ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Second Cycle Degree Programme in

Digital Humanities and Digital Knowledge

Dissertation Title

Introducing a Python library to perform time-agnostic queries on datasets compliant with the OpenCitations' provenance model.

Final Dissertation in

Computational Thinking

Supervisor Prof. Silvio Peroni

Presented by Arcangelo Massari

Session II

Academic Year

2020-2021

1 TABLE OF CONTENTS

2 ABSTRACT

Sommario.

3 INTRODUCTION

Giustificazione del problema. Introduzione al caso d'uso. Dico cos'è OpenCitations.

4 LITERATURE REVIEW

4.1 PROVENANCE FOR THE SEMANTIC WEB

In his book *Weaving the Web: the original design and ultimate destiny of the World Wide Web*, Tim Berners Lee, the inventor of the WWW, states:

> *I have a dream for the Web [in which computers] become capable of analysing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which makes this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialise.* (Berners-Lee, Weaving the Web: the original design and ultimate destiny of the World Wide Web 1999)

In this vision, which represents the first formulation of the Semantic Web, it is already possible to identify the criticality that in the following years would have led to discuss the provenance topic. The data must be reliable in a world where automatic data analysis systems manage trade, bureaucracy, and daily lives. However, the Web is an open and inclusive dimension in which it is possible to find contradictory and questionable information. Therefore, it is essential to own indications such as the primary source of data, who created or modified it, and when that happened. However, the underlying technologies of the Semantic Web (RDF, OWL, SPARQL) were not originally intended to express such information.

In order to revise state of the art and develop a roadmap on provenance for Semantic Web technologies, the Provenance Incubator Group (Provenance Incubator Group Charter 2010) was established in 2010. One of the first problems was identifying a shared and universal definition of "provenance", a task that proved impossible given its broad and multisectoral nature. Therefore, a working definition was accepted, restricted the context of the Web:

> *Provenance of a resource is a record that describes entities and processes involved in producing and delivering or otherwise influencing that resource. Provenance provides a critical foundation for assessing authenticity, enabling trust, and allowing reproducibility. Provenance assertions are a form of contextual metadata and can themselves become important records with their own provenance*. (Gil, Cheney, et al. 08 December 2010)

Starting from this working definition, the group has compiled 33 use cases to formulate scenarios and requirements. Topics covered included eScience, eGovernment, business, manufacturing, cultural heritage, and library science, to name a few. The analysis of these use cases led to the elaboration of three scenarios: a news aggregator, the study of an epidemic and a business contract. The second scenario, the study of the epidemic, is particularly interesting for the case study of this work because it focuses on the reuse of scientific data. Alice is an epidemiologist studying the spread of a new disease called owl flu. Alice needs to integrate structured and unstructured data from different sources, to understand how data has evolved through provenance and version information. In addition, she needs to justify the results obtained by supporting the validity of the sources used, reusing data published by others in a new context and using the provenance to repeat previous analyses with new data. Introducing the problem with a concrete and complex example is helpful to understand how multifaceted and multidimensional it is. Specifically, provenance can be evaluated under three categories: content, management, and usage, each with various dimensions summarised in Table 1.

Category	**Dimension**	**Description**
Content	Object	The artefact that a provenance statement is about.
	Attribution	The sources or entities that contributed to creating the artefact in question.
	Process	The activities (or steps) that were carried out to generate or access the artefact at hand.
	Versioning	Records of changes to an artefact over time and what entities and processes were associated with those changes.
	Justification	Documentation recording why and how a particular decision is made.
	Entailment	Explanations showing how facts were derived from other facts.
Management	Publication	Making provenance available on the Web.
	Access	The ability to find the provenance for a particular artefact.
	Dissemination	Defining how provenance should be distributed and its access be controlled.
	Scale	Dealing with large amounts of provenance.
Use	Understanding	How to enable the end-user consumption of provenance.

	Interoperability	Combining provenance produced by multiple different systems.
	Comparison	Comparing artefacts through their provenance.
	Accountability	Using provenance to assign credit or blame.
	Trust	Using provenance to make trust judgments.
	Imperfections	Dealing with imperfections in provenance records.
	Debugging	Using provenance to detect bugs or failures of processes.

Table 1 Dimensions of provenance

Historically, many data models, annotation frameworks, vocabularies and ontologies have been introduced to meet the above requirements. A complete list of all existing strategies will be drawn up in section 4.2, and their advantages and disadvantages will be explained.

4.2 REPRESENTING PROVENANCE IN RDF

The landscape of strategies to formally represent provenance in RDF data is vast and fragmented (Table 2). There are many approaches, varying in semantics, tuple typology, standard compliance, dependence on external vocabulary, blank node management, granularity and scalability. For an in-depth study of this topic, consult the article *Provenance-Aware Knowledge Representation: A Survey of Data Models and Contextualized Knowledge Graphs* (Sikos and Philp 2020). First, the annotation syntaxes and, subsequently, the knowledge organisation systems related to provenance will be discussed in sections 4.2.1 and 4.2.2.

Type of approach	**Metadata Representation Models**
Quadruples	Named graphs, RDF/S graphsets, RDF triple coloring
Extension of the RDF data model	Notation 3 Logic, RDF^{+}, annotated RDF (aRDF) and Annotated RDF Schema, SPOTL(X), RDF*
Encapsulating Provenance with RDF Triples	PaCE, singleton property
Data models alternative to RDF	GSMM, mapping entities to vectors
Knowledge organisation system	OPM, PML, Provenir, PREMIS, SWAN, DC, PROV, OCDM

Table 2 Annotation frameworks for RDF provenance

4.2.1 ANNOTATION SYNTAXES FOR RDF PROVENANCE

To date, the only standard syntax for annotating triples' provenance is *RDF reification* and is the only one to be compatible with all RDF-based systems. Included since RDF 1.0 (RDF Primer 2004), it consists in associating a statement to a new node of type *rdf:Statement*, which is connected to the triple by the predicates *rdf:subject*, *rdf:predicate*, and *rdf:object*. For example, consider the statement (*exproducts:item10245*, *exterms:weight*, *"2.4" xsd:decimal*) (Figure 1). Using the reification vocabulary, a new node *exproducts:triple12345* is introduced and associated with the following properties:

- (*exproducts:triple12345*, *rdf:type*, *rdf:Statement*)
- (*exproducts:triple12345*, *rdf:subject*, *exproducts:item10245*)
- (*exproducts:triple12345*, *rdf:predicate*, *exterms:weight*)
- (*exproducts:*item10245, rdf:*object*, *"2.4" xsd:decimal*).

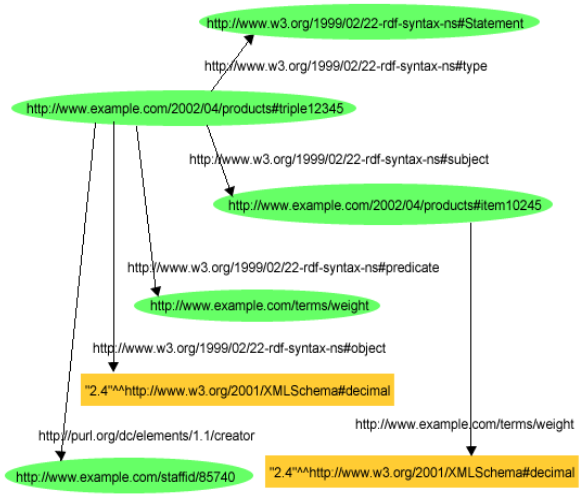

Figure 1 A statement, its reification, and its attribution

Finally, this new URI can become the subject of new provenance triples, as the responsible agent, expressed through the property (*exproducts:item10245*, *dc:creator*, *exstaff:85740*).

