph name. In that scenario, the graph name is used as a subject that can store the metadata about the associated triples. In [51] authors present a model of nanopublications along with a named graph notation.

Example 11. *This example presents a named graph with metadata.*

```
<g1> { <#js> foaf:name "John Smith" }
<g1> ex2:certainty 0.5 .
```

In Table 1 we present features of the above-mentioned standardized serializations, namely: Having W3C Recommendation, a special syntax that is an extension of RDF, and the number of extra RDF statements required to represent a RDF triple (i.e., $O(n)$).

Table 1. RDF reifications. This table presents the features of reifications, namely: Having W3C Recommendation (yes–☑, no–⊠), a RDF-compatible syntax (yes–☑, partial–□, no–⊠) and the number of extra statements.

Feature	sr	nr	rdr	sp	ng
Standard	☑	☑*	⊠	⊠	☑
RDF syntax only	☑	☑	⊠	⊠	□
Extra statements	$4n$	$2n$	$n^{\star\star}$	$2n$	$n^{\star\star\star}$

* W3C Working Group Note
** Counted as n-tripleX
*** Counted as n-quads

3. Modeling Blank Nodes

The standard semantics for blank nodes interprets them as existential variables. We provide an alternative formulation for the blank nodes, and look at theoretic aspects of blank nodes.

Following [52], blank nodes give the capability to:

- Define the information to encapsulate the N-ary association;
- describe reification;
- offer protection of the inner data;
- describe multi-component structures (e.g., RDF containers);
- represent complex attributes without having to name explicitly the auxiliary node.

The complexity of the problem of deciding whether or not two RDF graphs with blank nodes are isomorphic is GI-complete as noticed in [10]. Graph isomorphism complexity with a total absence of blank nodes is PTIME [53].

There is a complication in the notion of RDF graphs, caused by blank nodes. Blank nodes are proposed to be locally-scoped terms that can be seen as existential variables. Blank nodes are shared by graphs only if they are derived from the ones described by documents or RDF datasets that provide for the blank nodes sharing between different graphs. Performing a document download is not synonymous with the blank nodes in a resulting graph being identical to the blank nodes coming from the same source or other downloads of the same file. This makes a notion of isomorphism between RDF graphs that are the same up to blank node relabeling: Isomorphic RDF graphs can be studied as containing the same content.

Definition 8 (Graph isomorphism)**.** *Two RDF graphs G_1 and G_2 are identical in form (are isomorphic, denoted $G_1 \cong G_2$) if there is a bijective function M between the nodes sets of the two graphs, such that:*

1. $M : \mathcal{B} \to \mathcal{B}$;
2. $M_{|\mathcal{L}} = \mathrm{id}_{\mathcal{L}}$, *for all literals that are nodes of graph* G_1;
3. $M_{|\mathcal{I}} = \mathrm{id}_{\mathcal{I}}$, *for all RDF IRIs that are nodes of graph* G_1;
4. $\forall_{s,p,o}(\langle s, p, o\rangle \in G_1 \Leftrightarrow \langle M(s), p, M(o)\rangle \in G_2)$.

Moreover, merging two or more RDF graphs is crucial to ensure that there are no conflicts in blank node labels. A merging operation performs the union after forcing all of shared blank nodes that are present in two or more graphs to be distinct in each RDF graph. The graph after this operation is called the merge. The result of this operation on RDF graphs can produce more nodes than the original graphs.

Definition 9 (RDF merge)**.** *Given two graphs, G_1 and G_2, an RDF merge of these two graphs, denoted by $G_1 \uplus G_2$, is defined as the set union $G_1' \cup G_2'$, where G_1' and G_2' are isomorphic copies of G_1 and G_2 respectively such that the copies do not share any blank nodes.*

As in the RDF graphs we can compare RDF datasets.

Definition 10 (Dataset isomorphism)**.** *Two RDF datasets DS_1 and DS_2 are dataset-isomorphic if there is a function $M : \mathcal{B} \to \mathcal{B}$ between the triples, graphs and nodes in the two datasets, respectively, such that:*

1. *M is the identity map on RDF literals and IRIs, i.e.,* $M_{|\mathcal{I}\cup\mathcal{L}} = \mathrm{id}_{\mathcal{I}\cup\mathcal{L}}$;
2. *for every* $G \in DS_1$ *such that* $G = \{t_1, t_2, \ldots, t_n\}$, $M(G) = \{M(t_1), M(t_2), \ldots, M(t_n)\} \in DS_2$,;
3. *for every* $G \in DS_1$, $\forall_{s,p,o}\langle s, p, o\rangle \in G \Leftrightarrow \langle M(s), M(p), M(o)\rangle \in M(G)$;
4. $\forall_{u,G}(\langle u, G\rangle \in NG_1 \Leftrightarrow \langle M(u), M(G)\rangle \in NG_2)$.

Blank nodes identifiers cannot be found in the RDF abstract syntax. Giving a constant name to the blank nodes can be helped by the skolemization mechanism [54]. In cases where more powerful identification is needed, some or all of the blank nodes can be replaced with IRIs. Systems that wish to do so ought to create a globally unique IRI (called a skolem IRI) for every blank node so replaced. This conversion does not significantly change the graph meaning. It permits the possibility of other RDF graphs by subsequent use the skolem IRIs that is impossible for blank nodes. Systems use a well-known IRI [55] when they need skolem IRIs to be distinguishable outside of the system boundaries with the registered name `genid`.

Definition 11 (Skolemization)**.** *Assume that G is a graph including blank nodes, $\xi : \mathcal{B} \to \mathcal{I}_{skolem}$ is a skolemization injective function and $\mathcal{I}_{skolem}$ is a subset of skolem IRIs which is substituted for a blank node and not occur in any other RDF graph.*

From the above definition it follows that $\mathcal{I}_{skolem} \cap \mathcal{I}_G = \emptyset$, where $\mathcal{I}_G$ is a set of IRIs that are used in G.

On the other hand, in [53], authors propose two skolemization schemes: Centralized and decentralized. The first one is very similar to what the URL-shortening service does. Formally, there is a distinguished subset of the URIs. Whenever the service receives a request, it returns an element that is in Skolem constants such that it has not been used before. The second proposal resembles the first but with no central service. As a result, each publisher will generate its constants locally. Another proposal [54] focuses on a scheme to produce canonical labels for blank nodes, which maps them to globally canonical IRIs. It guarantees that two skolemized graphs will be equal if and only if the two RDF graphs are isomorphic.

NP-completeness originate from cyclic blank nodes. In [56], authors discuss a number of possible alternatives for blank nodes, such as:

1. Disallow blank node;
2. ground semantics;
3. well-behaved RDF.

The first alternative disallows the use of blank nodes in RDF. However, blank nodes are a useful convenience for publishers. The second alternative proposes to assign blank nodes a ground semantics, such that they are interpreted in a similar fashion to IRIs. The third alternative is also presented in [57]. The core motivation for this proposal is to allow implementers to develop tractable lightweight methods that support the semantics of blank nodes for an acyclical case. Following [57], a well-behaved RDF graph is a graph that conforms to the restrictions, which we present below.

Definition 12 (Well-behaved RDF graph)**.** *A well-behaved RDF graph is a RDF graph that conforms to the following limitations:*

1. *It can be serialized as Turtle without the use of explicit blank node identifiers;*
2. *it uses no deprecated features of RDF.*

Note that the first version of RDF published in 1999 did not have named blank nodes and thus was per definition well-behaved.

Another important concept is leanness, which is checking if a RDF graph contains redundancy.

Definition 13 (Subgraph and lean graph)**.** *A subgraph is a subset of a graph G. Assume that a function* $\nu : \mathcal{I} \cup \mathcal{B} \cup \mathcal{L} \rightarrow \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$ *preserving IRIs and literals. A RDF graph G is lean if there is no function* ν *such that* $\nu(G)$ *is a subgraph of G.*

Please note that a subgraph can be a graph with fewer triples. The complexity of the problem of verifying whether or not a RDF graph is lean is coNP-complete as noticed in [58]. Alongside the notion of graphs being non-lean, we also naturally refer to blank nodes as being non-lean. Non-lean blank nodes are the cause of excessive triples in non-lean graphs. A graph is non-lean if and only if it contains one or more non-lean blank nodes.

Example 12. *This example presents that the top graph in Figure 3 is lean, because there is no proper map into itself. The bottom graph in Figure 3 is not lean.*

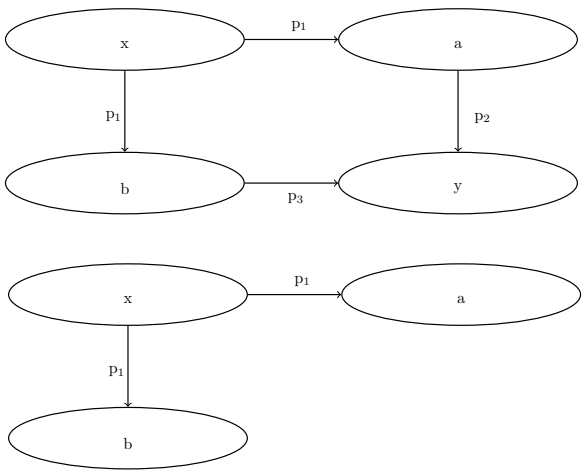

Figure 3. The top graph is lean and the bottom graph is non-lean.

4. Entailments

An interpretation in RDF is a function from literals and IRIs into a set, together with restrictions upon the mapping and the set. In this section we introduce different interpretation notions from the RDF area, each corresponding to an entailment regime in a standard way.

Following [59], a simple interpretation I is a structure which we present below.

Definition 14 (Simple interpretation)**.** *A simple interpretation is* $I = \langle R^I, P^I, EXT^I, INT^I \rangle$*, where:*

1. R^I *is a (nonempty) set of named resources (the universe of I),*
2. P^I *is a set, called the set of properties of I,*
3. EXT^I *is an extension function used to associate properties with their property extension,* $EXT^I : P^I \rightarrow 2^{R^I \times R^I}$*,*
4. INT^I *is the interpretation function which assigns a resource or a property to every element of V such that* INT^I *is the identity for literals,* $INT^I : V \rightarrow R^I \cup P^I$*.*

The interpretation is a map from expressions i.e., triples, graphs and names to truth values and universe elements. Following terminology, we say that I satisfies a RDF graph H when $I(H) = true$, that H is satisfiable if it is a simple interpretation that satisfies it. Moreover, a RDF graph G entails a graph H when every interpretation that satisfies G, will satisfy H as well. In this case, we shall write $G \models H$. The graphs G and H are logically equivalent if each entails the other. Simple entailment can be directly stated in terms of graph homomorphism as noticed in [59].

Ter Horst also proved the NP-completeness of simple entailment by reduction from the clique problem [60]. If the RDF graph H lacks of blank nodes, this issue is in PTIME, because one should only check that each triple in H is in G too.

Following [59], blank nodes are seen as simply indicating a thing's existence, without handling an IRI to identify any name of a resource. We need to describe a version of the simple interpretation mapping that includes the collection of blank nodes as part of its domain.

Definition 15 (Semantic condition for blank nodes)**.** *Let G be a graph and I be a simple interpretation. Let* $\alpha : \mathcal{B} \rightarrow R^I$ *be a map from blank nodes to resources and let* INT_α *denote an amended version of* INT^I *that includes* $\mathcal{B}$ *as part of its domain such that* $INT^I(x) = \alpha(x)$ *for* $x \in \mathcal{B}$ *and* $INT_\alpha(x) = INT^I(x)$ *for* $x \in \mathcal{I} \cup \mathcal{L}$*. We say that I is a model of graph G if there exists a function* α *such that for each* $\langle s, p, o \rangle \in G$*, it holds that* $INT^I(p) \in P^I$ *and* $(INT_\alpha(s), INT_\alpha(o)) \in EXT^I(INT^I(p))$*.*

When two RDF graphs share a blank node, their meaning is not fully captured by treating them in isolation. RDF graphs can be seen as conjunctions of simple atomic sentences in First-Order Logic (FOL) [61], where blank nodes are free variables that are existential.

