BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 03/10 pp288–300
Volume 6, Number 3, September 2023
DOI: 10.26599/BDMA.2022.9020044

τSQWRL: A TSQL2-Like Query Language for Temporal Ontologies Generated from JSON Big Data

Zouhaier Brahmia*, Fabio Grandi, and Rafik Bouaziz

Abstract: Temporal ontologies allow to represent not only concepts, their properties, and their relationships, but also time-varying information through explicit versioning of definitions or through the four-dimensional perdurantist view. They are widely used to formally represent temporal data semantics in several applications belonging to different fields (e.g., Semantic Web, expert systems, knowledge bases, big data, and artificial intelligence). They facilitate temporal knowledge representation and discovery, with the support of temporal data querying and reasoning. However, there is no standard or consensual temporal ontology query language. In a previous work, we have proposed an approach named τJOWL (temporal OWL 2 from temporal JSON, where OWL 2 stands for "OWL 2 Web Ontology Language" and JSON stands for "JavaScript Object Notation"). τJOWL allows (1) to automatically build a temporal OWL 2 ontology of data, following the Closed World Assumption (CWA), from temporal JSON-based big data, and (2) to manage its incremental maintenance accommodating their evolution, in a temporal and multi-schema-version environment. In this paper, we propose a temporal ontology query language for τJOWL , named τSQWRL (temporal SQWRL), designed as a temporal extension of the ontology query language—Semantic Query-enhanced Web Rule Language (SQWRL). The new language has been inspired by the features of the consensual temporal query language TSQL2 (Temporal SQL2), well known in the temporal (relational) database community. The aim of the proposal is to enable and simplify the task of retrieving any desired ontology version or of specifying any (complex) temporal query on time-varying ontologies generated from time-varying big data. Some examples, in the Internet of Healthcare Things (IoHT) domain, are provided to motivate and illustrate our proposal.

Key words: temporal big data; temporal ontology; temporal query language; temporal OWL 2 from temporal JSON (τJOWL); Semantic Query-enhanced Web Rule Language (SQWRL); Temporal SQL2 (TSQL2); Internet of Healthcare Things (IoHT)

1 Introduction

1.1 Context of work

Time is pervasive in computer applications, whether they are using classical technologies[1,2], like management information systems and database applications, or modern ones[3,4], like blockchains, Internet of Things (IoT), Internet of Healthcare Things (IoHT)[5], and internet of vehicles. A temporal database is a database

• Zouhaier Brahmia and Rafik Bouaziz are with the Department of Computer Science, Faculty of Economics and Management, University of Sfax, Sfax 3029, Tunisia. E-mail: zouhaier.brahmia@fsegs.rnu.tn; rafik.bouaziz@usf.tn.
• Fabio Grandi is with the Department of Computer Science and Engineering, University of Bologna, Bologna 40136, Italy. E-mail: fabio.grandi@unibo.it.
* To whom correspondence should be addressed.
Manuscript received: 2022-08-26; revised: 2022-10-15; accepted: 2022-10-28

with built-in support for managing time-varying data[6,7]. In the temporal database literature, there are two well-known temporal dimensions which have been proposed for timestamping temporal data: (1) transaction time[8], which denotes when some datum is current in the database, and (2) valid time[9], which denotes when some datum is valid in the modeled reality. The applications' requirements concerning temporal data management and querying are omnipresent regardless of both the application field (e.g., business, banking, weather monitoring and forecasting, meteorology, and health and social services) and the underlying database model (e.g., relational, object-oriented, relational-object, document-oriented, graph-oriented, etc.).

In the ontology[10] research field, which is closely related to databases, temporal ontologies[11–13] allow not only to represent concepts, their properties, and their relationships, but also time-varying knowledge, via versioning (e.g., Refs. [14–16]) or by adopting the four-dimensional (4D) perdurantist view[17]. Temporal ontologies are widely used to formally represent temporal data semantics in several applications belonging to different domains (e.g., Semantic Web, expert systems, knowledge bases, big data, and artificial intelligence). They facilitate temporal knowledge representation and discovery, and make easy temporal data querying and reasoning. However, to the best of our knowledge, there is a lack of standard or consensual query language for temporal ontologies, like Temporal SQL2 (TSQL2)[18] for temporal relational databases. Therefore, knowledge base administrators, ontology engineers, and Semantic Web application developers are proceeding in an ad hoc manner when querying time-varying ontologies.