Such methodology has a considerable disadvantage: the size of the dataset is at least quadrupled since subject, predicate, and object must be repeated to add at least one provenance's information. There is a shorthand notation, the rdf:ID attribute in RDF/XML (Figure 2), but it is not present in other serializations.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:exterms="http://www.example.com/terms/"
    xml:base="http://www.example.com/2002/04/products">
    <rdf:Description rdf:ID="item10245">
        <exterms:weight rdf:ID="triple12345" rdf:datatype="&xsd;decimal">
            2.4
        </exterms:weight>
    </rdf:Description>
    <rdf:Description rdf:about="#triple12345">
        <dc:creator rdf:resource="http://www.example.com/staffid/85740"/>
    </rdf:Description>
</rdf:RDF>
```

Figure 2 Generating reifications using rdf:ID

Finally, composing SPARQL queries to obtain provenance annotated through *RDF Reification* is cumbersome: to identify the URI of the reification, it is necessary to explicit the entire reference

triple. For all the mentioned reasons, there are several deprecation proposals for this syntax, including that by David Beckett, one of the editors of RDF in 2004 and RDF/XML (Revised) W3C Recommendation:

> *There are a few RDF model parts that should be deprecated (or removed if that seems possible), in particular reification which turned out not to be widely used, understood or implemented even in the RDF 2004 update* (Beckett 2010).

After *RDF Reification*, in 2006, the W3C published a note that suggested a new approach to express provenance, called *n-ary relations* (Defining N-ary Relations on the Semantic Web 2006). In RDF and OWL, properties are always binary relationships between two URIs or a URI and a value. However, sometimes it is convenient to connect a URI to more than one other URI or value, such as expressing the provenance of a certain relationship. The *n-ary relations* allow this behaviour through the instance of a relationship in the form of a blank node. Taking the above example, (*exproducts:item10245*, *exterms:weight*, *"2.4" xsd:decimal*) becomes (*exproducts:item10245*, *exterms:weight*, *_:Weight*). Then, *_:Weight* can be associated with the value by (*_:Weight*, *exterms:weight*, *"2.4"xsd:decimal*), and to the provenance by (*_:Weight*, *dc:creator*, *exstaff:85740*). From an ontological point of view, the *_:Weight* class is a "reified relationship". Therefore, there is a clear similarity between *n-ary relations* and *RDF Reification*, with the difference that the latter reifies the statement, the first the predicate, with the advantage of not having to repeat all the triple elements but only the predicate. The second similarity, which is the main disadvantage of *n-ary relations*, is the need to introduce blank nodes, which cannot be globally dereferenced.

In summary, *RDF Reification* and *n-ary relations* are the only standard and the only alternative recommended by W3C to describe the provenance in RDF and have fatal design flaws. For this reason, different approaches have been proposed since 2005, starting with Named Graphs and *formulae* in Notation 3 Logic, as will be clarified in the following paragraphs.

Named Graphs are graphs associated with a name in the form of a URI. They allow RDF statements describing graphs, with multiple advantages in numerous applications. For example, in Semantic Web publishing, named graphs allow a publisher to sign its graphs so that different information consumers can select specific graphs based on task-specific trust policies. Different tasks require different levels of trust. A naive information consumer may, for example, decide to accept any graph, thus collecting more information as well as false information. Instead, a more cautious consumer may require only graphs signed by known publishers, collecting less but more accurate data (Carroll, et al. 2005). From a syntactical point of view, named graphs are quadruples, where the fourth element is the graph URI that acts as context to triples. It is a solution compatible with the RDF data model, does not rely on

terms or ontologies to capture the provenance, does not cause *triple bloat*, and is therefore scalable and suitable for Big Data applications. On the other hand, concerning serialisation, it is possible to implement named graphs using extensions of RDF/XML, Turtle and N-Triples, called respectively TriX, TriG and N-Quads, all standardised and compatible with the SPARQL algebra.

The above advantages have led the Web Alliance to propose named graphs as a format to express the provenance of scientific statements. The suggested model is called “nanopublications” and represents a fundamental scientific statement with associated context. Precisely, a nanopublication consists of three named graphs: one on data, one on provenance and one on publication metadata (Groth, Gibson and Velterop 2010).

However, named graphs have a limit: they do not allow managing the provenance of implicit triples in the presence of update queries. RDFS allows the addition of semantics to RDF triples, so it is possible, through inference rules, to derive new implicit triples that are not explicitly declared. The moment an update query erases a named graph, all the logic of the triple associates get lost along with the data, and there is no way to separate the two aspects. *RDF/S graphsets*, and its evolution *RDF triple coloring*, extend named graphs to allow RDFS semantics. A graphset is a set of named graphs. It is associated with a URI, preserving provenance information lost following an update, and registering co-ownership of multiple named graphs (Pediaditis, et al. 2009). Similarly, *RDF triple coloring* allows managing scenarios in which the same data has different resources, but co-ownership is implicit (Flouris, et al. 2009). To better understand the problem, Table 2 shows four quadruples whose fourth element is the "color", the source of the triple.

S	**P**	**O**	**“Color”**
TheWashingtonPost	rdf:type	Newspaper	C_4
Newspaper	rdf:type	rdfs:Class	C_3
Newspaper	rdfs:subClassOf	MassMedia	C_3
MassMedia	rdfs:subClassOf	Media	C_5

Table 3 RDF triple coloring example

From the statements in Table 2, it is possible to infer that *Newspaper* is a subclass of *Media* since *Newspaper* is a subclass of *Massmedia* and *Massmedia* is a subclass of *Media*. Thus the origin of the implicit statement *(Newspaper, rdfs:subClassOf, Massmedia)* is C_3 and C_5. Named graphs could express this double provenance only with two separate quadruples. However, the query "returns the triple coloured C_3" would falsely return *(Newspaper, rdfs:subClassOf, Massmedia)*, ignoring that the provenance is not C_3 but C_3 and C_5. *RDF triple coloring* solves the problem by introducing the

operator +, such that $C_{3,5} = C_3 + C_5$. $C_{3,5}$ is a new URI assigned to those triples that have as their source both C_3 and C_5.

Both *RDF/S graphsets* and *RDF triple coloring* are serialisable in TriG, TriX and N-Quads, do not need proprietary terms or external vocabularies and are scalable. However, *RDF/S graphsets* does not comply with either the RDF data model or the SPARQL algebra, unlike *RDF triple coloring*, which is fully compatible.

On the other hand, quadruples are not the only strategy to correlate RDF triples with provenance information. Additionally, the RDF data model can be extended to achieve this goal. The first proposal of this kind was Notation 3 Logic, which introduced the *formulae* (Berners-Lee, Notation 3 Logic 2005). *Formulae* allow producing statements on N3 sentences, which are encapsulated by the syntax {...}. Berners-Lee and Connolly also proposed a *patch file format* for RDF deltas, or three new terms, using N3 (Berners-Lee and Connolly, Delta: an ontology for the distribution of differences between RDF graphs 2004):