Further interpretations are depending on which IRIs are known as datatypes. Hayes and Patel-Schneider [59] propose the use of a parameter D (the set of known datatypes) on simple interpretations. The next considered interpretation is D-interpretation.

Definition 16 (D-interpretation)**.** *Let D be a set of IRIs describing datatypes. A D-interpretation is a simple interpretation additionally satisfying the following conditions:*

1. *If* `rdf:langString` $\in D$*, then for every language-tagged string E with lexical form* s_l *and language tag* t_l*,* $L^I(E) = \langle s_l, t'_l \rangle$*, where* t'_l *is* t_l *transformed to lower case,*
2. *For every other IRI* $d \in D$*,* $I(d)$ *is the datatype identified by d, and for every literal* $"s_l"\text{^^}d$*,* $L^I("s_l"\text{^^}d) = L2V(I(d))(s_l)$*, where L2V is a function from datatypes to their lexical-to-value mapping.*

Note that in the RDF 1.0 specification, datatype D-entailment was described as a RDFS-entailment semantic extension. In RDF 1.1 it is defined as a simple direct extension. Moreover, in RDF 1.1 datatype

entailment formally refers to a set of recognized datatypes IRIs. RDF 1.0 used the concept of a datatype map: in the new semantic description, this is the mapping from recognized IRIs to the datatypes they identify.

A graph is satisfiable recognizing D (or simply D-satisfiable) if it has the true in some D-interpretation, and G entails recognizing D (or D-entails) H when every D-interpretation satisfies G, will satisfy satisfies H as well.

In [60] ter Horst proposes the $D*$ semantics, which is a weaker variant of RDFS 1.0 D semantics. This semantics generalizes the RDFS 1.0 semantics [62] to add reasoning with datatypes.

RDF interpretation imposes additional semantic conditions on part of the (infinite) set of IRIs with the namespace prefix `rdf:` and `xsd:string` datatype. In RDF there are three key terms:

- `rdf:Property` (**P**)–the class of RDF properties;
- `rdf:type` (**a**)–the subject is an instance of a class;
- `rdf:langString` (**ls**)–the class of language-tagged string literal values.

Definition 17 (RDF interpretation)**.** *An RDF interpretation is a D-interpretation I where D has* `xsd:string` *and* `rdf:langString`*, which satisfies the following conditions:*

1. $x \in P^I \Leftrightarrow \langle x, I(\mathbf{P})\rangle \in EXT^I(\mathbf{a})$;
2. $\forall_{d \in I}(\langle x, I(d)\rangle \in EXT^I(I(\mathbf{a})) \Leftrightarrow x \in VS(I(d)))$*, where VS is a value space.*

This RDF vocabulary is defined by the RDF Semantics [59] in terms of the RDF model theory. A selection of the inference rules are presented in Table 2. As earlier G RDF entails H recognizing D when RDF interpretation recognizing D that satisfies G, will satisfy H as well. When D is {`xsd:string`, `rdf:langString`} then G RDF entails H.

Table 2. A selection of RDF rules. A rule of the form body ⇒ head has predicates to the left of the ⇒ symbol are the premise of the rule, and predicates to the right of the ⇒ symbol are the consequent.

Rule ID	Body	Head
rdf1	$?s\ ?p\ "s_l"\text{^^}d\ .\ d \in D$	⇒ $?s\ ?p$ _:n . _:n **a** d .
rdf2	$?s\ ?p\ ?o$.	⇒ $?p$ **a P** .

RDF Schema [47] extends RDF to additional vocabulary with more complex semantics. RDFS semantics introduces a class. It is a resource which constitutes a set of things that all have class as a value of their `rdf:type` predicate (so-called property). They are outlined to be things of type `rdfs:Class`. We introduce C^I which is the set of all classes in an interpretation. Additionally, the semantic conditions are stated in terms of a function $CEXT^I : C^I \rightarrow 2^{R^I}$. In RDFS there are ten key terms:

- `rdfs:Class` (**C**)–the class of classes;
- `rdfs:Literal` (**Lit**)–the class of literal values;
- `rdfs:Resource` (**Res**)–the class resource, everything;
- `rdfs:Datatype` (**Dt**)–the class of RDF datatypes;
- `rdfs:subPropertyOf` (**spo**)–the property that allows for stating that all things related by a given property x are also necessarily related by another property y;
- `rdfs:subClassOf` (**sco**)–the property that allows for stating that the extension of one class X is necessarily contained within the extension of another class Y;
- `rdfs:domain` (**dom**)–the property that allows for stating that the subject of a relation with a given property x is a member of a given class X;
- `rdfs:range` (**rng**)–the property that allows for stating that the object of a relation with a given property x is a member of a given class X;

- `rdfs:ContainerMembershipProperty` (**Cmp**)–the class of container membership properties, `rdf:_i`;
- `rdfs:member` (**m**)–a member of the subject resource.

Definition 18 (RDFS interpretation)**.** *An RDFS interpretation is a RDF interpretation which additionally satisfies the following conditions:*

1. $CEXT^I(y) = \{x : \langle x, y\rangle \in EXT^I(INT^I(\mathbf{a}))\}$,
2. $IC = CEXT^I(INT^I(\mathbf{C}))$,
3. $LV = CEXT^I(INT^I(\mathbf{Lit}))$,
4. $CEXT^I(INT^I(\mathbf{Res})) = R^I$,
5. $CEXT^I(INT^I(\mathbf{ls})) = \{INT^I(E) : E \text{ is a language-tagged string }\}$,
6. $\forall_{d\in\mathcal{I}\setminus(\mathbf{Res}\cup\mathbf{ls})} CEXT^I(INT^I(d)) = VS(INT^I(d))$, *where VS is a value space,*
7. $\forall_{d\in\mathcal{I}} INT^I(d) \in CEXT^I(INT^I(\mathbf{Dt}))$
8. $\langle x, y\rangle \in EXT^I(INT^I(\mathbf{dom})) \wedge \langle u, v\rangle \in EXT^I(x) \Rightarrow u \in CEXT^I(y)$,
9. $\langle x, y\rangle \in EXT^I(INT^I(\mathbf{rng})) \wedge \langle u, v\rangle \in EXT^I(x) \Rightarrow v \in CEXT^I(y)$,
10. $\forall_{x,y,z\in P^I}(\langle x, y\rangle \in EXT^I(INT^I(\mathbf{spo})) \wedge \langle y, z\rangle \in EXT^I(INT^I(\mathbf{spo}))) \Rightarrow \langle x, z\rangle \in EXT^I(INT^I(\mathbf{spo}))) \wedge \forall_{x\in P^I}\langle x, x\rangle \in EXT^I(INT^I(\mathbf{spo}))$
11. $\langle x, y\rangle \in EXT^I(INT^I(\mathbf{spo})) \Rightarrow x \in P^I \wedge y \in P^I \wedge CEXT^I(x) \subset CEXT^I(y)$,
12. $x \in C^I \Rightarrow \langle x, INT^I(\mathbf{Res})\rangle \in EXT^I(INT^I(\mathbf{sco}))$
13. $\forall_{x,y,z\in P^I}(\langle x, y\rangle \in EXT^I(INT^I(\mathbf{sco})) \wedge \langle y, z\rangle \in EXT^I(INT^I(\mathbf{sco}))) \Rightarrow \langle x, z\rangle \in EXT^I(INT^I(\mathbf{sco}))) \wedge \forall_{x\in P^I}\langle x, x\rangle \in EXT^I(INT^I(\mathbf{sco}))$
14. $\langle x, y\rangle \in EXT^I(INT^I(\mathbf{sco})) \Rightarrow x \in C^I \wedge y \in C^I \wedge CEXT^I(x) \subset CEXT^I(y)$,
15. $x \in CEXT^I(INT^I(\mathbf{Cmp})) \Rightarrow \langle x, INT^I(\mathbf{m})\rangle \in EXT^I(INT^I(\mathbf{spo}))$,
16. $x \in CEXT^I(INT^I(\mathbf{Dt})) \Rightarrow \langle x, INT^I(\mathbf{Lit})\rangle \in EXT^I(INT^I(\mathbf{sco}))$.

This RDFS vocabulary is defined by the RDF Semantics [59] in terms of the RDF model theory. Selections of the RDFS inference rules are presented in Table 3.

As earlier G RDFS entails H recognizing D when every RDFS interpretation recognizing D which satisfies G, will satisfy H as well.

Table 3. A selection of RDFS rules. A rule of the form body ⇒ head has predicates to the left of the ⇒ symbol are the premise of the rule, and predicates to the right of the ⇒ symbol are the consequent.

Rule ID	Body	Head
rdfs1	any IRI $?p \in D$	⇒ $?p$ **a Dt** .
rdfs2	$?p$ **dom** $?x$. $?y$ $?p$ $?z$.	⇒ $?y$ **a** $?x$.
rdfs3	$?p$ **rng** $?x$. $?y$ $?p$ $?z$.	⇒ $?z$ **a** $?x$.
rdfs4a	$?x$ $?p$ $?y$.	⇒ $?x$ **a Res** .
rdfs4b	$?x$ $?p$ $?y$.	⇒ $?y$ **a Res** .
rdfs5	$?x$ **spo** $?y$. $?y$ **spo** $?z$.	⇒ $?x$ **spo** $?z$.
rdfs6	$?x$ **a P** .	⇒ $?x$ **spo** $?x$.
rdfs7	$?p$ **spo** $?q$. $?x$ $?p$ $?y$.	⇒ $?x$ $?q$ $?y$.
rdfs8	$?x$ **a C**	⇒ $?x$ **sco Res** .
rdfs9	$?x$ **sco** $?y$. $?z$ **a** $?x$.	⇒ $?z$ **a** $?y$.
rdfs10	$?x$ **a C** .	⇒ $?x$ **sco** $?x$.
rdfs11	$?x$ **sco** $?y$. $?y$ **sco** $?z$.	⇒ $?x$ **sco** $?z$.
rdfs12	$?x$ **a Cmp** .	⇒ $?x$ **spo m** .
rdfs13	$?x$ **a Dt** .	⇒ $?x$ **sco Lit** .

Example 13. *This example presents a RDFS interpretation for the vocabulary presented in Example 2.*

$$
\begin{array}{rcccc}
R^I & = & \{\Upsilon, \Phi, \Psi, \Omega, \alpha, \beta, \gamma, \delta, \epsilon\} & & \\
P^I & = & \{\Upsilon, \Phi, \Psi, \Omega\} & & \\
EXT^I & = & \Upsilon & \mapsto & \{\langle \alpha, \beta \rangle\} \\
 & & \Phi & \mapsto & \{\langle \alpha, \delta \rangle\} \\
 & & \Psi & \mapsto & \{\langle \alpha, \gamma \rangle\} \\
 & & \Omega & \mapsto & \{\langle \gamma, \epsilon \rangle\} \\
INT^I & = & \textit{rdf:type} & \mapsto & \Upsilon \\
 & & \textit{foaf:name} & \mapsto & \Phi \\
 & & \textit{foaf:workplaceHomepage} & \mapsto & \Psi \\
 & & \textit{rdfs:label} & \mapsto & \Omega \\
 & & \textit{<\#js>} & \mapsto & \alpha \\
 & & \textit{foaf:Person} & \mapsto & \beta \\
 & & \textit{"John Smith"} & \mapsto & \gamma \\
 & & \textit{<http://univ.com/>} & \mapsto & \delta \\
 & & \textit{"University"} & \mapsto & \epsilon
\end{array}
$$

In Table 4 we summarize results about complexity of entailment problems mentioned above.

Table 4. Computational complexity of entailments for RDF and RDF Schema (RDFS).

