Earth Science Informatics
https://doi.org/10.1007/s12145-021-00574-w

REVIEW ARTICLE

RDF for temporal data management – a survey

Fu Zhang [1] · Zhiyin Li [1] · Dunhong Peng [1] · Jingwei Cheng [1]

Received: 15 November 2020 / Accepted: 4 January 2021

Abstract

Over the years, a large amount of *temporal data* needs to be shared and exchanged on the Web. Resource Description Framework (RDF) has been widely accepted and has rapidly gained popularity to al RDF) model as mentioned in represent and share data in many application domains (e.g., the Data of Web, Linked Data, and Knowledge Graph). Accordingly, efficient management of *temporal data based on RDF* is of increasing importance. Much work has been devoted to the representation, querying, storage, and management of temporal RDF data. Therefore, to investigate and more importantly serve as helping researchers grasp the main ideas and results of temporal RDF data management, and to highlight an ongoing research on them, in this paper, we *provide a full up-to-date overview of the current state of the art in temporal RDF data management, including representation, querying*, *storage, and other directions*.

In detail, regarding to the *representation* of temporal RDF models, we *first* classify the existing temporal RDF models into two main categories according to their focuses (including the new RDF extension models and the original RDF models based on timestamp). We summarize and compare each model from the perspectives of syntax, semantics, and etc. *Then*, we further summarize the temporal RDF *querying* and *storage* techniques in detail. *Moreover*, to well introduce the main idea of each approach, we use some examples to explain each approach. *In addition*, the directions for future research and some comparisons and analyses are discussed in our whole survey. This survey will help readers understand and catch some key techniques about the issue and also identify some future research directions.

Keywords Resource description framework (RDF) · Temporal data management · Temporal RDF · Survey

Introduction

The management of *temporal information* has been widely investigated in the fields of temporal databases (Radhakrishna et al. 2015), Web applications (Bry and Spranger 2003), and recent Semantic Web (also called the Data of Web) (Wang and Tansel 2017), Linked Data (Rula et al. 2012) and Knowledge Graph (Chekol et al. 2017b). As an example, the Wikidata contains millions of temporal facts associated with validity intervals (i.e., start and end time) covering a variety of domains (Wudage Chekol et al. 2019). In these applications, data cannot be assumed to be static and they change often very frequently. Therefore, many applications have to represent and further process temporal information for effectually modeling, retrieving, and managing information evolving over time (Rula et al. 2012).

In order to represent and manage temporal data, kinds of temporal knowledge representation formalisms including logic-based (first-order logic, modal logic, description logic, and etc. (Al-Dhaheri 2016)) and non-logic based (database models (Radhakrishna et al. 2015), XML (Faisal and Sarwar 2014), and etc.) are developed. Among various of knowledge representation formalisms, a relatively new data representation format called *Resource Description Framework* (RDF) (Manola et al. n.d.) has been widely accepted and has rapidly gained popularity to represent and share data in many domains (e.g., social networking, medicine, particularly in the Data of Web, Linked Data, and Knowledge Graph (Candan et al. 2001)). As the W3C (World Wide Web Consortium) recommendation normative language, RDF can describe not only the data but also the semantics of the data, so that information can be shared and exchanged among applications without loss of semantics. Recently, several survey reports have been made to review the state of the art in RDF data representation and

Communicated by: H. Babaie

✉ Fu Zhang
zhangfu@cse.neu.edu.cn; zhangfu216@126.com

[1] School of Computer Science and Engineering, Northeastern University, Shenyang, China

Published online: 08 March 2021

management (Ma et al. 2016; Özsu 2016; Tzitzikas et al. 2017). Further, some extensions of RDF are proposed in order to deal with various of information (e.g., imprecise information (Mazzieri and Dragoni 2008; Straccia 2009), trust and provenance (Dividino et al. 2009; Hartig 2009; Schenk 2008)).

With the popularity of RDF, RDF is naturally employed to represent and process massive temporal application data. Currently, much work has been done in *temporal RDF data management* involving representation, querying, and storage of temporal RDF data. Lots of temporal RDF *representation* models (tRDFGraph (Gutierrez et al. 2007; 2005), TA-RDF (Rodríguez et al. 2009), tRDF (Pugliese et al. 2008), TS-RDF (Rula et al. 2014), etc.) are proposed. Accordingly, some temporal RDF extension *querying* languages (SPARQL^T (Gao et al. 2016; Zaniolo et al. 2018), τ-SPARQL (Tappolet and Bernstein 2009), TA-SPARQL (Rodríguez et al. 2009), etc.) are developed. Also, the *storage* of temporal RDF data is the infrastructure for temporal RDF data management, and several proposals are devoted to the storage of temporal RDF data (Bereta et al. 2013; Gergatsoulis and Lilis 2005; McBride and Butler 2009), etc. More details can be found in our survey of this paper.

Although up to now a huge number of temporal RDF data management techniques have been proposed in the literature, to the best of our knowledge, *detailed and in-depth reviews on these studies are scarce*. Analyti and Pachoulakis (Analyti and Pachoulakis 2012) provide a survey on the models and query languages for temporally annotated RDF, where several temporally annotated RDF models and query techniques are introduced. Wang and Tansel (Wang and Tansel 2017; 2019) give a survey of temporal aspects of major Semantic Web data models and query languages, focusing on temporal RDF and temporal OWL. They give a taxonomy to classify them according to the forms of explicit reification and implicit reification. But they only focus on the models based on the valid time and also the other temporal models are not considered. Moreover, temporal RDF query languages are briefly discussed, and the issue of storage is still missed. *Compared with* (Analyti and Pachoulakis 2012; Wang and Tansel 2017; 2019), *we make more contributions in our survey*:

- We consider and summarize more approaches about the representation, querying, and storage of temporal RDF models. In particular, the issue of storage is missed.
- More importantly, we *first* give an Overall Comparison of all approaches. *Then*, regarding all the existing approaches, we make more detailed and in-depth comparisons and discussions, and we classify the existing temporal RDF models into two main categories according to their focuses, and then we further summarize and compare each approach from the perspectives of syntax, semantics, and etc. *Further*, we summarize the temporal RDF *querying* and *storage* techniques in detail.
- Moreover, besides of some abstract formal notions, we use examples to explain each approach for illustrating the main idea of each approach. In addition, the directions for future research and some comparisons and analyses are discussed in our whole survey.

The rest of this paper is organized as follows. RDF and temporal data section introduces the basic knowledge. Temporal RDF representation models section makes detailed and in-depth comparisons and discussions about the representation of temporal RDF data. Temporal RDF querying techniques and Temporal RDF storage techniques sections focus on the querying and storage techniques of temporal RDF data. Discussion and future research directions section makes some discussions to identify some suggestions for possible research directions, and Conclusion section concludes the paper.

RDF and temporal data

In this section, we recap some basic knowledge on RDF and temporal data that are the basis for the techniques discussed in the later sections.

RDF

Resource Description Framework (RDF) (Manola et al. n.d.) is a W3C Recommendation for the notation of metadata on the World Wide Web (WWW). Here we recap some basic notions of RDF (Manola et al. n.d.). The basic idea of RDF is a *triple* model, where:

- Anything is described as *resource*, such as abstract concepts, physical things, strings and numbers. The resource identified by an IRI (International Resource Identifier) is called its referent. Moreover, the URLs (Uniform Resource Locators) that people use as Web addresses are one form of IRI. The notion of IRI is a generalization of URI (Uniform Resource Identifier), allowing non-ASCII characters to be used in the IRI character string. In addition, the resource denoted by a literal is called its literal value. *Literals* are basic values that are not IRIs, such as strings, numbers, and dates (note that the datatypes used in RDF is compatible with XML Schema datatypes (W3C XML Schema Definition Language (XSD) 1 2012)).
- Further, RDF allows us to make statements about resources. A *statement* expresses a relationship between two resources *subject* and *object*, and always has the *triple* structure:

  <subject> <predicate> <object>

  The relationship (i.e., *predicate*) from *subject* to *object* is also called in RDF a *property*. The triple is usually

abbreviated as <*s*, *p*, *o*>. A set of triples constitutes an RDF graph.

- In a triple, three types of RDF data may occur: *IRIs*, *literals* and *blank nodes*. The blank nodes are used to denote resources without explicitly naming them with IRIs. In more detail, IRIs may appear in all three positions of a triple; Literals can appear in the object position only; Blank nodes can appear in the subject or object position.

There are several syntaxes to represent RDF data, such as Turtle, JSON-LD (JSON-based RDF syntax), RDFa (for HTML and XML embedding), and RDF/XML (XML syntax for RDF) (Manola et al. n.d.).

As mentioned in (Manola et al. n.d.), RDF does not make any assumptions about what resource IRIs stand for. Therefore, to further define semantic characteristics of RDF data, *RDF Schema* (RDFS) (Brickley and Guha 2014), as a semantic extension of RDF, provides a data-modelling vocabulary for RDF data. For instance, we can declare that an IRI *http://www.example.org/friendOf* denotes a property (i.e., *predicate*), and its *domain* (i.e., *subjects*) and *range* (i.e., *objects*) must be resources of a class *http://www.example.org/Person*.

Moreover, the *semantics* of RDF and RDFS are introduced in (Hayes and Patel-Schneider 2014) to impose significant formal constraints on RDF and RDFS. Several main semantics constraints including *interpretation*, *entailment*, and *reification* are considered in most of work. An *interpretation* is a mapping from IRIs and literals into a set, together with some constraints upon the set and the mapping. *Entailment* makes it possible to make logical inferences. For example, a graph G_1 entails a graph G_2 when every interpretation which satisfies G_1 also satisfies G_2. If two graphs G_1 and G_2 each entail the other then they are logically equivalent. *Reification* defines statements about statements within RDF and allows an RDF graph to act as metadata describing other RDF triples. Note that here we will not classify the kinds of interpretation and entailment, such as simple interpretation and simple entailment.

In addition, SPARQL (Harris and Seaborne 2013) is a standard query language for querying RDF data. It is based on graph pattern matching. In brief, triple patterns are the basic graph patterns, and a *variable* (e.g.,? *title* in the following example) may appear in the subject, predicate, or object position in a triple pattern. Further, more complex graph patterns can be combined from triple patterns. In the example, the SELECT clause identifies the variables to appear in the query results; the WHERE clause provides the basic graph pattern to match against the data graph; if the OPTIONAL part does not match, it creates no bindings but does not eliminate the solution; the FILTER restricts solutions to those for which the filter expression evaluates to TRUE. The results of SPARQL queries can be result sets or RDF graphs.

```
PREFIX  dc:   <http://purl.org/dc/elements/1.1/>
PREFIX  ns:   <http://example.org/ns#>
SELECT  ?title ?price
WHERE   { ?x dc:title ?title .
          OPTIONAL { ?x ns:price ?price . FILTER (?price < 30) }
        }
```

Please refer to (Harris and Seaborne 2013; Manola et al. n.d.) for more detailed introduction about RDF, RDFS, and SPARQL.

Temporal data types

As mentioned in most of work, the basic *temporal data types* include *time point* and *time interval*. *Time point* refers to a precise point in time at which the event occurred or was occurring. *Time interval* is a collection of pairs of time points where the former is the start time point and the latter is the end one.

Moreover, time is multi-dimensional, including valid time, transaction time, publication time, efficacy time, and etc. Being similar to the field of relational databases, two main types of *temporal dimensions* (i.e., *valid time* and *transaction time*) are normally considered in most of work (Bereta et al. 2013; Böhlen et al. 2017; Grandi 2009; Motik 2012), and etc.

- Valid time is the time when data are valid in the modeled world or a fact is true in the application domain (e.g., the time 2000–2012 when John is a professor).
- Transaction time is the time when data are actually stored in the database or information system (the fact "John was professor from 2000-2012" was stored in the database on May 6, 2003, and has been deleted on June 1, 2012).

In more detail, valid time captures the time-varying states of the modeled world, and basically each fact has a valid time even though it might not be recorded in the system (i.e., the valid time is independent of the recording of the fact in a system (Böhlen et al. 2017). Validity time can affect most of aspects of knowledge representation and reasoning in many advanced application scenarios. Transaction time captures the time-varying states of the database or system, and is important in the context of data management, but is not part of an application domain's description (Motik 2012).

In addition, much work also considers some more complex temporal problems. Allen in (Allen 1983) presents the well-known Allan's interval algebra relations, which defines possible 13 relations between time intervals and provides a composition table that can be used as a basis for reasoning about temporal descriptions of events.

Temporal RDF representation models

As pointed out in (Gutierrez et al. 2005; McBride and Butler 2009), time is present in almost any Web and e-

business applications. The representation of time is one of the key primitives needed in management for Web and semi-structured data. Therefore, there is a clear need of applying temporal to RDF to allow metadata navigation across time.

In this section, *we summarize and classify the existing temporal RDF representation models*, these models extend the RDF triple in different forms to represent temporal information in RDF. In Category and overall comparison section, we first give the classification principle and provide an Overall Comparison of all approaches. Further, according to the classification principle, in Temporal RDF models based on new RDF extension syntaxes and Temporal RDF models based on the original form of RDF triple by adding timestamp sections we summarize and compare each category of approaches in more detail.

Category and overall comparison

By investigating the existing approaches, we roughly classify the existing temporal RDF models into two main categories (C_1-C_2) according to their focuses as shown in Table 1:

- C_1: the models based on the new RDF extension syntaxes. That is, this kind of models *define some new RDF syntaxes* by extending the RDF triple to other forms (e.g., RDF quad-tuple form or other *n*-tuple forms) for modeling temporal information.
- C_2: the models based on the RDF triple syntax by adding timestamp information. That is, the basic syntax forms of the models *are similar to the RDF triple*, but they further re-define the abstract semantics of the triple.

Based on the categories above, Temporal RDF models based on new RDF extension syntaxes and Temporal RDF models based on the original form of RDF triple by adding timestamp sections will further introduce each category in detail. The basic ideas in Temporal RDF models based on new RDF extension syntaxes and Temporal RDF models based on the original form of RDF triple by adding timestamp sections are as follows:

(i) For *each approach* in C_1-C_2, *we summarize its main contribution*.
(ii) At the same time, *we make in-depth comparative studies and discussions* among these approaches from several main aspects, including *Contribution*, *Representation form*, *Storage*, *Querying*, and *Implementation*. It should be noted that more details about the *querying* and *storage* for each approach can be found in the subsequent Temporal RDF querying techniques and Temporal RDF storage techniques sections.
(iii) Besides of some abstract formal notions, we use *examples* to explain each approach for illustrating the main idea of each approach.

Note that, in our survey each approach is named according to the abbreviation of the approach followed a bracket including the first authors' name and the year of publication.

Before introducing each category, we first provide an *Overall Comparison* of all approaches in Table 2. Here we choose some common techniques mentioned in these temporal RDF representation models and make a direct comparison.

Temporal RDF models based on new RDF extension syntaxes

This kind of models define some new RDF syntaxes by extending the RDF triple to represent temporal information. Table 3 first summarizes and compares these temporal RDF models from several aspects, including *Contribution*, *Representation form*, *Storage*, *Querying*, and *Implementation*. Also, the *examples* are provided in the table. Note that, more details about the *querying* and *storage* for each approach can be found in the subsequent Temporal RDF querying techniques and Temporal RDF storage techniques sections. After the table, we further introduce each approach in detail.

Table 1 Categories of the existing temporal RDF models

Categories	*Basic Representation Forms*	*Focuses & Objectives*
C_1 (Section 3.2)	RDF quad-tuple form or other *n*-tuple forms ($n \neq 3$)	This kind of models define *some new RDF syntaxes* by extending the RDF triple to other forms (e.g., RDF quad-tuple form or other *n*-tuple forms) for modeling temporal information
C_2 (Section 3.3)	RDF triple	This kind of models are based on the RDF triple syntax by adding timestamp information. The basic syntax forms of the models *are similar to the RDF triple*, but they further re-define the abstract semantics of the triple

Table 2 An Overall Comparison of the existing temporal RDF representation models

	Version /Transaction time	Valid time	User-defined time[a]	RDF streams with timestamps[b]	Anonymous timestamps[c]	Allen's 13 interval relations (Allen 1983)	Entailment	Modification operations[d]	Other meta knowledge[e]
VersioningRDF (Ognyanov and Kiryakov 2002)	✓							✓	
tRDFGraph (Gutierrez et al. 2007; 2005; Hurtado and Vaisman 2006)	✓	✓			✓	✓	✓		
TKB (Bykau et al. 2012; Rizzolo et al. 2009)		✓						✓[f]	
Multi-tempRDF (Grandi 2011; 2009)	✓	✓						✓	
NG-tempRDF (Tappolet and Bernstein 2009)	✓	✓				✓			
RDF+ (Dividino et al. 2009; Schueler et al. 2008)		✓							✓
T-YAGO (Wang et al. 2010)		✓							
RDFstreams (Barbieri et al. 2009; 2010a; 2010b)				✓					
UncertainTemRDF (Dylla et al. 2011) (2013)		✓				✓			✓
AnnotatedRDF (Zimmermann et al. 2012)		✓			✓	✓	✓		✓
ValidityTimeRDF (Motik 2012)		✓				✓	✓		
stRDF (Bereta et al. 2013)		✓	✓			✓			
WeightedTemporalRDF (Huber 2014)		✓				✓	✓		✓
UTKGs (Chekol et al. 2017a)		✓				✓	✓		✓
TRDFS (Yang and Yan 2018)	✓	✓							
TS-RDF (Rula et al. 2014; 2019)		✓				✓			
Ontop-temporalRDF (Kalayci et al. 2019)		✓				✓			
TKG (Huang et al. 2020)		✓				✓			
MRDF (Gergatsoulis and Lilis 2005)	✓	✓						✓	✓
TempCRM (Liao and Tu 2007)		✓							
tRDF (Pugliese et al. 2008)		✓					✓		
TA-RDF (Rodríguez et al. 2009)		✓		✓			✓		
TempRDF (McBride and Butler 2009)		✓							
aRDF (Udrea et al. 2010)		✓					✓	✓	✓
CNTRO (Tao et al. 2010)		✓				✓			✓
TempRDFLod (Rula et al. 2012)		✓							✓
SingletonPropertyRDF (Nguyen et al. 2014)		✓							✓
LORI (Robatjazi et al. 2015)		✓							✓
RDFt (Zhang et al. 2019)	✓	✓					✓		

[a] *user-defined time*: which has no special semantics (e.g., January 1st, 1963 when John has his birthday)

[b] *RDF streams*: are defined as ordered sequences of pairs, each pair being made of an RDF triple and a timestamp τ, and the timestamps are monotonically non-decreasing in the stream ($\tau_i \leq \tau_{i+1}$)

[c] *anonymous timestamps*: a graph may contain triples of the form $<s, p, o>$ [t], where t is an anonymous timestamp, stating that the triple $<s, p, o>$ is valid in some unknown time

[d] *modification operations:* means addition/insert, removal/delete, or update

[e] *Other meta knowledge:* Source or provenance denotes that "where is this RDF triple from?"; (un)certainty or probability or fuzzy denotes that the triple is associated with a degree θ, where $0 < \theta \leq 1$; or user-defined dimensions or other temporal meta-information

[f] specially, the operations denote the special four evolution terms: join, split, merge, and detach