1. diff:replacement, that allows expressing any change. Deletions can be written as {…} diff:replacement {}, and additions as {} diff:replacement {…}.
2. diff:deletion, which is a shortcut to express deletions as {…} diff:deletion {…}.
3. diff:insertion, which is a shortcut to express additions as {…} diff:insertion {…}.

The main advantage of this representation is its economy: given two graphs G1 and G2, its cost in storage is directly proportional to the difference between the two graphs. Therefore, it is a scalable approach. However, while conforming to the SPARQL algebra, N3 does not comply with the RDF data model and relies on the N3 Logic Vocabulary.

Adopting a completely different perspective, RDF$^+$ solves the problem by attaching a provenance property and its value to each triple, forming a quintuple (Table 4). In addition, it extends SPARQL with the expression "WITH META *Metalist*", which includes graphs specified in *Metalist*, containing RDF$^+$ meta knowledge statements (Dividino, et al. 2009). To date, RDF$^+$ is not compliant with any standard, neither the RDF data model, nor SPARQL, nor any serialisation formats.

Subject	**Predicate**	**Object**	**Meta-property**	**Meta-value**
:ra/15519	foaf:name	“Silvio Peroni”	:accordingTo	orcid:0000-0002-8420-0696

Table 4 An RDF$^+$ quintuple

Also, *SPOTL(X)* allows expressing a triple provenance through quintuple (Hoffart, et al. 2013). Indeed, the framework's name means Subject Predicate Object Time Location. Optionally, it is possible to create sextuples that add context to the previous elements. *SPOTL(X)* is concretely implemented in YAGO, a knowledge base automatically built from Wikipedia, given the need to specify in which time, space and context a specific statement is true. Outside of YAGO, *SPOTL(X)* does not follow either the RDF data model or the SPARQL algebra, and there is no standard serialisation format.

Similarly, *annotated RDF* (aRDF) does not currently have any standardisation. A triple annotation has the form (s, p: λ, o), where λ is the annotation, always linked to the property (Udrea, Recupero and Subrahmanian 2010). *Annotated RDF Schema* perfects this pattern by annotating an entire triple and presenting a SPARQL extension to query annotations, called AnQL (Zimmermann, et al. 2012). In addition, it specifies three application domains: the temporal, fuzzy and provenance domains (Table 5).

Domain	**Annotated triple**	**Meaning**
Temporal	(niklasZennstrom, ceoOf, skype): [2003, 2007]	Niklas was CEO of Skype during the period 2003 to 2007
Provenance	(niklasZennstrom, ceoOf, skype): wikipedia	Niklas was CEO of Skype according to Wikipedia
Fuzzy	(skype, ownedBy, bigCompany): 0.3	Skype is owned by a big company to a degree not less than 0.3
Temporal, provenance and fuzzy	(niklasZennstrom, ceoOf, skype): <[2003, 2007], 1, wikipedia>	Niklas was without doubt CEO of Skype during the period 2003 to 2007, according to Wikipedia

Table 5 Annotated RDF Schema application examples

The most recent proposal in extending the RDF data model to handle provenance information was RDF*, which embeds triples into triples as the subject or object (Hartig and Thompson 2019). Its main goal is to replace *RDF Reification* through less verbose and redundant semantics. Since there is no serialisation to represent such syntax, Turtle*, an extension of Turtle to include triples in other triples within << and >>, has also been introduced. Similarly, SPARQL* is an RDF*-aware extension

for SPARQL. Later, RDF* was proposed to allow statement-level annotations in RDF streams by extending RSP-QL to RSP-QL* (Keskisärkkä, et al. 2019). YAGO4 has adopted RDF* to attach temporal information to its facts, expressing the temporal scope through *schema:startDate* and *schema:endDate* (Tanon, Weikum and Suchanek, YAGO 4: A Reason-able Knowledge Base 2020). For example, to express that Douglas Adams, *Hitchhiker's Guide to the Galaxy*'s author, lived in Santa Barbara until 2001 when he died, YAGO4 records << *Douglas Adams schema:homeLocation Santa Barbara* >> *schema:endDate 2001*.

After discussing possible RDF extension, two strategies encapsulate provenance in RDF triples: PaCE and singleton properties. Provenance Context Entity (Pace) is an approach concretely implemented in the Biomedical Knowledge Repository (BKR) project at the US National Library of Medicine (Sahoo, Bodenreider, et al. 2010). Its implementation is flexible and varies depending on the application. It allows three granularity levels: the provenance can be linked to the subject, predicate and object of each triple, only to the subject or only to the subject and predicate, through the property *provenir:derives_from*. Therefore, such a solution depends on the Provenir ontology, and it is not scalable because it causes *triple bloat*. Apart from these two flaws, it has several advantages: it leads to 49% less triple than *RDF Reification*, does not involve blank nodes, is fully compatible with the RDF data model and SPARQL, and allows serialisation in any RDF format (RDF/XML, N3, Turtle, N-Triples, RDF-JSON, JSON-LD, RDFa and HTML5 Microdata).

Conversely, singleton properties are inspired by set theory, where a singleton set has a single element. Similarly, a singleton property is defined as "a unique property instance representing a newly established relationship between two existing entities in one particular context" (Nguyen, Bodenreider and Sheth 2014). This goal is achieved by connecting subjects to objects with unique properties that are singleton properties of the generic predicate via the new *singletonPropertyOf* predicate. Then, meta-knowledge can be attached to the singleton property (Table 6). This strategy has been shown to have advantages in terms of query size and query execution time over PaCE (tested on BKR) but disadvantages in terms of triples' number where multiple predications share the same source. Beyond that, singleton properties have the same advantages and disadvantages as PaCE: they rely on a non-standard term, are not scalable, adhere to the RDF data model and SPARQL, and are serialisable in any RDF format.

Subject	Predicate	Object
:ra/15519	:name#1	“Silvio Peroni”
:name#1	:singletonPropertyOf	foaf:name
:name#1	:accordingTo	orcid:0000-0002-8420-0696

Table 6 Singleton property and its meta knowledge assertion example

Table 7 summarises all the considerations made so far on the advantages and disadvantages of the listed RDF-based strategies.

Approach	Tuple type	Compliance with the RDF data model	Compliance with SPARQL	RDF serialiSations	External vocabulary	Scalable
Named graphs	Quadruple	+	+	TriG, TriX, N-Quads	-	+
RDF/S graphsets	Quadruple	-	-	TriG, TriX, N-Quads	-	+
RDF triple coloring	Quadruple	+	+	TriG, TriX, N-Quads	-	+
N3Logic	Triple (in N3)	-	+	N3	N3 Logic Vocabulary	+
aRDF	Non-standard	-	-	-	-	+
Annotated RDF Schema	Non-standard	-	-	-	-	+
RDF^{+}	Quintuple	-	-	-	-	+
SPOTL(X)	Quintuple/sextuple	-	-	-	-	Depends on implementation

RDF*	Non-standard	-	-	Turtle* (non-standard)	-	-
PaCE	Triple	+	+	RDF/XML, N3, Turtle, N-Triples, RDF-JSON, JSON-LD, RDFa, HTML5 Microdata	Provenir ontology	-
Singleton property	Triple	+	+	RDF/XML, N3, Turtle, N-Triples, RDF-JSON, JSON-LD, RDFa, HTML5 Microdata	*singletonPropertyOf* property	-

Table 7 Advantages and disadvantages of all metadata representations models to add provenance information to RDF data.