Entailment	Current Semantics	No Blank Nodes
simple	NP-complete	PTIME
D*	NP-complete	PTIME
RDF	NP-complete	PTIME
RDFS	NP-complete	PTIME

Example 14. *This example presents that the top graph entails the bottom graph in Figure 4. When we blank the node* `<#js>` *from the top, the bottom graph still has a node that represents a person and preserves semantics. Similarly, when we delete the node* `"John Smith"` *from the top graph, the bottom graph still preserves the graph's semantics.*

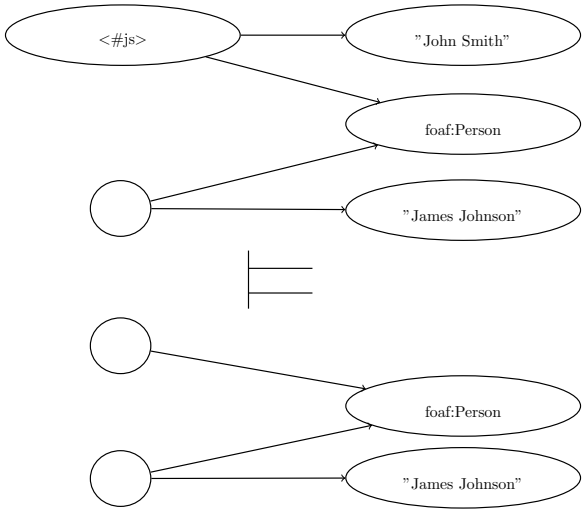

Figure 4. The top graph entails the bottom graph.

5. RDF Data Integration

The role of RDF as an integration system for data from different sources is one of the crucial motivations for research efforts. It is important to provide integration methods to bridge the gap between the RDF and other environments. Following [5], the definition of data integration systems is provided below.

Definition 19 (Data integration system)**.** *A data integration system is a tuple* $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ *where* $\mathcal{G}$ *is the global schema,* $\mathcal{S}$ *is the source schema and* $\mathcal{M}$ *is the mapping between* $\mathcal{G}$ *and* $\mathcal{S}$*, constituted by a set of assertions.*

In this section the approaches of mapping of relational databases to RDF are discussed (Section 5.1) as well as approaches of mapping XML to RDF (Section 5.2). However, there are also more general approaches [63–69].

5.1. Bringing Relational Databases into the RDF

This subsection contains an overview and comparison of the approaches for mapping from a relational database into RDF. Table 5 presents the key approaches from related work. It presents the features of the below-mentioned proposals, namely: Mapping representation (SQL, RDF, XML, etc.), schema representation (RDF, OWL [70] and F-Logic [14]) and level of automation.

At the beginning, we focus on solutions [71–73] based on SQL as mapping representation. Triplify [71] is based on the mapping of HTTP requests onto database queries expressed in SQL queries, which are used to match subsets of the store contents and map them to RDFS classes and properties. It converts the resulting relations into RDF triples and subsequently publishes it in various RDF syntaxes. That proposal includes an approach for publishing update logs in RDF which contain all RDF resources in order to enable incremental crawling of data sources. An additional advantage is that it can be certainly integrated and deployed with the numerous, installed applications on the Web. The next approach is StdTrip [72], which proposes a structure-based framework using existing ontology alignment software. The approach finds ontology mappings between simple vocabulary that is generated from a database. The results of the vocabulary and ontology alignment algorithm are presented as suggestions to the user, who matches the most appropriate ontology mapping. RDOTE [73] also uses SQL for the specification of the data subset. In that proposal, the suitable SQL query is stored in a file. That approach transforms data residing in the database into a RDF graph dump using classes and properties.

Table 5. Relational databases mapping. This table presents the features of transformation approaches, namely: Mapping representation, schema representation and level of automation (automatic–☑, semi-automatic–□ and manual–⊠).

Approaches	Mapping Represent.	Schema Represent.	Automatic
[74,75]	n/a	RDFS, OWL, F-Logic	□
[71]	SQL	RDFS	⊠
[76,77]	Constraint rules	RDFS, OWL	□
[78]	D2RQ	RDFS	☑
[79]	RDF, Rel.OWL	RDFS, OWL	☑
[80]	n/a	RDFS, OWL	□
[81]	FOL, Horn	RDFS, OWL	☑
[82]	SQL	RDFS, OWL	⊠
[83]	Logic rules	RDFS, OWL	□
[84]	n/a	RDFS	□
[85]	XML	RDFS, OWL	⊠
[86]	SPARQL	RDFS	☑*
[87]	R_2O	RDFS, OWL	☑
[88]	RDF	RDFS	⊠
[89]	RDF, Rel.OWL	RDFS, OWL	☑
[90]	D2RQ	RDFS, OWL	□
[91]	XPath (XSLT)	RDFS, OWL	⊠
[72]	SQL	RDFS, OWL	□
[92]	n/a	RDFS, OWL	☑
[93]	n/a	RDFS, F-Logic	□
[94]	FOL	RDFS, OWL	☑
[73]	SQL	RDFS, OWL	⊠
[95]	RDF/XML	RDFS, OWL	⊠
[96]	RDF (Direct)	RDFS	☑*
[97,98]	XQuery	n/a	□

* possible manual level.

The next group of approaches [78,90] uses D2RQ as the mapping representation. D2RQ [78] supports both automatic and manual operation modes. In the first mode, RDFS vocabulary is created, in accordance with the reverse engineering methodologies, for the translation of foreign keys to properties. In the second mode, the contents of the database are exported to a RDF in accordance with mappings stored in RDF. It allows RDF applications to treat non-RDF stores as virtual RDF graphs. One of the disadvantages of D2RQ is a read-only RDF view of the database. Another D2RQ-based proposal is AuReLi [90], which uses several string similarity measures associating attribute names to existing vocabulary entities in order to complete the automation of the transformation of databases. It also tries to link database values with ontology individuals. RDF Views [86] has similar functionality to D2RQ. It supports both automatic and manual operation modes. That solution uses the table of RDFS class and column as a predicate and takes into account cases such as whether a column is part of the unique key. The data is represented as virtual RDF graphs without the physical formation of RDF datasets.

Another group of proposals [88,96] use RDF. The first approach is OntoAccess [88], which is a vocabulary-based write access to data. That paper consists of the relational database to RDF mapping language called R3M, which consists of a RDF format and algorithms for translating queries to SQL. The next proposal is SquirrelRDF [96], which extracts data from a number of databases and integrates that data into a business process. That proposal supports RDF views and allows for the execution of queries against it. In that group we can also distinguish solutions [79,89] based on Relational.OWL [99]. ROSEX [79] uses Relational.OWL to represent the relational schema of a database as an OWL ontology. The created database schema annotation and documentation is mapped to a domain-specific vocabulary, which is achieved automatically by reverse-engineering the schema. An additional advantage is that it supports automatic query translation. DataMaster [89] also uses Relational.OWL for the importing of schema structure and data from relational databases.

R2RML [100] is the best-known language based off RDF for expressing mappings from relational databases to RDF datasets because it is a W3C recommendation and has a lot of implementations [101–103]. R2RML is a language for specifying mappings from relational to RDF data. A mapping takes as input a logical table (`logicalTable` predicate), i.e., a database table, an SQL query, or a database view. In the next step, a logical table is mapped to a triples map that is a set of triples. A triples map has two main parts. The first part is a subject map (`subjectMap` predicate) that generates the subject of all RDF triples that will be generated from a logical table row. The second part is a predicate-object map (`predicateObjectMap` predicate) that specifies the target property and the generation of the object via `objectMap`.

Yet other approaches are DartGrid [95] and MASTRO [85]. DartGrid [95] describes a database integration architecture. It uses a visual mapping system to align the relational database to existing vocabulary. Correspondences between components of the models which are defined via a graphical user interface are created and stored in a RDF/XML syntax (see Section 6.1). The next proposal is MASTRO [85], which is a framework that enables the definition of mappings between a relational database and vocabulary.

Another group of proposals [81,94] use First-Order Logic (FOL) [61]. Tirmizi et al. [94] present formal rules in FOL to transform column to predicate and table to class. The authors present the system which is complete with respect to a space of the possible primary key and foreign key mixtures. The next proposal is MARSON [81], which uses mappings based on virtual documents (called vectors of coefficients). It removes incorrect mappings via validating mapping consistency. Moreover, a special type of semantic mappings (called contextual mappings) is presented in the paper.

Some approaches [76,83,87,91,94,97] adopt other mapping representations. RDBToOnto [76,77] is a tool that simplifies the implementation of methods for vocabulary acquisition from databases. It provides a visual interface for manual modification and adjustment of the learning parameters. RDBToOnto links the data analysis with heuristic rules and generates an ontology. DB2OWL [87] creates local vocabulary from a relational database that is aligned to reference vocabulary. The vocabulary generated in that

proposal reflects the database semantics. The mappings are stored in an R_2O [104] document. The next proposal is SOAM [83], which uses the column to predicate and the table to class approach. It creates an initial schema, which is refined by referring to a dictionary. Constraints are mapped to constraints in the vocabulary schema. That approach tries to establish the quality of the constructed vocabulary. Another proposal is [91]. It is a domain semantics-driven mapping generation approach. Mapping is created on the XSLT [105] and XPath [106]. Yet another approach is XSPARQL [97], which can be used both in relational databases and XML (see Section 5.2).

There are also several approaches [74,80,84,92,93] that do not have a defined mapping representation. Astrova [74,75] discusses correlation among key, data in key attributes between two relations and non-key attributes. In these papers, the quality of transformation is considered. Buccella et al. [80], Shen et al. [92] and Stojanovic et al. [93] examine heuristic rules. Byrne [84] proposes a domain-specific approach for the generic design of cultural heritage data and discusses the options for including published heritage thesauri.

5.2. Bringing XML into the RDF

This subsection overviews and compares the approaches for mapping from XML into RDF. Table 6, presents key approaches from related work. It presents features of the below-mentioned proposals, namely: Existing vocabulary, schema representation (RDF, OWL and DAML+OIL [70]) and level of automation.

Table 6. XML mapping. This table presents the features of transformation approaches, namely: Existing vocabulary (yes–☑, no–☒), schema representation and level of automation (automatic–☑, semi-automatic–□ and manual–☒).

Approaches	Existing Vocabulary	Schema Representation	Auto-Matic
[107]	☒	RDFS, DAML+OIL	☒
[108]	☒	RDFS, OWL	☑
[109]	☒	RDFS, OWL	☑
[110]	☒	n/a	☑
[111]	☒	RDFS, OWL	☑
[98,112]	☒	n/a	□
[113]	☒	RDFS, OWL	☑
[114]	☒	RDFS, OWL	□
[115]	☑	RDFS, OWL	☑
[116]	☑	RDFS, OWL	□
[69]	☒	RDFS, OWL	□
[117]	☒	n/a	□
[118]	☒	n/a	□
[119]	☒	RDFS, OWL	☑
[120]	☒	RDFS, OWL	☑
[121]	☑	RDFS, OWL	☑
[122]	☒	RDFS	☑
[123]	☒	RDFS	☑
[124]	☒	RDFS, OWL	☑
[125]	☒	RDFS, OWL	□
[126]	☑	RDFS, OWL	☑
[127]	☑	RDFS, OWL	☑
[128]	☒	RDFS, OWL	☑
[129]	☒	RDFS, OWL	☑
[130,131]	☒	RDFS, OWL	□
[132]	☒	n/a	☑
[133]	☒	RDFS	☑
[134]	☒	RDFS, OWL	☑

At the beginning, we focus on solutions [115,116,121,126,127] that use existing vocabulary and/or ontology. This means that the XML data is transformed according to the mapped vocabularies. Cruz et al. [115] propose basic mapping rules to specify the transformation rules on properties, which are defined in the XML Schema. Deursen et al. [116] propose the method for the transformation of XML data into RDF instances in an ontology-dependent way. X2OWL [121] is a tool that builds an

OWL ontology from an XML data source and a set of mapping bridges. That proposal is based on an XML Schema that can be modeled using different styles to create the vocabulary structure. The next proposal is WEESA [126], which is an approach for Web engineering techniques and developing semantically tagged applications. Another tool is JXML2OWL [127]. It supports the conversion from syntactic data sources in XML language to a commonly shared global model defined by vocabulary.