Table 3 The existing temporal RDF models based on new RDF extension syntaxes

	Contribution	*Representation form*	*Example*	*Storage*	*Query*	*Implementation*
VersioningRDF (Ognyanov and Kiryakov 2002)	*Tracking the changes* in RDF(S) repositories. They indicate an RDF statement can only be added and removed. And each addition or removal turns the repository into a new state. They used versions as the labeled states of the repository.	*Syntax*: <*s*, *p*: nn_{add}-nn_{remove}, *o*>, and the history of the repository can be represented via events in format: *UID*:*nn* {*add* | *remove*} <*s*, *p*, *o*> *Comment*: nn_{add}, $nn_{remove} \in nn$, which is an integer variable that increases its value each time when the repository is added or removed. Each state of the repository is identified by identifier *UID*:*nn*.	A statement <*A*, r_1, *B*>is added into a repository KB_1 and then is removed. The history of the repository is represented via events in format: *UID*:1 *add*<*A*, r_1, *B*> *UID*:2 *remove*<*A*, r_1, *B*> In this case, the repository KB_1 can be represented as: <*A* r_1:1–2 *B*>.	No	No	No
tRDFGraph (Gutierrez et al. 2007; 2005; Hurtado and Vaisman 2006)	Proposing a temporal extension of RDF, based on the idea of *assigning timestamps to RDF triples*, which allows anonymous unknown timestamps in temporal RDF graphs. They also propose rules to *translate* temporal RDF into RDF graph.	*Syntax*: <*s*, *p*, *o*>[*t*], $t \in T$ *Comment*: *T* is a discrete and linearly ordered time domain. An ordered pair $[t_s, t_e]$ of time points, with $t_s \leq t_e$, denotes the closed interval from t_s to t_e. Moreover, a set of temporal RDF triples with consecutive time points $\{<s, p, o>[t] \mid t_s \leq t \leq t_e\}$ are encoded using the interval-based expression as: $<s, p, o>[t_s, t_e]$.	<*John*, *rdf:type*, *Undergraduate*> [0, 10], which means that the *John* (subject) was *Undergraduate* (object) throughout the entire time frame 0–10.	No	A sketch of temporal query language	No
TKB (Bykau et al. 2012; Rizzolo et al. 2009)	Presenting a framework by extending an RDF-like model with temporal features and evolution operators to manage the *evolution of a concept*. The model is to answer historical queries.	*Syntax*: <*U*, *L*, *T*, τ>, *Comment*: $U \subseteq \mathbf{U}$ is an infinite set of *resources*, $L \subseteq \mathbf{L}$ is a set of *literals*, $T \subseteq \mathbf{U} \times \mathbf{U} \times \{\mathbf{U} \cup \mathbf{L}\}$, and τ is function that maps each resource $r \in U$ into a temporal interval [*r.start*, *r.end*], which can represent the lifespan of the resource.	<*Kingdom-of-Netherlands*, *detach*, *Belgiumi*>with τ(*Belgium*).*start*=1831, which denotes that a new concept *Belgium* is formed at a time 1831 with at least one part from *Kingdom-of-Netherlands*.	No	A historical query language based on nested regular expressions	Yes (the system, called TrenDS, is implemented in Java)
Multi-tempRDF (Grandi 2011; Grandi 2009)	Presenting a Multi-temporal RDF Database Model in order to manage *temporal versions* of an ontology.	*Syntax*: (*s*, *p*, *o* | *T*) *Comment:* where $T \subseteq \mathcal{T}$ is a timestamp assigning a temporal pertinence. $\mathcal{T}$ is a *N*-dimensional time domain that $\mathcal{T} = \mathcal{T}_1 \times \ldots \times \mathcal{T}_N$ where $\mathcal{T}_i = [0, UC)_i$ is the *i*-th time domain, and *UC* means "Until Changed".	(*e*, *Dept*, "*Toys*" |? *t*) which denotes that an employee *e* works in the toys department at *t*.	No	No	No
NG-tempRDF (Tappolet and Bernstein 2009)	Presenting an approach to express the temporal RDF model by using *Named Graph*.	*Syntax*: *ng*=(*n*, *g*) [*s*, *e*] *Comment:* a named graph is a pair *ng*=(*n*, *g*), where *n* is URI	:*person rdf:type*: *Person* [?*s*,? *e*] represents all: *Persons* that were valid (i.e. alive) in [?*s*,? *e*].	No	τ-SPARQL query language	Yes (using the ng4j named graph API)

Table 3 (continued)

	Contribution	*Representation form*	*Example*	*Storage*	*Query*	*Implementation*
		reference and g is an RDF graph, s and e express time interval with s, $e \in \tau \mid s \leq e$, τ is 1-dimensional discrete value.				
RDF+ (Schueler et al. 2008) (Dividino et al. 2009)	Presenting an embedded model for managing many *dimensions of meta knowledge*, like *timestamp*, source, and (un)certainty.	*Syntax*: (K, M) *Comment: K* is a set of literal statements and M is a set of meta knowledge statements.	K=(G_1, *James*, *researchTopic*, *SemanticWeb*, θ_1) and M=(θ_1, *timestamp*, "5/5/2007") represent that the information of a person at a timepoint.	No	A small extension of standard SPARQL	Yes (an initial prototype)
RDFstreams (Barbieri et al. 2010a) (2010b) (2009)	Presenting an RDF streams model to support *streams* in RDF format.	*Syntax*: (<$subj_i$, $pred_i$, obj_i>, τ_i) (<$subj_{i+1}$, $pred_{i+1}$, obj_{i+1}>, τ_{i+1}) *Comment*: the RDF streams are defined as ordered sequence of pairs, each pair being made of an RDF triple and with a timestamp τ.	(<*c*:*Usr1 sd*:*accesses c*:*movie1*>, *t400*) (<*c*:*Usr1 sd*:*likes c*:*movie1*>, *t401*) which denote that the users may access a movie or express their opinion.	No	C-SPARQL for continuous queries over streams of RDF data	No
T-YAGO (Wang et al. 2010)	Presenting T-YAGO (an extension version of YAGO with temporal aspects) by introducing the concept of *temporal facts*.	*Syntax*: #*id*: *s p o* #*id on* time point *or* … *Comment*: where #*id* is a *fact identifier* to the *primary fact* (*s p o*), and then the associated facts are represented as the relation between the identifier and the remaining arguments.	#1: *DavidBeckham hasWonPrize UEFAClubPlayerOfTheYear* #1 *on* 1999 which denotes "*David Beckham* has won the *UEFA Club Player* of the Year in 1999".	No	A simple time-aware query language	Yes (a simple demo)
UncertainTemRDF (Dylla et al. 2011) (2013)	Presenting a way to resolve *temporal conflicts* in *inconsistent* RDF knowledge bases.	*Syntax*: $p(s, o, i)_d$ *Comment*: where $p(s, o)$ is an RDF triple, i is a (half-open) temporal interval of the form $[t_b, t_e)$, and $d \in [0, 1]$ is a confidence degree that $p(s, o)$ is true during interval i.	*fbornBL*: = *bornIn*(*David Beckham*, *Leytonstone*, [1975, 1976))$_{0.9}$, which denotes *David Beckham* was born in *Leytonstone* in *1975* with weight 0.9.	No	queries via first-order logical predicates	Yes (Experiments are tested on T-YAGO (Wang et al. 2010))
Annotated RDF (Zimmermann et al. 2012)	Proposing an RDF *annotation* framework to support statements annotated with *fuzzy*, *temporal*, and *provenance*.	*Syntax*: τ: λ *Comment*: where τ is a triple and λ is an annotation value in a non-empty set L. L can be defined as L=$\{t \mid t$ is a finite set of disjoint temporal intervals$\} \cup \{\varnothing, [-\infty, +\infty]\}$.	(*niklasZennstrom*, *ceoOf*, *skype*): [2003, 2007] means that "Niklas was CEO of Skype during the period 2003 to 2007".	No	AnQL query language of supporting features of SPARQL 1.1.	Yes (based on the constraint logic programming techniques)
Validity TimeRDF (Motik 2012)	Proposing a *logic-based* approach to represent and query *validity time* in RDF.	*Syntax*: <s, p, o>[t] or<s, p, o>[t_1, t_2] *Comment*: where t is a time instant, t_1 is a time instant or$-\infty$, and t_2 is a time instant or$+\infty$.	<:*LHR*,: *flightTo*,: *MUC*>[50, 120] states that there is a flight from *LHR* to *MUC*.	No	An extension of SPARQL	Yes
stRDF (Bereta et al. 2013)	Proposing a formal extension of RDF for the representation and	*Syntax:* (s, p, o, t)	*corine*:*Area4 corine*:*hasLandCover*	Storing triples in the form of an		Yes (support in Strabon)

Table 3 (continued)

	Contribution	*Representation form*	*Example*	*Storage*	*Query*	*Implementation*
	querying of *linked data that changes over time*.	*Comment:* a temporal triple is a quad (s, p, o, t), where t is a time instant or a time period called the valid time of a triple.	*corine:coniferousForest* "[2000,2006)"^^*strdf:period* represents the land cover of an area for the time periods.	N-Quads document	A query language stSPARQL using linear constraints	
WeightedTemporalRDF (Huber 2014)	Presenting an approach for carrying out reasoning and resolving *inconsistencies* in the *temporal probabilistic* knowledge bases.	*Syntax*: *conf* (s, p, o) [t] *Comment*: where *conf* denotes the confidence value that the statement (s, p, o) is true, t is a time point or interval.	0.9 (*John*, *attended*, *University_1*) [2002, 2006] denotes that the confidence value of John attended the *University_1* from 2002 to 2006 is 0.9.	No	No	Yes (evaluated on the temporal facts extracted from DBPedia dataset)
UTKGs (Chekol et al. 2017a)	Proposing a formal characterization of *uncertain temporal* knowledge graphs (UTKGs) to represent uncertainties and reason on temporal knowledge graphs.	*Syntax*: a UTKG is a tuple (D, U), D is a deterministic temporal KG and U is an uncertain temporal KG. *Comment*: where f=(s, p, o, [t_1, t_2]) is a deterministic temporal triple in D, (f, w) is an uncertain temporal triple in U, and w is a real-valued weight assigned to the temporal triple.	(*ClaudioRanieri*, *coach*, *Chelsea*, [2000, 2004], 0.9), represents that a weight 0.9 is assigned to the temporal triple (*ClaudioRanieri*, *coach*, *Chelsea*, [2000, 2004]).	No	No	Yes
TRDFS (Yang and Yan 2018)	Proposing a method to *transform* temporal XML model into temporal RDF model.	*Syntax*: (s, p, o): [v] [t] *Comment*: where (s, p, o) is an RDF triple, [v] is the valid time of the triple, and [t] is the transaction time of the triple.	(:*FoxSports*,: *hasWebpage*,: *www.foxsports.com*): [3, *Now*] [3, *UC*] denotes valid and transaction time of the triple. *UC* means "Until Changed".	No	No	Yes (a prototype transformation tool TX-TR)
TS-RDF (Rula et al. 2014) (2019)	Presenting an approach for extracting time information and further detecting the *temporal scope of the triple* begins and ends.	*Syntax*: <f, [t_i, t_j]> *Comment*: where f is a fact (i.e., a relation acknowledged to hold between subject and object of a triple) and [t_i, t_j] is a time interval.	<(*Balotelli*, *team*, *Inter Milan*), [2007, 2010]>expresses the fact that occurrs from 2007 to 2010.	No	No	Yes (evaluated on DBpedia and Freebase)
Ontop-temporalRDF (Kalayci et al. 2019)	Proposing a framework for temporal ontology-based data access (OBDA), which can extract information about *temporal events* in RDF format.	*Syntax*: p (s, o)@ $\langle t_{\text{begin}}, t_{\text{end}} \rangle$ *Comment*: the triple can be further modelled by a named graph and a distinguished graph including many triples.	mainFlame(mf01, 2.3)@[12:32:30, 12:32:37) states that the main flame sensor mf01 was registering the value 2.3 in the interval [12:32:30, 12:32:37).	No	Use the τ-SPARQL query language (Tappolet and Bernstein 2009)	Yes (an architecture of extending the OBDA platform Ontop)
TKG (Huang et al. 2020)	Proposing a *five-tuple* to formulate a temporal knowledge graph.	*Syntax*: G=<V, L_V, E, L_E, T_E> *Comment*: V is a set of subject and object vertices, L_V is a set of vertex labels, E is a set of directed edges, L_E is a set of edge labels, and T_E is a set of time frames on edges.	<Bush, bePresident, United States> was a fact between 2001 and 2008 can be expressed as the basic form <Bush, bePresident, United States, [2001, 2008]>, which can be further formulated as a five tuple.	A storage model from logical view and physical view	A cluster query method on temporal knowledge graph	Yes (evaluated on the datasets Soccer player, Twitter, and DBLP)

VersioningRDF (Ognyanov and Kiryakov 2002)

Ognyanov & Kiryakov (2002) present a model to track the changes in RDF repositories. The model is based on the assumption that an RDF statement is the smallest manageable piece of knowledge in an RDF repository. They argue that an RDF statement can only be added and removed. Each addition or removal turns the repository into a new state. The history of changes in the repository could be defined as sequence of states, as well, as a sequence of updates.

In more detail, formally, for each RDF repository *KB*, there is an update counter *nn* – an integer variable that increases its value each time when the repository *KB* is updated (i.e., when an RDF statement is added and removed). For each statement in the repository *KB* an update identifier *UID*:*nn* is followed to indicate the statement was added and removed. The history of the repository *KB* is presented via events in format: *UID*:*nn* {*add* | *remove*} <*s*, *p*, *o*>. The repository *KB* is represented as a graph where the lifetime of each statement is given with nn_{add}-nn_{remove} after the predicate *p*, where nn_{add}, $nn_{remove} \in nn$.

For example, a statement <*A*, r_1, *B*> is added into a repository KB_1 and then is removed. The history of the repository is represented via events in format:

UID:1 *add* <*A*, r_1, *B*>
UID:2 *remove* <*A*, r_1, *B*>

In this case, the repository KB_1 can be represented as <*A* r_1:1–2 *B*>. A more detailed example and its graphical representation in (Ognyanov and Kiryakov 2002) can be found in Fig. 1.

tRDFGraph (Gutierrez et al. 2007) (2005) (Hurtado and Vaisman 2006)

Gutierrez et al., (2007; 2005; Hurtado and Vaisman 2006) point out there are two mechanisms for adding the time dimension to RDF graphs: *labeling* and *versioning* (following the timestamp and snapshot models, respectively). The former consists of labeling the elements subject to changes (i.e., triples). The latter is based on maintaining a snapshot of each state of the graph. For instance, each time when a triple changes, a new version of the RDF graph is created, and the past states are stored somewhere. The idea is similar to the cases in temporal databases, and there are at least two temporal dimensions to consider when dealing with temporal databases: valid and transaction times. Valid time is the time when data is valid in the modeled world; transaction time is the time when data is actually stored in the database. The *versioning* approach captures *transaction* time, while *labeling* is mostly used when representing *valid* time. They believe that for RDF data, labeling is better than versioning because: *i*) it preserves the spirit of the distributed and extensible nature of RDF, and *ii*) in scenarios where changes are frequent and only affecting a few elements of the document, creating a new physical version of the graph each time an update occurs may lead to large overheads when processing temporal queries that span multiple versions.

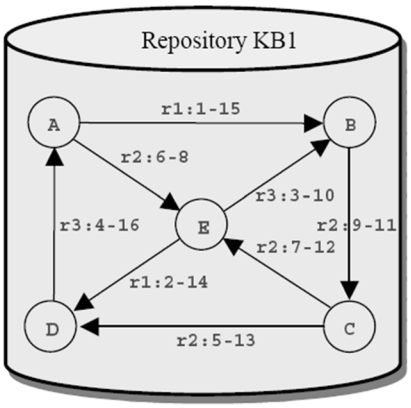
On this basis, they propose a temporal RDF graph model (tRDFGraph, for short). In a tRDFGraph, each RDF triple is annotated with a temporal element to represent the time when this triple is valid. They consider time as a discrete, linearly ordered domain, as usual in virtually all temporal database applications. An ordered pair [t_s, t_e] of time points, with $t_s \le t_e$, denotes the closed interval from t_s to t_e. Formally, given a point-based temporal domain *T*, a tRDFGraph consists of a set of temporal RDF triples where each temporal RDF triple <*s*, *p*, *o*> [*t*] is an RDF triple <*s*, *p*, *o*> annotated with a temporal element $t \in T$. A set of temporal RDF triples with consecutive time points {<*s*, *p*, *o*> [*t*] | $t_s \le t \le t_e$} are encoded using the interval-based expression as: <*s*, *p*, *o*> [t_s, t_e].

For example, the temporal RDF triple <*John*, *rdf:type*, *Undergraduate*> [0, 10] means that the *John* (subject) was *Undergraduate* (object) throughout the entire time frame 0–10.

Note that, the model tRDFGraph allows *anonymous timestamps*. That is, a graph may contain triples of the form <*s*, *p*, *o*> [*t*], where *t* is an anonymous timestamp, stating that the triple <*s*, *p*, *o*> is valid in some unknown time. For

Fig. 1 A temporal RDF model based on the versioning

```
History:
UID:1 add <A, r1, B>
UID:2 add <E, r1, D>
UID:3 add <E, r3, B>
UID:4 add <D, r3, A>
UID:5 add <C, r2, D>
UID:6 add <A, r2, E>
UID:7 add <C, r2, E>
UID:8 remove <A, r2, E>
UID:9 add <B, r2, C>
UID:10 remove <E, r3, B>
UID:11 remove <B, r2, C>
UID:12 remove <C, r2, E>
UID:13 remove <C, r2, D>
UID:14 remove <E, r1, D>
UID:15 remove <A, r1, B>
UID:16 remove <D, r3, A>
```



example, assuming that it is not certain about the time when *John* was changed from *Undergraduate* to *Master*. In this case, the triple <*John*, *rdf:type*, *Undergraduate*> is now labeled with an anonymous interval i_1 (instead of [0, 10]). Analogously, <*John*, *rdf:type*, *Master*> is labeled with an anonymous interval i_2 (instead of (Bry and Spranger 2003; Carroll et al. 2004)). Now, one can use the basic Allen's 13 interval relations (Allen 1983) to place constraints over the anonymous intervals. Adding constraints allows reasoning about temporal RDF graphs in order to infer useful knowledge. For example, one can use the constraint i_1 *meets* i_2 (*meets* is one kind of Allen's 13 interval relations (Allen 1983)) to state that *John* started *Master* immediately after *Undergraduate*.

Moreover, since the triples in tRDFGraph do not belong to the RDF standard syntax. They further specify rules to *translate temporal RDF as RDF graph* by making use of reification plus some simple additional vocabulary (e.g., temporal, instant, interval, initial and final) and rules. For example, the temporal RDF graph at the left-hand side of the Fig .2 can be translated as the RDF graph at the right-hand side of the Fig .2.

In addition, they give the semantics for the notion of entailment for temporal graphs based on the corresponding notion for RDF graphs, and show that the former does not yield extra asymptotic complexity with respect to nontemporal RDF graphs.

TKB (Bykau et al. 2012; Rizzolo et al. 2009)

Bykau et al., (2012); Rizzolo et al., (2009) present a framework by extending an RDF-like model with temporal features and evolution operators to model, query and manage the evolution of a concept. The model can be applied for answering historical queries such as "How has a concept evolved over a time period".

The RDF temporal model is an extension of the temporal model (Gutierrez et al. 2005) with additional constructs and consistency conditions to model merges, splits, and other forms of concept evolution. Formally, an RDF temporal knowledge base Σ_T is a tuple <U, L, T, τ>, where $U \subseteq \mathbf{U}$ is an infinite set of *resources*, $L \subseteq \mathbf{L}$ is a set of *literals*, $T \subseteq \mathbf{U} \times \mathbf{U} \times \{\mathbf{U} \cup \mathbf{L}\}$, and τ is function that maps each resource $r \in U$ into a temporal interval [*r.start*, *r.end*], which can represent the lifespan of the resource.

For example, an evolution term *detach*(Kingdom-of-Netherlands, Belgium, 1831), which denotes that a new concept Belgium is formed at a time 1831 with at least one part from Kingdom-of-Netherlands, is represented as an RDF temporal triple: <Kingdom-of-Netherlands, detach, Belgium> with τ(Belgium).start = 1831. Similarly, the other several terms *join*, *split*, and *merge* are defined for modeling kinds of evolution concepts.

Moreover, they define a query language that exploits the extensions and supports historical queries. And a system called TrenDS is implemented in Java, and the approach is applied in two real world scenarios (i.e., the evolution of biotechnology and the history of Germany), and several query examples are answered based on the query language.

Multi-tempRDF (Grandi 2011; Grandi 2009)

Grandi (2011; 2009) present a Multi-temporal RDF Database Model in order to manage temporal versions of an ontology. The model contains a N-dimensional time domain $\mathcal{T} = \mathcal{T}_1 \times \ldots \times \mathcal{T}_N$, where $\mathcal{T}_i = [0, UC)_i$ is the i-th time domain. Right-unlimited time intervals are expressed as [t, UC), where UC means "*Until Changed*". Moreover, the transaction time of any modification is always, by definition, [*NOW*, *UC*), where *NOW* is the transaction time when the modification is executed.

On this basis, they further define a multi-temporal RDF triple as (s, p, o | T), where (s, p, o) is a standard RDF triple and $T \subseteq \mathcal{T}$ is a timestamp assigning a *temporal pertinence* to the RDF triple. The temporal pertinence of a triple is a subset of the multidimensional time domain.

For example, a triple with temporal information about an employee has been working in the toys department can be expressed as (:*e*: *Dept* "*Toys*" |? *t*), where the employee was denoted by the blank node _:*e*, the variable? $t \subseteq \mathcal{T}$ binds to the timestamp of a temporal triple.