Big data[19–22] are being widely used in several application fields like online social networks, IoT, IoHT, telecommunications, energy consumption control and grid management, water resources monitoring, studies of climate change effects on agriculture and environment, fighting pandemics like COVID-19, food and nutrition security, and nuclear industry. Big data can be stored according to some specific data format like JavaScript Object Notation (JSON)[23], and require to be efficiently managed, to satisfy the requirements of end users in performing daily tasks concerning creation, deletion, and update of big data instances. Moreover, they also require to be efficiently queried[24] and analyzed[25], to help/assist the decision-makers in taking appropriate decisions. From one hand, the semantics of big data[26] is very helpful for both big data management[27] and analytics[28], and, from the other hand, ontologies are widely used to formally represent data semantics of applications. Hence, we have proposed in a previous work[29] an approach, named τJOWL (temporal OWL 2 from temporal JSON, where OWK 2 stands for OWL 2 Web Ontology Language), which allows (1) to automatically produce a temporal OWL 2[30] ontology of data, following the Closed World Assumption (CWA)[31], starting from temporal JSON-based big data, and (2) to manage the incremental maintenance of such a temporal ontology by accommodating the evolution of temporal big data, in a temporal and multi-schema-version environment. Notice that OWL 2 is the World Wide Web Consortium (W3C) standard ontology language for the Semantic Web ontologies.

1.2 Problems

However, the state-of-the-art of temporal OWL 2 ontology management and querying does not include any standard or consensual language for querying temporal OWL 2 ontologies, either generated from temporal big data or from temporal databases. The few available query languages for OWL ontologies, like OWL-QL[32], Semantic Web Rule Language (SWRL)[33], Semantic Query-enhanced Web Rule Language (SQWRL)[34], and OWL 2 QL[35], do not provide any built-in support for querying temporal ontologies. Moreover, some temporal instance support has been added to the SQWRL language, as shown in Refs. [36] and [37], and to the Protégé tool, as shown in Ref. [38], but such a support is very limited and does not fulfill all users' requirements since it does not allow them to express powerful and complex temporal queries on temporal OWL 2 ontologies. Furthermore, there is neither a language nor a tool (among those mentioned in Refs. [36–38]) that has tried to take advantage of the expressiveness of the consensual temporal query language TSQL2[18], which is well known in the temporal database community (for querying bitemporal relational databases).

In a previous work[39], we have proposed a temporal OWL 2 ontology framework, temporal OWL 2 called τOWL , to construct and validate time-varying OWL 2 documents via the use of a temporal ontology schema. Nevertheless, τOWL does not support querying (temporal) ontology instances since it has not been equipped with a temporal ontology query language yet.

1.3 Objectives, choices, and contributions

For the reasons presented above, a new temporal

ontology query language, temporal SQWRL named τSQWRL, is proposed in this paper to be used in the τJOWL environment. The aim of the proposal is to facilitate the retrieval of any desired temporal OWL 2 ontology version and to express (complex) temporal query on time-varying OWL 2 ontologies that are generated from temporal JSON-based big data. The new language is designed as a temporal extension of the ontology query language SQWRL, inspired by the features of TSQL2 to take advantage of its widely acknowledged strengths. Notice that we chose to extend the SQWRL language[34] and not the W3C-endorsed OWL 2 QL language[35] for the following reasons:

- SQWRL is simpler and more practical than OWL 2 QL whose specification is very long, complex and, thus, a bit impractical.
- SQWRL is far more used in the ontology querying literature than OWL 2 QL.
- Since we are especially interested in medical/health applications, we prefer SQWRL, which has been developed for the biomedical field and, thus, it is more often than OWL 2 QL used in the development of such applications.

It is worth mentioning that this paper does not deal with temporal reasoning on ontologies[40], which is applied in an Open World Assumption (OWA)[41], but focuses on temporal querying of OWL 2 ontologies that are produced from JSON big data, in a CWA environment.

Dealing with the general issue of querying temporal ontologies generated from temporal big data, the contribution of the present paper is to focus on the data side, assuming a separation of environments/concerns:

(1) The τJOWL framework to manage the temporally versioned ontology, to be used as a (versioned) schema, and

(2) A temporal database, equipped with a TSQL2-like query language, to manage the JSON-based big data (e.g., generated by an IoT or an IoHT platform).

1.4 Organization

The rest of this paper is structured as follows. Section 2 presents the background of the present work. Section 3 proposes the new temporal ontology query language τSQWRL, and illustrates its use through some examples of τSQWRL queries in the IoHT domain. Section 4 gives some information on the implementation of the current proposal. Section 5 discusses related work and clarifies the contribution with respect to the state-of-the-art. A summary of the present contribution and some remarks concerning its continuation in future work are provided in Section 6.

2 Background

In this section, we first introduce the temporal ontology instance data model on which the new proposed language τSQWRL is based. Then, adoption of this model is illustrated through an application example. Finally, the main features of the SQWRL ontology language are briefly recalled.