Another group of proposals [107,109,111,119,120,122–125,130,133,134] do not support mappings between XML Schemas and existing vocabularies. Amann et al. [107] discuss a data integration system, where XML is mapped into vocabulary that supports roles and inheritance. That tool focuses on offering the appropriate high-level primitives and methods for representing the semantics of data expressed in XML. Janus [109] is a framework that focuses on an advanced logical representation of XML Schema components and a set of patterns that enable the transformation from XML Schema into the vocabulary. It supports a set of patterns that enable the translation from XML Schema into ontology. SPARQL2XQuery [111] is a framework that transforms SPARQL [135] query into a XQuery [136] using mapping from vocabulary to XML Schema. It allows query XML databases. Ferdinand et al. [119] propose two independent mappings: From XML to RDF graphs and XML Schema to OWL. That proposal allows items in XML documents to be mapped to different items in OWL. Garcia et al. [120] present a domain-specific approach that maps the MPEG-7 standard to RDF. Klein [122] proposes a procedure to transform the XML tree using the RDF primitives by annotating the XML by RDFS. This procedure can multiply the availability of semantically annotated RDF data. SWIM [123] is an integration middleware for mediating high-level queries to XML sources using RDFS. Lehti et al. [124] show how the ontologies can be used for mapping data sources to a global schema. In this work, the authors introduce how the inference rules can be used to check the consistency of such mappings. In this paper, a query language based on XQuery is presented. O'Connor et al. [125] propose an OWL-based language that can transform XML documents to arbitrary ontologies. It extends it with Manchester syntax for XPath to support references to XML fragments. Another framework is DTD2OWL [130], which changes XML into vocabularies. It also allows transforming specific XML instances into OWL individuals. Xiao et al. [133] propose the mappings between XML schemas and local RDFS vocabularies and those between local vocabulary and the global RDFS vocabulary. The authors discuss the problem of query containment and present a query rewriting algorithm for RDQL [137] and XQuery. Yahi et al. [134] propose an approach which covers schema level and data level. In this proposal XML Schema documents are generated for XML documents with no schema using the trang tool.

Solutions [110,113,117] using XSLT are separate from the ones mentioned above. Berrueta et al. [110] discuss an XSLT+SPARQL framework, which allows to perform SPARQL queries from XSLT. It is a collection of functions for XSLT, which allows to transform the XML results format. Yet another tool is XML2OWL [113], which uses XSLT for mapping from XML to ontology. Droop et al. [117] propose another XSLT solution, which allows embedding XPath into SPARQL. Shapkin et al. [128] propose a transformation language, which is not strictly based on XLST but inspired by it. That proposal focuses on matching of types of RDF resources.

Another subgroup of approaches [98,108,114,118,129,132] supports mutual transformation. XSPARQL [98,112] is a query language based on SPARQL and XQuery for transformations from RDF into XML and back. It is built on top of XQuery in a syntactic and semantic view. Gloze [108] is another tool for bidirectional mapping between XML and RDF. It uses information available in the XML schema for describing how XML is mapped into RDF and back again. GRDDL [114] is a markup language that obtains RDF data from XML documents. It is represented in XSLT. SAWSDL [118] is also a markup language but it proposes a collection of new attributes for the WSDL [138] and XML Schema. Other tools include SPARQL2XQuery [129] and XS2OWL [132].

6. RDF Serializations

Several of RDF syntax formats exist for writing down graphs. RDF 1.1 introduces a number of serialization formats, such as: Turtle, N-Triples, TriG, N-Quads, JSON-LD, RDFa, and RDF/XML. Note that in RDF 1.1 RDF/XML is no longer the only recommended serialization format.

In Section 6.1, we present serializations that support single graphs. In Section 6.2, we briefly introduce serialization which supports multiple graphs. Moreover, in this section we show the above-mentioned formats in examples.

6.1. Single Graph Support

RDFa [139] (denoted rdfa in Table 7) is a RDF syntax which embeds RDF triples in HTML and XML documents. The RDF data is mixed within the Document Object Model. This implies that document content can be marked up with RDFa. It adds a set of attribute-level extensions to HTML and different types of XML documents for embedding rich metadata within these documents. What is more, RDFa allows for free intermixing of terms from multiple vocabularies. It is also designed in such a way that the format can be processed without information of the specific vocabulary being used. It is common in contexts where data publishers are able to change Web templates but have little additional control over the publishing infrastructure.

Following [139], we provide the most important attributes that can be used in RDFa, such as:

- `about`–an attribute that is an IRI or CURIE [41] specifying the resource the metadata is about (a RDF subject);
- `rel` and `rev`–attributes that expresses (reverse) relationships between two resources (a RDF predicate);
- `property`–an attribute that expresses relationships between a subject and some literal value (a RDF predicate);
- `resource`–an attribute for expressing a relationship's partner resource that is not intended to be navigable (a RDF object);
- `href`–an attribute that expresses the partner resource of a relationship (a RDF resource object);
- `src`–an attribute that expresses a relationship's partner resource when the resource is embedded (a RDF object that is a resource);
- `content`–an attribute that overrides the content of the element when using the `property` (a RDF object that is a literal);
- `datatype`–an attribute that specifies the datatype of a literal;
- `typeof`–an attribute that specifies the RDF types of the subject or the partner resource;
- `inlist`–an attribute that specifies that the object associated with `property` or `rel` attributes on the same element is to be pushed onto the list for that predicate;
- `vocab`–an attribute that specifies the mapping to be used when a RDF term is assigned in a value of attribute.

Table 7. RDF serializations. This table presents the features of serializations, namely: Having W3C Recommendation (yes–☑, no–☒), human-friendly syntax (yes–☑, partial–□, no–☒), easy to process (yes–☑, partial–□, no–☒), compact form (yes–☑, partial–□, no–☒), similarity to Turtle syntax and XML-based syntax (yes–☑, no–☒) and multigraph support (yes–☑, no–☒).

Feature	ttl	nt	tg	nq	jld	rdfa	xml
Standard	☑	☑	☑	☑	☑	☑	☑
Human readable	☑	□	☑	□	☑	☑	□
Efficient	☒	☒	☒	☒	□	☒	□
Normalized	☒	☑	☒	☑	□	☒	☒
Turtle family	☑	☑	☑	☑	☒	☒	☒
XML family	☒	☒	☒	☒	☒	☑	☑
Multiple graphs	☒	☒	☑	☑	☑	☒	☒

Example 15. *This example presents a RDFa 1.1 serialization that represents the RDF triples of Example 2.*

```
<div xmlns="http://www.w3.org/1999/xhtml"
prefix="
foaf: http://xmlns.com/foaf/0.1/
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs: http://www.w3.org/2000/01/rdf-schema#"
>
<div typeof="foaf:Person" about="http://example.com/p#js">
<div property="foaf:name">John Smith</div>
<div rel="foaf:workplaceHomepage">
<a typeof="rdfs:Resource" href="http://univ.com/">
<span property="rdfs:label">University</span>
</a>
</div>
</div>
</div>
```

RDF/XML [140] (denoted xml in Table 7) is a syntax to serialize a RDF graph as an Extensible Markup Language (XML) document. Nevertheless, the syntax is viewed as problematic to read and write for humans so one should consider using other syntaxes for editors. In order to process a graph in RDF/XML serialization, there must be a representation of nodes and predicates in the terms of XML XML–names of attributes, names of elements, values of attributes, and contents of elements. This syntax uses qualified names (so-called QNames) to represent IRI references. There are some limitations imposed on RDF/XML by the XML format and the use of the XML namespace, which prevents the storage of all RDF graphs as some IRI references are forbidden by the specifications of these standards.

Following [140], we provide the most important elements and attributes that can be used in RDF/XML, such as:

- `rdf:RDF`–a root element of RDF/XML documents;
- `rdf:Description`–an element that contains elements that describe the resource, it contains the description of the resource identified by the `rdf:about` attribute;
- `rdf:Alt`, `rdf:Bag` and `rdf:Seq`–elements that are containers used to describe a group of things (see Section 2);
- `rdf:parseType="Collection"`–an attribute that describe groups that can only contain the specified members;
- `rdf:parseType="Resource"`–an attribute that is used to omit blank nodes;
- `xml:lang`–an attribute that is used to allow content language identification;
- `rdf:datatype`–an attribute that is used to define a typed literal;
- `rdf:nodeID`–an attribute that identifies a blank node;
- `rdf:ID` and `xml:base`–attributes that abbreviate IRIs.

Example 16. *This example presents a RDF/XML serialization that represents RDF triples of Example 2.*

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
>
<rdf:Description rdf:about="http://univ.com/">
<rdfs:label>University</rdfs:label>
```

```
</rdf:Description>
<rdf:Description rdf:about="http://example.com/p#js">
<foaf:name>John Smith</foaf:name>
<rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
<foaf:workplaceHomepage rdf:resource="http://univ.com/"/>
</rdf:Description>
</rdf:RDF>
```

Another RDF syntax refers to Terse RDF Triple Language (Turtle) [141] (denoted ttl in Table 7). That solution offers textual syntax that enables recording RDF graphs in a compact form, including abbreviations that use data patterns and datatypes. Following [141], we provide the most important rules for constructing the Turtle document:

- The simplest triple statement consists of a sequence of subject, predicate, and object, separated by space, tabulation or other whitespace and terminated by a dot after each triple.
- Often, the same subject will be referenced by several predicates. In this situation, a series of predicates and objects are separated by a semicolon.
- As with predicates, objects are often repeated with the same subject and predicate. In this case, a comma should be used as a separator.
- IRIs may be written as relative or absolute IRIs or prefixed names. Both absolute and relative IRIs are enclosed in less-than sign and greater-than sign.
- Quoted literals have a lexical form followed by a datatype IRI, a language tag or neither. Literals should be delimited by apostrophe or double quotes.
- Blank nodes are expressed as underscore, colon and a blank node label that is a series of name characters. Blank nodes can be nested, abbreviated, and delimited by square brackets.
- Collections are enclosed by parentheses.

Example 17. *This example presents a Turtle serialization that represents RDF triples of Example 2.*

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
<http://example.com/p#js> a foaf:Person ;
foaf:name "John Smith" ;
foaf:workplaceHomepage <http://univ.com/> .
<http://univ.com/> rdfs:label "University" .
```

N-Triples [142] (denoted nt in Table 7) is a line-based, plain text serialization format and a subset of the Turtle format minus features such as shorthands. It means that there is a lot of redundancy, and its files can be larger than Turtle and RDF/XML. N-Triples was designed to be a simpler format than Turtle, and therefore easier for software to parse and generate. Following [142], we provide the most important rules for constructing the N-Triples document:

- The triple statement consists of a sequence of subject, predicate, and object, divided by whitespace, and terminated by a dot after each triple.
- IRIs should be represented as absolute IRIs and they are enclosed in less-than sign and greater-than sign.
- The representation of the lexical form is a sequence of a double quote (an initial delimiter), a list of characters or escape sequence, and a double quote (a final delimiter).
- Blank nodes are expressed as underscore, colon and a blank node label that is a series of name characters.

There are some changes in RDF 1.1 N-Triples, e.g., encoding is UTF-8 rather than US-ASCII and blank node labels may begin with a digit. Syntactically, N-Triples is a subset of Turtle.

Example 18. *This example presents a N-Triples serialization that represents RDF triples of Example 2. Note that we change some RDF term because of legibility.*

```
<http://example.com/p#js> <http://...#type> <http://...foaf/0.1/Person> .
<http://example.com/p#js> <http://.../workplaceH...>  <http://univ.com/> .
<http://example.com/p#js> <http://.../name> "John Smith" .
<http://univ.com/> <http://...#label> "University" .
```

6.2. Multiple Graphs Support

JSON-LD [143] (denoted jld in Table 7) is a JSON-based format to serialize structured data such as RDF. The syntax of this serialization is created to simply integrate into deployed systems that use JSON and provides a smooth upgrade path from JSON to JSON-LD. The use of RDF in JSON makes RDF data accessible to Web developers without the obligation to install additional software libraries, parsers or other tools for changing RDF data. Like JSON, JSON-LD uses human-readable text to transmit data objects consisting of key-value pairs.