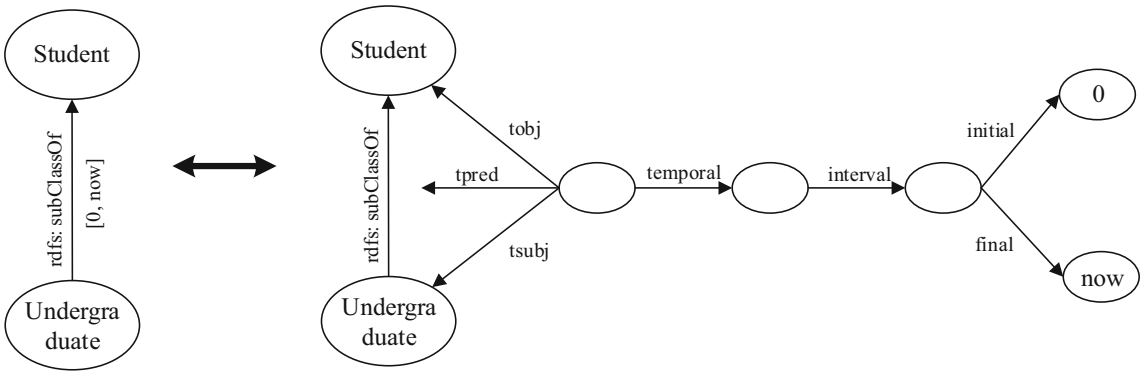


Fig. 2 An example of translating temporal RDF as RDF graph

They also define several modification operations (including Insertion and Deletion) for the maintenance of a set of multi-temporal RDF triples.

NG-tempRDF (Tappolet and Bernstein 2009)

Tappolet & Bernstein (2009) present an approach to express the temporal RDF model by using named graph (Carroll et al. 2005). They define two different representations: one is for internal use only and facilitates the addressing and indexing of temporal entities, and anther one provides the basis for a machine processable exchange format considering the semantics of time.

This model expresses time as a 1-dimensional discrete value, and each valid time point is defined as $\tau \in \mathbb{N}_0$. Further, they use two time points s (start) and e (end) to express time interval where s, $e \in \tau \mid s \leq e$. They also define three special time interval forms $[0, e]$, $[s,\infty]$ and $[0, \infty]$, which 0 stands for the minimum time value, and ∞ stands for maximum time value respectively.

Formally, let U be the set of all URI references, B an infinite set of RDF blank nodes, and L the set of all legal RDF literals (U, B and L are pairwise disjoint). Further, let $V = U \cup B \cup L$ be the set of *nodes*, $T = V \times U \times V$ be the set of RDF triples. The set of RDF graphs G is the power set of T. A *Named Graph* is a pair $ng = (n, g)$ with n in U and g in G. Further, in order to express the temporal RDF model by named graph, each named graph has a valid time interval, e.g., $ng = (n, g)$ $[s, e]$ or $[s, e]$ (n, g). Note that, for blank nodes B, they assign a URI to any blank node in order to avoid make a confusion in different named graphs. For example,: *person rdf:type*: *Person* [?*s*,? *e*] represents all: *Persons* that were valid (i.e. alive) in [?*s*,? *e*].

Moreover, they introduce an extended syntax for SPARQL (called τ-SPARQL) to query time point and interval information. In addition, they evaluate the approach on two different data sources based on in-memory graphs using the ng4j named graph API.

RDF+ (Dividino et al. 2009; Schueler et al. 2008)

Dividino et al., (2009); Schueler et al., (2008) present an embedded model call RDF+ for managing several dimensions of meta knowledge, like timestamp, source, and (un)certainty. Formally, an RDF+ model is a pair (K, M) referring to a set of literal statements K and a set of meta knowledge statements M. A literal statement in K is a quintuple (g, s, p, o, θ), where g is a graph name, s, p, o are elements of a standard RDF triple, and θ is a statement identifier. A meta knowledge statement in M is a tuple (θ, π, ω), where θ is the literal statement identifier as mentioned above, π is the meta knowledge property, and ω is the value ranges for the meta knowledge property.

For example, the following statements in an RDF+ model represent that the extracted knowledge about a person from different sources at different timepoints.

(G_1, JamesHendler, researchTopic, SemanticWeb, θ_1)
(G_1, JamesHendler, affiliatedWith, RensselaerPI, θ_2)
(G_2, JamesHendler, researchTopic, Robotics, θ_3)
(G_2, JamesHendler, affiliatedWith, UnivMaryland, θ_4)
(θ_1, mk:source, {<www.rpi.edu/report.doc>})
(θ_1, mk:timestamp, "5/5/2007")
(θ_2, mk:source, {<www.cs.umd.edu/survey.pdf>})

Further, they provide a generic semantic framework for meta knowledge in RDF, which includes a standard interpretation Is and a Π-interpretations $I\pi$. They also propose the bidirectional mapping approach between RDF and RDF+. Moreover, they present a small extension of standard SPARQL syntax and define how the extension of SPARQL can be applied to an RDF+ model. Finally, an initial prototype is implemented and several querying examples using artificial data are provided.

RDFstreams (Barbieri et al. 2009; 2010a; 2010b)

Barbieri et al., (2009); Barbieri et al., 2010a, 2010b) propose an RDFstream model to support streams in RDF format to guarantee interoperability and open up important applications, in which reasoners can deal with evolving knowledge over time.

Similar to RDF graph, RDF stream are depended on IRIs to identify the source of stream data, and are made up some of sequences of triples continuously with timestamps. The RDF streams are defined as ordered sequence of pairs, each pair being made of an RDF triple and with a timestamp τ. Such as:

($<subj_i, pred_i, obj_i>, \tau_i$)
($<subj_{i+1}, pred_{i+1}, obj_{i+1}>, \tau_{i+1}$)

Timestamp are monotonically non-decreasing in the streams ($\tau_i \leq \tau_{i+1}$). However, timestamps are not strictly increasing because they are not required to be unique. Any (unbounded, though finite) number of consecutive triples can have the same timestamp, meaning that they occur at the same time.

For example, in a social data analysis scenario, users may access a movie or express their opinion about it, such example of possible triples in a stream of interactions and opinions is given below.

(<c:Usr1 sd:accesses c:movie1>, t400)
(<c:Usr1 sd:accesses c:movie1>, t401)
(<c:Usr1 sd:likes c:movie2>, t402)

Moreover, they develop a continuous query language C-SPARQL for querying the RDF data streams model.

T-YAGO (Wang et al. 2010)

Wang et al., (2010) introduce Timely YAGO (T-YAGO), which extends the knowledge base YAGO (Fabian et al. 2007) with temporal aspects. They introduce the concept of *temporal facts*. A temporal fact is a relation with an associated *validity time*. The fact may be valid at a time point or within a time interval. In YAGO, temporal facts cannot be directly represented. Facts are limited to binary relations while temporal facts have more than two arguments. To support temporal facts in a binary relation model, they decompose the *n*-ary fact into a primary fact and several associated facts. They assign a *fact identifier* to the *primary fact*, and then the associated facts are represented as the relation between the identifier and the remaining arguments. For temporal facts valid at a *time point*, they use the relation *on* to describe the validity time. For temporal facts that are valid during a *time period*, they use two relations to represent the interval: the relation *since* for the begin time point, and the relation *until* for the end time point. Formally, a *temporal fact* is represented as:

#id: *s p o*
#id on time point
or
#id since begin time point
#id until end time point

For example, for "David Beckham has won the UEFA Club Player of the Year in 1999", they use the original time-agnostic fact as a primary fact with identifier #1. Then they represent the temporal property of the primary fact as #1 *on* 1999 as follows:

#1: DavidBeckham *hasWonPrize* UEFAClubPlayerOfTheYear
#1 on 1999

For example, for the fact that Beckham played for Real Madrid from 2003 to 2007 is represented as:

#2: DavidBeckham *playsFor* RealMadrid
#2 *since* 2003
#2 *until* 2007

Moreover, sometimes it is impossible to extract accurate time points, say the exact day, and sometimes ones may know only the begin or the end of a fact's validity interval but not both. For these situations, they adopt the time expression for time points with the *earliest* and *latest* possible time to constrain the range of the true time point. For example, if Beckham started playing for Real Madrid in July 2003 and terminated his contract in 2007, the fact can be represented as:

#2: DavidBeckham *playsFor* RealMadrid
#2 *since* [1-July-2003, 31-July-2003]
#2 *until* [1-January-2007, 31-December-2007]

In addition, they provide a time-aware query language on the T-YAGO knowledge base of temporal facts. They also implement a demo to support several types of temporal queries.

UncertainTemRDF (Dylla et al. 2011; 2013)

Dylla et al., (2011); (2013) present a way to resolve temporal conflicts in inconsistent RDF knowledge bases. An uncertain temporal knowledge base is a pair $KB = <F, C>$, where F is a set of weighted and temporal RDF triples and C is a set of first-order temporal consistency constraints. A fact in F has the form: $p(s, o, i)_d$, where $p(s, o)$ is an RDF triple, i is a (half-open) temporal interval of the form $[t_b, t_e)$ (where t_b is the begin time, t_e is the end time, and $t_b < t_e$), and $d \in [0, 1]$ is a confidence degree that p (s, o) is true during interval i.

For example, David Beckham was born in Leytonstone in 1975 with weight 0.9 can be represented by the fact $fbornBL$: = $bornIn$(*David Beckham*, *Leytonstone*, [1975, 1976))$_{0.9}$.

Moreover, they develop a query engine that is capable of scaling to very large temporal knowledge bases, with millions of uncertain facts and hundreds of thousands of grounded rules.

AnnotatedRDF (Zimmermann et al. 2012)

Zimmermann et al., (2012) present an RDF based annotation framework for representing, reasoning and querying data on the Semantic Web based on two works Annotated RDF (Straccia et al. 2010) and SPARQL extension AnQL (Lopes et al. 2010). In particular, the framework supports statements annotated with fuzzy, temporal, and provenance.

Formally, an annotated triple is an expression τ: λ, where τ is a triple and λ is an annotation value in a non-empty set L. An annotated graph is a finite set of annotated triples. Regarding the temporal domain, a temporal interval is a non-empty interval $[\alpha_1, \alpha_2]$, where α_i are time points. An empty interval is denoted as $\varnothing$. A partial order on intervals is defined as $I_1 \leq I_2$ if and only if $I_1 \subseteq I_2$. The intuition here is that if a triple is true at time points in I_2 and $I_1 \leq I_2$ then, in particular, it is true at any time point in $I_1 \neq \varnothing$. Now the set of intervals would be a candidate for L. Moreover, in order to represent the upper bound interval of τ: [1, 5] and τ: [8, 9], the union of intervals denoted {[1, 5], [8, 9]} is defined, meaning that a triple is true both in the former as well as in the latter interval. Now, L can be defined as (where $\perp = \{\varnothing\}$, $\top = \{[-\infty, +\infty]\}$) $L = \{t \mid t$ is a finite set of disjoint temporal intervals$\} \cup \{\perp, \top\}$.

For example, the annotated triple (niklasZennstrom, ceoOf, skype): [2003, 2007] means that "Niklas was CEO of Skype during the period 2003 to 2007", and the annotated triple (toivo, type, paypalEmp): {[1999, 2004], [2006, 2008]} means that toivo is type of paypalEmp during both the time intervals [1999, 2004] and [2006, 2008].

Further, they specify the semantics by conservatively extending the RDFS semantics, where an interpretation will assign to a triple τ an element of the annotation domain $\lambda \in L$. Moreover, they describe the implementationof AnQL based on constraint logic programming techniques along with a practical experiment for representing sensor data as Annotated RDF.

ValidityTimeRDF (Motik 2012)

Motik, (2012) present a logic-based approach to represent and query validity time in RDF. Unlike the existing proposals, their approach is applicable to nondeterministic entailment relations and/or entailment relations that involve existential quantification.

Formally, a temporal triple is an expression of the form $<s, p, o>[t]$ or $<s, p, o>[t_1, t_2]$ such that t is a time instant, t_1 is a time instant or $-\infty$, and t_2 is a time instant or $+\infty$. A temporal graph G is a finite set of temporal triples. They define the semantics of temporal triples by mapping them into formulae of many-sorted first-order logic. As an example, the triples <:LHR,: flightTo,: MUC > [50, 120] and <:LHR,: flightTo,: MUC > [100, 150] state that there is a flight from LHR to MUC, and this information may have been gathered from two distinct sources, so validity times of the two triples overlap.

Moreover, temporal axioms such as $(A \sqsubseteq \exists R.B)$ (Ao et al. 2020; Barbieri et al. 2009) can be encoded using temporal RDF triples as follows (where the symbol _:x denotes a blank node).

<A, rdfs:subClassOf, _:x> [5, 8]
<_:x, rdf:type, owl:Restriction>[$-\infty$,+ ∞]
<_:x, owl:onProperty, R > [$-\infty$,+ ∞]
<_:x, owl:someValuesFrom, B > [$-\infty$,+∞]

In addition, they define an extension of SPARQL to query temporal RDF and implement the approach in ExperienceOn's backend system and further test the approach with datasets consisting of tens of millions of triples to take as confirmation that the approach is amenable to practical implementation.

stRDF (Bereta et al. 2013)

Bereta et al., (2013) propose a formal extension of RDF, called stRDF, for the representation and querying of linked geospatial data that changes over time. In stRDF, the time line assumed is the (discrete) value space of the datatype *xsd*:*dateTime* of XML-Schema (W3C XML Schema Definition Language (XSD) 1 2012). The stRDF supports two kinds of time primitives, time instants and time periods, and the time instants can also be presented as time periods with the same beginning and ending time. Formally, a temporal triple is a quad (s, p, o, t), where t is a time instant or a time period called the valid time of a triple. Moreover, they introduce two temporal constants NOW and UC. NOW represents the current time and can appear in the beginning or the ending point of a period. UC (Until Changed) is used for introducing valid times of a triple that persist until they are explicitly terminated by an update.

For example, the following stRDF graph consists of temporal triples that represent the land cover of an area in Spain for the time periods [2000, 2006) and [2006, UC) and triples which encode other information about this area.

```
corine:Area_4   rdf:type    corine:Area.
corine:Area_4   corine:hasID    "EU-101324".
corine:Area_4   corine:hasLandCover    corine:coniferousForest "[2000-01-01T00:00:00,
        2006-01-01T00:00:00)"^^strdf:period.
corine:Area_4   corine:hasLandCover    corine:naturalGrassland "[2006-01-01T00:00:00,
        UC)"^^strdf:period.
corine:Area_4   corine:hasGeometry    "POLYGON((-0.66 42.34, ...))"^^strdf:WKT.
```

The above temporal triples denote that the area corine: Area_4 has been a coniferous forest area until 2006, when the newer version of CORINE showed it to be natural grassland. Until the CORINE Land cover dataset is updated, UC is used to denote the persistence of land cover values of 2006 into the future. The last triple of the stRDF graph gives the WKT serialization of the geometry of the area.

Moreover, they implement the approach support in Strabon, and evaluate the implementation with two different datasets named GovTrack dataset and the above CORINE Land Cover changes dataset.

WeightedTemporalRDF (Huber 2014)

Huber (2014) focus on carrying out reasoning and resolving inconsistencies in the temporal probabilistic knowledge bases containing erroneous facts to improve the data quality. The focus of their work is not on developing a data model for temporal and probabilistic facts. In their work, the basic facts are modeled as triples that express relations between entities. Moreover, a fact can be annotated with a confidence value and temporal information. A confidence value states the probability that the statement is true while temporal information indicates when the statement holds. In addition, they use well-known Allen's interval algebra (Allen 1983) to express the temporal relations of facts in order to enable temporal reasoning.

Being similar to the idea in temporal RDF graph model (Gutierrez et al. 2007; 2005; Hurtado and Vaisman 2006) as mentioned above, Huber (Huber 2014) propose a formalism that is suitable to express temporal and non-temporal as well as probabilistic and non-probabilistic facts and constraints. Formally, a fact can be represented as: *conf* (s, p, o) $[t]$, where *conf* denotes the confidence value that the statement (s, p, o) is true, t is a time point or interval. For example, 0.9 (Jack,

birthYear, 1961) [1961] or 0.9 (John, attended, University_1) [2002, 2006].

Further, based on the reification in RDF, they expressed a weighted temporal statement in RDF. For example, the statement 0.9 (John, attended, University_1) [2002, 2006] can be expressed as:

```
John attended University_1.
[rdf:type rdf:Statement;
rdf:subject John;
rdf:predicate attended;
rdf:object University_1;
weight "0.9"^^xsd:decimal;
start "2002";
end "2006" ].
```

These RDF triples then are converted to the corresponding ground predicate in Markov Logic for carrying out the reasoning: quadW("John", "attended", "Universtity_1", [2002, 2006], 1.0). Finally, their approach is evaluated on the temporal facts extracted from DBPedia dataset.

UTKGs (Chekol et al. 2017a)

Chekol et al., (2017a) propose a formal characterization of uncertain temporal knowledge graphs (UTKGs) to represent uncertainties and reason on temporal knowledge graphs. The syntax and semantics of UTKGs are formalized based on a numerical extension of Markov Logic Networks (MLNs), which combine first-order logic and Markov networks by attaching weights to first-order formulas). A UTKG is a temporal knowledge graph (KG) where each fact has a valid-time and an associated weight or confidence.

Formally, a UTKG is a tuple (D, U), D is a deterministic temporal KG and U is an uncertain temporal KG U with $D \cap U = \varnothing$. In detail, a deterministic temporal KG D is a set of temporal facts $f = (s, p, o, [t_1, t_2])$, where (s, p, o) is an RDF triple, $[t_1, t_2]$ is an ordered pair $[t_1, t_2]$ of time points with $t_1 \leq t_2$. An uncertain temporal KG U is a set of $\{f_i, w_{fi}\}$, where f_i is a temporal fact as mentioned above, w_{fi} is a real-valued weight assigned to f_i.

For example, a UTKG contains several temporal facts with associated weights {(ClaudioRanieri, coach, Chelsea, [2000, 2004], 0.9), (ClaudioRanieri, coach, Leicester, [2015, 2016], 0.7)}, which represent sport's personality career.

Further, they provide the semantics of a UTKG on the basis of a joint probability distribution over the uncertain part of the UTKG. Also, some temporal RDF inference rules are defined for reasoning on temporal knowledge graphs. Finally, they test the efficiency and scalability of the proposed approach based on their datasets by extracting temporal facts from footballdb.com and wikidata.org. Subsequently, they develop a system called TeCoRe in (Chekol et al. 2017b) for temporal inference and conflict resolution in the UTKGs.

TRDFS (Yang and Yan 2018)

Yang & Yan (2018) propose some mapping rules and algorithms for transforming the temporal XML (eXtensible Markup Language) Schema and document into temporal RDF Schema and temporal RDF triples, respectively. They further extend the temporal RDF graph model (Gutierrez et al. 2007) with both valid and transaction dimensions by labeling. They define a temporal RDF triple as a tuple (s, p, o): $[v]$ $[t]$, where (s, p, o) is an RDF triple, $[v]$ is the valid time of the triple, and $[t]$ is the transaction time of the triple. The expression (s, p, o): $[v_1, v_2]$ $[t_1, t_2]$ is a notation for $\{(s, p, o)$: $[v]$ $[t] \mid v_1 \leq v \leq v_2, t_1 \leq t \leq t_2\}$. In the model, the temporal labels are applied at the level of triples, not the level of the subject, property or object of triples.

Further, several new vocabularies are added to represent a temporal RDF graph. A temporal RDF graph, which is a set of (temporal) triples, consists of (s, p, o), $(X, tsubj, s)$, $(X, tpred, p)$, $(X, tobj, o)$, $(X, valid, V)$, and $(X, trans, T)$, where X is the triple (s, p, o), $tsubj$, $tpred$ and $tobj$ denote the subject, predicate and object of the triple X, and V and T represent the valid and transaction time, respectively.

For example, a temporal RDF triple (:FoxSports,: hasWebpage,: www.foxsports.com): [3, Now] [3, UC] denotes the valid and transaction time of the triple is [3, Now] and [3, UC]), respectively.

Based on the temporal RDF, they define a mapping function to transform temporal XML to temporal RDF data, and a prototype system called TX-TR is implemented to complete the automatic transformation.

TS-RDF (Rula et al. 2014; 2019)

Rula et al., (2014; 2019) present an approach for extracting time information from the sources including the document Web and the Linked Data Web, and further detecting the temporal scope of facts referred to by triples (i.e., the time points at which the temporal scope of the triple begins and ends). They define a *temporal annotation* of a fact a couple $<f, [t_i, t_j]>$, where f is a fact (i.e., a relation acknowledged to hold between subject and object of a triple) and $[t_i, t_j]$ is a time interval delimited by a starting time point t_i and an ending time point t_j. They regard time as a discrete and linearly ordered domain. Moreover, they state that the intervals $[t_i, t_j]$ and $[t_h, t_k]$ are disconnected iff $t_k < t_i$ or $t_j < t_h$. On this basis, they further define the *temporal scope* of a fact f as a set of *temporal annotations* of f with disconnected time intervals.

For example, <Balotelli, team, Inter Milan> refers to a fact occurring from 2007 to 2010 can be expressed as <(Balotelli, team, Inter Milan), [2007, 2010]>. Moreover, they evaluate the approach about the facts extracted from DBpedia and Freebase against the Yago2 knowledge base. The final experimental evaluations show that the approach can detect

temporal information for facts from DBpedia with an F-measure of up to 70%.