2.1 Data model for temporal OWL 2 ontology instances

A temporal data model[14] (e.g., BCDM, TEMPOS, XBiT, TempoJCM, tOWL, and tRDF) is a data model for representing temporal data, which can be of type state (i.e., with a continuous persistence over a temporal interval) or event (i.e., with an instantaneous occurrence).

The temporal OWL 2 ontology data model, on which the proposed language is based, has been defined for the τJOWL framework. In such a framework, temporal OWL 2 ontology instances are stored in an eXtensible Markup Language (XML) document called the "temporal instances document". In the following, it is explained how a temporal instances document is created.

Each time τJOWL generates a new OWL 2 ontology schema (i.e., an OWL 2 file), this latter is considered as a new conventional (i.e., non-temporal) ontology schema; to each conventional schema corresponds a conventional ontology document (i.e., an OWL 2 file) which stores conventional ontology instances conforming to that schema. Consequently, the τJOWL base administrator specifies temporal aspects of this conventional ontology schema by annotating this ontology schema with a set of temporal features, such that each feature is associated with a component of this conventional ontology schema. There are two types of temporal features: logical features and physical features. Logical features allow the administrator to specify which components (e.g., a class, a data property, and an object property) of a conventional ontology schema can vary over transaction time and/or valid time. Physical features allow the administrator to specify where timestamps should be placed and how the temporal aspects should be represented. The whole set of temporal features is stored in a temporal feature document (which is an XML file). Then, once the τJOWL base administrator asks the system to

commit his/her temporal annotation of the conventional ontology schema, the system automatically generates the following files:

(1) The temporal ontology schema is an XML file which ties together the conventional ontology schema and the temporal feature document;

(2) A temporal ontology document is an XML file which links each conventional ontology document, which is conformant to the conventional ontology schema, to its temporal ontology schema and, consequently, to its set of temporal logical and physical features; notice that the temporal ontology document (a) represents a sequence of versions of the same conventional OWL 2 ontology instance document, such that each version, also termed "ontology slice", has a distinct timestamp, and (b) specifies the temporal ontology schema associated to these conventional ontology document versions; hence, an ontology slice is a version of a temporal ontology document during some given time interval. For example, if a Temporal Ontology Document (TOD) is composed of three conventional ontology instance documents OntoDoc1, OntoDoc2, and OntoDoc3, which are valid during time intervals $[t_1, t_2)$, $[t_2, t_3)$ and $[t_3, \text{UC})$, respectively, then the slice at time t_x such that $t_1 \leqslant t_x < t_2$, is OntoDoc1;

(3) The temporal instances document is an OWL file associated to each generated temporal ontology document; it stores the temporal ontology instances that result from the application of the temporal features to the conventional ontology instances. For example, the temporal instances document TID corresponding to the TOD mentioned above is the "squashed"/combined version of the three documents OntoDoc1, OntoDoc2, and OntoDoc3. Notice that τJOWL deals with OWL 2 ontologies with an RDF/XML syntax[42] (RDF stands for Resource Description Framework), which is the only syntax that must mandatorily be supported by OWL 2 tools as recommended in the W3C OWL 2 specification document[43]. Notice also that the management of temporal features in τJOWL is similar to the management of temporal annotations in the τOWL framework[39].

2.2 Illustrative example

In order to illustrate the underlying data model, let us consider the example of a "temporal instances document" that stores the medical data of a patient (i.e., his/her SSN, name, birthdate, body temperature, and heart rate), in a τJOWL base of a hospital. Furthermore, let us assume that the τJOWL base administrator has chosen to manage the evolution over time of body temperature and heart rate along the valid time dimension[9]. Figure 1 shows an instance of the temporal OWL 2 ontology instance data model, which stores the information of one patient with two valid-time versions of body temperature and two valid-time versions of heart rate. Notice that the temporal instances document of Fig. 1 combines/squashes two versions of the same conventional ontology instance document, such that the first is valid during [2022-05-14, 2022-05-15] and the second during [2022-05-16, now).