Keywords in JSON-LD start with the @ sign. Following [143], we provide the most important keywords that can be used in JSON-LD, such as:

- `@context`–set the short-hand names that are used throughout a document;
- `@id`–uniquely identify things that are being described in the document with blank nodes or IRIs;
- `@value`–specify the data that is associated with a particular property;
- `@language`–define the language for a particular string value or the default language of a document;
- `@type`–set the data type of an IRI, a blank node, a JSON-LD value or a list;
- `@container`–set the default container type for a short-hand string that expands to an IRI or a blank node identifier;
- `@list`–define an ordered set of data;
- `@set`–define an unordered set of data (values are represented as arrays);
- `@reverse`–used for reverse relationship expression between two resources;
- `@index`–specify that a container is used to index information;
- `@base`–define the base IRI against which relative IRIs are resolved;
- `@vocab`–expand properties and values in `@type` with a common prefix IRI;
- `@graph`–express a graph.

Example 19. *This example presents a JSON-LD serialization that represents the RDF triples of Example 3.*

```
{ "@id": "http://example.com/#people",
"@graph": [
{
"@id": "http://univ.com/",
"http://www.w3.org/2000/01/rdf-schema#label": "University"
},
{
"@id": "http://example.com/p#js",
"@type": "http://xmlns.com/foaf/0.1/Person",
"http://xmlns.com/foaf/0.1/name": "John Smith",
"http://xmlns.com/foaf/0.1/workplaceHomepage": {
"@id": "http://univ.com/"
}
}
]
}
```

JSON-LD is a RDF syntax. However, it also extends the RDF data model, e.g., in JSON-LD predicates can be IRIs or blank nodes whereas in RDF have to be IRIs. JSON-LD can serialize generalized RDF triples, where subjects, predicates, and objects can be IRIs, blank nodes or literals.

Example 20. *This example presents a JSON-LD serialization that represents a RDF triple:* `<#a>` *as a subject, blank node as a predicate and* `"Alice"` *as an object.*

```
{
"@context": {
"name": "_:b"
},
"@id": "#a",
"name": "Alice"
}
```

Another RDF syntax refers to TriG [144] (denoted trig in Table 7). It is a plain text syntax for RDF datasets serialization. Syntactically, Turtle is subset of TriG. A document consists of:

1. A sequence of directives;
2. RDF triples;
3. graph statements which contain triple-generating statements.

Graph statements are a pair of blank node label or an IRI with a group of RDF triples surrounded by curly brackets. The blank node label or IRI of the graph statement may be used in another graph statement which implies taking the union of the triples generated by each graph statement. A blank node label or IRI used as a graph label may also reoccur as part of any RDF triples.

Example 21. *This example presents a TriG serialization that represents RDF triples of Example 3.*

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
<http://example.com/#people> {
<http://example.com/p#js> a foaf:Person ;
foaf:name "John Smith" ;
foaf:workplaceHomepage <http://univ.com/> .
<http://univ.com/> rdfs:label "University" .
}
```

N-Quads [145] (denoted nq in Table 7) is another line-based syntax for serializing RDF datasets. That plain text format is a RDF syntax similar to N-Triples. N-Quads line statement is a sequence of RDF terms representing the RDF triple (subject, predicate and object line in N-Triples) and graph label, which can be an IRI or blank node. The graph label is also a part of a dataset, and this sequence is terminated by a dot.

Example 22. *This example presents a N-Quads serialization that represents RDF triples of Example 3. Note that we change some RDF term because of legibility.*