Finally, there are data models alternative to RDF to organise knowledge. The General Semistructured Meta-model (GSMM) is a meta-model to aggregate heterogeneous data models into a single formalism to manage them in a homogeneous way or compare (Damiani, et al. 2019). A triple-based database can be converted to a GSMM graph by introducing nodes for subjects, predicates and objects, where predicates have an incoming edge labelled < TO >, and an incoming edge labelled < FROM >. Since predicates are modelled as nodes, GSMM supports reification and allows to represent provenance. On the other hand, research exists to convert knowledge bases' entities into embeddings in a vector space (Suchanek, et al. 2019). With embeddings, impossible operations in RDF representations can be performed, such as predicting links (using neural networks) or new facts (using logical rules).

4.2.2 KNOWLEDGE ORGANISATION SYSTEMS FOR RDF PROVENANCE

Historically, many vocabularies and ontologies have been introduced to represent provenance information, either upper ontologies, domain ontologies, and provenance-related ontologies. Among the upper ontologies, the Open Provenance Model stands out because of its interoperability. It describes the history of an entity in terms of processes, artefacts and agents (Moreau, Clifford, et al. 2011), a pattern that will be discussed later about the PROV Data Model (2013). On the other hand, the Proof Markup Language (PML) is an ontology designed to support trust mechanisms between heterogeneous web services (Pinheiro da Silva, McGuinness and Fikes 2006).

About domain-relevant models, there is the Provenir Ontology for eScience (Sahoo and Sheth, Provenir Ontology: Towards a Framework for eScience Provenance Management 2009), PREMIS for archived digital objects, such as files, bitstreams and aggregations (Caplan 2017) and Semantic Web Applications in Neuromedicine (SWAN) Ontology to model a scientific discourse in the context of biomedical research (Ciccarese, et al. 2008). Finally, the Dublin Core Metadata Terms allows to express the provenance of a resource and specify what is described (e.g. dct:BibliographicResource), who was involved (e.g. dct:Agent), when the changes occurred (e.g. dct:dateAccepted), and the derivation (e.g. dct:references), sometimes very precisely (DCMI Metadata Terms 2020).

All the requirements and ontologies mentioned have been merged into a single data model, the PROV Data Model (Moreau, Clifford, et al. 2011), translated into the PROV Ontology using the OWL 2 Web Ontology Language (PROV-O: The PROV Ontology 2013). It provides several classes, properties, and restrictions, representing provenance information in different systems and contexts. Its level of genericity is such that it is even possible to create new classes and data model-compatible properties for new applications and domains. Just like the Open Provenance Model, PROV-DM captures the provenance under three complementary perspectives:

- *Agent-centred provenance*, which people, organisations, software, inanimate objects, or other entities are involved in the generation, manipulation, or influence of a resource. For example, concerning a journal article, it is possible to distinguish between the author, the editor, and the publisher. PROV-O maps the responsible agent with prov:Agent, the relationship between an activity and the agent with prov:wasAssociatedWith and an entity's attribution to an agent with prov:wasAttributedTo.
- *Object-centred-provenance*, which is the origin of a document's portion from other documents. Taking the example of the article, a fragment of it can quote an external document.

PROV-O maps a resource with prov:Entity, whether physical, digital, or conceptual, while the predicate prov:wasDerivedFrom expresses a derivation relationship.

- *Process-centred provenance*, or the actions and processes necessary to generate a resource. For example, an editor can edit an article to correct spelling errors using the previous version of the document. PROV-O expresses the concept of action with prov:Activity, creating an entity with the predicate prov:wasGeneratedBy and the use of another entity to complete a passage with prov:used.

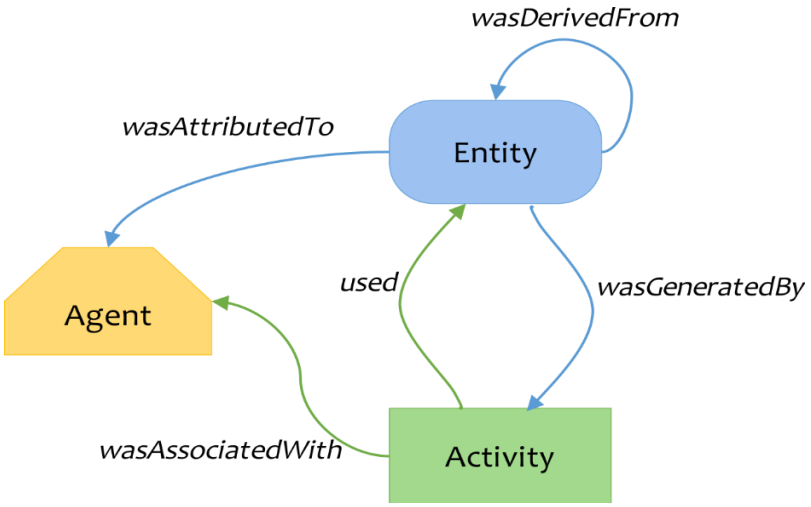

Figure 3 High level overview diagram of PROV records (PROV Model Primer 2013)

The diagram in Figure 1 provides a high-level view of the discussed concepts' structure, constituting the so-called "starting point terms". PROV-O is much more extensive and provides sophisticated entities, agents, activities, and relationships in a modular way, namely "expanded terms" and "qualified terms".

The OpenCitations Data Model, used in this research, relies on the flexibility of PROV-O to record the provenance of bibliographic datasets (Daquino, et al. 2020). Each bibliographical entity described by the OCDM is annotated with one or more snapshots of provenance. The snapshots are of type prov:Entity and are connected to the bibliographic entity described through prov:specializationOf, predicate present in the mentioned "expanded terms". Being the specialisation of another entity means sharing every aspect of the latter and, in addition, presenting more specific aspects, such as an abstraction, a context or, in this case, a time. In addition, each snapshot records the validity dates (prov:generatedAtTime, prov:invalidatedAtTime), the agents responsible for both creation and modification of the metadata (prov:wasAttributedTo), the primary sources (prov:hadPrimarySource) and a link to the previous snapshot in time (prov:wasDerivedFrom). The model is summarised in Figure 2.

In addition, OCDM extends the Provenance Ontology by introducing a new property called

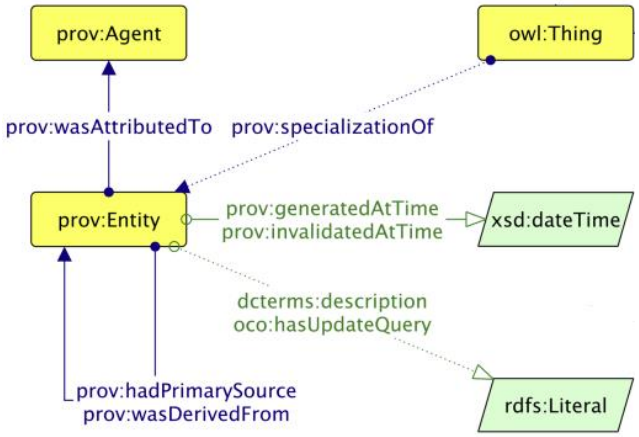

Figure 4 Provenance in the OpenCitations Data Model

hasUpdateQuery, a mechanism to record additions and deletions from an RDF graph with a SPARQL INSERT and SPARQL DELETE query string. The *snapshot-oriented* structure, combined with a system to explicitly indicate how a previous snapshot was modified to reach the current state, makes it easier to recover the current statements of an entity and restore an entity to a specific snapshot. The current statements are those available in the present dataset, while recovering a snapshot s_i means applying the reverse operations of all update queries from s_n to s_{i+1} (Peroni, Shotton and Vitali 2016).

This expedient, initially adopted for the OpenCitations Corpus, was designed to foster reusability in other contexts and is the added value of the provenance model proposed in the OCDM, which is the basis for the library to perform time agnostic queries presented in this work.

In the next section, the existing literature on tracking changes in RDF data will be deepened, focusing on the sources that inspired the OpenCitations provenance model.

4.3 TRACKING CHANGES OF RDF DATA

In section 4.2, several strategies have been introduced to correlate RDF data with provenance information. Up to now, the discourse has been generic and extended to all the "content" category dimensions summarized in Table 1, that is, to all the possible provenance types: the attribution, the processes, the justification, the entailment and the versioning. The Time Agnostic Library, which is the main contribution of this work, deals with versioning and the temporal dimension of provenance.

However, even time is a multidimensional concept, which can be evaluated from two points of view. On the one hand, the *transaction time*, defined in the temporal databases literature as "the time a fact was present in a database as stored data". On the other, the *valid time*, which is instead "the time a fact was true in reality" (Snodgrass 1986). For example, the statement (*dbr:Rome*, *dbo:capital*, *dbr:Roman_Empire*) is currently found in DBPedia[1]. Nevertheless, in the present, that is a false statement because its valid time goes from 27 B.C. to 395 A.D. when the Empire split into the Western Roman Empire and the Eastern Roman Empire and the Western capital was moved to Milan. "Validity" is a dense concept, especially in Digital Humanities, subject to conjectures that transcend the boundaries of the current discourse (Barabucci, Tomasi and Vitali 2021). Instead, we will focus exclusively on transaction time and, in particular, on how to keep track of the changes in RDF datasets.