2.3 SQWRL ontology query language

SQWRL[34] is a high-level query language that is based on the low-level SWRL language[33]. It enables not only writing temporal rules, but also temporal queries on an OWL ontology. It provides a set of SQL-like query operators that are based on some proprietary built-

```
<?xml version="1.0"?>

<rdf:RDF
 xmlns:rdf=
     "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:pt="http://www.hospital.cure/patient#">

<rdf:Description
 rdf:about=
   "http://www.hospital.cure/patient/Layla-Ahmad">
  <SSN>111222333</SSN>
  <name>Layla Ahmad</name>
  <birthdate>1989-05-16</birthdate>
  <bodyTemperature_RepItem>
    <bodyTemperature_Version>
      <timestamp_ValidExtent
           begin="2022-05-16" end="now" />
      <pt:bodyTemperature>37</pt:bodyTemperature>
    </bodyTemperature_Version>
    <bodyTemperature_Version>
      <timestamp_ValidExtent
           begin="2022-05-14" end="2022-05-15" />
      <pt:bodyTemperature>40</pt:bodyTemperature>
    </bodyTemperature_Version>
  </bodyTemperature_RepItem>
  <heartRate_RepItem>
    <heartRate_Version>
      <timestamp_ValidExtent
           begin="2022-05-16" end="now" />
      <pt:heartRate>130</pt:heartRate>
    </heartRate_Version>
    <heartRate_Version>
      <timestamp_ValidExtent
           begin="2022-05-14" end="2022-05-15" />
      <pt:heartRate>140</pt:heartRate>
    </heartRate_Version>
  </heartRate_RepItem>
</rdf:Description>
</rdf:RDF>
```

Fig. 1 An example of a temporal ontology instance document in the τJOWL framework.

ins. An SQWRL query consists of two components, the antecedent (or the body) and the consequent (or the head), and is specified as follows:

```
antecedent -> consequent.
```

The antecedent and consequent consist of a positive conjunction of atoms:

```
atom ^ atom ^ ··· ^ atom
    -> atom ^ atom ^ ··· ^ atom.
```

An atom is an expression having the following form:

p(arg$_1$, arg$_2$, ..., arg$_n$) such that:

- $p()$ is a predicate symbol, representing an OWL class, property, or data type, and
- arg$_i$ ($i = (1, 2, \ldots, n)$) is an argument of $p()$, representing an OWL individual, a data value, or a variable referring to one of them.

For example, the following query returns all medicines that have been used by any patient (in all his/her diseases):

```
Patient(?p)^hasDisease(?p,?d)
  ^ Medicine(?m)^useMedicine(?p,?m,?d)
  ->sqwrl:select(?p,?m).
```

Moreover, SQWRL also supports set-based queries. In fact, it provides some operators to create and to manipulate sets like the following ones:

- sqwrl:makeSet(S,$elem$): it creates a set S and adds the element elem to it;
- sqwrl:size(z,S): it returns the size z of the set S;
- sqwrl:union(U,S_1,S_2): it returns the union U between the sets S_1 and S_2;
- qwrl:difference(D,S_1,S_2): it returns the difference D between the sets S_1 and S_2.

Aggregation operators, like sqwrl:min(), sqwrl:max(), sqwrl:sum(), and sqwrl:avg(), are also supported by SQWRL. Furthermore, similarly to the GROUP BY clause of SQL, the sqwrl:groupBy() operator allows to group related sets, but only one grouping could be applied to each set. The first argument of sqwrl: groupBy(set,group) is the set and the second and (optional) subsequent arguments are the entities to group by. Finally, SQWRL offers a clause, represented by character °, for containing the set management operators. To illustrate these operators, we provide the following SQWRL query that returns the number of diseases of each patient:

```
Patient(?p)^hasDisease(?p,?d)
°sqwrl:makeSet(?s,?d)^sqwrl:groupBy(?s,?p)
^ sqwrl:size(?n,?s)->sqwrl:select(?p,?n).
```

3 Temporal Ontology Query Language τSQWRL

In this section, first, the basic assumptions of this work are presented (in Section 3.1). Then, the goals of the approach are provided (in Section 3.2). Next, the syntax and semantics of the temporal ontology query language τSQWRL are defined (in Section 3.3). Finally, the use of the language is illustrated by providing some examples of τSQWRL queries in the IoHT context (in Section 3.4). Notice that a temporal query language[44] (e.g., TSQL2, SQL/Temporal, TTXPath, τXQuery, TOQL, and T-SPARQL) is a query language that provides built-in support for querying temporal data. It is associated with a temporal data model that specifies the underlying data structures that the query language acts on.

3.1 Basic assumptions

The τSQWRL proposal is based on the following four assumptions:

(A1) The ontologies are managed/queried in a CWA environment, which means that "what is not known to be true must be false". Notice that although the OWA, which means that "what is not known to be true or false is unknown", is the classical approach adopted for managing ontologies in environments where data are supposed to be incomplete (e.g., artificial intelligence, expert systems, knowledge bases, and Semantic Web applications), the CWA can be used to manage ontologies in environments where data are supposed to be complete (e.g., traditional databases) via the Data Box (DBox) concept[45]. It is important to note that, with the CWA, an ontology definition (concepts, relationships between concepts, axioms, ...) operates like a database schema and, thus, can be named ontology schema; the ontology individuals are instances of such an ontology schema.

(A2) Time is used as part of data/instances, as in temporal databases and, thus, temporal reasoning is not required to query temporal data.