In addition, they propose a generic approach for deciding the time intervals in which the fact is valid in (Rula et al. 2019), and the results show that the approach can detect temporal information for facts from DBpedia with an *f*-measure of up to 80%.

Ontop-temporalRDF (Kalayci et al. 2019)

Kalayci et al., (2019) propose a framework for temporal ontology-based data access (OBDA). The framework uses extended mapping languages to extract information about temporal events in the RDF format. Being similar to the NG-tempRDF (Tappolet and Bernstein 2009), in order to represent temporal facts, for each temporal interval they introduce a graph identifier and collect the triples that hold within this interval into the respective graph. Further, the details of the interval (i.e., the beginning and the end) of the graph identifier are described in the distinguished graph using the vocabulary from the W3C TIME ontology, where two new data properties *isBeginInclusive* and *isEndInclusive* are added.

For example, a temporal fact about the the main flame sensor mf01 was registering the value 2.3 in the interval [12:32:30, 12:32:37) can be represented as mainFlame(mf01, 2.3)@[12:32:30, 12:32:37). Further, the temporal RDF triple can be modelled by a named graph g_0 in the form of GRAPH g_0 {(mf01, mainFlame, 2.3)} and a distinguished graph including some triples such as:

(g_0, time:hasTime, i_0),
(i_0, rdf:type, time:Interval),
(i_0, time:isBeginningInclusive, true),
(i_0, time:isEndInclusive, false),
(i_0, time:hasBeginning, b_0),
(b_0, time:inXSDDateTimeStamp, '2017-06-06 12:32:30'),
(i_0, time:hasEnd, e_0),
(e_0, time:inXSDDateTimeStamp, '2017-06-06 12:32:37').

Further, they use the classical ontology and rule languages to reflect static information, as well as a temporal rule language to describe events, and finally they develop an architecture of system implementation extending the state-of-the-art OBDA platform Ontop.

TKG (Huang et al. 2020)

Huang et al., (2020) propose a five-tuple to formulate a temporal knowledge graph. In a temporal knowledge graph, each fact is the form of <s, p, o, [t_1, t_2]>, where s is an entity or a class, p denotes the predicate, o is an entity, a class, or a literal value, and [t_1, t_2] is the corresponding time frame of the predicate. For example, <Georage W. Bush, bePresident, United States> was a fact only between 2001 and 2008 can be represented as <Georage W. Bush, bePresident, United States, [2001, 2008]>.

On this basis, a temporal knowledge graph can be formally defined as a five-tuple $G = <V, L_V, E, L_E, T_E>$, where $V = V_c \cup V_e \cup V_l$ is a collection of vertices including all subjects and objects in knowledge graphs, where V_c, V_e, and V_l are collections of class vertices, entity vertices, and literal vertices, respectively; L_V is a collection of vertex labels; $E = <u_i, u_j, [t_s, t_e]>$ is a collection of directed edges that connect the corresponding subjects and objects; L_E is a collection of edge labels; T_E is a collection of time frames on edges, and there are two kinds of time frame on edges, one is $[t_s, t_e]$ ($t_s < t_e$), where t_s and t_e are exact value, the other is $[*, t_e]$ or $[t_s, *]$, where $*$ means past or present. Further, they also consider Allen's 13 temporal relations (Allen 1983) and divide the thirteen types into three classes, i.e., non-intersect, overlap and including. Moreover, they also give the definition of Time Frame Compatible. That is, give two time frames [t_1, t_2] and [t_s, t_e], [t_1, t_2] is said to be compatible with [t_s, t_e] while the time relationship between [t_1, t_2] and [t_s, t_e] is overlap or including.

Based on the definition of temporal knowledge graphs, they investigate the cluster query on temporal knowledge graph in depth, and some experiments demonstrate the effectiveness of the proposed methods.

Temporal RDF models based on the original form of RDF triple by adding timestamp

This kind of models are based on the conventional form of RDF triple <s, p, o> by adding timestamp information. Table 4 *first* summarizes and compares these temporal RDF models from several aspects. *Subsequently*, we further introduce each approach in detail.

MRDF (Gergatsoulis and Lilis 2005)

Gergatsoulis & Lilis (2005) present a Multidimensional RDF (or MRDF) for the main purpose of representing context-dependent RDF data. MRDF extends the RDF model so that the objects of triples may have values that are context dependent. Contexts form an n-dimensional space, so a context is specified by defining its coordinates in that space, one of which may be time. Moreover, they also demonstrate that MRDF graphs can be used as a formalism for tracking the history of conventional RDF-graphs. They also discuss three primitive change operations on RDF graphs, namely update, delete, and insert.

Table 4 The existing temporal RDF models based on new RDF extension syntaxes

	Contribution	*Representation form*	*Example*	*Storage*	*Query*	*Implementation*
MRDF (Gergatsoulis and Lilis 2005)	Tracking the *time history of conventional RDF-graphs*. Discussing three primitive change operations on RDF graphs, namely update, delete, and insert.	*Syntax*: <*s p o*>, where *o* is the form of $[(d_1, v_1), \ldots, (d_n, v_n)]$ *Comment*: The object *o* of an RDF triple (*s p o*) is annotated with a temporal interval of the form $\{t_{start} .. t_{now}\}$. Formally, the object *o* contains pairs of values $[(d_1, v_1), \ldots, (d_n, v_n)]$, where d_i is a dimension specifier such as [*time* in $\{t_1 .. t_2\}$)] or [*customer_type*=library], v_i is the object (resource/literal).	LP_book author _:abc. _:abc email **[**([*time* in {*start* .. 2010}], "mge@ionio.gr"), ([*time* in {2011 .. *now*}], "mge@otenet.gr"**)]**. _:abc telephone **[**([*time* in {2011 .. *now*}, *customer_type*=library], +8,683,688,888**)]**.	A simple relational database schema is designed for storing the History Graphs	No	No
Temp CRM (Liao and Tu 2007)	Presenting a *temporal context reasoning model* (TempCRM) to infer the dangerous level of a smart home based on RDF.	*Syntax:* <*s p o*> *Comment:* A temporal event is expressed as a *context predicate*, which is a 4-tuple (SVOT) ContextType (<subject>, <verb>, <object>, <time>). The 4-tuple context predicate can be transformed into the conventional RDF triple <*s p o*> .	Location (Lily, entering, kitchen, 10:20:40). Device (Gas_switch#1, is, on, 10:25:40). These 4-tuples are transformed into RDF triples: <Gas_switch#1 rdf:type Device>. <Gas_switch#1 trdf:eventTime 10:25:40>.	No	No	No
tRDF (Pugliese et al. 2008)	Extending the work of (Gutierrez et al. 2007; 2005) to the case of *indeterminate* triples, and proposing a temporal RDF model tRDF.	*Syntax*: a triple can take one of the forms: (*i*) (*s*, *p*: {*T*}, *o*). It indicates that the triple (*s*, *p*, *o*) holds at *every* time point in *T* (*T* denotes the set of all possible time intervals). (*ii*) (*s*, *p*: <*n*: *T*>, *o*), where *n* is a natural number. It indicates that the triple (*s*, *p*, *o*) holds *at least n distinct* time points within *T*. (*iii*) (*s*, *p*: [*n*: *T*], *o*). It indicates that the triple (*s*, *p*, *o*) holds *at most n distinct* time points within *T*.	Regarding the form (*ii*), for example, (*people/B000711*, *campaign*: <8:2004>, *campaign/2004/S2CA00286*), which denotes that the politician with identifier *people/B000711* campaigned for *campaign/2004/S2CA00286* at least 8 months in 2004.	A single relation table (subject, property, object, annotation) in a PostgreSQL 8.0 database	A tRDF querying technique	Yes
TA-RDF (Rodríguez et al. 2009)	Presenting TA-RDF model to express not only that the properties between resources change over time, but also that *the resources themselves change*, where resources are optionally annotated with a time value.	*Syntax*: <*s* [*t*], *p* [*t*], *o* [*t*]> *Comment*: where $t \in T \cup \{Nil\}$. The time domain *T* is a discrete, ordered infinite set intended to represent discrete time and will be assumed to be the set of natural numbers. Intuitively, a time annotated resource *r*[*t*] represents the state of the resource *r* at the point *t* in time. The marker *Nil* indicates a time invariant, used to express a property of the resource that does not change.	<urn:OHARE, urn:hasTemperatureSensor, 25["2008-06-16"^^xsd:dateTime]>, which denotes that the temperature on Chicago O'Hare at 2008-06-16 is 25.	No	A query language TA-SPARQL	Yes (the prototype was implemented on top of the semantic content management middleware Tupelo)

Table 4 (continued)

	Contribution	*Representation form*	*Example*	*Storage*	*Query*	*Implementation*
Temp RDF (McBride and Butler 2009)	Presenting an approach for representing *valid time* that complies with the existing RDF abstract model and semantics.	*Syntax*: (*s*, *p*: (*begin--end*), *o*) *Comment*: where *p*: (*begin--end*) is a shorthand for a URI that identifies a temporal property which has base property *p*.	(p:_12345 f:memberOf: (2000-11-27--2009-10-01) f:HPLabs), which represents an employee was a member of HPLabs from the 27th November 2000 to the 1st Oct 2009.	TDB storage (Owens et al. 2009), which is a clustered triple store for Jena (Carroll et al. 2004)	Using the normal SPARQL query language	Yes (the prototype was developed using Jena (Carroll et al. 2004) and TDB (Owens et al. 2009))
aRDF (Udrea et al. 2010)	Presenting the aRDF model by attaching an annotation to the property of a triple based on the *annotated logic*.	*Syntax*: (*s*, *p*: *a*, *o*) *Comment*: where $a \in A_{\text{time}} \cup A_{time-int}$, $A_{\text{time}}=\mathbf{N}$ could be the set of all non-negative integers (denoting time points), and $A_{time-int}=\{[x, y] \mid x, y \in \mathbf{N}\}$ could be the set of all time intervals.	(*people/B000711*, *role*: [1987, 1988], *congress/100/ca*) denotes the member people/ B00711 was a representative in the 100th Congress between 1987 and 1988.	The aRDF datasets were stored in flat binary files on disk.	An aRDF query language	Yes
CNTRO (Tao et al. 2010)	Proposing a Clinical Narrative Temporal Relation ontology (CNTRO). Based on CNTRO, temporal information can be expressed as triples.	*Syntax*: (*s*, *p*, *o*) Comment: several classes *Event, Time, Duration, Granularity, TemporalRelationStatement* are added. And temporal instances and temporal relations are represented as RDF triples (*s*, *p*, *o*).	<*event*$_i$>*rdf:type Event*; *hasTimeStamp*<*tInst*$_i$>. <tInst$_i$>*rdf:type TimeInstant*; *hasOrigTime* "June 10, 2004"; *hasGranularity* "day". denotes that the event *event*$_i$ has a time stamp *tInst*$_i$.	No	No	Yes (the ontology was evaluated on clinical notes)
Temp RDFLod (Rula et al. 2012)	Proposing an abstract definition of temporal information and further investigating its representation based several temporal RDF models in Linked Data.	*Syntax*: (*s*, *p*, *o*) *Comment*: where a temporal information is described at the abstract level as a temary relation *T*(*x*, *a*, *t*), where *x* is a resource, a statement, or a graph, *a* is a predicate symbol, and *t* is a temporal entity likes temporal interval [t^b; t^e] with a starting point t^b and an ending point t^e. Further, the abstract definition can be represented by the different forms of the RDF triple (*s*, *p*, *o*).	s^{st} rdf:type rdf:Statement . s^{st} rdf:subject: s001 . s^{st} rdf:predicate: study . s^{st} rdf:object: neu . s^{st} a_S^b 1993 . s^{st} a_S^e 1997 . which denote the student: s001 studied in neu from 1993 to 1997.	No	No	Yes (the experiments are tested on Billion Triple Challenge 2011 dataset)
SingletonProperty RDF (Nguyen et al. 2014)	Presenting a *Singleton Property* model for representing statements about RDF statements. The model can express additional information about RDF triples, such as *time* and *certainty*.	*Syntax*: (*s*, *p*, *o*) *Comment*: each relationship between the subject and the object of one statement in one particular context is uniquely represented by using a singleton property, and some properties (e.g., *singletonPropertyOf*, *hasStart*, and *hasEnd*) are added for expressing temporal or other information.	Bobn isMarriedTo#1 Sara. isMarriedTo#1 *hasStart* 1965-11-22. isMarriedTo#1 *hasEnd* 1977-06-29. which denote the marital status of Bobn and Sara.	No	Using the standard SPARQL query language	Yes
	Proposing simple *fuzzy* and *temporal* RDF/RDFS ontologies to exploit	*Syntax*: (*s*, *p*, *o*)	For example, a predicate *approx_at_instant*(?*degree1*,? *event*,?	No	Using the standard	Yes (a Linguistically

Table 4 (continued)

	Contribution	*Representation form*	*Example*	*Storage*	*Query*	*Implementation*
LORI (Robatjazi et al. 2015)	temporal data containing complex relations.	*Comment*: some concepts and predicates are added for representing temporal as well as fuzzy information based on the RDF and RDFS.	*timeInstant*) represents the degree of temporal overlapping between an event and an instant in time and is represented as RDF format.		SPARQL query language	Oriented RDF Interface LORI)
RDFt (Zhang et al. 2019)	Proposing an RDFt model for representing temporal data with both the *time* information and the *update count* information.	*Syntax*: (*s*, *p*[*t*]-*n*, *o*) or (*s*, *p*[*ts*, *te*]-*n*, *o*) *Comment*: the sequential information in RDFt data model refers to valid time, which can be either a time point *t* or a time interval [*ts*, *te*], and *n* describes the update count information.	For example, "John" was born in "City_A" on "1980-12-30" can be represented as (John, hasBirthCity[1980-12-30]-1, City_A)	Neo4J database	A SPARQL[t] query language	Yes (a prototype system was implemented for querying the temporal data)

Formally, the object *o* of an RDF triple <*s p o*> is annotated with a temporal interval of the form $\{t_{start} .. t_{now}\}$. In detail, the object *o* contains pairs of values $[(d_1, v_1), \ldots, (d_n, v_n)]$, where d_i is a dimension specifier such as [*time* in $\{t_1 .. t_2\}$)] or [*customer_type* = library], v_i is the object (resource/literal) corresponding to the specifier.

As an example, the email of the person *manolis* is "manolis@ionio.gr"in 2010 while he change the email as "mgerg@otenet.gr" in 2011, such facts can be represented by a triple <:*manolis email* [([*time in* {*start* .. 2010}], "manolis@ionio.gr"), ([*time in* {2011 .. *now*}], "mgerg@otenet.gr")]>. A more detailed example from (Gergatsoulis and Lilis 2005) and its graphical representation can be found in Fig. 3.

TempCRM (Liao and Tu 2007)

Liao & Tu (2007) present a temporal context reasoning model (TempCRM) to infer the dangerous level of a smart home. In a smart home, a potentially dangerous situation is caused by a series of temporal events. A temporal event is expressed as a context predicate, which is a four tuples (SVOT) ContextType (<*subject*>, <*verb*>, <*object*>, <*time*>). ContextType is the type of a temporal event, e.g., location, temperature, and device. <*subject*> is the thing or person giving the context. <*verb*> is the action or relation to the object. <*object*> is the value of the context. <*time*> is the occurrence time of the event. For example:

Location (Lily, entering, kitchen, 10:20:40).
Device (Gas_switch#1, is, on, 10:25:40).

Further, they use the RDF triple to represent the context predicate. For example, the above temporal predicates can be transformed into the following RDF triples as shown in Fig. 4. In addition, they define a smart home ontology and temporal reasoning rules to infer and computer the dangerous level.

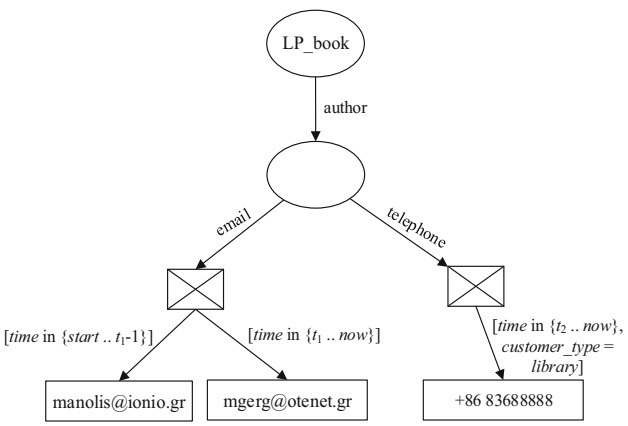


Fig. 3 An example of Multidimensional RDF model

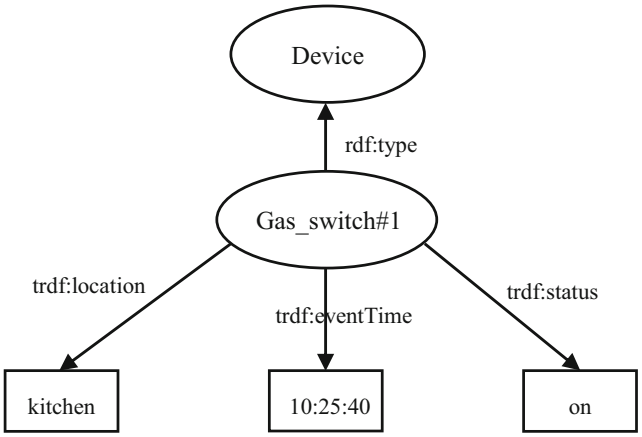


Fig. 4 An example of TempCRM RDF model

tRDF (Pugliese et al. 2008)

Pugliese et al., (2008) extend the work of (Gutierrez et al. 2007; 2005) to the case of *indeterminate* triples. They call triples that are valid continuously during a time interval *determinate triples* as mentioned in (Gutierrez et al. 2007; 2005). For instance, the *USA* (subject) may have *Bill Clinton* (object) as the value of its *President* property throughout the entire time frame 1992–2000. Likewise, some triples may only be known to occur at certain time points during an interval, rather than be true continuously throughout the interval. For example, company *A* (subject) may have a *supplier* (property) relationship with company *B* (object) at some points of time during the time interval 10–20. They call such triples *indeterminate triples*.

A temporal RDF (tRDF for short) database consists of a set of temporally annotated RDF triples of the form (*subject*, *property*: *annotation*, *object*). The annotation of a tRDF triple can take one of the following forms:

- $(s, p: \{T\}, o)$. It indicates that the triple (s, p, o) holds at *every* time point in T (T denotes the set of all possible time intervals). For example, (*Senate Joint Resolution* 37, *inCommittee*: {05/2002}, *Senate Finance Committee*) denotes that *Senate Joint Resolution 37* was under discussed in the *Senate Finance Committee* throughout May 2002.
- $(s, p: <n: T>, o)$, where n is a natural number. It indicates that the triple (s, p, o) holds *at least n distinct* time points within T. For example, (*people/B000711*, *campaign*: <8:2004>, *campaign/2004/ S2CA00286*) denotes that the politician with identifier *people/B000711* campaigned for *campaign/2004/S2CA00286* at least 8 months in 2004.
- $(s, p: [n: T], o)$. It indicates that the triple (s, p, o) holds *at most n distinct* time points within T. For example, (*Carol*, *contributed*: [2:2004], *campaign/2004/S2CA00286*) denotes that Carol contributed to the campaign of politician people/B000711 at most two times in 2004.

Moreover, the semantic of tRDF is defined by an interpretation function I which associates a set of RDF triples with every time-point. They also point that not all interpretations make sense for a given tRDF database, and thus the definition of a satisfying interpretation is also provided. Further, they state that a tRDF database D entails a tRDF triple e (written as $D \models e$), iff for each tRDF interpretation I such that $I \models D$. In addition, they investigate the corresponding tRDF query techniques and defined the tGRIN index structure for temporal RDF. Finally, they store the tRDF data in a single relation (*subject*, *property*, *object*, *annotation*) in a PostgreSQL 8.0 database and implement a framework in Java.

TA-RDF (Rodríguez et al. 2009)

Rodríguez et al., (2009) present a Time-Annotated RDF (TA-RDF for short) to represent and query time-series streaming data. TA-RDF is an extension of the RDF model where resources are optionally annotated with a time value, i.e., a time-annotated resource is a pair of the form *resource*[*time*], where *resource* ∈ URI References ∪ Blank Nodes ∪ Literal and *time* $\in T \cup \{Nil\}$. The time domain T is a discrete, totally ordered infinite set intended to represent discrete time and will be assumed to be the set of natural numbers. Intuitively, a time annotated resource $r[t]$ represents the state of the resource r at the point t in time. The time marker *Nil* indicates a time invariant, used to express a property of the resource that does not change. All literals are considered to be annotated with a time of *Nil* as it goes against the spirit of the model to consider that literals change over time. The resource $r[*]$, or the state of r for each instant, represents a data stream composed by the ordered sequence of frames $r[t]$ for all $t \in T \cup \{Nil\}$. Notice that this allows for easily expressing not only that the relationship between entities change over time, but that the entities themselves change, and that they might do so in different time scales.