```
<http://univ.com/> <http://...#label> "University" <http://...#pe..>.
<http://example.com/p#js> <http://..#type> <http://../P..> <http://...#pe.>.
<http://example.com/p#js> <http://.../work...>  <.../> <http://...#pe..>.
<http://example.com/p#js> <http://.../name> "John ..." <http://...#pe..>.
```

In Table 7 we present features of the above-mentioned standardized serializations, namely: Having W3C Recommendation, human-friendly syntax (partial support means that some fragments may be difficult to read), easy to process, compact form (partial support means that there are several different forms and not all are normalized), similarity to Turtle syntax and XML-based syntax and multigraph support. Furthermore, there are a few RDF serializations that are not standardized, such as TriX [146], RDF/JSON [147,148].

7. RDF Compression

RDF compression has been widely addressed recently. However, there is no leading standard for RDF compression.

A recent work [149] points out that RDF datasets are highly compressible because of the RDF graph structure and RDF syntax verbosity. In that paper, different compression approaches are analyzed, including:

1. Direct compression;
2. adjacency list compression;
3. a RDF split into the element dictionaries and the statements.

The conclusions of that paper suggest that RDF is highly compressible.

Definition 20 (RDF compression processor)**.** An RDF compression processor is used by application programs for encoding their RDF data into compressed RDF data and/or to decode compressed RDF data to make the data accessible.

Header-Dictionary-Triples [150] (denoted hdt in Table 8), a binary format that is based on three main parts:

1. A header, which includes metadata describing the RDF dataset;
2. a dictionary, which organizes all the identifiers in the graph (it provides a list of the RDF terms such as literals, IRIs and blank nodes);
3. a triples component, which consists of the underlying RDF graph pure structure.

HDT achieves high levels of compression and provides retrieving features to the compressed data. That approach works on the complete dataset, with non-negligible processing time. That idea is extended in [151]. In that thesis, the author proposes techniques to compress rich-functional RDF dictionaries and triple indexing. That thesis shows the use of a succinct data configuration to browse HDT-encoded datasets. HDT can be used as a backend of Triple Pattern Fragments [152] and natively supports fast triple-pattern extraction.

In [153,154] (denoted eri in Table 8), authors apply a feature of RDF data streams, which is the symmetry and regularity of their structure and values. They propose a compressed Efficient RDF Interchange format, which can reduce the amount of data transmitted when processing RDF streams. ERI considers a RDF stream as a continuous flow of blocks of triples. A standard compressor can be used in each channel and impact on its data regularities to produce better compression results.

Another approach is the RDF Differential Stream compressor based on Zlib [155] (denoted rdsz in Table 8), which is a proposal for RDF streaming compression. It applies the general-purpose stream compressor Zlib to RDF streams. It uses differential encoding to obtain structural similarities. The results of this process are compressed with Zlib library to exploit extra redundancies. Furthermore, that approach achieves gains in compression at the cost of increasing the processing time.

The interest on RDF compression over streaming data has been indirectly covered by RDF stream processing systems such as Continuous Query Evaluation over Linked Streams Cloud [156] (denoted cqels in Table 8) and Ztreamy [157] (denoted ztr in Table 8). These papers emphasize the significance of compression for scalable transmission of RDF streams over the Web. In [156], authors propose an

approach to deal with this issue by dictionary encoding. In [157], authors discuss a scalable middleware for stream publishing.

Several research areas have emerged around MapReduce and RDF compression, e.g., scalable compression of large RDF datasets [158] and large RDF data compression and decompression efficiency [159] (denoted mr in Table 8). The first paper presents an approach based on providing another dictionary-based compression on top of MapReduce [160]. In the second one, authors expand [159] and achieve linear scalability concerning the number of nodes and input size. Another proposal is presented in [161], where HDT-MR is introduced. HDT-MR uses MapReduce technique to process a huge RDF and build the HDT serialization.

It is worth noting that EXI (Efficient XML Interchange) [162] can be used to RDF compression but it can only serialize XML [163], i.e., RDF/XML or TriX or JSON [164] i.e., JSON-LD. In Table 8 we present features of the above-mentioned proposals, namely: Having W3C Recommendation, binary syntax, ability to stream, ability to scale and support of a software library used for data compression (Zlib).

Table 8. RDF compression. This table presents the features of serializations, namely: Having W3C Recommendation (yes–☑, no–☒), binary syntax (yes–☑, no–☒), ability to stream (yes–☑, no–☒), ability to scale (yes–☑, no–☒), and support of a software library used for data compression (yes–☑, no–☒).

Feature	hdt	eri	rdsz	cqels	ztr	mr
Standard	☑*	☒	☒	☒	☒	☒
Binary format	☑	☑	☑	☑	☑	☑
Streamable	☒	☑	☑	☑	☑	☑
Scalable	☑	☑	☑	☑	☑	☑
Zlib	☒	☒	☑	☒	☑	☑

* W3C Member Submission.

8. Conclusions

Standards are instrumental in achieving a significant level of interoperability. W3C recommendations provide people and institutions a basis for mutual understanding. The recommendations that define RDF are used as tools to facilitate various providers to interact with one another. Despite the achievements of current RDF recommendations, they are not sufficient for achieving full end-to-end interoperability. The standards leave several areas vulnerable to variations in interpretation. In this article, we outlined various RDF recommendations, scientific papers that extend and clarify them, and presented a summarised formal description that we hope will clarify some of interpretative differences.

We specifically provided insights on the interpretation of the handling of blank nodes and reification. We presented several interpretative differences, each corresponding to an entailment regime in a standard way. We surveyed various RDF serializations, RDF compression proposals, and RDF mapping approaches to highlight their differences. Finally, we presented a summarized formal definition of RDF 1.1 and emphasized changes between RDF versions 1.0 and 1.1.

We argue that knowledge representation and data integration on the Web faces some of the same challenges faced ten years ago, in spite of the significant work being accomplished by both researchers and implementers. We hope that this review contributes to a better understanding of RDF 1.1, and provides the basis for a discussion of interpretative differences. We hope some of these gaps may be able to be fixed in a future version of RDF, such as selection of concise reification, and a formal description of the data model that addresses practical experiences with reification, and blank nodes.

Funding: This research received no external funding. The APC was funded by the University of Bialystok.

Acknowledgments: The authors gratefully acknowledges the members of the RDF 1.1 Working Group who defined RDF version 1.1. We thank Anna Gomolińska for comments that greatly improved the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CURIE	compact URI expressions
DOM	Document Object Model
dRDF	domain-restricted RDF
EXI	Efficient XML Interchange
FOAF	Friend of a Friend
FOL	First-Order Logic
HDT	Header-Dictionary-Triples
HTTP	Hypertext Transfer Protocol
IRI	Internationalized Resource Identifier
JSON	JavaScript Object Notation
OWL	Web Ontology Language
QNames	qualified name
R2RML	RDB to RDF Mapping Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SPARQL	SPARQL Protocol And RDF Query Language
SQL	Structured Query Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language
XQuery	XML Query language
XSLT	Extensible Stylesheet Language Transformations

References

1. Sowa, J.F. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*; Brooks/Cole Publishing Co.: Pacific Grove, CA, USA, 2000.
2. Raimond, Y.; Schreiber, G. *RDF 1.1 Primer*: *W3C Note*; World Wide Web Consortium: Cambridge, MA, USA, 2014.
3. Schwitter, R.; Tilbrook, M. Controlled Natural Language Meets the Semantic Web. In Proceedings of the Australasian Language Technology Workshop, Sydney, Australia, 8 December 2004; Volume 2, pp. 55–62.
4. Crystal, D. *The Cambridge Encyclopedia of Language*, 3rd ed.; Cambridge University Press: Cambridge, UK, 2018. [CrossRef]
5. Lenzerini, M. Data Integration: A Theoretical Perspective. In Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '02, Madison, WI, USA, 3–5 June 2002; ACM: New York, NY, USA, 2002; pp. 233–246. [CrossRef]
6. Ceri, S.; Tanca, L.; Zicari, R.V. Supporting Interoperability Between New Database Languages. In Proceedings of the 5th Annual European Computer Conference (CompEuro), Bologna, Italy, 13–16 May 1991; pp. 273–281. [CrossRef]
7. Lassila, O.; Swick, R.R. Resource Description Framework (RDF) Model and Syntax Specification. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 1999.
8. Decker, S.; Mitra, P.; Melnik, S. Framework for the Semantic Web: An RDF Tutorial. *IEEE Int. Comput.* **2000**, *4*, 68–73. [CrossRef]
9. Champin, P.A. *RDF Tutorial*; Semantic Web Best Practices and Deployment Working Group; World Wide Web Consortium: Cambridge, MA, USA, 2001.
10. Carroll, J.J. Matching RDF Graphs. In *The Semantic Web—ISWC 2002*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 5–15. [CrossRef]
11. Pan, J.Z.; Horrocks, I. RDFS(FA) and RDF MT: Two semantics for RDFS. In *The Semantic Web-ISWC 2003*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 30–46. [CrossRef]
12. Grau, B.C. A Possible Simplification of the Semantic Web Architecture. In Proceedings of the 13th International Conference on World Wide Web, WWW '04, New York, NY, USA, 17–20 May 2004; ACM: New York, NY, USA, 2004; pp. 704–713. [CrossRef]
13. Yang, G.; Kifer, M. Reasoning about Anonymous Resources and Meta Statements on the Semantic Web. *J. Data Semant.* **2003**, *1*, 69–97. [CrossRef]

14. Kifer, M.; Lausen, G. F-logic: A Higher-order Language for Reasoning About Objects, Inheritance, and Scheme. *SIGMOD Rec.* **1989**, *18*, 134–146. [CrossRef]
15. Berners-Lee, T.; Connolly, D.; Hawke, S. Semantic Web Tutorial Using N3. In Proceedings of the Twelfth International World Wide Web Conference, Budapest, Hungary, 20–24 May 2003.
16. Marin, D. A Formalization of RDF. Master's Thesis, École Polytechnique, Palaiseau, France, 2004.
17. Manola, F.; Miller, E. RDF Primer. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2004.
18. Gutierrez, C.; Hurtado, C.A.; Mendelzon, A.O. Foundations of Semantic Web Databases. In Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Paris, France, 14–16 June 2004; ACM: Paris, France, 2004; pp. 95–106. [CrossRef]
19. de Bruijn, J.J.; Franconi, E.; Tessaris, S. Logical Reconstruction of RDF and Ontology Languages. In *Principles and Practice of Semantic Web Reasoning*; Springer: Berlin, Germany, 2005; pp. 65–71. [CrossRef]
20. Franconi, E.; de Bruijn, J.; Tessaris, S. Logical Reconstruction of Normative RDF. In Proceedings of the CEUR Workshop Proceedings, Galway, Ireland, 11–12 November 2005; Volume 188.
21. Feigenbaum, L.; Herman, I.; Hongsermeier, T.; Neumann, E.; Stephens, S. The Semantic Web in Action. *Sci. Am.* **2007**, *297*, 90–97. [CrossRef]
22. Muñoz, S.; Pérez, J.; Gutierrez, C. Minimal Deductive Systems for RDF. In *The Semantic Web: Research and Applications*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 53–67. [CrossRef]
23. Munoz, S.; Perez, J.; Gutierrez, C. Simple and Efficient Minimal RDFS. *Web Semant.* **2009**, *7*, 220–234. [CrossRef]
24. Pichler, R.; Polleres, A.; Wei, F.; Woltran, S. dRDF: Entailment for Domain-restricted RDF. In *The Semantic Web: Research and Applications*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 200–214. [CrossRef]
25. Hitzler, P.; Krotzsch, M.; Rudolph, S. *Foundations of Semantic Web Technologies*, 1st ed.; Chapman & Hall/CRC: London, UK, 2011.
26. Antoniou, G.; Groth, P.; Van Harmelen, F.; Hoekstra, R. *A Semantic Web Primer*, 3rd ed.; MIT Press: Cambridge, MA, USA, 2004.
27. Curé, O.; Blin, G. *RDF Database Systems: Triples Storage and SPARQL Query Processing*; Morgan Kaufmann: Burlington, MA, USA, 2014.
28. Zimmermann, A.; Lopes, N.; Polleres, A.; Straccia, U. A General Framework for Representing, Reasoning and Querying with Annotated Semantic Web Data. *J. Web Semant.* **2012**, *11*, 72–95. [CrossRef]
29. Buneman, P.; Kostylev, E. Annotation algebras for RDFS. In Proceedings of the The Second International Workshop on the Role of Semantic Web in Provenance Management (SWPM-10), CEUR Workshop Proceedings, Shanghai, China, 7 November 2010; p. 32.
30. Udrea, O.; Recupero, D.R.; Subrahmanian, V.S. Annotated RDF. In Proceedings of the 3rd European Conference on The Semantic Web: Research and Applications, ESWC'06, Budva, Montenegro, 11–14 June 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 487–501. [CrossRef]