(A3) A separation is made between the ontology definition/schema and the ontology instances (or the individuals), which are stored in a temporal database according to such a schema.

(A4) Considering OWL 2 ontologies that are automatically constructed from JSON big data (received from an IoHT system), we assume that such big data are well structured, cleaned and validated, and such ontologies are consistent and faithfully encode the underlying big data; we assume that both big data and

ontologies do not require specific processing tasks (e.g., repair actions) to be ready for use/exploitation (e.g., querying).

3.2 Goals

The τSQWRL language is proposed to achieve the following six goals:

(G1) τSQWRL must be designed as a TSQL2-like query language, which means that it has to take advantage of all strengths of TSQL2 and to provide temporal atoms that are similar to the temporal operators of TSQL2.

(G2) τSQWRL should be fully backwards-compatible with SQWRL, which means that any SQWRL query is also a valid τSQWRL query.

(G3) The extensions to SQWRL should be both minimal and efficient. Hence, the definition of τSQWRL should reuse SQWRL whenever possible.

(G4) The extensions to SQWRL should include all functions that could be useful to query temporal OWL 2 ontology instances.

(G5) τSQWRL should be scalable when the volume of the queried ontologies grows, since we are dealing with temporal ontologies that are generated from temporal big data. In fact, querying in the ontology world is considered a deduction/reasoning task, which is usually computationally intractable for large ontologies even in absence of data and introducing temporal reasoning usually implies a blow-up in complexity. Considering the management of IoHT big data, scalability is an issue that should be seriously taken into account.

(G6) The completeness and the expressiveness of τSQWRL should be guaranteed.

It must be noted that, due to space limitations, goals G5 and G6 will be dealt with in a future work.

3.3 Syntax and semantics of τSQWRL

The proposed temporal extension of SQWRL consists of augmenting the its syntax with a set of valid-time atoms, as shown below, in order to allow querying valid-time OWL 2 ontology instances that are generated from valid-time JSON big data. Such new temporal atoms have been inspired from the specification of the TSQL2 language[18] (and, as for TSQL2, the resulting language is capable of testing all the thirteen possible relationships between intervals[46]).

Hence, in order to have a TSQL2-like query language, we decided to exploit the full power of temporal querying along valid time using the following atoms to extract and compare timestamps (a timestamp could be either a time point, or a time interval):

- `sqwrl:validAt(?x,?t)`: it returns true if the argument x is valid at the time t, and false otherwise;
- `sqwrl:validBetween(?x,?vts,?vte)`: it returns true if the argument x is valid between the time point *vts* (for valid-time period start) and the time point *vte* (for valid-time period end), and false otherwise;
- `sqwrl:precedes(?`i_1`,?`i_2`)`: it returns true if the ending point of interval i_1 is earlier than the starting point of interval i_2, and false otherwise;
- `sqwrl:meets(?`i_1`,?`i_2`)`: it returns true if the interval i_1 precedes the interval i_2 and there are no time points between them, and false otherwise;
- `sqwrl:overlaps(?`i_1`,?`i_2`)`: it returns true if the intervals i_1 and i_2 have some point in common, and false otherwise;
- `sqwrl:starts(?`i_1`,?`i_2`)`: it returns true if the intervals i_1 and i_2 have the same starting point, and false otherwise;
- `sqwrl:contains(?`i_1`,?`i_2`)`: it returns true if every point of the interval i_2 also belongs to the interval i_1, and false otherwise;
- `sqwrl:finishes(?`i_1`,?`i_2`)`: it returns true if the intervals i_1 and i_2 have the same ending point, and false otherwise;
- `sqwrl:equals(?`i_1`,?`i_2`)`: it returns true if the intervals and i_2 are identical, and false otherwise;
- `sqwrl:valid(?t)`: this atom is always specified with the `sqwrl:select(?x)` atom. It returns the valid-time timestamp of the argument x of the `sqwrl:select` atom;
- `sqwrl:VTbegin(?t,?b)`: this atom is always specified with the `sqwrl:select(?x)` atom. It returns the starting point b of the valid-time timestamp t of the argument x of the `sqwrl:select` atom;
- `sqwrl:VTend(?t,?e)`: this atom is always specified with the `sqwrl:select(?x)` atom. It returns the ending point e of the valid-time timestamp t of the argument x of the `sqwrl:select` atom;
- `sqwrl:duration(?t,?d,?g)`: it returns the duration d, at a given granularity g (e.g., "year", "month", "day", and "hour"), of the time expression t;
- `sqwrl:firstVers(?fv,?s)`: it returns the first version *fv* of the set of versions s (i.e., the version having the first valid-time interval among all versions belonging to s);
- `sqwrl:firstNVers(?fnv,?s,n)`: it returns the first n versions (*fnv*) of the set of versions s;
- `sqwrl:lastVers(?lv,?s)`: it returns the last

version *lv* of the set of versions *s* (i.e., the version having the last valid-time interval among all versions belonging to *s*);

- `sqwrl:lastNVers(?lnv,?s,n)`: it returns the last *n* versions (*lnv*) of the set of versions *s*;
- `sqwrl:NthVers(?nthv,?s,n)`: it returns the *n*-th version (*nthv*) of the set of versions *s* (i.e., the version having the *n*-th valid-time interval among all versions belonging to *s*).