Moreover, the semantics of TA-RDF extends directly of the RDF semantics. In addition, they define a Time-annotated SPARQL (TA-SPARQL) query language, and develop an implementation of TA-RDF and TA-SPARQL.

Further, to translate a TA-RDF graph into a regular RDF graph, a data stream vocabulary is used, where (*i*) *dvs:belongsTo* is a property indicating a resource is a frame in a stream, (*ii*) *dvs:hasTimestamp* is a property indicating the timestamp of a frame, and (*iii*) *dvs:Nil* is a resource corresponding to the *Nil* timestamp. The TA-RDF model remains translatable/compatible with regular RDF by means of a special RDF vocabulary, which allows the use of existing RDF tools and technologies.

Noted that, they use time-annotating resources instead of triples based on the consideration that RDF is often employed for metadata annotations of data stored outside of RDF. These data or resources may be time varying which makes useful to

possess a mechanism to refer to them according to their position in time.

TempRDF (McBride and Butler 2009)

McBride & Butler (2009) present an approach for representing valid time that complies with the existing RDF abstract model and semantics. They create a new property for each time interval for which a base property is asserted. A temporal RDF graph is a set of triples of the form (s, p: (*begin–end*), o), where p: (*begin–end*) is a shorthand for a URI that identifies a temporal property which has the base property p.

For example, they represent the assertion that an employee was a member of HPLabs from the 27th November 2000 to the 1st October 2009 as follows:

p:_12345 f:memberOf: (2000-11-27–2009-10-01) *f:HPLabs* .

The notation *f:memberOf*: (2000-11-27–2009-10-01) is shorthand for a URI that identifies a temporal property. A temporal property has a *base property*, in this example *f:memberOf*, that is asserted for a time interval, in this example (2000-11-27–2009-10-01). The temporal property in this example has the following properties:

f:memberOf: (2000-11-27--2009-10-01)
a rdf:Property;
tb:property f:memberOf;
tb:begin "2000-11-27"^^xsd:date;
tb:end "2009-10-01"^^xsd:date .

The *tb:property* triple identifies the base property of the time bounded property, the *tb:begin* property identifies the start of the interval for which it is valid and the *tb:end* property identifies the end of the interval for which it is valid.

The semantic of temporal RDF graph is also called temporal interpretation by extending RDF and RDF Schema for temporal properties. Moreover, they state that the temporal RDF graphs are ordinary RDF graphs and so can be queried in the normal manner using SPARQL.

aRDF (Udrea et al. 2010)

Udrea et al., (2010) present *Annotated RDF* (aRDF for short), which is built on top of annotated logic, attaches an annotation to the property of a triple, such as timestamp (time point or interval), a probability or indicator of the provenance associated with the triple.

In aRDF, one can start with any partially ordered set that you like as long as it has a bottom element. For a given partially order set (A, $\leq$), an element $\perp \in A$ is the "bottom element" of A iff $\perp \leq x$ for all $x \in A$. A could capture timestamps, fuzzy values, pedigree information or temporal fuzzy information, and so on. Assuming the existence of a partially ordered set (A, $\leq$) where elements of A are called *annotations*, $\leq$ is a partial ordering on A, and A has a unique bottom element. Then, one could have any of the following scenarios (here we only introduce the scenarios related to the temporal aspects): (*i*) $A_{\text{time}} = \mathbf{N}$ could be the set of all non-negative integers (denoting time points) with the usual "less than or equals" ordering on it. (*ii*) $A_{time-int} = \{[x, y] \mid x, y \in \mathbf{N}\}$ could be the set of all time intervals. The interval $[x, y]$ as usual denotes the set of all $t \in \mathbf{N}$ such that $x \leq t \leq y$. The inclusion ordering $\subseteq$ is a partial ordering on this set. On this basis, an annotated RDF theory (aRDF-theory for short) is a finite set of triples (s, p: a, o), where $a \in A$.

For example, the triple (*people/B000711*, *role*: [1987,1988], *congress/house/100/ca*) denotes the fact that the member of Congress identified by people/B00711 was a representative of the state of California in the 100th Congress between 1987 and 1988.

Further, they present a formal declarative semantics for aRDF and develop algorithms to check consistency of aRDF theories and to answer queries to aRDF theories.

CNTRO (Tao et al. 2010)

Tao et al., (2010) propose a Semantic Web ontology called Clinical Narrative Temporal Relation ontology (CNTRO). Based on the CNTRO ontology, temporal information in clinical narratives can be expressed as RDF triples. In the ontology, several classes *Event*, *Time*, *Duration*, *Granularity*, and *TemporalRelationStatement* are added.

- The class *Event* denotes any event of occurring on a time line.
- The class *Time* includes four sub-classes for representing temporal informaton: *TimeInstant* (i.e., time point), *TimeInterval* (i.e., time interval), *TimePhase*, and *TimePeriod*. The sub-class *TimePhas* represents each occurrence of a repeating interval, and the sub-class *TimePeriod* is used to measure the frequency at which the *TimePhase* repeats. For example, "every five minutes for one day starting from tomorrow" is a *TimePhase*, and its *TimePeriod* is "every five minutes".
- The class *Duration* denotes the time length of a TimeInterval, and the class *Granularity* specifies the granularity of each time instant.
- The class *TemporalRelationStatement* describes temporal relations between an event and an instance of *Time* or between two events.

All of these classes, temporal instances and temporal relations can be represented as RDF triples (s, p, o). For example, the following RDF triples represent an event with a time instant.

```
<event_i> rdf:type Event;
    hasTimeStamp <tInst_i>.
<tInst_i> rdf:type TimeInstant;
    hasOrigTime "June 10, 2004";
    hasNormalizedTime 2004-06-10;
    hasGranularity "day".
```

In the example, the triple (<$event_i$>, *hasTimeStamp*, <$tInst_i$>) denotes that the event $event_i$ has a time stamp $tInst_i$. Further, the $tInst_i$ belongs to the class *TimeInstant* and has several properties. Finally, the ontology is evaluated on real clinical notes, and several experimental results denote that the ontology can represent the temporal-related information in real clinical notes.

TempRDFLod (Rula et al. 2012)

Rula et al., (2012) investigate several temporal RDF models and perform both a quantitative and a qualitative analysis about the characterisation and availability of these models in Linked Data at large scale. They first propose an abstract definition of temporal information, where a temporal information can be described as a ternary relation $T(x, a, t)$, where x is a resource, a statement, or a graph, a is a predicate symbol, and t is a temporal entity likes temporal interval $[t^b; t^e]$ with a starting point t^b and an ending point t^e.

Further, they define three core perspectives adopted for the concrete representation of temporal meta-information in RDF such as Document-centric perspective and Fact-centric perspective (including two subcategories Sentence-centric perspective and Relationship-centric perspective). In Document-centric perspective, RDF documents (graphs) associated with temporal-meta information can be represented by the protocol-based approach (e.g., HTTP) and the metadata-based approach (e.g., Dublin Core).

In Fact-centric perspective, facts can be represented by the temporal RDF model (Gutierrez et al. 2005) $<s, p, o>[t^b: t^e]$. Further, in order to make statements about statements in the RDF data model, they introduce several approaches that are able to implement the time dimension. For example, in Sentence-centric perspective, the temporal RDF-based representation $<s, p, o>[t^b: t^e]$ can be further transformed into the following reification form:

s^{st} rdf:type rdf:Statement.
s^{st} rdf:subject s .
s^{st} rdf:predicate p .
s^{st} rdf:object o .
s^{st} a_S^b t^b .
s^{st} a_S^e t^e .

where s^{st} is a modelling of a statement as a resource, a_S^b, a_S^e are two temporal properties with respect to the beginning and the ending point of a time interval $[t^b{:}t^e]$. In Relationship-centric perspective, the temporal RDF-based representation $<s, p, o>[t^b: t^e]$ can be transformed into the following N-ary relationship form:

$s\ \overline{p}\ o^p$.
$o^p\ p\ o$.
$o^p\ a_R^b\ t^b$.
$o^p\ a_R^e\ t^e$.

where a_R^b and a_R^e are two temporal properties with respect to the beginning and the ending point of a time interval $[t^b{:}t^e]$, o^p is the instance of the class property, p is the old property with the new object.

SingletonPropertyRDF (Nguyen et al. 2014)

Nguyen et al., (2014) present a Singleton Property model for representing statements about RDF statements. The model can express additional information about RDF triples, such as time, trust, provenance, certainty, and location. The basic idea is that each relationship is universally unique, and such uniqueness of the relationship can be a key for any statement using a singleton property. A URI will be assigned for each singleton property.

In detail, a *singleton property* is represented as an instance of its generic property by a new property *singletonPropertyOf*, which is defined as a subproperty of *rdf:type*. For example, a generic relationship *isMarriedTo* can be established between Bob and Sara according to two Wiki pages. In this case, each Wiki page will be considered as a context associated with the relationship. Thus, two singleton properties *isMarriedTo#1* and *isMarriedTo#2* are created to represent the relationships associated with these two contexts, and the following several statements are defined:

isMarriedTo#1 *singletonPropertyOf* isMarriedTo.
isMarriedTo#2 *singletonPropertyOf* isMarriedTo.
Bobn *isMarriedTo*#1 Sara.
Bob *isMarriedTo*#2 Sara.

Further, in order to represent metadata for a statement, such as provenance, certainty, location, or time, some meta properties may be added. Taking the *time* for example, the validity of a statement may be associated with a time point or a time span. The following statements represent the time span of the marriage between Bob and Sara using two properties *hasStart* and *hasEnd*:

isMarriedTo#1 *hasStart* 1965-11-22.
isMarriedTo#1 *hasEnd* 1977-06-29.

Moreover, they also explain the new properties mentioned above and provide a formal semantics by adding additional

criteria for supporting the singleton property. The case studies on two datasets show that the singleton property results in a decent performance in the aspects of the number of triples and querying execution time.

LORI (Robatjazi et al. 2015)

Robatjazi et al., (2015) propose simple fuzzy and temporal RDF/RDFS ontologies to exploit temporal data containing complex relations. Some concepts and built-in predicates are added for representing temporal as well as fuzzy information based on the RDF and RDFS. Several concepts including *ValidTime*, *Event*, and *Granularity* are defined. *ValidTime* represents possible "moments" when information is true and is a superclass for two classes Instant and Interval. *Event* represents a class of entities occurring in time. *Granularity* represents time granularity of individuals of the class ValidTime such as Minutes, Hours, and Days. Alao, some predicates such as *approx_at_instant*(?*degree1*,? *event*,? *timeInstant*), *approx_in_interval*(?*degree*,? *event*,? *timeInterval*), and *approx_at_instant_before*(?*degree*,? *event*, *n*, *granularity*,? *referenceInstant*) are added. For example, the predicate *approx_at_instant* can determine the degree of temporal overlapping between an event and an approximated instant in time.

Further, they develop a Linguistically Oriented RDF Interface–LORI, which uses a standard SPARQL query language and can construct temporal queries and support imprecise phrases describing time and data features, and high-level predicates built based on them. Some case studies focus on querying time-based events and their performance evolutions are made.

RDFt (Zhang et al. 2019)

Zhang et al., (2019) propose an RDFt model for representing temporal data with both the time information and the update count information. The RDFt model retains the RDF triple form and then extends the triple by adding the time information and update count information into the predicate p of the triple (s, p, o).

Formally, a temporal RDF triple may be (s, p[t]-n, o) for valid time point or (s, p[ts, te]-n, o) for valid time interval, where $t \in T$ (T is the time domain), $n \in$ N denotes the update count information, i.e., the triple records are updated n times. The change of n is based on the change of transaction time, and the largest n represents the most recent historical record of the triple, and the boundary value of the time interval is expressed by the start time point (*ts*) and the end time point (*te*). For example, "John" was born in "City_A" on "1980-12-30" can be represented as (John, hasBirthCity[1980-12-30]-1, City_A).

They also define the semantics of the RDFt model and give a temporal data query language SPARQL[t] by extending RDF standard query language SPARQL. A prototype system is implemented for querying the temporal data.

Temporal RDF querying techniques

As surveyed in Temporal RDF representation models section, there are lots of temporal RDF representation models for modeling and expressing time information. Accordingly, in order to further achieve efficient query on the temporal information, some temporal RDF querying techniques were subsequently proposed. To help readers understand and catch key techniques about the issue of temporal RDF querying techniques, in this section we make a detailed survey on the existing techniques and make some discussions.

Table 5 first summarizes and compares the temporal RDF querying techniques from the aspects of purposes and query language forms. We also provide some examples for well introducing each approach. Then, each technique will be further discussed in detail in the subsequently text.

QtRDFGraph (Gutierrez et al. 2007; 2005; Hurtado and Vaisman 2006)

Gutierrez et al., (2007; 2005; Hurtado and Vaisman 2006) present a sketch of temporal query language for temporal RDF graphs $<s, p, o>[t]$ as mentioned in Table 3. They introduce the query language by providing some query examples, along with its semantics. For example, a simple query: "Students taking Master courses between t_1 and t_2" can be expressed as:

$$(?X, type, Student) \leftarrow (?X, takes, ?C): [?T], (?C, type, Master): [?T], t_1 \leq ? T, ? T \leq t_2.$$

Moreover, they present a brief study of the complexity of query processing. They state that the temporal labeling over the triples does not introduce any complexity overhead. In addition, they consider the problem of emptiness of the query answer set, including the Query Complexity Version QCV (i.e., for a fixed database D, given a query q, is $q(D)$ nonempty?) and the Data Complexity Version DCV (i.e., for a fixed query q, given a database D, is $q(D)$ nonempty?). They prove that the evaluation problem is NP-complete for the QCV and polynomial for the DCV.

However, they do not provide query processing algorithms and an implementation or experimental details of the language.

QtRDF (Pugliese et al. 2008)

Pugliese et al., (2008) present a formal definition of tRDF queries and answers for the tRDF model as mentioned in

Table 5 The existing Querying techniques for temporal RDF models

	Purpose	*Query language form*	*Example*
QtRDFGraph (Gutierrez et al. 2007) (Gutierrez et al. 2005; Hurtado and Vaisman 2006)	Presenting a sketch of temporal query language by providing some query examples for temporal RDF graph model <*s*, *p*, *o*>[*t*] as mentioned in Table 3.	Let *V* be a set of variables [disjoint from *U* (RDF URI references), *B* (Blank nodes), *L* (RDF literals), and *T* (Temporal Vocabulary)]. Individual variables will be denoted? *X*,? *Y*, etc. There are temporal variables $V_t \subset V$. A *query* is a *temporal tableau*, which is a pair (H, $B \cup A$), where *H* and *B* are temporal RDF graphs with some elements of $U \cup B \cup L$ replaced by variables in *V* and with some elements of *T* replaced with variables in V_t; *B* has no blank nodes and all the variables in *H* occur also in *B*. The set *A* has the usual arithmetic built-in predicates such as <, >, =, over elements in V_t and *T*.	A query: "*Students* taking *Master* courses between t_1 and t_2" (i.e. starting and completing their studies within this interval) can be expressed as: (?*X*, rdf:type, *Student*) ← (?*X*, *takes*,? *C*): [?*T*], (?*C*, rdf:type, *Master*): [?*T*], $t_1 \leq$? *T*,? $T \leq t_2$.
QtRDF (Pugliese et al. 2008)	Presenting a tRDF query language for temporal tRDF model in Table 4, and a tGRIN *index* that builds a specialized index for temporal RDF.	A tRDF query is a conjunctive SPARQL query that is augmented with *temporal annotations on the edges* (either variable or constant). Formally, a tRDF query is a 5-tuple (N, E, V, λ_n, λ_t). The details can be found in the subsequent text of our review.	A query triple in the tRDF query tuple can be represented as? v_1 *supported*: <1: 2000–2005> congress/107/bills/sj37.
TA-SPARQL (Rodríguez et al. 2009)	Presenting a query language *Time-annotated* SPARQL (TA-SPARQL) for their TA-RDF model as mentioned in Table 4. Also, providing the rules for *translating* TA-SPARQL to SPARQL.	The new constructs in TA-SPARQL are listed in (Rodríguez et al. 2009) in detail, here we take several TA-SPARQL query forms as examples: {… *resource*[t_0] *predicate object* …}, which specifies resource in a triple by timestamp {… *resource*[t_0..t_1] *predicate object* …}, which specifies resource in a triple by time range.	A query "what was the humidity on SY at 2019-06-28?" can be expressed as: SELECT? humidity WHERE {<urn:SY><urn:hasHumiditySensor>? *v*["2019-06-28"^^xsd:dateTime]. ?*v*<urn:hasReading>? humidity.}
QTempRDF (McBride and Butler 2009)	Using SPARQL for querying the temporal RDF model (*s*, *p*: (*begin*--*end*), *o*) as mentioned in Table 4.	A query triple can be directly described as a normal SPARQL query triple such as <?*s*,? *p*,? *o*>, where it allows predicates? *p* in triple patterns to specify a begin time and an end time.	A query "who worked in HPLabs in 2008" can be described as: DESCRIBE? labbie WHERE {? labbie f:memberOf:(2008–2009) f:HPLabs.}
QMulti-tempRDF (Grandi 2011; 2009; 2010)	Presenting a T-SPARQL temporal query language for the Multi-temporal RDF model (*s*, *p*, *o* | *T*) as mentioned in Table 3.	A graph pattern of the WHERE clause in a T-SPARQL query is extended with an optional timestamp position by extending the SPARQL WHERE clause in the SELECT statement such as <?*s*,? *p*,? *o* |? *ts*>.	A query "what are the names of employees who worked in the Sales department for more than two years" can be described as: SELECT? ename WHERE { ?emp rdf:type ex:emp; ex:Name? ename; ex:Dept "Sales" |? t. FILTER (xs:duration (VALID(?t))>"P2Y"^^xs:duration).}
τ- SPARQL (Tappolet and Bernstein 2009)	Presenting a temporal query language τ-SPARQL for querying the temporal RDF model based on *named graph* as mentioned in Table 3 and *transforming* τ-SPARQL into SPARQL.	Two kinds of query forms (i.e., time point queries and temporal queries) are considered in τ-SPARQL. For *time point queries*, a FROM SNAPSHOT τ expression is added by extending a FROM statement in SPARQ, where τ is a time point such as: SELECT? *x* FROM SNAPSHOT τ WHERE{} For *temporal queries*, a time interval is allowed in a graph pattern such as: [?*s*,? *e*]? *x*? *y*? *z* or? *x*? *y*? *z* [?*s*,? *e*]	The query of retrieving all foaf:Persons that were valid (i.e. alive) in 1995 can be expressed as follows. SELECT? person FROM SNAPSHOT 1995 WHERE{?person *a* foaf:Person.}
QaRDF (Udrea et al. 2010)	Presenting a query language for the *Annotated* RDF (aRDF) model in Table 4, including *simple queries* and *conjunctive queries*.	A *simple* aRDF query is a triple (*R*, *P*: *A*, *V*) where *R*, *P*, *A*, *V* are all terms over resources, properties, annotations and values respectively. A *conjunctive* query is set of simple aRDF queries that have common variables.	For example, the query "What committees was *people/B000711* a member of between 1997 and 2001?" can be expressed as an atomic aRDF query: (*people/B000711*, *member*:[1997, 2001],? *v*)
QT-YAGO (Wang et al. 2010)	Giving a simple introduction about how to make queries for the T-YAGO model as mentioned in Table 3.	Several new time predicates such as *before, after, equal, during, overlaps,* and *sameYear* are defined for representing the *on*, *since*,	

Table 5 (continued)