Before discussing how RDF datasets currently track changes, the importance of implementing such policies must be understood. The Web of Data has an intrinsic dynamic nature, and the Dynamic Linked Data Observatory was born to measure how it evolves. The project was launched in 2012 with the aim of releasing weekly snapshots of the Semantic Web based on 791 domains[2] (Umbrich, et al. 2010). In 2013, the project monitored the evolution of 86,696 RDF documents for 29 weeks (Käfer, et al. 2013). Among the most significant conclusions, it was found that 37.8% of documents were modified during that period, that about 20% of the documents were temporarily unavailable and that 5% disappeared permanently. Without a system to understand when a resource was altered, why it was updated and who was responsible for its revision, the information's reliability is seriously questioned.

[1] https://dbpedia.org/page/Rome
[2] http://km.aifb.kit.edu/projects/dyldo/data/

4.3.1 STORING AND QUERYING DYNAMIC LINKED OPEN DATA

In order to store and query how an RDF dataset evolves, various archiving policies have been elaborated, namely *independent copies*, *change-based* and *timestamp-based* policies (Fernández, Polleres and Umbrich 2015). Table 8 lists the main knowledge bases, version control systems and archives for RDF, divided by storage policy. They will be deepened in the following paragraphs, and the allowed query typologies will be discussed.

Archiving policy	**Datasets / Software**
Independent copies (IC)	DBPedia, Wikidata, YAGO, Dynamic Linked Data Observatory, SemVersion, PromptDiff
Change-based (CB)	(Im, Lee and Kim 2012), (Papavasileiou, et al. 2013), R&Wbase
Timestamp-based (TB)	x-RDF-3X, v-RDFCSA
Hybrid	OSTRICH (CB/TB), OpenCitations Corpus (CB/TB), (Tanon and Suchanek 2019) (IC/CB/TB)

Table 8 Datasets and software divided by storage policy.

Two query and focus types are identified in (Fernández, Umbrich, et al. 2016). On the one hand, a query can be a materialization or structured; on the other, the focus can affect a version or a delta. Combining query and focus types results in six possible retrieval functionalities (Table 9). 1) Version materialisation: the request to obtain a full version of a specific resource. This feature is the most common, provided by any version control system for RDF. 2-3) Single-version structured query and cross-version structured query: queries made on a specific version or through different versions. The latter is also called a time-traversal query. 4) Delta materialisation is to get the differences between two versions of a specific resource. This feature is handy for RDF authoring applications and operations in version control systems, such as merge or conflict resolution. 5-6) Single-delta structured queries and cross-delta structured queries: the equivalent of 2-3), but satisfied on deltas instead of versions.

	Materialization	Structured queries	
		Single time	**Cross time**
Version	Version materialization *Get snapshot at time t_i*	Single-version structured queries *Articles written by a specific author at time t_i*	Cross-version structured queries *Articles associated with the same DOI simultaneously*
Delta	Delta materialization *Get delta at time t_i*	Single-delta structured queries *DOI modified between two consecutive snapshots*	Cross-delta structured queries *The most significant change in the number of articles in the history of the dataset*

Table 9 Retrieval functionalities according to (Fernández, Umbrich, et al. 2016).

The policy called *independent copies* consists of storing each version separately. Two levels of granularity are possible: either a copy of the entire dataset is saved, or only resources that change. This strategy is sometimes defined as *physical snapshots* in the literature (Peroni, Shotton and Vitali 2016). It is the most straightforward model to implement and allows obtaining versions materializations with great ease. However, the disadvantages are much more consistent: first, a massive amount of space and time is needed; furthermore, given the different statements' versions, further diff mechanisms are needed to identify what has changed. Nevertheless, to date, this is the archiving policy adopted by most systems and knowledge bases.

The first version control systems for RDF were PromptDiff (Noy and Musen 2002) and SemVersion (Völkel, et al. 2005), specially tailored for ontologies. Inspired by CVS, the classic version control system for text documents, they save each version of an ontology in a separate space. In addition, PromptDiff provides diff algorithms to compute deltas between two versions and see what has changed, applying ten heuristic matchers. The results of a matcher become the input for others until they produce no more changes. On the other hand, SemVersion provides two diff algorithms: one *structure-based*, which returns the difference between explicit triples in two graphs, the other *semantic-aware*, which also considers the triples inferred through RDFS relations. Differences are calculated on the fly in both approaches, while all ontology's versions take up space on the disk. For this reason, SemVersion and PromptDiff are classified as having *independent-copies* archiving policies, despite the article from which this classification is taken consider them as *changed-based* systems (Fernández, Polleres and Umbrich, Towards Efficient Archiving of Dynamic Linked 2015). As for the allowed queries, they are limited to the delta end version materialization in both cases.

Concerning knowledge bases, DBpedia (Lehmann, et al. 2015) publicly releases snapshots of the entire dataset at regular intervals. Therefore, in the specific case of DBpedia, a further problem arises: many changes may not be reflected in the snapshots, that is, all statements with a lifespan shorter than the interval between snapshots. There are proposals to fill this gap, such as exploiting Wikipedia revisions history information (Orlandi and Passant 2011). Similarly, Yago releases backups of the whole dataset, downloadable in the Downloads section of the website[3]. Since the Yago data model has changed significantly from the first to the fourth edition, each version can be downloaded separately.

On the other hand, Wikidata does not save the whole dataset but only the resources that change (F. Erxleben, et al. 2014). Wikibase, the database used for Wikidata, creates a revision associated with a specific entity every time the related page is modified (Dooley and Božić 2019). Within each revision, in the "text" field, there is a complete copy of that page after the change. Some metadata are also saved, such as the timestamp, contributor's username and id and a comment summarizing what was modified (Figure 5). This information is stored in compressed XML files and made available for download on the Wikidata website[4]. However, the content of the text field is not in XML format, but in JSON format, with all non-ASCII characters escaped. On the Wikidata site, through the user interface, it is possible to explore the content of a single revision and compute on the fly the delta between two or more versions. Though, there is no way to perform SPARQL queries on revisions.

[3] https://yago-knowledge.org/downloads
[4] https://www.wikidata.org/wiki/Wikidata:Database_download