Moreover, for use in a temporal and multi-version ontology setting, the SQWRL language does not include expressions/statements/clauses to specify the ontology version(s) on which the queries are formulated and have to be executed. Therefore, the τJOWL base administrator writes his/her query while considering only one big fat ontology that includes all temporal versions in the background. Hence, in the proposed extension, the following SET clauses can be used to select the desired ontology version(s) along transaction time:

- `SET ontology version anOntology.owl as of tt`: It selects the schema version of the ontology anOntology.owl, whose transaction-time interval includes the temporal value (i.e., a time point) *tt*. Notice that this clause will select the current schema version of the ontology anOntology.owl if the user specifies the string "current_time" or "CT" or "now" as a value of the parameter *tt*.
- `SET ontology version anOntology.owl between b and e`: It selects the consecutive schema versions of the ontology anOntology.owl, which were current between the temporal value *b* (for beginning) and the temporal value *e* (for ending).

Recall that, in the τJOWL framework, ontology schemas are versioned along transaction-time[29]; for that reason, each ontology schema version has only a transaction-time timestamp.

3.4 Examples of τSQWRL queries

In Refs. [18, 47], the authors have defined the following types of temporal queries: time-point selection, history selection, rollback (with transaction time only), snapshot (with valid time only), temporal slicing, temporal join, temporal aggregate, and restructuring. In the rest of this section, τSQWRL usage is illustrated via the specification of eight queries. Notice that queries 5, 6, 7, and 8 are set-based temporal queries[34]. Temporal atoms are colored in red.

Let us assume to deal with a temporal knowledge base that is being exploited in an IoHT environment, and that is storing temporal information on patients, medicines, doctors, and diseases. Assume also that the creation and the evolution over time of such a knowledge base has been managed via the use of the τJOWL framework. Therefore, collected big data that are received from connected medical devices (e.g., monitors, sensors) are stored in JSON format and ontologies associated to these big data are automatically generated and stored in OWL 2 files. Furthermore, the evolution over time of such big data, when new values that replace old ones are received, leads to two inseparable modifications: (1) temporal updates to the involved JSON-based big data instances, executed in order to produce new temporal versions of such instances, and (2) temporal changes to the involved OWL 2 ontology schema, executed in order to generate a new temporal version of such schema.

Query 1: Find medicines which appeared in 2020 and exactly when

```
Medicine(?m)
  ^ sqwrl:validAt(?m,?t)
  ^ sqwrl:VTbegin(?t,?b)
  ^ sqwrl:contains('2020',?b)
 -> sqwrl:select(?m)
  ^ sqwrl:valid(?b)
```

Query 2: Find patients who were hospitalized with COVID-19 in 2021.

```
Patient(?p)
  ^sqwrl:validAt(hasDisease(?p,?d),?t)
  ^ Disease(?d,'COVID-19')
  ^ sqwrl:contains('2021',?t)
  -> sqwrl:select(?p)
```

Query 3: Find patients who changed doctor and when (this query could be motivated by, for example, the emigration of Tunisian doctors to European countries, which intensifies from one year to the next).

```
Patient(?p)
  ^ sqwrl:validAt(followedBy(?p,?d1),?t1)
  ^ sqwrl:validAt(followedBy(?p,?d2),?t2)
  ^ sqwrl:isDifferent(?d1,?d2)
  ^ sqwrl:meets(?t1,?t2)
  -> sqwrl:select(?p)
  ^ sqwrl:valid(?t)
```

Query 4: Find medicines which have been used by any patient during the period of his/her hospitalization with COVID-19 during 2020.

```
Patient(?p)
  ^sqwrl:validAt(hasDisease(?p,?d),?t)
  ^ Disease(?d,'COVID-19')
```

```
  ^ Medicine(?m)
  ^ sqwrl:contains('2020',?t)
  ^ sqwrl:validAt(useMedicine(?p,?m,?d),?t)
  -> sqwrl:select(?p,?m)
```

Query 5: Find the time of the first hospitalization of each patient.

```
Patient(?p)
  ^ sqwrl:validAt(hospitalized(?p,?h),?t)
  ^ sqwrl:VTbegin(?t,?begin)
  ° sqwrl:makeSet(?s,?begin)
  ^ sqwrl:groupBy(?s,?p,?h)
  ^ sqwrl:min(?first,?s)
  ^ sqwrl:equals(?first,?begin)
  -> sqwrl:select(?p,?begin)
```