	Purpose	*Query language form*	*Example*
		and *until* relations in the T-YAGO model. Each predicate takes time points or periods as input and returns a Boolean value.	A query "teammates of Beckham who played for the same club as Beckham during overlapping periods" can be expressed as: ?id1: "David Beckham" *playsForClub?* x. ?id2:? a *playsForClub?* x. ?id1 *since?* t1.? id1 *until*? t2. ?id2 *since?* t3.? id2 *until?* t4. [?t1-?t2] *overlaps* [?t3-?t4]. ?a *notEqual* "David Beckham".
C-SPARQL (Barbieri et al. 2009; 2010a; 2010b)	Presenting a *continuous* query language C-SPARQL for querying the RDF *data streams* model (<*s*, *p*, *o*>, τ) as mentioned in Table 3.	Some new elements are introduced into C-SPARQL for representing RDF *streams*, windows, *timestamps*, and language atoms. Moreover, a timestamp function is defined which can retrieve and bound the timestamp of a stream element to a variable. The details can be found in the subsequent text of our review.	A detailed query example can be found in the subsequent text of our review, and thus omitted here for the limitation of the table.
AnQL (Zimmermann et al. 2012)	Presenting an extension query language AnQL for querying the *annotated* RDF model τ: λ as mentioned in Table 3.	An AnQL query is a tuple Q=(P, G, V, A) where P is a graph pattern, G is an annotated RDF graph, V is the result form, A is the set of annotation variables.	The query of looking for *Ebay* employees during some time period and that optionally owned a car can be posed: SELECT? *p*? *l*? *c* WHERE { (?*p* type ebayEmp):?*l* OPTIONAL{(?*p* hasCar? *c*):?*l*}}
QRDF+ (Dividino et al. 2009; Schueler et al. 2008)	Presenting a small extension to standard SPARQL and then defining how SPARQL can be applied to an RDF+ model as mentioned in Table 3.	An additional expression "WITH META *MetaList*" is introduced into SPARQL syntax. On this basis, two querying forms are added into the SPARQL extension queries: SPARQL SELECT query and SPARQL CONSTRUCT query.	A SELECT query on the example of the RDF+ model in Section 3.2 is expressed as: SELECT? x WITH META G_3, G_4 FROM G_1 FROM G_2 WHERE { {GRAPH? h1 {?x affiliatedWith? y}}AND {GRAPH? h2 {?x researchTopic 'SW'}}}
QValidity TimeRDF (Motik 2012)	Presenting an extension of SPARQL to query temporal RDF< *s*, *p*, *o*>[*t*] or<*s*, *p*, *o*>[t_1, t_2] as mentioned in Table 3.	The *maximality* and *minimality* notions are introduced into the extension query language, which can support the query such as "the maximal interval in which some triple holds" or "the minimal time time at which some triple holds".	A query "retrieving events in London having at least one time point in common with Chinese Teachers' Day" is expressed as: SELECT? events WHERE { {(:China,: hosts,: TeachersDay)} *maxint* [?t_1,?t_2]. (:London,: hosts,? events)} *occurs* [?t_1,?t_2].} where *maxint* and *occurs* are the elements of a temporal graph pattern.
QTKB (Bykau et al. 2012)	Presenting a *navigational* query language for querying the RDF model with temporal features and evolution operators as mentioned in Table 3.	The query language extends the navigational language nSPARQL (Perez et al. 2010) (a SPARQL extension language based on nested regular expressions) and re-define five evolution axes *join*, *split*, *merge*, *detach*, and *becomes* based on the nSPARQL axes *self*, *next*, *edge*, and *node*. A detailed grammar and formal semantics can be found in (Bykau et al. 2012).	A query "who was the head of the German government before and after the unification of 1990?" can be expressed as: SELECT? *Y* (?*X*, *self*::Reunified_Germany /*join*$^{-1}$[1990]/*next*::head[1990],? *Y*) AND (?*Z*, *self*::Reunified_Germany/ *next*::head[1990],? *Y*)
st SPARQL (Bereta et al. 2013)	Presenting a language called stSPARQL for querying the valid time of triples in linked geospatial data in the stRDF model as mentioned in Table 3.	stSPARQL is an extension of SPARQL 1.1, and adds several new features, including temporal triple patterns, temporal extension functions, and temporal constants. It also allows to define the temporal extension functions in SELECT, FILTER, and HAVING clause of a query.	A query for "retrieving the current land cover of each area?" can be expressed: SELECT? Area? *c* WHERE { ?Area rdf:type: Area. ?Area: hasLandCover? *c*? *t*. FILTER(strdf:during(NOW,? *t*))}

Table 5 (continued)

	Purpose	*Query language form*	*Example*
LSPARQL (Bellamy-McIntyre 2018)	Presenting a query language LSPARQL that allows for temporal queries over *different source code versions*.	A LSPARQL query is a quintuple form (t, c, s, p, o), where t is a timestamp, and c is a change type. The valid change types are +, to denote an added triple, − to denote a deleted triple, e which means 'exists' and which is used for queries on a specific snapshot.	A query about the triple (Bob, likes, Alice) which is valid in the interval (1, 4) may be expressed as: (?*t*, *e*,? *x*, likes,? *y*)
TSPARQL toTSQL2 (Guo et al. 2018)	Proposing an approach to *translate* the temporal SPARQL T-SPARQL (Grandi 2010) into temporal database query language TSQL2.	They considered the basic graph pattern, group graph pattern, multi-graph pattern, and optional graph pattern of T-SPARQL (Grandi 2010).	A temporal query is expressed as follows: SELECT ? *x*, ? *date* FROM <http://example.org/foaf/congressFoaf> WHERE{{?*x* foaf:name ? *n*} OPTIONAL{?*x* foaf:type congressman}:?*date*}
SPARQL^T (Gao et al. 2016) (Zaniolo et al. 2018)	Proposing a temporal extension of SPARQL called SPARQL^T based on the point-based temporal model (Gutierrez et al. 2007).	A SPARQL^T query is a quad {s, p, o, t}. It also supports basic operations such as *select*, *join*, *aggregate*, *negation*, *duration*, and some temporal *constraints*. Here, the *negation* is expressed by introducing several new constructs such as *attime*(?*t*) and *notattime*(?*t*). The *duration* is supported by a built-in function *LENGTH*.	A query: "retrieving the total population of Beijing in 2019" can be expressed as follows: SELECT? *popul* WHERE { :Beijing: totalPopul? *popul*? *t* . FILTER (? *t*=2019)}

Temp SPARQL (Kalayci et al. 2019)	Proposing a query language that is an extension of SPARQL similar to the τ-SPARQL (Robertson 2004).	A temporal predicate in query is followed by a suffix @ 〈?e_1,? v_1,? v_2,? e_2〉 , where? e_1 is a Boolean variable, depending on whether the interval where the predicate holds is left-closed or left-open (and? e_2 indicating right-closedness or right-openness), while? v_1 and? v_2 are variables over date/time whose values respectively indicate from when and until when the predicate holds.	A query "finding the periods of time when the main flame was on for some sensor" can be expressed as follows:
			SELECT? *l*? *begin*? *end*? *r* WHERE {
			{?*ts* rdf:type
			:MainFlameOn}@ 〈?*l*,? *begin*,? *end*,? *r*〉
			}

iSPARQL (Wudage Chekol et al. 2019)	Proposing a lightweight temporal extension of SPARQL called iSPARQL, which leverages algorithms for efficient computation of interval joins and coalescing.	An iSPARQL query is a query of the form: SELECT V WHERE {QP}, where the syntax of the query patterns (QP) is based on a SPARQL query language called SPARQL*.	A query "Select regions containing Bazoncourt before R. Poincare came to power" can be expressed as follows: SELECT? *x* WHERE { (Bazoncourt locatedin? *x*)? *s*? *e*. (?*x* locatedin? *y*)? *s1*? *e1*. (?*z* containsTerritory? *y*)? *s2*? *e2*. (?*z* headOfState RPoincare)? *s3*? *e3*. FILTER (OVERLAPS(?*s*,? *e*,? *s1*,? *e1*) && OVERLAPS(?*s1*,? *e1*,? *s2*,? *e2*) && BEFORE(?*s*,? *e*,? *s3*,? *e3*))}

Table 4. In brief, a tRDF query is a conjunctive SPARQL query that is augmented with temporal annotations on the edges (either variable or constant). They refer to each edge in the query graph pattern as a query atom. Formally, a tRDF query is a tuple (N, E, V, λ_n, λ_t), where:

- N is a set of vertices.
- V is a set of variables.
- $E \subseteq N \times N \times (V \cup P)$ is a set of edges, where P is a set of properties in RDF triples (*subject*, *property*, *object*).
- λ_n: $N \rightarrow R \cup V$ is a vertex labeling function, where $R = U \cup L$ is the set of resources from URI references U and literals L.
- λ_t is an edge labeling function that associates with every edge in E, an expression of the form $\{T\}$, $<n: T>$ or $[n: T]$, where T is either a constant time interval or a variable and n is a natural number.

For example, a simple query triple in the tRDF query tuple can be represented as? v_1 *supported*: <1: 2000–2005 > congress/107/bills/sj37.

Further, they present a tGRIN index structure that builds a specialized index for temporal RDF that is physically stored in a relational database management system (RDBMS). They state that the index structure is shown to be a better match for tRDF queries than the use of standard indexes for temporal data.

TA-SPARQL (Rodríguez et al. 2009)

Rodríguez et al., (2009) present a query language Time-annotated SPARQL (TA-SPARQL) for their TA-RDF model as mentioned in Table 4. TA-SPARQL is designed to stay as similar as possible to its preexisting counterpart in syntax and semantics while introducing the capabilities required to easily express the one-time time-dependent queries common when handling streams and time-varying data. They present the language using several examples before listing all the constructs that are new in TA-SPARQL vs SPARQL. They give several TA-SPARQL query forms (here we take two forms for example):

- {… *resource*[t_0] *predicate object* …}, which specifies resource in a triple by timestamp
- {… *resource*[t_0..t_1] *predicate object* …}, which specifies resource in a triple by range

Further, a query: "What was the humidity on Shenyang at 2019-06-28?" can be expressed as:

```
SELECT ?humidity
WHERE {
<urn:Shenyang > <urn:hasHumiditySensor> ?v["2019-06-28"^^xsd:dateTime].
?v <urn:hasReading> ?humidity. }
```

Moreover, TA-SPARQL also supports the aggregation for functions SUM, AVG, MIN, MAX and COUNT. The semantics of TA-SPARQL queries are informally described by providing translations of TA-SPARQL queries into SPARQL queries.

QTempRDF (McBride and Butler 2009)

McBride & Butler (2009) use SPARQL for querying the temporal RDF model (*s*, *p*: (*begin–end*), *o*) as mentioned in Table 4. They only briefly provide several querying examples instead of giving the querying language in detail. The queries are described by using a syntax that allows predicates in triple patterns to specify a begin time and an end time. For example, a query "Who worked in HPLabs in 2008" can be described as:

```
DESCRIBE ?labbie
WHERE {
?labbie f:memberOf:(2008-01-01--2009-01-01) f:HPLabs. }
```

where *f:memberOf*:(2008-01-01–2009-01-01) is shorthand for a URI that identifies a temporal property as introduced in Temporal RDF models based on the original form of RDF triple by adding timestamp section. Therefore, the query of a triple pattern can be done in the normal manner using SPARQL.

QMulti-tempRDF (Grandi 2011; 2009; 2010)

Grandi (2011; 2009; 2010) present a temporal query language T-SPARQL for the Multi-temporal RDF model (s, p, $o \mid T$) as mentioned in Table 3. T-SPARQL is designed based on the temporal relational database query language TSQL2 (Snodgrass 1995) and an extended set of the SPARQL temporal datatypes, functions and operators.

T-SPARQL adds the temporal selection capabilities to SPARQL by extending the SPARQL WHERE clause in the SELECT statement. That is, a graph pattern of the WHERE clause in a T-SPARQL query will be extended with an optional position. They introduce the syntax and semantics of T-SPARQL through several example queries and don't provide the detailed formal syntax and semantics of T-SPARQL language. For example, a query "What are the names of the employees who have works in the Toys department longer than Ann has made $20,000" can be represented as:

```
Select ?ename WHERE{
  ?emp1 rdf:type ex:emp ;
        ex:Name "Ann" ;
        ex:Salary ?salary | ?ts .
  ?emp2 rdf:type ex:emp;
        ex:Name ?ename;
        ex:Dept "Toys" | ?tt .
  FILTER (?salary > 20000 && xs:duration(VALID(?tt)) > xs:duration(VALLID(?ts))) . }
```

where VALID(T) expresses conditions on the valid time component of T, *xs*:*duration* is the XML Schema primitive datatype corresponding to the base temporal type *interval*.

τ-SPARQL (Tappolet and Bernstein 2009)

Tappolet & Bernstein (2009) present a temporal query language τ-SPARQL for querying the temporal RDF model NG-tempRDF based on named graph as mentioned in Table 3. They consider two kinds of τ-SPARQL query forms: time point queries and temporal queries. The former retrieves information valid at a time point, and the latter retrieves information valid at a time interval. Regarding to the time point, τ-SPARQL extends a FROM statement in SPARQL to a FROM SNAPSHOT τ expression, which will match the graph patterns valid at the time point τ. For example, the query of retrieving all foaf:Persons that were valid (i.e. alive) in 2001 can be expressed as follows.

```
SELECT ?x FROM SNAPSHOT 2001 WHERE{
?x rdf:type foaf:Person. }
```

Similarly, the temporal queries (i.e., the interval queries) are defined as shown in the following example, which retrieves a person and his birthday.

```
SELECT ?s ?x WHERE{
[?s, ?e] ?x rdf:type foaf:Person.     // or ?x rdf:type foaf:Person [?s, ?e]. }
```

In addition, they propose some rules for transforming τ-SPARQL queries into SPARQL queries, which denote that τ-SPARQL can be reconstructed without any loss of expressivity to SPARQL. They also propose a specific index structure called keyTree index for temporal intervals which improves the retrieval time of time point queries.

QaRDF (Udrea et al. 2010)

Udrea et al., (2010) presents a query language for the Annotated RDF (aRDF) model as mentioned in Table 4, including simple queries and conjunctive queries.

Regarding to the simple queries, that is, the annotated triples in which any of the subject, property, value or annotation can be either constant or variable. Given the sets of variables ranging over resources R, properties P, values V, and annotation A as introduced in Temporal RDF models based on the original form of RDF triple by adding timestamp section, a *simple query* is a triple (R, P: A, V). In particular, a query is atomic if at most one term in the above query triple is a variable. For example, an atomic aRDF query "What committees was *person/A03* a member of between 2002 and 2005?" can be represented as "(*person/A03*, *member*:[2002, 2005],? *v*). Further, a *conjunctive query Q* is a set of simple aRDF queries that have common variables.

They present some algorithms to answer the simple and conjunctive queries. They first discuss the atomic queries and then investigate how the atomic query algorithms can be generalized to simple queries. They also provide two approaches for the conjunctive queries. Finally, they implement a prototype and make some evaluations on several datasets.

QT-YAGO (Wang et al. 2010)

Wang et al., (2010) give a simple introduction about how to make queries for the T-YAGO model as mentioned in Table 3. For representing the *on*, *since*, and *until* relations in the T-YAGO model, they define several new time predicates such as *before, after, equal, during, overlaps,* and *sameYear*. Each predicate can take time points or periods as input and return a Boolean value.

For example, a query "who played for the same club as the soccer player Messi during overlapping periods" can be expressed as follows.

```
?id1: "Messi" pfClub ?c.
?id2: ?player pfClub ?c.
?id1 since ?t1. ?id1 until ?t2.
?id2 since ?t3. ?id2 until ?t4.
[?t1-?t2] overlaps [?t3-?t4].      // [?t1-?t2] is a time interval
?player notEqual "Messi".
```

More recently, a Question Answering system using Graph-Pattern Association Rules (called QAGPAR) on YAGO knowledge base is developed in (Wahyudi et al. 2018), where the answer as output of the system is provided based on a user question as input. If the answer is missing or unavailable in the database, then graph-pattern association rules are used to get the answer.

C-SPARQL (Barbieri et al. 2009; 2010a; 2010b)

Barbieri et al., (2009; 2010a; 2010b) present a continuous query language C-SPARQL for querying the RDF data streams model (<s, p, o>, τ) as mentioned in Table 3. C-SPARQL extends SPARQL by adding support for window and aggregation operations and adds some new elements for representing RDF streams, windows, timestamps, and language atoms for aggregating stream information. The following example can directly illustrate the C-SPARQL query form.

For example, in a city, we want to count the number of cars of entering the city center through the tollgates. We know that each tollgate will register each entered car. In this case, the query "how many cars has been registering to each tollgate in the last 30 minutes" can be expressed as follows.

```
REGISTER QUERY NumOfCarsTollgate
        COMPUTED EVERY 1m AS. //  The window is adjusted every minute.
SELECT DISTINCT ?tollgate ?passages
FROM STREAM <RDF streams dataset>
        [RANGE 30m STEP 1m]
WHERE {?tollgate :registers ?car.} AGGREGATE {(?passages, COUNT, {?tollgate})}
```

Please refer to (Barbieri et al. 2010a) for the detailed formal semantics of the new elements in C-SPARQL. Similarly, the other several studies (Anicic et al. 2011; Le-Phuoc et al. 2011) also discuss the querying of Event Processing and Stream Reasoning, but they don't focus on the temporal RDF query and thus are not introduced in detail in our current review.

AnQL (Zimmermann et al. 2012)

Zimmermann et al., (2012) present an extension query language AnQL for querying the annotated RDF model τ: λ as mentioned in Table 3. In AnQL, triples in a query are replaced with annotated triples in which annotation variables. Formally, an AnQL query is a tuple $Q = (P, G, V, A)$ where P is a graph pattern, G is an annotated RDF graph, V is the result form, A is the set of annotation variables.

For example, a query: "who are the employees of the company C at the time interval and that optionally owned a house during the interval" can be represented as follows.

SELECT ?*person* ?*t* ?*h* WHERE {
(?*person* rdf:type CEmp): ?*t*
OPTIONAL {(?*person* :hasHouse ?*h*): ?*t* }}

Moreover, they describe the implementation of AnQL based on constraint logic programming techniques along with a practical experiment for representing sensor data as Annotated RDF. In addition, AnQL considers the features of SPARQL 1.1 including subqueries, aggregates, assignment, and solution modifiers.

QRDF+ (Dividino et al. 2009; Schueler et al. 2008)

Dividino et al., (2009); Schueler et al., (2008) introduce a small extension to standard SPARQL and then define how SPARQL can be applied to an RDF+ model as mentioned in Table 3. The main extension is that one additional expression "WITH META *MetaList*" is introduced into SPARQL syntax. This expression includes the named graphs specified in *MetaList* for treatment as meta knowledge (such as timestamp, source, and (un)certainty). It should be noted that this statement is optional. If there is an expression, then the SPARQL processor may digest the RDF+ meta knowledge statements derivable from the RDF named graphs appearing in the *MetaList*. The SPARQL processor will then use this meta knowledge to compute and output all the meta knowledge statements derivable by successful matches of RDF+ literal statements with the WHERE pattern.

In detail, one of the following two forms may be in the SPARQL extension queries: SPARQL SELECT query and SPARQL CONSTRUCT query. The former has the form:

SELECT SelectExpression
(WITH META *MetaList*)?
(FROM GraphName)+
WHERE *P*

And the latter has the form:

CONSTRUCT ConstructExpression
(WITH META *MetaList*)?
(FROM GraphName)+
WHERE *P*

Where P refers to a graph pattern that explains how RDF+ literal statements from named graphs specified using FROM statements are matched. Matches bind variables that are used to provide results based on the SelectExpression or the ConstructExpression.

QValidityTimeRDF (Motik 2012)

Motik (2012) present an extension of SPARQL to query temporal RDF $< s, p, o > [t]$ or $< s, p, o > [t_1, t_2]$ as mentioned in Table 3. They introduce the maximality and minimality notions into the extension query language, which can support the query such as "the maximal interval in which some triple holds" or "the minimal time time at which some triple holds.". For example, a query "retrieving all events in London having at least one time point in common with Chinese Teachers' Day" can be represented as follows.

SELECT? events WHERE {
{(:China,: hosts,: TeachersDay)} maxint [?t_1,?t_2].
(:London,: hosts,? events)} occurs [?t_1,?t_2].}

where (:China,: hosts,: TeachersDay) is a basic graph pattern, and *maxint* and *occurs* are the elements of a temporal graph pattern.

In detail, several types of questions that the language supports are listed in their work such as:

Q_1: Is B true in G at instant t?
Q_2: Is B true in G at all instants between t_1 and t_2?
Q_3: Is B true in G at some instant between t_1 and t_2?
Q_4. Is $[t_1, t_2]$ the maximal interval such that B is true in G for each time instant in the interval?
Q_5. Is t the smallest/largest time instant at which B is true in G?

where B is a basic graph pattern $<s, p, o>$, G is a temporal graph, and t, t_1, and t_2 are time instants.

Moreover, an algorithm that can be used with the entailment relations is presented. Further, two optimization techniques of the algorithm along with a set of deterministic rules are presented for deciding the Entailment.

QTKB (Bykau et al. 2012)

Bykau et al., (2012) present a navigational query language for querying the RDF model with temporal features and evolution operators as mentioned in Table 3, where they traverse temporal and evolution edges in an evolution graph. They extend the navigational language nSPARQL (Perez et al. 2010) (a SPARQL extension language based on nested regular

expressions) and re-define five evolution axes *join*, *split*, *merge*, *detach*, and *becomes* based on the nSPARQL axes *self*, *next*, *edge*, and *node*. A detailed grammar and formal semantics can be found in (Bykau et al. 2012). Here, the following example can directly explain the query language.