```
<page>
  <title>Q78189694</title>
  <ns>0</ns>
  <id>77644210</id>
  <revision>
    <id>1467205756</id>
    <parentid>1233484847</parentid>
    <timestamp>2021-07-26T18:45:13Z</timestamp>
    <contributor>
      <username>Twofivesixbot</username>
      <id>2691515</id>
    </contributor>
    <comment>/* wbeditentity-update-languages-short:0||bn */ KOI</comment>
    <model>wikibase-item</model>
    <format>application/json</format>
    <text bytes="19449" xml:space="preserve">{&quot;type&quot;:&quot[…]}</text>

    <sha1>jm79xfec7qbv4o5adf7umx1r94wblh4</sha1>
  </revision>
</page>
```

Figure 5 Wikidata revision example

The *change-based* policy was introduced to solve scalability problems caused by the *independent copies* approach. It consists of saving only the deltas between one version and the other. For this reason, delta materialization is costless. On the flip side, to support version-focused queries, additional computational costs for delta propagation is required.

The first proposal of this kind was described in the article *A Version Management Framework for RDF Triple Stores* (Im, Lee and Kim 2012). The idea is to store the original dataset and the deltas between two consecutive versions. However, as have been said, performing version queries requires rebuilding that state on the fly. In order to avoid performance problems, deltas are compressed in Aggregated Deltas to directly compute the version of interest instead of considering the whole sequence of deltas. In other words, all possible deltas are stored in advance, and duplicated or unnecessary modifications are deleted. Finally, the article analyzes the performance for structured queries on a single version, on a single delta and cross-delta. However, no mention is made of possible queries on multiple versions.

If the dataset's data model includes RDFS, reducing the deltas' size or generating high-level deltas is possible. The article *High-Level Change Detection in RDF(S) KBs* introduces the *language of change*, where every possible change has well-defined semantics (Papavasileiou, et al. 2013). In particular, it proposes 132 types of change, 54 of which are basic, 51 are composite, and 27 are heuristic changes. Deltas are computed on the fly from added and removed triples – that is, from low-level deltas – and are both human-readable and machine-interpretable. See Table 7 for an example of a heuristic change.

Low-Level Delta		High-Level Delta
Added Triples	**Deleted Triples**	**Detected Changes**
(Stuff,subClassOf, Persistent)	(Stuff,subClassOf, Existing)	Rename Class(Existing, Persistent)
(started on,domain, Persistent)	(started on,domain, Existing)	
(Persistent,type,class)	(Existing,type,class)	

Table 10 High-level changes compared to low-level changes.

However, low-level deltas are easier to compute and manage, although they take up more disk space and are less expressive. Moreover, it is impossible to generate high-level deltas without underlying semantics based on RDFS and OWL. Therefore, such a solution can not be implemented in any context. Finally, the article makes no mention of possible structured queries.

A concrete example of a *change-based* policy application is R&Wbase, a version control system inspired by Git but designed for RDF (Sande, et al. 2013). Triples are stored in quads, where the context identifies the version and whether the triple was added or removed. More specifically, each delta is associated with a version number higher than all previous values. Additions have an even value of 2y, while deletions have an odd value of 2y+1. Finally, insertions-related graphs store metadata, such as the date, the responsible agent and the parent delta. The main advantage of this approach is that it allows single-version structured queries at query-time: a so-called interpretation layer is responsible for translating SPARQL queries to find all the ancestors of a resource at a specific time. In other words, to answer a query on a $2y_n$ version, the interpreter finds all ancestors $A_{2yn} = \{2_{yi},...,2_{yj}\}$. The query specifies the time via FROM <version_graph_URI>, where the graph's path is either a hash or "master". In order to speed up the process, triples in both the additions and deletions graphs are excluded, and the most frequent queries can be cached. The article does not mention any other type of query and whether it is possible to indicate more than one graph for cross-version structured queries. In any case, since a not human-readable version's URI must be known, that could be considered as a cumbersome solution.

On the other hand, the *timestamp-based* policy consists of annotating each triple with the timestamp of the version in which that statement was in the dataset. Annotated RDF Schema can be used to achieve this, combined with AnQL to perform queries, as seen in chapter 4.2.1 (Zimmermann, et al. 2012). However, implementations of that solution are not known. On the contrary, x-RDF-3X is a database for RDF designed to manage high-frequency online updates, versioning, time-travel queries

and transactions (Neumann and Weikum 2010). The triples are never deleted but are annotated with two fields: the insertion and deletion timestamp, where the last one has zero value for currently living versions. Afterwards, updates are saved in a separate workspace and merged into various indexes at occasional savepoints. A dictionary is used to encode strings in short IDs and compressed clustered B+ trees to index data in lexicographic order. Time-travel queries are speedy because of indexes, but no approach to return deltas or query them is mentioned.

v-RDFCSA uses a similar strategy but excels particularly for reducing space requirements, managing to compress 325 GB of storage into 5.7 - 7.3GB (Cerdeira-Pena, et al. 2016). To achieve that result, it compresses both the RDF archive and the timestamps attached to the triples. All types of queries are explicitly allowed.

Finally, there are hybrid storage policies that combine the changed-based approach with the timestamp-based approach. For example, OSTRICH is a triplestore that retains the first version of a dataset and subsequent deltas, as seen in (Im, Lee and Kim 2012). However, it merges changesets based on timestamps to reduce redundancies between versions, adopting a change-based and timestamp-based approach simultaneously (Taelman, Sande and Verborgh 2018). OSTRICH supports version materialization, delta materialization and single-version queries.

The OpenCitations Corpus embraces a similar hybrid approach, which is mirror-like and opposite to that seen in (Im, Lee and Kim 2012) and OSTRICH: the present state is the only one stored, not the original one. For each entity, a provenance graph is generated as a result of an update. The delta versus the next version is expressed as a SPARQL query in the property *oco:hasUpdateQuery*. In addition, each provenance graph contains transactional time information, expressed via *prov:generatedAtTime* and *prov:invalidatedAtTime*, that is, the insertion and deletion timestamps. The advantage is that the most interesting dataset's state, the current one, is immediately available and must not be reconstructed. It is worth mentioning that, to date, the OpenCitations Corpus is the only bibliographical database to implement change-tracking mechanisms. Among the leading players in the field, neither Web of Science nor Scopus have adopted solutions in this regard.