Query 6: Find, for each patient, the hospitalization having the minimum duration and that having the maximum duration, among his/her hospitalizations (durations should be presented in days).

```
Patient(?p)
  ^ sqwrl:validAt(hospitalized(?p,?h),?t)
  ° sqwrl:makeSet(?s,?t)
  ^ sqwrl:groupBy(?s,?p,?h)
  ^ sqwrl:min(?mi,?s)
  ^ sqwrl:max(?ma,?s)
  ^ sqwrl:duration(?mi,minDur,'day')
  ^ sqwrl:duration(?ma,maxDur,'day')
  -> sqwrl:select(?p,?minDur,?maxDur)
```

Query 7: Find the most recent disease of each patient together with medicines used by this patient for such a disease.

```
Patient(?p)
  ^ sqwrl:validAt(hasDisease(?p,?d),?t1)
  ^ Disease(?d)
  ^ Medicine(?m)
  ^ sqwrl:validAt(useMedicine(?p,?m,?d),?t2)
  ° sqwrl:makeSet(?s,?d)
  ^ sqwrl:groupBy(?s,?p)
  ^ sqwrl:lastVers(?l,?s)
  ^ sqwrl:equals(?t1,?t2)
  -> sqwrl:select(?p,?d,?m)
```

Query 8: Find the average number of medicines used by each patient for each one of his/her diseases.

```
Patient(?p)
  ^ sqwrl:validAt(hasDisease(?p,?d),?t1)
  ^ Disease(?d)
  ^ Medicine(?m)
  ^ sqwrl:validAt(useMedicine(?p,?m,?d),?t2)
  ° sqwrl:makeSet(?s,?m)
  ^ sqwrl:groupBy(?s,?p,?d)
  ^ sqwrl:avg(?a,?s)
  ^ sqwrl:equals(?t1,?t2)
  -> sqwrl:select(?p,?d,?a)
```

4 Implementation

In order to show the feasibility of the proposed language, a tool that supports τSQWRL, named τSQWRL-Processor, is under development.

The τSQWRL-Processor will be integrated in the τJOWL-Manager system that is being developed to support the whole τJOWL framework[29]. The τSQWRL-Processor is a temporal stratum[48], written in Java, programmed to run on top of a traditional (i.e., non-temporal) SQWRL engine (like the SWRLTab plugin[34] developed in Protégé-OWL), as shown in Fig. 2. The stratum is composed of four modules: "Query Syntax Checker", "Query Mapper", "Query Optimizer", and "Query Result Processor".

As far as its functioning is concerned, the τJOWL base administrator first specifies his/her τSQWRL query with a graphical user interface that allows him/her to edit and manage the execution of τSQWRL queries. Then, the "Query Syntax Checker" module carefully checks the syntax of the input τSQWRL query and guarantees that it is syntactically valid. After that, the "Query Mapper" module converts the syntactically checked τSQWRL query into a valid SQWRL query; in particular, temporal atoms are transformed into semantically equivalent

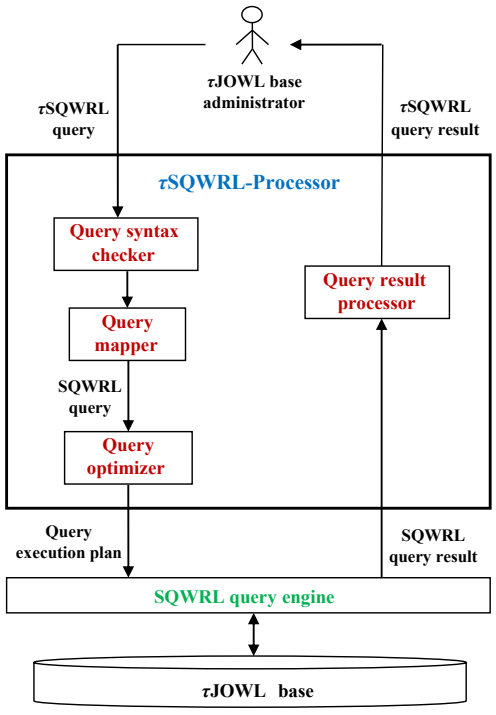

Fig. 2 Architecture of the τSQWRL-Processor tool.

SQWRL expressions. Next, the "Query Optimizer" generates a query execution plan[49] for the SQWRL query that has been produced by the "Query Mapper". Finally, the generated execution plan is submitted to the SQWRL engine to be executed on the underlying τJOWL base. Once the query result is returned by the SQWRL engine to the temporal stratum, it is managed by the "Query Result Processor", which checks if the results of the query necessitate temporal coalescing[50] (i.e., merging value-equivalent data that have adjacent or overlapping timestamps).