For example, a query "who was the head of the German government before and after the unification of 1990?" can be expressed as follows:

SELECT? *Y*.
(?*X*, *self*::Reunified_Germany/*join*$^{-1}$[1990]/*next*::-head[1990],? *Y*) AND
(?*Z*, *self*::Reunified_Germany/*next*::head[1990],? *Y*)

where the first triple finds all the heads of state of the Reunified Germany before the unification by following *join*$^{-1}$[1990] and then following *next*::head[1990]. The second triple finds the heads of state of the Reunified Germany. Finally, the join on? *Y* will bind the variable only to those heads of state that are the same in both triples, hence returning the one before and after the mentioned unification.

Moerover, a system called TrenDS has been implemented, and more case studies can be found in (Bykau et al. 2012).

stSPARQL (Bereta et al. 2013)

Bereta et al., (2013) present a language called stSPARQL for querying the valid time of triples in linked geospatial data in the stRDF model as mentioned in Table 3. stSPARQL is an extension of SPARQL 1.1, and adds several new features, including temporal triple patterns, temporal extension functions, and temporal constants. A temporal triple pattern is a tuple (*s*, *p*, *o*, *t*) as mentioned in the stRDF model in Table 3, where *t* is a time period or a variable. Nine temporal extension functions are defined to express temporal relations. Two temporal constants NOW and UC (Until Changed) are allowed in a query in order to query triples whose valid time has not ended or we do not know when it ends. In particular, stSPARQL allows to define the temporal extension functions in SELECT, FILTER, and HAVING clause of a query.

For example, a temporal selection query for "retrieving the current land cover of each area?" can be expressed as follows:

SELECT? Area? *c* WHERE {
?Area rdf:type: Area.
?Area: hasLandCover? *c*? *t*.
FILTER(strdf:during(NOW,? *t*))}

where the FILTER function represents the valid time of the triple pattern.

LSPARQL (Bellamy-McIntyre 2018)

Bellamy-McIntyre (2018) propose a method to use RDF for modeling versioned source code. They first parse all the classes of a project into abstract syntax trees, and then generate an abstract semantic graph represented as a set of RDF triples. They also develop a triplestore that maintains an index of all changes made to the store. Further, they present a query language LSPARQL that allows for temporal queries using that index and which can be applied to typical triplestore transaction logs.

Formally, given the quintuple representation of triples of the form (*t*, *c*, *s*, *p*, *o*), where *t* is a timestamp, and *c* is a change type. For LSPARQL queries, the valid change types are +, to denote an added triple, − to denote a deleted triple, *e* which means 'exists' and which is used for queries on a specific snapshot. The temporal variable? *t* is assigned a single timestamp denoting a point in time in the case of patterns with change types +, −, and + −, and a pair of timestamps denoting a valid interval in the case of *e* patterns. Further, a temporal join query with two single timepoints requires them to be the same time point, and a temporal join query on two intervals will get the intersection of those intervals. For example, given the triple (Bob, likes, Alice) valid in the interval (1, 4), and (Alice, likes, Bob) valid in the interval (2, 5), the LSPARQL query (?*t*, *e*,? *x*, likes,? *y*), (?*t*, *e*,? *y*, likes,? *x*) then the corresponding mapping μ would give $\mu(?t) = (2, 4)$.

Currently, LSPARQL is limited to conjunctive queries with FILTER. They also develop a prototype focusing on Java source code to show that the approach can be practical.

TSPARQLtoTSQL2 (Guo et al. 2018)

Guo et al., (2018) propose an approach to translate the temporal SPARQL T-SPARQL (Grandi 2010) into TSQL2 (a standard query language for temporal relational database management system RDBMS). They realize the functions from different graph patterns in T-SPARQL (including basic graph pattern, group graph pattern, multi-graph pattern, and optional graph pattern) to TSQL2. For example, a temporal query can be expressed as follows:

SELECT? *x*,? *date*
FROM <http://example.org/foaf/congressFoaf>
WHERE{{?*x* foaf:name? *n*}
OPTIONAL{?*x* foaf:type congressman}:?*date*}

SPARQL^{T} (Gao et al. 2016; Zaniolo et al. 2018)

Gao et al., (2016); Zaniolo et al., (2018) propose a temporal extension of SPARQL called SPARQL^{T}. It is based on the point-based temporal model (Gutierrez et al. 2007) as

mentioned in Table 3 and inherits the SPARQL syntax and extends the SPARQL query pattern with one temporal construct.

Formally, the syntax structure of SPARQL^T is a quad $\{s, p, o, t\}$ from $(\mathcal{U}\cup\mathcal{L}\cup\mathcal{V}) \times(\mathcal{U}\cup\mathcal{L}) \times(\mathcal{U}\cup\mathcal{L}\cup\mathcal{V}) \times(\mathcal{T}\cup\mathcal{V})$ where $\mathcal{T}$, $\mathcal{U}$, $\mathcal{L}$, $\mathcal{V}$, is a set of Timestamps, Uniform Resource Identifiers, Literals, and Variables. SPARQL^T also supports basic operations such as *select*, *join*, *aggregate*, *negation*, *duration*, and some temporal *constraints*. The *select*, *join*, *aggregate*, and *constraints* are similar to the SPARQL. The *negation* is expressed by introducing several new constructs such as *attime*(?*t*) and *notattime*(?*t*). The *duration* is supported by a built-in function *LENGTH*.

For example, a query: "retrieving the total population of Beijing in 2019" can be expressed as follows:

SELECT? *popul* WHERE {
:Beijing: totalPopul? *popul*? *t*. FILTER (? *t* = 2019)}

Moreover, they describe an efficient system called RDF-TX for managing and querying temporal RDF data. The multiversion B+ tree is used in the system to store and index temporal RDF data. They also introduce a delta encoding scheme to reduce the storage overhead of indices. Finally, the history of the Wikipedia infoboxes for the last 13 years are collected and a historical knowledge base called Cliopedia is formed.

Similarly, Song (2015) also constructs a Wikipedia Infobox Temporal RDF Knowledge Base which includes 170,941,613 temporal triples. The author further explores the performances of Postgres SQL B-Tree, MySQL B + Tree, and Interval Tree to support temporal RDF queries.

SPARQL[t] (Zhang et al. 2019)

Zhang et al., (2019) propose a temporal data query language SPARQL[t] by extending RDF standard query language SPARQL for querying the time and update count information in RDFt model as mentioned in Temporal RDF models based on the original form of RDF triple by adding timestamp section.

Being similar to SPARQL, the query form of SPARQL[t] includes the declaration, result set, dataset, graph pattern, and result modifier. The basic graph pattern can be a query triple in the form of (*s*, *p*, *o*), (*s*, *p*[*t*]-*n*, *o*), or (*s*, *p*[*ts*, *te*]-*n*, *o*). For example, a query: "retrieving the time and score when John played for the club *A* at his third career" can be expressed as follows:

SELECT? *ts*? *te*? *score*
WHERE {
?SportName: Plays_For[?*ts*,? *te*]-?*n* "Club_A" .
?SportName: hasScore? *score* .
FILTER? SportName = "John" and? *n* = 3}

Moreover, to achieve compatibility with the existing SPARQL query engines, they also give the query transformation algorithm from SPARQL[t] to SPARQL. Finally, a prototype system is implemented for representing and querying the temporal data.

TempSPARQL (Kalayci et al. 2019)

Kalayci et al., (2019) propose a temporal ontology-based data access (OBDA) framework to support access to temporal data and reasoning over it. In order to retrieve temporal information within the framework, they propose a query language that is an extension of SPARQL similar to the τ-SPARQL (Tappolet and Bernstein 2009).

Formally, temporal predicates have to be followed by a suffix @ $\langle ?e_1,? v_1,? v_2,? e_2\rangle$, where? e_1 is a Boolean variable evaluating to either 'true' or 'false', depending on whether the interval where the predicate holds is left-closed or left-open (and similarly for? e_2 indicating right-closedness or right-openness), while? v_1 and? v_2 are variables over date/time whose values respectively indicate from when and until when the predicate holds.

For example, a temporal query "finding the periods of time when the main flame was on for some sensor" can be expressed as follows:

SELECT ?*l* ?*begin* ?*end* ?*r*
WHERE {
{?*ts* rdf:type :MainFlameOn}@⟨?*l*, ?*begin*, ?*end*, ?*r*⟩
}

The framework supports both the τ-SPARQL-based language and the plain SPARQL. The τ-SPARQL-based language is treated as syntactic sugar and handled by compiling it into the corresponding plain SPARQL query language.

iSPARQL (Wudage Chekol et al. 2019)

Wudage Chekol et al., (2019) present efficient algorithms to compute interval joins for the main Allen's relations (e.g., before, after, during, meets) (Allen 1983). They also address the problem of interval coalescing, which is used for merging contiguous or overlapping intervals of temporal facts, and propose an efficient algorithm. Further, they integrate the interval joins and coalescing algorithms into a light SPARQL extension called iSPARQL, which allows to write queries in a more succinct way than SPARQL.

Formally, let *I* and *L* be disjoint infinite sets denoting the set of IRIs (identifying resources) and literals (character strings or some other type of data), respectively. The union of these sets ($I \cup L$) is abbreviated as *IL*. A discrete time domain *T* is considered as a linearly ordered finite sequence

of time points, e.g., days, minutes, or milliseconds. Let V be a set of variables. An iSPARQL query is a query of the form:

> SELECT V WHERE $\{QP\}$, where the syntax of query patterns (QP) is as follows:
> $QP ::= IV \times IV \times ILV \times TV \times TV \mid QP_1$ AND $QP_2 \mid \{QP_1\}$ UNION $\{QP_2\} \mid \{QP_1\}$ MINUS $\{QP_2\} \mid QP_1$ OPT $\{QP_2\} \mid QP$ FILTER (R)

For example, a query "Select regions containing Bazoncourt before R. Poincare came to power" can be expressed as follows:

> SELECT? x WHERE {
> (Bazoncourt locatedin? x)? s? e.
> (?x locatedin? y)? $s1$? $e1$.
> (?z containsTerritory? y)? $s2$? $e2$.
> (?z headOfState RPoincare)? $s3$? $e3$.
> FILTER (OVERLAPS(?s,? e,? $s1$,? $e1$) && OVERLAPS(?$s1$,? $e1$,? $s2$,? $e2$) && BEFORE(?s,? e,? $s3$,? $e3$))}

They also carry out a number of experiments to showcase the performance of the proposed algorithms, and the results show that iSPARQL is a very good alternative for querying large temporal knowledge graphs.

TempClusterQ (Huang et al. 2020)

Huang et al., (2020) explore cluster queries in temporal knowledge graph. Comparing to the snapshot queries, which can only give separate answers (e.g., assuming there are two queries, one is who was the president of the United States in 2002, the other is who was the president in 2010. The answers are George W. Bush and Barack Obama, respectively.). But if one wants to query who are the presidents between 2002 and 2010, the snapshot queries can not answer the query. In this case, the cluster query can do it well.

Formally, a cluster query on the temporal knowledge graph $G = <V, L_V, E, L_E, T_E>$ (see Temporal RDF models based on new RDF extension syntaxes section) is defined on the basis of a query graph. A query graph is a five-tuple $Q = <V^Q, L_V^Q, E^Q, L_E^Q, [t_s, t_e]>$, where $V = V_c^Q \cup V_e^Q \cup V_l^Q \cup V_p^Q$ is a collection of vertices including all subjects and objects, and V_c^Q, V_e^Q, and V_l^Q are collections of class vertices, entity vertices, and literal vertices, respectively, V_p^Q is a collection of parameter vertices; L_V^Q is a collection of vertex labels. If $v \in V_p^Q$, the label of v is $\varnothing$; $E^Q = E_k^Q \cup E_p^Q$ is a collection of edges, where E_k^Q is a collection of edges that correspond to the known predicates in query graph, and E_p^Q is a collection of parameter edges; L_E^Q is a collection of edge labels. If the label of edge is $\varnothing$, it means the edge is parameter edge; $[t_s, t_e]$ is the time frame of the query graph, where t_s is start time and t_e is end time. On this basis, given a temporal knowledge graph G and a query graph Q, a cluster query is to retrieve the set of time frame matches of Q, which can answer a question with both facts and expired facts over the period of time $[t_s, t_e]$. The corresponding cluster query algorithms are proposed.

For further accelerating the query processing, two indexing techniqus, named Temporal Vertex tree (TV-tree) for indexing vertices and Temporal Predicate hash (TP-hash) for indexing the temporal predicates of the edges which link two entity vertices or class vertices, are developed. Finally, some experiments are conducted on three real datasets (i.e., Soccer Player, Twitter, and DBLP), and the results demonstrate the effectiveness and the outstanding performance of the proposed cluster query algorithms.

Temporal RDF storage techniques

Over the years, the temporal extension models based on RDF are increasingly used in many applications (e.g., Linked Data, Data Stream Management Systems, and the Semantic Web) as mentioned in the previous sections. Accordingly, some temporal RDF datasets have been created and these datasets tend to become very large to huge (e.g., the Cliopedia Historical Knowledge Base (Gao et al. 2016; Zaniolo et al. 2018) and YAGO (Hoffart et al. 2013)). Therefore, the efficient storage of temporal RDF data is considered that has arisen from practical needs.

In this section we further summarize and discuss the temporal RDF storage techniques. Table 6 first summarizes the temporal RDF storage techniques from the aspects of the data format of the stored temporal RDF model, the storage medium, the index technique, and the main idea. Then, each technique will be further discussed in detail in the subsequently text.

In general, there may be several techniques to store temporal RDF datasets. The one is to use the well-known relational, object or object-relational databases. In particular, on the basis of the widespread use and mature techniques of relational databases, lots of works have pointed out the suitability of storing temporal RDF datasets in the existing relational database management systems. The another one is considered for storing temporal RDF datasets by some special systems, such as main-memory or RDF engines (e.g., Jena (Carroll et al. 2004)).

Storage of MRDF model (Gergatsoulis and Lilis 2005)

Gergatsoulis & Lilis (2005) present a technique for storing the MRDF (Multidimensional RDF) model as mentioned in Table 4 into *relational database management system*

Table 6 The existing Storage techniques for temporal RDF models

	The data format of the temporal RDF models	***Storage techniques***		
		Storage medium	*Index*	*Main idea*
Storage of MRDF model (Gergatsoulis and Lilis 2005)	(*s p o*), where *o* is the form of [(d_1, v_1), …, (d_n, v_n)] as mentioned in Table 4.	Relational Database Management System		The temporal RDF MRDF data is stored in *Relational Database Management System RDBMS*. Two Relational Database Schema *statement* and *context* is designed to store the statement and the context triples respectively. The details can be found in the subsequent text of our review.
Storage of tRDF model (Pugliese et al. 2008)	(*subject*, *property*: *annotation*, *object*) as mentioned in Table 4.	Relational Database Management System	temporal Graph-based RDF INdex (tGRIN)	The tRDF data is stored in a single relation (*subject*, *property*, *object*, *annotation*) in a PostgreSQL 8.0 relational database. The tGRIN index is stored separately on disk.
Storage of TempRDF model (McBride and Butler 2009)	(*s*, *p*: (*begin--end*), *o*) as mentioned in Table 4.	Disk		The prototype is developed using the Jena (Carroll et al. 2004) and TDB (Owens et al. 2009) for persistent storage of RDF graphs. They extend the Jena API with a new API to support the representation of temporally varying information.
Storage of stRDF (Bereta et al. 2013)	(*s*, *p*, *o*, *t*) as mentioned in Table 3.	Relational Database Management System	GiST index	A RDBMS technique is developed to store their stRDF data in the system Strabon, which is a semantic geospatial database management system and has been implemented by extending Sesame and using an RDBMS as a backend. The stRDF data (*s*, *p*, *o*, *t*) is stored in Strabon in the form of an N-Quads document.
Storage of PDStore (Bellamy-McIntyre 2018)	(*t*, *c*, *s*, *p*, *o*) where *t* is a timestamp and *c* is a change type that is either an add change + or a removal change –.	Memory	hash based index	PDStore which uses a hash based indexing scheme, and its key feature is that by default it indexes all changes that are made in the transaction log so that queries can be run against any historic state. Moreover, a pair of hash based indexes is defined.
Storage of SPARQL^T model (Gao et al. 2016; Zaniolo et al. 2018)	<*s*, *p*, *o*>[*t*] or <*s*, *p*, *o*> [t_s, t_e] as mentioned in (Gutierrez et al. 2007) in Table 3.	Main-memory	Multiversion B+tree (MVBT) index	The point-based temporal RDF triples (Gutierrez et al. 2007) is stored in the main-memory, and the multiversion B+tree (MVBT) and some data compression techniques are used to store and index the temporal data.

(RDBMS). In detail, the following relational database schema is designed for storing the temporal RDF data:

- *Statement* (*Subject*, *Predicate*, *MultidimensionalNode*), where *Subject* is the subject of the triple, *Predicate* is the predicate and *MultidimensionalNode* is the multidimensional node identifier.
- *Context* (*MultidimensionalNode*, *Object*, *S*, *E*), where *MultidimensionalNode* is the multidimensional node identifier, *Object* is the object, *S* is the start time point and *E* is the end time point of the time interval.

Notice that this schema is appropriate only for MRDF which has only one dimension which takes as values time intervals.

Storage of tRDF model (Pugliese et al. 2008)

Pugliese et al., (2008) propose an approach and a corresponding temporal Graph-based RDF INdex (tGRIN) for storing the annotated RDF triples (*subject*, *property*: *annotation*, *object*) into a PostgreSQL 8.0 *relational database management system* (RDBMS). In detail, the tRDF data is stored in a single relation table (*subject*, *property*, *object*, *annotation*). The tGRIN index is stored separately on disk and is independent of the relational storage model. R+ trees, SR-trees, ST-index, and MAP21, the tGRIN index exhibits superior performance. Based on the storage techniques, they built the experimental temporal RDF prototype and perform the evaluation on several datasets. The results show that their storage and index structure are a better match for tRDF queries.

Storage of TempRDF model (McBride and Butler 2009)

McBride & Butler, (2009) develop a prototype implementation using the Jena (Carroll et al. 2004) and Tuple Database (TDB) (Owens et al. 2009) for persistent storage of temporal RDF graphs. Jena is a leading Semantic Web toolkit for Java programmers (the Jena was first released in 2000 and the revised version Jena was released in August 2003). TDB is a component of Jena for persistent storage of RDF on disk. The Jena API is further extended along with a new API for representing temporally varying information. The new temporal API allows to add new properties to a resource at some time, to modify the value of a resource's property at some time. Noted that the new temporal API is independent on the temporal RDF models, which means that it can be used to any temporal RDF model.

Storage of stRDF (Bereta et al. 2013)

Bereta et al., (2013) propose a *relational database management system* (RDBMS) technique for storing their stRDF data in the system Strabon, which is a semantic geospatial database management system and has been implemented by extending Sesame and using a RDBMS as a backend. The stRDF data (*s*, *p*, *o*, *t*) is stored in Strabon in the form of an N-Quads document, which is divided into many temporal triples and each temporal triple is handled separately by the storage manager following some rules as mentioned in (Bereta et al. 2013). In particular, regarding to the temporal literals in the temporal triples, a schema *period_values*(*id int*, *value period*) is designed to store the instances of the *strdf*:*period* datatype, where *id* is an identifier to identify uniquely each period, and *value* is an instance value of the period datatype and a GiST index on the *value* column is constructed.

Storage of PDStore (Bellamy-McIntyre 2018)

Bellamy-McIntyre (2018) propose an RDF triplestore method called PDStore that uses a hash based index for all triples added and deleted from the store. The RDF triple (*s*, *p*, *o*) is instead represented with a quintuple (*t*, *c*, *s*, *p*, *o*) where *t* is a timestamp and *c* is a change type that is either an add change + (i.e., the addition of a new triple) or a removal change – (i.e., the deletion of the original triple). This representation corresponds to the minimum amount of information one would expect from a standard triplestore's transaction log. Internally String literal values such as URI identifiers are represented with 128 bit integers, with the strings mapped to them using a dictionary. Further, a pair of hash based indexes called the InstanceInstance index and the RoleInstance index are defined. The former is a hashmap that takes as a key both a subject and object pair, and has as its value a list of all triples added and deleted that used both that subject and object in temporal order. The latter takes the predicate as a key, and has a second hashmap as its value (where the second hashmap takes either a subject or object as a key, and once again returns a list of all corresponding triples that were added or deleted in temporal order).

Storage of SPARQL^T model (Gao et al. 2016) (Zaniolo et al. 2018)

Gao et al., (2016); Zaniolo et al., (2018) propose a *main-memory* storage mechanism which uses multiversion B + tree (MVBT) to store and index the point-based temporal RDF triples (Gutierrez et al. 2007). In detail, an MVBT, which is a forest of trees, contains multiple root nodes and each one corresponds to a temporal partition of data. An entry in the MVBT is expressed as (*key*, *start version*, *end version*, *data value*/*pointer*) where *key* is unique for a *version*, and *start version* and *end version* represent the period of a triple. Moreover, in order to further reduce the storage overhead of indices, a delta encoding scheme is designed, and the results show that the size of compressed MVBT is 15%–25% of

standard MVBT. Finally, they implement a system RDF-TX in Java as a single-thread main memory query engine, and the experiments on datasets show that most RDF datasets can be easily fit in main memory.