31. Straccia, U. A minimal deductive system for general fuzzy RDF. In *Web Reasoning and Rule Systems*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 166–181. [CrossRef]
32. Gutierrez, C.; Hurtado, C.; Vaisman, A. Temporal RDF. In *The Semantic Web: Research and Applications*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 93–107. [CrossRef]
33. Koubarakis, M.; Kyzirakos, K. Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL. In *ESWC (1)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6088, pp. 425–439. [CrossRef]
34. Tomaszuk, D.; Pak, K.; Rybinski, H. Trust in RDF Graphs. In *ADBIS (2)*; Advances in Intelligent Systems and Computing; Springer: Berlin/Heidelberg, Germany, 2012; Volume 186, pp. 273–283.
35. Sequeda, J.F.; Tirmizi, S.H.; Corcho, O.; Miranker, D.P. Survey of directly mapping SQL databases to the Semantic Web. *Knowl. Eng. Rev.* **2011**, *26*, 445–486. [CrossRef]
36. Spanos, D.E.; Stavrou, P.; Mitrou, N. Bringing Relational Databases into the Semantic Web: A Survey. *Semant. Web* **2012**, *3*, 169–209. [CrossRef]
37. Harth, A.; Hose, K.; Schenkel, R. *Linked Data Management*; CRC Press: Boca Raton, FL, USA, 2014.
38. Thakkar, H.; Angles, R.; Tomaszuk, D.; Lehmann, J. Direct Mappings between RDF and Property Graph Databases. *arXiv* **2019**, arXiv:1912.02127.

39. Cyganiak, R.; Lanthaler, M.; Wood, D. RDF 1.1 Concepts and Abstract Syntax. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2014.
40. Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; Ives, Z. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 722–735. [CrossRef]
41. Birbeck, M.; McCarron, S. CURIE Syntax 1.0: A syntax for expressing Compact URIs. In *W3C Working Group Note*; World Wide Web Consortium: Cambridge, MA, USA, 2010.
42. Bray, T.; Hollander, D.; Layman, A.; Tobin, R. Namespaces in XML 1.1 (Second Edition). In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2006.
43. Sperberg-McQueen, M.; Thompson, H.; Peterson, D.; Malhotra, A.; Biron, P.V.; Gao, S. W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2012.
44. Brickley, D.; Miller, L. FOAF Vocabulary Specification 0.99; Technical Report, FOAF Project. Available online: http://xmlns.com/foaf/spec/20140114.html (accessed on 1 December 2019).
45. Carroll, J.J.; Bizer, C.; Hayes, P.; Stickler, P. Named Graphs, Provenance and Trust. In Proceedings of the 14th International Conference on World Wide Web, WWW '05, Chiba, Japan, 10–14 May 2005; ACM: New York, NY, USA, 2005; pp. 613–622. [CrossRef]
46. Zimmermann, A. RDF 1.1: On Semantics of RDF Datasets. In *W3C Note*; World Wide Web Consortium: Cambridge, MA, USA, 2014.
47. Brickley, D.; Guha, R. RDF Schema 1.1. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2014.
48. Noy, N.; Rector, A. Defining N-ary Relations on the Semantic Web. In *W3C Working Group Note*; World Wide Web Consortium: Cambridge, MA, USA, 2006.
49. Hartig, O.; Thompson, B. Foundations of an Alternative Approach to Reification in RDF. *arXiv* **2014**, arXiv:1406.3399.
50. Nguyen, V.; Bodenreider, O.; Sheth, A. Don't Like RDF Reification?: Making Statements About Statements Using Singleton Property. In Proceedings of the 23rd International Conference on World Wide Web, WWW '14, Seoul, Korea, 7–11 April 2014; ACM: New York, NY, USA, 2014; pp. 759–770. [CrossRef]
51. Groth, P.; Gibson, A.; Velterop, J. The anatomy of a Nano-publication. *Inf. Serv. Use* **2010**, *30*, 51–56. [CrossRef]
52. Chen, L.; Zhang, H.; Chen, Y.; Guo, W. Blank Nodes in RDF. *J. Softw.* **2012**, *7*, 1993–1999. [CrossRef]
53. Mallea, A.; Arenas, M.; Hogan, A.; Polleres, A. On Blank Nodes. In *The Semantic Web–ISWC 2011*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 421–437. [CrossRef]
54. Hogan, A. Skolemising Blank Nodes While Preserving Isomorphism. In Proceedings of the 24th International Conference on World Wide Web, WWW '15, Florence, Italy, 18–22 May 2015; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2015; pp. 430–440. [CrossRef]
55. Nottingham, M.; Hammer-Lahav, E. Defining Well-Known Uniform Resource Identifiers (URIs). In *RFC 5785, Request for Comments*; Internet Engineering Task Force: Fremont, CA, USA, 2010.
56. Hogan, A.; Arenas, M.; Mallea, A.; Polleres, A. Everything You Always Wanted to Know About Blank Nodes. *J. Web Semant.* **2014**, *27–28*, 42–69. [CrossRef]
57. Booth, D. Well Behaved RDF: A Straw-Man Proposal for Taming Blank Nodes. Available online: http://dbooth.org/2013/well-behaved-rdf/Booth-well-behaved-rdf.pdf (accessed on 1 December 2019).
58. Gutierrez, C.; Hurtado, C.A.; Mendelzon, A.O.; Jorge, P. Foundations of Semantic Web Databases. *J. Comput. Syst. Sci.* **2011**, *77*, 520–541. [CrossRef]
59. Patel-Schneider, P.; Hayes, P. RDF 1.1 Semantics. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2014.
60. ter Horst, H.J. Completeness, Decidability and Complexity of Entailment for RDF Schema and a Semantic Extension Involving the OWL Vocabulary. *J. Web Semant.* **2005**, *3*, 79–115. [CrossRef]
61. Data, C. *An Introduction to Database Systems*; Addison-Wesley Publ.: Boston, MA, USA, 1975.
62. Hayes, P. RDF 1.0 Semantics. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2004.
63. Corby, O.; Faron-Zucker, C. A Transformation Language for RDF based on SPARQL. In *Web Information Systems and Technologies*; Springer: Cham, Switzerland, 2016; pp. 318–340. [CrossRef]

64. Corby, O.; Faron-Zucker, C.; Gandon, F. A Generic RDF Transformation Software and Its Application to an Online Translation Service for Common Languages of Linked Data. In *International Semantic Web Conference (2)*; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2015; Volume 9367, pp. 150–165.
65. Alkhateeb, F.; Laborie, S. Towards extending and using SPARQL for modular document generation. In Proceedings of the Eighth ACM Symposium on Document Engineering, DocEng '08, Sao Paulo, Brazil, 16–19 September 2008; ACM: New York, NY, USA, 2008; pp. 164–172. [CrossRef]
66. Quan, D.; Karger, D.R. Berners-Lee, T.; Connolly, D.; Hawke, S. Xenon: An RDF Stylesheet Ontology. In Proceedings of the International World Wide Web Conference, Chiba, Japan, 10–14 May 2005.
67. Peroni, S.; Vitali, F. RSLT: RDF Stylesheet Language Transformations. In Proceedings of the ESWC Developers Workshop, CEUR Workshop Proceedings, Portoroz, Slovenia, 31 May–4 June 2015; Volume 1361, pp. 7–13.
68. Tandy, J.; Herman, I.; Kellogg, G. Generating RDF from Tabular Data on the Web. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2015.
69. Dimou, A.; Vander Sande, M.; Colpaert, P.; Verborgh, R.; Mannens, E.; Van de Walle, R. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In Proceedings of the 7th Workshop on Linked Data on the Web, CEUR Workshop Proceedings, Seoul, Korea, 7–11 April 2014; Volume 1184.
70. Horrocks, I.; Patel-Schneider, P.F.; Harmelen, F.V. Reviewing the Design of DAML+OIL: An Ontology Language for the Semantic Web. In *Eighteenth National Conference on Artificial Intelligence*; American Association for Artificial Intelligence: Menlo Park, CA, USA, 2002; pp. 792–797.
71. Auer, S.; Dietzold, S.; Lehmann, J.; Hellmann, S.; Aumueller, D. Triplify: Light-weight Linked Data Publication from Relational Databases. In Proceedings of the 18th International Conference on World Wide Web, WWW '09, Madrid, Spain, 20–24 April 2009; ACM: New York, NY, USA, 2009; pp. 621–630. [CrossRef]
72. Salas, P.E.; Breitman, K.K.; Viterbo F., J.; Casanova, M.A. Interoperability by Design Using the StdTrip Tool: An a Priori Approach. In Proceedings of the 6th International Conference on Semantic Systems, I-SEMANTICS '10, Graz, Austria, 1–3 September 2010; ACM: New York, NY, USA, 2010; pp. 43:1–43:3. [CrossRef]
73. Vavliakis, K.N.; Grollios, T.K.; Mitkas, P.A. RDOTE-Publishing Relational Databases into the Semantic Web. *J. Syst. Softw.* **2013**, *86*, 89–99. [CrossRef]
74. Astrova, I. Reverse Engineering of Relational Databases to Ontologies. In *The Semantic Web: Research and Applications*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 327–341. [CrossRef]
75. Astrova, I. Rules for Mapping SQL Relational Databases to OWL Ontologies. In *MTSR*; Springe: Boston, MA, USA, 2007; pp. 415–424. [CrossRef]
76. Cerbah, F. Learning Highly Structured Semantic Repositories from Relational Databases: The RDBToOnto Tool. In *The Semantic Web: Research and Applications*; ESWC'08; Springer: Berlin/Heidelberg, Germay, 2008; pp. 777–781. [CrossRef]
77. Cerbah, F. Mining the Content of Relational Databases to Learn Ontologies with Deeper Taxonomies. Web Intelligence. *IEEE Comput. Soc.* **2008**, *1*, 553–557. [CrossRef]
78. Bizer, C.; Cyganiak, R. D2RQ-Lessons Learned. In Proceedings of the W3C Workshop on RDF Access to Relational Databases, Cambridge, MA, USA, 25–26 October 2007.
79. Curino, C.; Orsi, G.; Panigati, E.; Tanca, L. Accessing and Documenting Relational Databases Through OWL Ontologies. In Proceedings of the 8th International Conference on Flexible Query Answering Systems, FQAS '09, Roskilde, Denmark, 26–28 October 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 431–442. [CrossRef]
80. Buccella, A.; Penabad, M.R.; Rodriguez, F.J.; Farina, A.; Cechich, A. From relational databases to OWL ontologies. In Proceedings of the 6th National Russian Research Conference, Pushchino, Russia, 29 September–1 October 2004.
81. Hu, W.; Qu, Y. Discovering Simple Mappings Between Relational Database Schemas and Ontologies. In *The Semantic Web*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 225–238. [CrossRef]
82. Būmans, G.; Čerāns, K. RDB2OWL: A Practical Approach for Transforming RDB Data into RDF/OWL. In Proceedings of the 6th International Conference on Semantic Systems, I-SEMANTICS '10, Graz, Austria, 1–3 September 2010; ACM: New York, NY, USA, 2010; pp. 25:1–25:3. [CrossRef]
83. Li, M.; Du, X.; Wang, S. A Semi-automatic Ontology Acquisition Method for the Semantic Web. In *Advances in Web-Age Information Management, WAIM'05*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 209–220. [CrossRef]

84. Byrne, K. Having Triplets–Holding Cultural Data as RDF. In Proceedings of the ECDL 2008 Workshop on Information Access to Cultural Heritage, Aarhus, Denmark, 14–19 September 2008.
85. Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodriguez-Muro, M.; Rosati, R.; Ruzzi, M.; Savo, D.F. The MASTRO System for Ontology-based Data Access. *Semant. Web* **2011**, *2*, 43–53. [CrossRef]
86. Erling, O.; Mikhailov, I. RDF Support in the Virtuoso DBMS. In *Networked Knowledge-Networked Media: Integrating Knowledge Management, New Media Technologies and Semantic Systems*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 7–24. [CrossRef]
87. Ghawi, R.; Cullot, N. Database-to-Ontology Mapping Generation for Semantic Interoperability. In Proceedings of the Third International Workshop on Database Interoperability (InterDB 2007), Dijon, France, 23–27 September 2007.
88. Hert, M.; Reif, G.; Gall, H.C. Updating Relational Data via SPARQL/Update. In Proceedings of the 2010 EDBT/ICDT Workshops, EDBT '10, Lausanne, Switzerland, 22–26 March 2010; ACM: New York, NY, USA, 2010; pp. 24:1–24:8. [CrossRef]
89. Nyulas, C.; OConnor, M.; Tu, S. DataMaster–a plug-in for importing schemas and data from relational databases into Protege. In Proceedings of the 10th International Protege Conference, Budapest, Hungary, 15–18 July 2007.
90. Polfliet, S.; Ichise, R. Automated Mapping Generation for Converting Databases into Linked Data. In Proceedings of the ISWC Posters & Demos, CEUR Workshop Proceedings, Shanghai, China, 7–11 November 2010; Volume 658.
91. Sahoo, S.S.; Bodenreider, O.; Rutter, J.L.; Skinner, K.J.; Sheth, A.P. An Ontology-driven Semantic Mashup of Gene and Biological Pathway Information: Application to the Domain of Nicotine Dependence. *J. Biomed. Inf.* **2008**, *41*, 752–765. [CrossRef]
92. Shen, G.; Huang, Z.; Zhu, X.; Zhao, X. Research on the Rules of Mapping from Relational Model to OWL. In Proceedings of the OWLED, CEUR Workshop Proceedings, Athens, GA, USA, 10–11 November 2006; Volume 216.
93. Stojanovic, L.; Stojanovic, N.; Volz, R. Migrating Data-intensive Web Sites into the Semantic Web. In Proceedings of the 2002 ACM Symposium on Applied Computing, SAC '02, Madrid, Spain, 11–14 March 2002; ACM: New York, NY, USA, 2002; pp. 1100–1107. [CrossRef]
94. Tirmizi, S.H.; Sequeda, J.; Miranker, D. Translating SQL Applications to the Semantic Web. In *Database and Expert Systems Applications*; DEXA '08; Springer: Berlin/Heidelberg, Germany, 2008; pp. 450–464. [CrossRef]
95. Wu, Z.; Chen, H.; Wang, H.; Wang, Y.; Mao, Y.; Tang, J.; Zhou, C. Dartgrid: A Semantic Web Toolkit for Integrating Heterogeneous Relational Databases. In Proceedings of the Semantic Web Challenge at 4th International Semantic Web Conference, Beijing, China, 27–29 November 2005. [CrossRef]
96. Seaborne, A.; Steer, D.; Williams, S. SQL-RDF. Available online: http://www.w3.org/2007/03/RdfRDB/papers/seaborne.html (accessed on 1 December 2019).
97. Lopes, N.; Bischof, S.; Decker, S.; Polleres, A. On the Semantics of Heterogeneous Querying of Relational, XML and RDF Data with XSPARQL. In Proceedings of the 15th Portuguese Conference on Artificial Intelligence (EPIA 2011), Lisbon, Portugal, 10–13 October 2011.
98. Bischof, S.; Lopes, N.; Polleres, A. Improve Efficiency of Mapping Data Between XML and RDF with XSPARQL. In Proceedings of the 5th International Conference on Web Reasoning and Rule Systems, RR'11, Galway, Ireland, 29–30 August 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 232–237. [CrossRef]