To conclude, software exists that adopt all three archiving policies. For example, (Tanon and Suchanek 2019) propose a system to fill the already mentioned Wikidata gap, which provides provenance data but does not allow queries. XML dumps downloaded from Wikidata are organized into four graphs: a global state graph, which contains a named graph on the global state of Wikidata after each revision; an addition and deletion graphs, which contain all the added and deleted triples for revision; and a default graph, containing metadata for each revision, such as the author, the timestamp, the id of the modified entity, the previous version of the same entity and the URIs of the

additions, deletions and global state graphs. Since the sum of these graphs would weigh exabytes, they are not directly saved into a triplestore, but RocksDB[5] is used to store specific indexes. Four kinds of indexes are generated: dictionary indexes, in which each string is associated to an integer and vice versa; content indexes, which associate the permutations *spo*, *pos* and *osp* to the respective transaction time in the form [start, end[; revision indexes, which provides the set of added and removed triples for a given revision; and meta indexes, which provide the relevant metadata for each revision. The use of each storage policy allows managing all kinds of queries efficiently.

Table 11 summarizes all the considerations made regarding possible query categories for various software and whether these are computed on the fly or need an index. Knowledge bases and datasets, such as Dbpedia, Yago, Wikidata and the OpenCitations Corpus, were excluded from the table since they are interesting only for storage policies and separate software is required for queries. For the same reason, the proposal by Papavasileiou *et al.* is not categorized either (Papavasileiou, et al. 2013).

Software	**Version materialization**	**Delta materialization**	**Single-version structured query**	**Cross-version structured query**	**Single-delta structured query**	**Cross-delta structured query**	**On the fly**
PromptDiff	+	+	-	-	-	-	+
SemVersion	+	+	-	-	-	-	+
(Im, Lee and Kim 2012)	+	+	+	-	+	+	-
R&Wbase	+	+	+	-	-	-	+
x-RDF-3X	+	-	+	+	-	-	-

[5] https://rocksdb.org/

v-RDFCSA	+	+	+	+	+	+	-
OSTRICH	+	+	+	-	-	-	-
(Tanon and Suchanek 2019)	+	+	+	+	+	+	-

Table 11 Software catalogued by allowed query types and the need for indexing.

From Table 11, it is clear that all the existing solutions need indexes and pre-processing to manage time-travel queries efficiently. Software that performs operations on the fly, such as R&Wbase, do not allow cross-version structured queries. This flaw can prove fatal in dynamic open linked datasets that constantly receive many updates, such as Wikidata. This work aims to introduce a Python library to perform time-agnostic queries on the fly, exploiting the OpenCitations' provenance model.

5 MATERIALS AND METHODS

Descrizione generale e astratta dei metodi, deve essere comprensibile da persone non esperte di software. Non si parla minimamente di software.

6 THE TIME AGNOSTIC LIBRARY

Come ho sviluppato la libreria e come funziona. Non si entra nel dettaglio, si dà una visione generale.

7 LIBRARY IMPLEMENTATION

Only nerds are allowed.

8 DISCUSSION

Benchmark e confronto con soluzioni esistenti.

9 CONCLUSION

Ricapitolo tutto.

10 REFERENCES

Aggelen, Astrid van, Laura Hollink, and Jacco van Ossenbruggen. “Combining Distributional Semantics and Structured Data to Study Lexical Change.” Edited by Laura Hollink, Sándor Darányi, Albert Meroño Peñuela and Efstratios Kontopoulos. *Detection, Representation and Management of Concept Drift in Linked Open Data.* Springer, 2016. 40-49.

Barabucci, Gioele. “Introduction to the Universal Delta Model.” *Proceedings of the 2013 ACM Symposium on Document Engineering.* Florence, Italy: Association for Computing Machinery, 2013. 47–56.

Barabucci, Gioele, Francesca Tomasi, and Fabio Vitali. “Supporting Complexity and Conjectures in Cultural Heritage Descriptions.” *CEUR Workshop Proceedings* 2810 (2021): 104-115.

Barabucci, Gioele, Paolo Ciancarini, Angelo Di Iorio, and Fabio Vitali. “Measuring the quality of diff algorithms: a formalization.” *Computer Standards & Interfaces* 46 (2016): 52-65.

Beckett, David. “RDF Syntaxes 2.0.” *W3C.* 10 April 2010. https://www.w3.org/2009/12/rdf-ws/papers/ws11 (accessed 07 22, 2021).

Berners-Lee, Tim. “Notation 3 Logic.” *W3C.* August 2005. https://www.w3.org/DesignIssues/N3Logic (accessed 07 23, 2021).

—. *Weaving the Web: the original design and ultimate destiny of the World Wide Web.* San Francisco: Harper San Francisco, 1999.

Berners-Lee, Tim, and Dan Connolly. “Delta: an ontology for the distribution of differences between RDF graphs.” 2004.

Caplan, Priscilla. *Understanding PREMIS: an overview of the PREMIS Data Dictionary for Preservation Metadata.* Library of Congress, 2017.

Carroll, Jeremy J., Christian Bizer, Pat Hayes, and Patrick Stickler. “Named graphs, provenance and trust.” *Proceedings of the 14th international conference on the World Wide Web.* New York: Association for Computing Machinery, 2005. 613–622.

Cerdeira-Pena, Ana, Antonio Farina, Javier D Fernández, and Miguel A Martınez-Prieto. “Self-indexing rdf archives.” *Proceedings of IEEE Data Compression Conference.* 2016.

Ciccarese, Paolo, et al. “The SWAN Scientific Discourse Ontology.” *Journal of biomedical informatics* 41, no. 5 (2008): 739–751.

Damiani, Ernesto, Barbara Oliboni, Elisa Quintarelli, and Letizia Tanca . “A graph-based meta-model for heterogeneous data management.” *Knowledge and Information Systems* 61, no. 1 (2019): 107–136.

Daquino, Marilena, et al. “The OpenCitations Data Model.” Edited by Jeff Z. Pan, et al. *International Semantic Web Conference.* Springer, Cham, 2020. 447-463.

“DCMI Metadata Terms.” *Dublin Core Metadata Initiative.* 20 01 2020. http://dublincore.org/specifications/dublin-core/dcmi-terms/2020-01-20/ (accessed 07 16, 2021).

Noy, Natasha, and Alan Rector. “Defining N-ary Relations on the Semantic Web.” *W3C.* 12 04 2006. http://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/ (accessed 07 22, 2021).

Ding, Li, Tim Finin, Yun Peng, Paulo Pinheiro da Silva, and Deborah L. “Tracking RDF Graph Provenance.” Technical report, 2005.

Dividino, Renata, Sergej Sizov, Steffen Staab, and Bernhard Schueler. “Querying for provenance, trust, uncertainty and other meta knowledge in RDF.” *Journal of Web Semantics* 7, no. 3 (2009): 204-219.

Dooley, Paula, and Bojan Božić. “Towards Linked Data for Wikidata Revisions and Twitter.” *iiWAS2019: Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services.* New York, NY, USA: Association for Computing Machinery, 2019. 166–175.

Erxleben, F., M. Günther, M. Krötzsch, J. Mendez, and D Vrandečić. “Introducing Wikidata to the Linked Data Web.” *The Semantic Web – ISWC 2014.* Springer International Publishing, 2014. 50–65.

Erxleben, Fredo, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. “Introducing Wikidata to the Linked Data Web.” *The Semantic Web – ISWC 2014.* Cham: Springer, 2014. 50-65.

Fernández, Javier D., Axel Polleres, and Jürgen Umbrich. “Towards Efficient Archiving of Dynamic Linked.” *DIACRON@ESWC.* Portorož, Slovenia : Computer Science, 2015. 34–49.

Fernández, Javier D., Jürgen Umbrich, A. Polleres, and Magnus Knuth. “Evaluating Query and Storage Strategies for RDF Archives.” *Proceedings of the 12th International Conference on Semantic Systems.* 2016.