Discussion and future research directions

After reviewing most of the proposals of temporal extensions to RDF, it has been widely approved that temporal RDF could play a key role in the Semantic Web (also called the Data of Web) (Wang and Tansel 2017), Linked Data (Rula et al. 2012), Knowledge Graph (Chekol et al. 2017b), and other application domains by serving as a framework for temporal knowledge representation, querying, and so on. However, the researches on temporal extensions to RDF are still in a developing stage and still the much more potential of temporal RDF has not been exhaustively explored. The following issues may be important in order for temporal RDF technologies to be more widely adoptable in the future research directions.

RDF Schema extension or representation of temporal data

How to interact between temporal properties and the RDF Schema language is also an important issue for further exploration as also mentioned in (Ao et al. 2020; McBride and Butler 2009). *RDF Schema* (RDFS) (Brickley and Guha 2014), as a semantic extension of RDF, provides a data-modelling vocabulary for RDF data. RDFS provides some vocabularies to model class and subclass, domain and range constraints, and subproperty. When the temporal properties are added into the RDF models, whether we need to simultaneously extend the RDFS with the temporal properties. For example, is a class a temporal class (that is, can the instances of the class vary with time or is the class necessarily atemporal with instances fixed for all time?).

Semantics

Based on the introduction in the previous sections, it is shown that kinds of temporal syntactic extensions to RDF have been investigated. In this case, much more attention should be focused on the semantics of temporal RDF. As similarly mentioned in (Gutierrez et al. 2007), a basic unified semantics for the temporal RDF models may be useful for the further temporal knowledge inference, and also may allow closeness and full query composition in a temporal query language for RDF.

Algebraic operations

As evidenced by the relational database management system, the algebraic operations are important for applying standard database style optimization to query processing. Also, some work (Buneman and Kostylev 2010; Chen and Gupta 2005; Frasincar et al. 2002; Libkin et al. 2018; Robertson 2004) evidenced the power of having an algebraic approach to query RDF, but less research has been done in defining the algebraic operations of temporal RDF. The work in (Buneman and Kostylev 2010) developed an algebra of annotations for RDFS that differs from that developed for relational databases, and they showed how such an annotation algebra can be used for computing annotations on inferred triples that provide information on belief, trust and temporal aspects of data as well as providing a framework for default reasoning.

Inference complexity about time

The satisfiability problem of using simple time intervals for the representation of temporal annotations with some constraints is known to be NP-hard (Golumbic and Shamir 1993). Therefore, when temporal properties are introduced into RDF, another dimension of complexity to inference with RDF triples may also be added. As also discussed in (Dylla et al. 2011; Gergatsoulis and Lilis 2005), the issues of investigating inference complexity about temporal RDF models and developing query and inference systems will be interesting topics for future research so that temporal RDF technologies could be employed in more and more practical applications.

Combination of query languages and indexing

As shown in Temporal RDF querying techniques section, kinds of variants of SPARQL have been presented to query temporal information. Moreover, some temporal indexing techniques (such as SR-trees, ST-index, MAP21 (Pugliese et al. 2008; Yan et al. 2019)) are employed for the efficient implementation of the query system. The future research directions may include such an extension of SPARQL which encompasses some key features of these variants and indexing techniques which can be easily implemented and supports RDF datasets as also mentioned in (Malik et al. 2010).

More temporal RDF datasets and evaluation

With the emergence of temporal information in many applications and tasks, more and more temporal RDF datasets like the T-YAGO [118] and EventKG (Gottschalk and Demidova 2018) are expected to be defined and implemented in the future. This makes it possible that many datasets will be employed to deal with some practice problems and more temporal RDF techniques will also be developed. Moreover, as mentioned in (Fernández et al. 2019), with the emerging demand on efficiently archiving and (temporal) querying different versions of evolving Semantic Web data, the evaluation of

current archiving techniques including querying and storage is extremely important.

The other issues of temporal extensions to RDF

In our current review we focus on the issues of representation, querying, and storage of temporal extensions to RDF. Besides of the approaches mentioned in the previous sections, there are some work that discussing the temporal data management based on RDF while not to model, query and store the temporal RDF data. The work in (Gaha et al. 2015) investigates the use of temporal RDF system for power utilities. The work in (Schrag 2014) discusses how to exploit inference to improve temporal RDF annotations and queries for machine reading. With the development and requirement of temporal RDF data management in various of application domains, more issues about further extensions to temporal RDF (e.g., imprecise temporal RDF (Chekol and Stuckenschmidt 2018; Ghorbel et al. 2019)) are also interesting problems.

Conclusion

The previous sections have shown that RDF for temporal data management and related issues have been widely investigated in the context of the Semantic Web and other real-world applications. In the end of this paper, we once more provide a brief summarization of the main content of this paper to well gain a general understanding of the field. Firstly, in order for RDF to represent temporal information, many proposals for extensions to RDF with different formalisms such as based on the new extension syntaxes and the timestamp were proposed. Furthermore, in order to achieve efficient query on the temporal information, kinds of variants of temporal RDF querying and indexing techniques were subsequently developed. Also, several efficient storage methods of temporal RDF data are considered that has arisen from practical needs. Finally, some discussion and several future research directions are identified. The survey in this paper may help readers understand and catch some key techniques about the issue.

In our future work we will further track the technologies and cover more topics about RDF for temporal data management. Moreover, to make the issues discussed more focused and clearer, in this paper we do not cover the techniques of extending RDF with spatialtemporal information, which will be discussed in detail in our near future work.

Acknowledgements The authors really appreciate the hard work and the time that the reviewers for reviewing this paper. The work is supported by the Natural Science Foundation of Ningxia Province (NO. 2020AAC03212) and the National Natural Science Foundation of China (61672139).

References

Al-Dhaheri S (2016) Survey of temporal knowledge representation. The Graduate Center, CUNY

Allen JF (1983) Maintaining knowledge about temporal intervals. Communications of the ACM 26(11):832–843

Analyti A, Pachoulakis I (2012) A survey on models and query languages for temporally annotated RDF. Int J Adv Comput Sci Appl 3(9):28–35

Anicic D, Fodor P, Rudolph S et al (2011) EP-SPARQL: a unified language for event processing and stream reasoning, Proceedings of the 20th international conference on World Wide Web, pp 635–644

Ao J, Cheng Z, Chirkova R et al (2020) Temporal enrichment and querying of ontology-compliant data, European Conference on Advances in Databases and Information Systems, pp 129–139

Barbieri DF, Braga D, Ceri S, Valle ED, Grossniklaus M (2010b) Querying RDF streams with C-SPARQL. ACM SIGMOD Rec 39(1):20–26

Barbieri DF, Braga D, Ceri S et al (2009) C-SPARQL: SPARQL for continuous querying, Proceedings of the 18th international conference on World Wide Web, pp 1061–1062

Barbieri DF, Braga D, Ceri S et al (2010a) C-SPARQL: a continuous query language for RDF data streams. Int J Semantic Comput 4(01): 3–25

Bellamy-McIntyre J (2018) Modeling and querying versioned source code in RDF, European Semantic Web Conference (ESWC 2018) Satellite Events, pp 251–261

Bereta K, Smeros P, Koubarakis M (2013) Representation and querying of valid time of triples in linked geospatial data, Extended Semantic Web Conference, pp 259–274

Böhlen M, Dignös A, Jensen C, Gamper J (2017) Temporal data management - an overview, Business Intelligence and Big Data - 7th European Summer School (eBISS), pp 51–83

Brickley D, Guha R V. RDF Schema 1.1. W3C Recommendation, 2014. http://www.w3.org/TR/rdf-schema/

Bry F, Spranger S (2003) Temporal constructs for a Web language, Proc Fourth Workshop Interval Temporal Logics and Duration Calculi (ESSLLI '03), pp 1–10

Buneman P, Kostylev EV (2010) Annotation algebras for RDFS. Proceedings of the Second International Workshop on the role of Semantic Web in Provenance Management (SWPM-10), CEUR Workshop Proceedings, pp 1–6

Bykau S, Mylopoulos J, Rizzolo F, Velegrakis Y (2012) On modeling and querying concept evolution. J Data Semantics 1:31–55

Candan KS, Liu H, Suvarna R (2001) Resource description framework: metadata and its applications. Acm Sigkdd Explorations Newsletter 3(1):6–19

Carroll JJ, Bizer C, Hayes P, Stickler P (2005) Named graphs. J Web Semantics 3(3):247–267

Carroll JJ, Dickinson I, Dollin C, Reynolds D, Seaborne A, Wilkinson K (2004) Jena: implementing the Semantic Web recommendations, Proceedings of the 13th International World Wide Web Conference, pp 74–83

Chekol MW, Pirrò G, Schoenfisch J, Stuckenschmidt H (2017a) Marrying uncertainty and time in Knowledge Graphs, Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), pp 88–94

Chekol MW, Pirrò G, Schoenfisch J, Stuckenschmidt H (2017b) TeCoRe: temporal conflict resolution in Knowledge Graphs. Proc VLDB Endowment 10(12):1929–1932

Chekol MW, Stuckenschmidt H (2018) Towards probabilistic bitemporal knowledge graphs. WWW, pp 1757–1762

Chen L, Gupta A, Kurul ME (2005) A semantic-aware RDF query algebra, Proceedings of the International Conference on Management of Data, pp 1–12

Dividino RQ, Sizov S, Staab S, Schueler B (2009) Querying for provenance, trust, uncertainty and other meta knowledge in RDF. J Web Semantics 7(3):204–219
Dylla M, Miliaraki I, Theobald M (2013) A temporal-probabilistic database model for information extraction. Proc VLDB Endowment 6(14):1810–1821
Dylla M, Sozio M, Theobald M (2011) Resolving temporal conflicts in inconsistent RDF knowledge bases. Coord Chem Rev 2(1):474–493
Fabian MS, Gjergji K, Gerhard W (2007) Yago: a core of semantic knowledge unifying wordnet and Wikipedia, 16th International World Wide Web Conference, pp 697–706
Faisal S, Sarwar M (2014) Temporal and multi-versioned XML documents: a survey. Inf Process Manag 50(1):113–131
Fernández JD, Umbrich J, Polleres A et al (2019) Evaluating query and storage strategies for RDF archives. Semantic Web 10(2):247–291
Frasincar F, Houben GJ, Vdovjak R, Barna P (2002) RAL: an algebra for querying RDF, Proceedings of the Third International Conference on Web Information Systems Engineering, pp 173–181
Gaha M, Zinflou A, Bouffard A et al (2015) Temporal RDF system for power utilities, The Ninth International Conference on Advances in Semantic Processing, pp 32–37
Gao S, Gu J, Zaniolo C (2016) RDF-TX: a fast, user-friendly system for querying the history of RDF Knowledge Bases, Proc 19th International Conference on Extending Database Technology (EDBT), pp 269–280
Gergatsoulis M, Lilis P (2005) Multidimensional RDF, Proceedings of the OTM Confederated International Conferences on the Move to Meaningful Internet Systems, pp 1188–1205
Ghorbel F, Hamdi F, Métais E et al (2019) Ontology-based representation and reasoning about precise and imprecise temporal data: a fuzzy-based view. Data Knowl Eng 124:1–26
Golumbic MC, Shamir R (1993) Complexity and algorithms for reasoning about time: a graph-theoretic approach. J ACM 40(5):1108–1133
Gottschalk S, Demidova E (2018) EventKG: a multilingual event-centric temporal knowledge graph, European Semantic Web Conference (ESWC 2018), pp 272–287
Grandi F (2009) Multi-temporal RDF ontology versioning, Proc. of IWOD Workshop, CEUR-WS, pp 1–10
Grandi F (2010) T-SPARQL: a TSQL2-like temporal query language for RDF, Proceedings of the 1st international workshop on querying graph structured data (GraphQ 2010), pp 21–30
Grandi F (2011) Light-weight ontology versioning with multi-temporal RDF schema, Proceedings of the 5th international conference on advances in semantic processing (SEMAPRO 2011), pp 42–48
Guo S, Yan L, Hu Z (2018) Select query translation from temporal SPARQL to TSQL2, Proceedings of the 2018 2nd international conference on management engineering, Software Engineering and Service Sciences, pp 172–175
Gutierrez C, Hurtado C, Vaisman A (2005) Temporal RDF, European Semantic Web Conference, vol 93-107
Gutierrez C, Hurtado CA, Vaisman A (2007) Introducing time into RDF. IEEE Trans Knowledge Data Eng 19(2):207–218
Harris S, Seaborne A. SPARQL 1.1 query language. 2013, https://www.w3.org/TR/sparql11-query/
Hartig O (2009) Querying trust in RDF data with tSPARQL. ESWC, pp 5–20
Hayes P J, Patel-Schneider P F. RDF 1.1 Semantics. 2014. http://www.w3.org/TR/rdf11-mt/
Hoffart J, Suchanek FM, Berberich K, Weikum G (2013) Yago2: A spatially and temporally enhanced knowledge base from wikipedia. Artif Intell 194:28–61
Huang J, Chen W, Liu A et al (2020) Cluster query: a new query pattern on temporal knowledge graph, World Wide Web, pp 1–25
Huber J. Temporal reasoning for RDF(S): a Markov logic based approach. Arbeitspapier, 2014
Hurtado C, Vaisman A (2006) Reasoning with temporal constraints in RDF. PPSWR Workshop, pp 164–178
Kalayci E, Brandt S, Calvanese D, Ryzhikov V, Xiao G, Zakharyaschev M (2019) Ontology-based access to temporal data with Ontop: a framework proposal. Int J Appl Math Comput Sci 29(1):17–30
Le-Phuoc D, Dao-Tran M, Parreira JX et al (2011) A native and adaptive approach for unified processing of linked streams and linked data, International Semantic Web Conference, pp 370–388
Liao HC, Tu CC (2007) A RDF and OWL-based temporal context reasoning model for smart home. Inform Technol J 6(8):1130–1138
Libkin L, Reutter JL, Soto A, Vrgoc D (2018) Trial: a navigational algebra for RDF triplestores. ACM Trans Database Syst 43(1):1–46
Lopes N, Polleres A, Straccia U, Zimmermann A (2010) AnQL: SPARQLing up annotated RDF, Proceedings of the International Semantic Web Conference (ISWC-10), pp 518–533
Ma Z, Capretz MAM, Li Y (2016) Storing massive Resource Description Framework (RDF) data: a survey. Knowledge Eng Rev 31(4):391–413
Malik S, Goel A, Maniktala S (2010) A comparative study of various variants of sparql in semantic web, Computer Information Systems & Industrial Management Applications International Conf, pp 471–474
Manola F, Miller E, McBride B. RDF 1.1 primer. W3C Recommendation. https://www.w3.org/TR/rdf11-primer/
Mazzieri M, Dragoni AF (2008) A fuzzy semantics for the Resource Description Framework, ISWC International Workshops on Uncertainty Reasoning for the Semantic Web, pp 244–261
McBride B, Butler M (2009) Representing and querying historical information in RDF with application to e-discovery. ISWC, pp 1–13
Motik B (2012) Representing and querying validity time in RDF and OWL: A logic-based approach, Web semantics: science, Services and Agents on the World Wide Web, vol 12, pp 3–21
Nguyen V, Bodenreider O, Sheth A (2014) Don't like RDF reification? making statements about statements using singleton property, Proceedings of the 23rd international conference on World Wide Web, pp 759–770
Ognyanov D, Kiryakov A (2002) Tracking changes in RDF(S) repositories, International Conference on Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, pp 373–378
Owens A, Seaborne A, Gibbins N (2009) Clustered TDB: a clustered triple store for Jena. WWW, pp 1–10
Özsu MT (2016) A survey of RDF data management systems. Front Comput Sci 10(3):1–15
Perez J, Arenas M, Gutierrez C (2010) nSPARQL: a navigational language for RDF. J Web Semantics 8(4):255–270
Pugliese A, Udrea O, Subrahmanian VS (2008) Scaling RDF with time. WWW, pp 605–614
Radhakrishna V, Kumar PV, Janaki V (2015) A survey on temporal databases and data mining, Proceedings of the The International Conference on Engineering & MIS, pp 1–6
Rizzolo F, Velegrakis Y, Mylopoulos J, Bykau S (2009) Modeling concept evolution: a historical perspective, 28th International Conference on Conceptual Modeling, pp 1–15
Robatjazi M et al (2015) LORI: linguistically oriented RDF interface for querying fuzzy temporal data. In: Andreasen T (ed) Flexible Query Answering Systems
Robertson EL (2004) Triadic relations: an algebra for the Semantic Web, Proc. of the Second International Workshop on Semantic Web and Databases, pp 91–108
Rodríguez A, McGrath R, Liu Y, Myers J (2009) Semantic management of streaming data, 2nd International Workshop on Semantic Sensor Networks at the International Semantic Web Conference, pp 80–95
Rula A, Palmonari M, Harth A et al (2012) On the diversity and availability of temporal information in linked open data. ISWC, pp 492–507

Rula A, Palmonari M, Ngomo ACN et al (2014) Hybrid acquisition of temporal scopes for RDF data, The Semantic Web: Trends and Challenges. Springer International Publishing

Rula A, Palmonari M, Rubinacci S, Ngomo ACN, Lehmann J, Maurino A, Esteves D (2019) TISCO: temporal scoping of facts. J Web Semantics 54:72–86

Schenk S (2008) On the semantics of trust and caching in the Semantic Web, Proc. of 7th International Semantic Web Conference (ISWC'2008), pp 533–549

Schrag R (2014) Exploiting inference to improve temporal RDF annotations and queries for machine reading. CEUR Workshop Proceedings:104–111

Schueler B, Sizov S, Staab S, Tran DT (2008) Querying for meta knowledge, Proceedings of the 17th international conference on World Wide Web, pp 625–634

Snodgrass RT (1995) The TSQL2 temporal query language. Kluwer Academic Publishers

Song A (2015) Wikipedia infobox temporal RDF knowledge base and indices. University of California

Straccia U (2009) A minimal deductive system for general fuzzy RDF, Web reasoning and rule systems, third international conference, RR 2009, pp 166–181

Straccia U, Lopes N, Lukacsy G, Polleres A (2010) A general framework for representing and reasoning with annotated Semantic Web data, Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10), pp 1437–1442

Tao C, Wei WQ, Solbrig HR, Savova G, Chute CG (2010) Cntro: a semantic web ontology for temporal relation inferencing in clinical narratives, AMIA 2010 Symposium proceedings, pp 787–791

Tappolet J, Bernstein A (2009) Applied temporal RDF: efficient temporal querying of RDF data with SPARQL, 6th European Semantic Web Conference (ESWC-2009), pp 308–322

Tzitzikas Y, Manolis N, Papadakos P (2017) Faceted exploration of RDF/S datasets: a survey. J Intell Inf Syst 48:329–364

Udrea O, Recupero DR, Subrahmanian V (2010) S. Annotated RDF. ACM Trans Comput Logic 11(2):1–41

W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. 2012. W3C Recommendation. http://www.w3.org/TR/xmlschema11-2/

Wahyudi W, Khodra ML, Prihatmanto AS, Machbub C (2018) A question answering system using graph-pattern association rules (QAGPAR) on YAGO Knowledge Base. International Conference on Information Technology Systems and Innovation (ICITSI), pp 536–541

Wang HT, Tansel AU (2017) Temporality in Semantic Web. International Conference on Applied Mechanics and Mechanical Automation (AMMA 2017), pp 187–191

Wang HT, Tansel AU (2019) Temporal extensions to RDF. J Web Eng 18(1–3):25–168

Wang Y, Zhu M, Qu L, Spaniol M, Weikum G (2010) Timely YAGO: harvesting, querying, and visualizing temporal knowledge from Wikipedia, International Conference on Extending Database Technology, pp 697–700

Wudage Chekol M, Pirrò G, Stuckenschmidt H (2019) Fast interval joins for temporal SPARQL queries, Proceedings of the 2019 World Wide Web Conference (WWW2019), pp 1148–1154

Yan L, Zhao P, Ma Z (2019) Indexing temporal RDF graph. Computing 101(10):1457–1488

Yang D, Yan L (2018) Transforming XML to RDF(S) with temporal information. CIT 26(2):115–129

Zaniolo C, Gao S, Atzori M, Chen M, Gu J (2018) User-friendly temporal queries on historical knowledge bases. Inf Comput 259:444–459

Zhang F, Wang K, Li Z, Cheng J (2019) Temporal data representation and querying based on RDF. IEEE Access 7:85000–85023

Zimmermann A, Lopes N, Polleres A, Straccia U (2012) A general framework for representing, reasoning and querying with annotated semantic web data. Web Semantics Sci Serv Agents World Wide Web 11(3):72–95