99. de Laborda, C.P.; Conrad, S. Relational.OWL: A Data and Schema Representation Format Based on OWL. In Proceedings of the 2Nd Asia-Pacific Conference on Conceptual Modelling-Volume 43, APCCM '05, Newcastle, Australia, 30 January–4 February 2005; Australian Computer Society, Inc.: Darlinghurst, Australia, 2005; pp. 89–96.
100. Das, S.; Cyganiak, R.; Sundara, S. R2RML: RDB to RDF Mapping Language. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2012.
101. Priyatna, F.; Alonso-Calvo, R.; Paraiso-Medina, S.; Padron-Sanchez, G.; Corcho, O. R2RML-based Access and Querying to Relational Clinical Data with Morph-RDB. In Proceedings of the 8th Semantic Web Applications and Tools for Life Sciences International Conference, Cambridge, UK, 7–10 December 2015; pp. 142–151.
102. Sequeda, J.F.; Miranker, D.P. Ultrawrap Mapper: A Semi-Automatic Relational Database to RDF (RDB2RDF) Mapping Tool. In Proceedings of the International Semantic Web Conference (Posters & Demos), Bethlehem, PA, USA, 11–15 October 2015.

103. Erling, O.; Mikhailov, I. RDF Support in the Virtuoso DBMS. In *Networked Knowledge-Networked Media*; Springer: Berlin, Germany, 2009; pp. 7–24.
104. Barrasa, J.; Corcho, Ó.; Gómez-pérez, A. R2O, an Extensible and Semantically based Database-to-Ontology Mapping Language. In Proceedings of the 2nd Workshop on Semantic Web and Databases (SWDB2004), Toronto, ON, Canada, 29–30 August 2004; pp. 1069–1070. [CrossRef]
105. Kay, M. XSL Transformations (XSLT) Version 3.0. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2017.
106. Snelson, J.; Chamberlin, D.; Dyck, M.; Robie, J. XML Path Language (XPath) 3.1. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2017.
107. Amann, B.; Beeri, C.; Fundulaki, I.; Scholl, M. Ontology-Based Integration of XML Web Resources. In *The Semantic Web—ISWC 2002*; ISWC '02; Springer: Berlin/Heidelberg, Germany, 2002; pp. 117–131. [CrossRef]
108. Battle, S. Gloze: XML to RDF and back again. In Proceedings of the Jena User Conference, Bristol, UK, 10–11 May 2006.
109. Bedini, I.; Matheus, C.; Patel-Schneider, P.F.; Boran, A.; Nguyen, B. Transforming XML Schema to OWL Using Patterns. In Proceedings of the 2011 IEEE Fifth International Conference on Semantic Computing, ICSC '11, Palo Alto, CA, USA, 18–21 September 2011; IEEE Computer Society: Washington, DC, USA, 2011; pp. 102–109. [CrossRef]
110. Berrueta, D.; Labra, J.E.; Herman, I. XSLT+SPARQL: Scripting the Semantic Web with SPARQL Embedded into XSLT Stylesheets. In Proceedings of the 4th Workshop on Scripting for the Semantic Web, Tenerife, Spain, 1 June 2008.
111. Bikakis, N.; Gioldasis, N.; Tsinaraki, C.; Christodoulakis, S. Querying XML Data with SPARQL. In *Database and Expert Systems Applications*; DEXA '09; Springer: Berlin/Heidelberg, Germany, 2009; pp. 372–381. [CrossRef]
112. Bischof, S.; Decker, S.; Krennwallner, T.; Lopes, N.; Polleres, A. Mapping between RDF and XML with XSPARQL. *J. Data Semant.* **2012**, *1*, 147–185. [CrossRef]
113. Bohring, H.; Auer, S. Mapping XML to OWL Ontologies. *Leipz. Inf.-Tage* **2005**, *72*, 147–156.
114. Connolly, D. Gleaning Resource Descriptions from Dialects of Languages (GRDDL). In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2007.
115. Cruz, C.; Nicolle, C. Ontology Enrichment and Automatic Population From XML Data. In Proceedings of the 4th International VLDB Workshop on Ontology-Based Techniques for DataBases in Information Systems and Knowledge Systems, ODBIS 2008, Auckland, New Zealand, 34 August 23 2008; pp. 17–20.
116. Deursen, D.V.; Poppe, C.; Martens, G.; Mannens, E.; Walle, R.V.d. XML to RDF Conversion: A Generic Approach. In Proceedings of the 2008 International Conference on Automated Solutions for Cross Media Content and Multi-channel Distribution, AXMEDIS '08, Florence, Italy, 17–19 November 2008; IEEE Computer Society: Washington, DC, USA, 2008; pp. 138–144. [CrossRef]
117. Droop, M.; Flarer, M.; Groppe, J.; Groppe, S.; Linnemann, V.; Pinggera, J.; Santner, F.; Schier, M.; Schöpf, F.; Staffler, H.; et al. Bringing the XML and Semantic Web Worlds Closer: Transforming XML into RDF and embedding XPath into SPARQL. In *Enterprise Information Systems*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 31–45. [CrossRef]
118. Farrell, J.; Lausen, H. Semantic Annotations for WSDL and XML Schema. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2007.
119. Ferdinand, M.; Zirpins, C.; Trastour, D. Lifting XML schema to OWL. In *Web Engineering*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 354–358. [CrossRef]
120. Garcia, R.; Celma, O. Semantic Integration and Retrieval of Multimedia Metadata. In Proceedings of the 5th International Workshop on Knowledge Markup and Semantic Annotation, Galway, Ireland, 7 November 2005; pp. 69–80.
121. Ghawi, R.; Cullot, N. Building Ontologies from XML Data Sources. In Proceedings of the 2009 20th International Workshop on Database and Expert Systems Application, DEXA '09, Linz, Austria, 31 August–4 September 2009; IEEE Computer Society: Washington, DC, USA, 2009; pp. 480–484. [CrossRef]
122. Klein, M.C.A. Interpreting XML Documents via an RDF Schema Ontology. In Proceedings of the 13th International Workshop on Database and Expert Systems Applications, DEXA '02, Aix-en-Provence, France, 6 September 2002; IEEE Computer Society: Washington, DC, USA, 2002; pp. 889–894. [CrossRef]

123. Koffina, I.; Serfiotis, G.; Christophides, V.; Tannen, V. Mediating RDF/S Queries to Relational and XML Sources. *Int. J. Semant. Web Inf. Syst.* **2006**, *2*, 68. [CrossRef]
124. Lehti, P.; Fankhauser, P. XML Data Integration with OWL: Experiences and Challenges. In Proceedings of the 2004 International Symposium on Applications and the Internet, Tokyo, Japan, 26–30 January 2004; pp. 160–167. [CrossRef]
125. O'Connor, M.J.; Das, A. Acquiring OWL Ontologies from XML Documents. In Proceedings of the Sixth International Conference on Knowledge Capture, K-CAP '11, Banff, AB, Canada, 26–29 June 2011; ACM: New York, NY, USA, 2011; pp. 17–24. [CrossRef]
126. Reif, G.; Jazayeri, M.; Gall, H. Towards Semantic Web Engineering: WEESA-Mapping XML Schema to Ontologies. In Proceedings of the WWW Workshop on Application Design, Development and Implementation Issues in the Semantic Web, New York, NY, USA, 18 May 2004.
127. Rodrigues, T.; Rosa, P.; Cardoso, J. Moving from Syntactic to Semantic Organizations Using JXML2OWL. *Comput. Ind.* **2008**, *59*, 808–819. [CrossRef]
128. Shapkin, P.; Shumsky, L. A Language for Transforming the RDF Data on the Basis of Ontologies. In Proceedings of the 11th International Conference on Web Information Systems and Technologies, Lisbon, Portugal, 20–22 May 2015; pp. 504–511. [CrossRef]
129. Stavrakantonakis, I.; Tsinaraki, C.; Bikakis, N.; Gioldasis, N.; Christodoulakis, S. SPARQL2XQuery 2.0: Supporting Semantic-based queries over XML data. In Proceedings of the 2010 Fifth International Workshop Semantic Media Adaptation and Personalization, Limmassol, Cyprus, 9–10 December 2010; pp. 76–84.
130. Thuy, P.T.T.; Lee, Y.K.; Lee, S. DTD2OWL: Automatic Transforming XML Documents into OWL Ontology. In Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, ICIS '09, Seoul, Korea, 24–26 November 2009; ACM: New York, NY, USA, 2009; pp. 125–131. [CrossRef]
131. Thuy, P.T.; Lee, Y.K.; Lee, S. A Semantic Approach for Transforming XML Data into RDF Ontology. *Wirel. Pers. Commun.* **2013**, *73*, 1387–1402. [CrossRef]
132. Tsinaraki, C.; Christodoulakis, S. Interoperability of XML Schema Applications with OWL Domain Knowledge and Semantic Web Tools. In Proceedings of the 2007 OTM Confederated International Conference on On the Move to Meaningful Internet Systems: CoopIS, DOA, ODBASE, GADA, and IS-Volume Part I, OTM'07, Vilamoura, Portugal, 25–30 November 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 850–869. [CrossRef]
133. Xiao, H.; Cruz, I.F. Integrating and Exchanging XML Data Using Ontologies. In *Journal on Data Semantics VI*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 67–89.
134. Yahia, N.; Mokhtar, S.A.; Ahmed, A. Automatic Generation of OWL Ontology from XML Data Source. *arXiv* **2012**, arXiv:1206.0570.
135. Harris, S.; Seaborne, A. SPARQL 1.1 Query Language. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2013.
136. Robie, J.; Dyck, M. XQuery 3.1: An XML Query Language. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2017.
137. De Laborda, C.P.; Conrad, S. Querying Relational Databases with RDQL. In Proceedings of the Berliner XML Tage, Podebrady, Czech Republic, 2–6 October 2006; pp. 161–172.
138. Chinnici, R.; Moreau, J.J.; Ryman, A.; Weerawarana, S. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2007.
139. McCarron, S.; Adida, B.; Birbeck, M.; Herman, I. RDFa Core 1.1 - Third Edition. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2015.
140. Gandon, F.; Schreiber, G. RDF 1.1 XML Syntax. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2014.
141. Prud'hommeaux, E.; Carothers, G. RDF 1.1 Turtle. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2014.
142. Carothers, G.; Seaborne, A. RDF 1.1 N-Triples. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2014.
143. Sporny, M.; Lanthaler, M.; Kellogg, G. JSON-LD 1.0. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2014.

144. Seaborne, A.; Carothers, G. RDF 1.1 TriG. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2014.
145. Carothers, G. RDF 1.1 N-Quads. In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2014.
146. Carroll, J.J.; Stickler, P. RDF triples in XML. In Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, WWW Alt. '04, New York, NY, USA, 19–21 May 2004; ACM: New York, NY, USA, 2004; pp. 412–413. [CrossRef]
147. Tomaszuk, D. Named graphs in RDF/JSON serialization. *Zesz. Naukowe Politech. Gdań.* **2011**, *2*, 273–278.
148. Tomaszuk, D. Flat triples approach to RDF graphs in JSON. In *W3C Workshop–RDF Next Steps*; World Wide Web Consortium: Cambridge, MA, USA, 2010.
149. Fernández, J.D.; Gutierrez, C.; Martínez-Prieto, M.A. RDF Compression: Basic Approaches. In Proceedings of the 19th International Conference on World Wide Web, WWW '10, Raleigh, NC, USA, 26–30 April 2010; ACM: New York, NY, USA, 2010; pp. 1091–1092. [CrossRef]
150. Fernández, J.D.; Martínez-Prieto, M.A.; Gutiérrez, C.; Polleres, A.; Arias, M. Binary RDF Representation for Publication and Exchange (HDT). *J. Web Semant.* **2013**, *19*, 22–41. [CrossRef]
151. Fernández, J.D. Binary RDF for Scalable Publishing, Exchanging and Consumption in the Web of Data. In Proceedings of the 21st International Conference on World Wide Web, WWW '12 Companion, Lyon, France, 16–20 April 2012; ACM: New York, NY, USA, 2012; pp. 133–138. [CrossRef]
152. Verborgh, R.; Hartig, O.; De Meester, B.; Haesendonck, G.; De Vocht, L.; Vander Sande, M.; Cyganiak, R.; Colpaert, P.; Mannens, E.; Van de Walle, R. Querying Datasets on the Web with High Availability. In *The Semantic Web–ISWC 2014*; Springer: Cham, Switzerland, 2014; pp. 180–196. [CrossRef]
153. Fernández, J.D.; Llaves, A.; Corcho, O. Efficient RDF Interchange (ERI) Format for RDF Data Streams. In *The Semantic Web–ISWC 2014*; Springer: Cham, Switzerland, 2014; pp. 244–259. [CrossRef]
154. Álvarez-García, S.; Brisaboa, N.R.; Fernández, J.D.; Martínez-Prieto, M.A.; Navarro, G. Compressed Vertical Partitioning for Efficient RDF Management. *Knowl. Inf. Syst.* **2015**, *44*, 439–474. [CrossRef]
155. Fernández, N.; Arias, J.; Sánchez, L.; Fuentes-Lorenzo, D.; Corcho, Ó. RDSZ: An approach for lossless RDF stream compression. In *The Semantic Web: Trends and Challenges*; Springer: Cham, Switzerland, 2014; pp. 52–67. [CrossRef]
156. Le-Phuoc, D.; Quoc, H.N.M.; Le Van, C.; Hauswirth, M. Elastic and Scalable Processing of Linked Stream Data in the Cloud. In *The Semantic Web–ISWC 2013*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 280–297. [CrossRef]
157. Fisteus, J.A.; Garcia, N.F.; Fernandez, L.S.; Fuentes-Lorenzo, D. Ztreamy: A middleware for publishing semantic streams on the Web. *J. Web Semant.* **2014**, *25*, 16–23. [CrossRef]
158. Urbani, J.; Maassen, J.; Drost, N.; Seinstra, F.; Bal, H. Scalable RDF data compression with MapReduce. *Concurr. Comput. Pract. Exp.* **2013**, *25*, 24–39. [CrossRef]
159. Urbani, J.; Maassen, J.; Bal, H. Massive Semantic Web Data Compression with MapReduce. In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC '10, Chicago, IL, USA, 21–25 June 2010; ACM: New York, NY, USA, 2010; pp. 795–802. [CrossRef]
160. Dean, J.; Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM* **2008**, *51*, 107–113. [CrossRef]
161. Giménez-García, J.M.; Fernández, J.D.; Martínez-Prieto, M.A. HDT-MR: A scalable solution for RDF compression with HDT and MapReduce. In *The Semantic Web. Latest Advances and New Domains*; Springer: Cham, Switzerland, 2015; pp. 253–268. [CrossRef]
162. Käbisch, S.; Peintner, D.; Anicic, D. Standardized and Efficient RDF Encoding for Constrained Embedded Networks. In *The Semantic Web. Latest Advances and New Domains*; Springer: Cham, Switzerland, 2015; pp. 437–452. [CrossRef]
163. Schneider, J.; Kamiya, T.; Peintner, D.; Kyusakov, R. Efficient XML Interchange (EXI) Format 1.0 (Second Edition). In *W3C Recommendation*; World Wide Web Consortium: Cambridge, MA, USA, 2014.
164. Peintner, D.; Brutzman, D. EXI for JSON (EXI4JSON). In *W3C Working Group Note*; World Wide Web Consortium: Cambridge, MA, USA, 2018.