Flouris, Giorgos, Irini Fundulaki, Panagiotis Pediaditis, Yannis Theoharis, and Vassilis Christophides. “Coloring RDF Triples to Capture Provenance.” *The Semantic Web - ISWC 2009.* Springer, Berlin, Heidelberg, 2009.

Fokkens, Antske, Serge ter Braake, Isa Maks, and Davide Ceolin. “On the Semantics of Concept Drift: Towards Formal Definitions of Semantic Change.” Edited by Laura Hollink, Sándor Darányi, Albert Meroño Peñuela and Efstratios Kontopoulos. *Detection, Representation and Management of Concept Drift in Linked Open Data.* CEUR, 2016. 10-17.

Gil, Yolanda, et al. “Provenance XG Final Report.” W3C, 08 December 2010.

Groth, Paul, Andrew Gibson, and Jan Velterop. “The anatomy of a nanopublication.” *Information Services & Use* 30, no. 1-2 (09 2010): 51-56.

Hartig, Olaf, and Bryan Thompson. “Foundations of an Alternative Approach to Reification in RDF.” (arXiv) March 2019.

Hoffart, Johannes, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. “YAGO2: A spatially and temporally enhanced knowledge base.” *Artificial Intelligence* 194 (2013): 28-61.

Im, Dong-Hyuk, Sang-Won Lee, and Hyoung-Joo Kim. “A Version Management Framework for RDF Triple Stores.” *International Journal of Software Engineering and Knowledge Engineering* 22 (2012): 85-106.

Käfer, Tobias, Ahmed Abdelrahman, Jürgen Umbrich, Patrick O’ Byrne, and Aidan Hogan. “Observing Linked Data Dynamics.” *The Semantic Web: Semantics and Big Data.* Berlin, Heidelberg: Springer, 2013. 213-227.

Keskisärkkä, R., E. Blomqvist, L. Lind, and O. Hartig. “RSP-QL* : Enabling Statement-Level Annotations in RDF Streams.” *The Power of AI and Knowledge Graphs. SEMANTiCS 2019. Lecture Notes in Computer Science.* Karlsruhe, Germany: Springer, Cham, 2019.

Lehmann, Jens, et al. “DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia.” *Semantic Web* 6, no. 2 (2015): 167-195.

Moreau, Luc, et al. “The Open Provenance Model core specification (v1.1).” *Future Generation Computer Systems,* 27, no. 6 (2011): 743-756.

Neumann, Thomas, and Gerhard Weikum. “x-RDF-3X: Fast Querying, High Update Rates, and Consistency for RDF Databases.” *Proceedings of the VLDB Endowment.* 2010. 256–263.

Nguyen, Vinh, Olivier Bodenreider, and Amit Sheth. “Don't like RDF reification?: making statements about statements using singleton property.” *WWW '14: Proceedings of the 23rd international conference on World wide web.* New York, NY, USA: Association for Computing Machinery, 2014. 759–770.

Noy, N. F., and M. A. Musen. “Promptdiff: A Fixed-Point Algorithm for Comparing Ontology Versions.” *Proc. of IAAI.* 2002. 744–750.

Ognyanov, Damyan, and Atanas Kiryakov. “Tracking Changes in RDF(S) Repositories.” Edited by Gómez-Pérez A. and Benjamins V.R. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web.* Berlin, Heidelberg: Springer, 2002. 373-378.

Orlandi, Fabrizio, and Alexandre Passant. “Modelling provenance of DBpedia resources using Wikipedia contributions.” *Journal of Web Semantics* 9, no. 2 (2011): 149-164.

Papavasileiou, Vicky, Giorgos Flouris, Irini Fundulaki, Dimitris Kotzinos, and Vassilis Christophides. “High-level Change Detection in RDF(S) KBs.” *ACM Transactions on Database Systems* 38, no. 1 (2013).

Pediaditis, P., G. Flouris, I. Fundulaki, and V. Christophides. “On Explicit Provenance Management in RDF/S Graphs.” *First Workshop on the Theory and Practice of Provenance.* San Francisco, CA, USA: USENIX, 2009.

Peroni, Silvio, David Shotton, and Fabio Vitali. “A Document-inspired Way for Tracking Changes of RDF Data.” Edited by Laura Hollink, Sándor Darányi, Albert Meroño Peñuela and Efstratios Kontopoulos. *Detection, Representation and Management of Concept Drift in Linked Open Data.* Bologna: CEUR Workshop Proceedings, 2016. 26-33.

Pinheiro da Silva, Paulo, Deborah L. McGuinness, and Richard Fikes. “A proof markup language for Semantic Web services.” *Information Systems* 31, no. 4–5 (2006): 381-395.

Gil, Yolanda, and Simon Miles. “PROV Model Primer.” *W3C.* 30 04 2013. http://www.w3.org/TR/2013/NOTE-prov-primer-20130430/ (accessed 07 16, 2021).

Moreau, Luc, and Paolo Missier, . “PROV-DM: The PROV Data Model.” *W3C*. 30 04 2013. http://www.w3.org/TR/2013/REC-prov-dm-20130430/ (accessed 07 16, 2021).

Provenance Incubator Group Charter. 2010. https://www.w3.org/2005/Incubator/prov/charter (accessed July 15, 2021).

Lebo, Timothy, Satya Sahoo, and Deborah McGuinness. “PROV-O: The PROV Ontology.” *W3C*. 30 04 2013. http://www.w3.org/TR/2013/REC-prov-o-20130430/ (accessed 07 16, 2021).

Manola, Frank, and Eric Miller. “RDF Primer.” Vers. 1.0. *W3C*. 10 February 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ (accessed 07 22, 2021).

Recchia, Gabriel, Ewan Jone, Paul Nulty, John Regan, and Peter de Bolla. “Tracing Shifting Conceptual Vocabularies Through Time.” *Knowledge Engineering and Knowledge Management*. Cham: Springer International Publishing, 2017. 19-28.

Sahoo, Satya S., and Amit P. Sheth. “Provenir Ontology: Towards a Framework for eScience Provenance Management.” 2009.

Sahoo, Satya S., Olivier Bodenreider, Pascal Hitzler, Amit Sheth, and Krishnaprasad Thirunarayan. *Provenance Context Entity (PaCE): Scalable Provenance Tracking for Scientific RDF Data*. Vol. 6187, in *Scientific and Statistical Database Management*, by Gertz M. and Ludäscher B., 461-470. Berlin, Heidelberg: Springer, 2010.

Sande, Miel Vander, Pieter Colpaert, Ruben Verborgh, Sam Coppens, Erik Mannens, and Rik Van de Walle. “R&Wbase: Git for triples.” *Proceedings of the 6th Workshop on Linked Data on the Web*. CEUR Workshop Proceedings, 2013.

Sikos, L.F., and D. Philp. “Provenance-Aware Knowledge Representation: A Survey of Data Models and Contextualized Knowledge Graphs.” *Data Science and Engineering* 5, no. 3 (2020): 293-316.

Snodgrass, Richard. “Temporal Databases.” *IEEE Computer* 19 (1986): 35–42.

Suchanek, Fabian M., Jonathan Lajus, Armand Boschin, and Gerhard Weikum. “Knowledge Representation and Rule.” Edited by Markus Krötzsch and Daria Stepanova. *Reasoning Web. Explainable Artificial Intelligence: 15th International Summer School 2019, Bolzano, Italy, September 20-24, 2019, Tutorial Lectures*. Springer International Publishing, 2019. 110-152.

Taelman, Ruben, Miel Vander Vander Sande, and Ruben Verborgh. “OSTRICH: Versioned Random-Access Triple Store.” *Companion Proceedings of the Web Conference 2018*. 2018. 127-130.

Tanon, Thomas Pellissier, and Fabian M. Suchanek. “Querying the Edit History of Wikidata.” *Extended Semantic Web Conference.* Portorož, Slovenia, 2019. 161-166.

Tanon, Thomas Pellissier, Gerhard Weikum, and Fabian Suchanek. “YAGO 4: A Reason-able Knowledge Base.” *The Semantic Web. ESWC 2020.* Cham: Springer, 2020. 583-596.

Udrea, Octavian, Diego Reforgiato Recupero, and V. S. Subrahmanian. “Annotated RDF.” *ACM Transactions on Computational Logic* 11, no. 2 (January 2010): 1–41.

Umbrich, Jürgen, Michael Hausenblas, Aidan Hogan, Axel Polleres, and Stefan Decker. “Towards Dataset Dynamics: Change Frequency of Linked Open Data Sources.” Edited by Christian Bizer, Tom Heath, Tim Berners-Lee and Michael Hausenblas. *Proceedings of the WWW2010 Workshop on Linked Data on the Web.* Raleigh, USA: CEUR Workshop Proceedings, 2010.

Völkel, Max, W. Winkler, York Sure, S. Kruk, and Marcin Synak. “SemVersion: A Versioning System for RDF and Ontologies.” *Proc. of ESWC.* 2005.

Zimmermann, Antoine, Nuno Lopes, Axel Polleres, and Umberto Straccia. “A general framework for representing, reasoning and querying with annotated Semantic Web data.” *Journal of Web Semantics* 11 (2012): 72-95.