5 Related Work

Querying temporal data has been widely studied in the database field (e.g., Refs. [18, 44, 51, 52] just to cite a few). In the Semantic Web world, Grandi has provided, in Section 2.1 of his annotated bibliography on temporal and evolution aspects[53], a list of 97 references for works (published before December 2012) which have added time dimension(s) to some Semantic Web models (e.g., RDF) or languages, like the SPARQL protocol and RDF query language.

The study of the state-of-the-art shows that only few works focused on the problem of temporal ontology querying considered in the present work: Refs. [36, 37, 54, 55]. In Ref. [54], a high-level Temporal Ontology Querying Language (TOQL), has been proposed as an SQL-like query language. It considers ontologies as relational databases. TOQL represents temporal concepts of ontologies based on the 4D perdurantist approach. References [36] and [37] proposed an approach (a data model and a set of tools) to represent and query temporal information in biomedical ontologies that are specified in OWL. A lightweight temporal model is used to encode the temporal dimension of biomedical data, from one hand, and both the SWRL and SQWRL languages are used to query the temporal information that are conforming to that model, from the other hand. In Ref. [55], Grandi proposed T-SPARQL, a temporal extension of the SPARQL query language for RDF graphs to represent the evolving specification of an ontology. T-SPARQL is based on a multi-temporal RDF database model[11] which employs triple timestamping with temporal elements; such a model best preserves the scalability property enjoyed by triple storage technologies, especially in a multi-temporal setting. The temporal extensions proposed for SPARQL are aimed at embedding several features of TSQL2.

As for available technical support related to temporal ontology querying, the SWRLAPI[56] Protégé project (on GitHub) includes a software component named "SWRL Temporal Ontology and Library". Considering the documentation on the GitHub page[38] of such a component (on which the name of the component becomes SWRLAPI temporal), it can be said that the Protégé tool (1) allows to specify in OWL a valid-time ontology named SWRL Temporal Ontology[57], and (2) provides some temporal support, via a library named SWRL temporal Built-ins[58], which allows ontology engineers or knowledge base administrators to query the SWRL Temporal Ontology. Moreover, it is worth mentioning that SQWRL has been used in the Protégé tool as a Tab, called SWRLTab, acting on the ontology that has been already opened in the editor, as indicated in Ref. [59].

Moreover, some researchers have focused on querying spatio-temporal ontologies like Refs. [60] and [61]. In Ref. [60], a spatio-temporal query language, called SOWL QL, was proposed to query Spatio-temporal OWL (SOWL) ontologies. An SOWL ontology represents spatio-temporal information in OWL. SOWL QL enables to query not only quantitative information (e.g., exact dates, times, and locations), but also qualitative spatio-temporal information (specified through natural language expressions like "east of", "west of", "north of", "south of", "before", and "after"). SOWL QL augments SPARQL with a set of temporal and spatial operators, like temporal Allen operators[46], spatial directional, and topological operations. In Ref. [61], a spatiotemporal data model based on RDF was defined and spatiotemporal algebraic operations were studied. The authors proposed five types of graph algebras and define a spatiotemporal RDF syntax to allow browsing, querying, and reasoning with spatiotemporal RDF graphs.

Some other works have dealt with temporal reasoning on (temporal) ontologies, like Refs. [40] and [62]. In fact, Ref. [40] proposed an approach for reasoning over temporal OWL ontologies: first they defined an OWL ontology that represents both temporal qualitative information and temporal quantitative information, and after that they introduced and discussed a temporal query language, which is based on SPARQL, for querying such an ontology. In Ref. [62], a reasoner that supports an extension of the TOQL language[54] to temporal reasoning aspects, was developed. Notice that, contrarily to temporal querying, which deals

with retrieving information/knowledge explicitly represented in the underlying temporal ontology, temporal reasoning focuses on inferring/deducing new information/knowledge from those represented in a temporal ontology. Hence, TOQL supports both querying and reasoning. Furthermore, since (temporal) OWL 2 ontologies could be considered as semi-structured data collections, the research works that have dealt with temporal semi-structured data (e.g., XML data and JSON data) querying could also be considered as related. In the literature, only a few works can be found that have studied temporal XML or JSON data querying: Refs. [63–66]. In Ref. [63], Dyreson proposed TTXPath, a temporal extension of the XML Path language (XPath)[67] to locate and query transaction-time XML data. Gao and Snodgrass[64] extended the XQuery language[68] to support querying bitemporal XML data in a τXSchema (temporal XML Schema) environment[69]. Rizzolo and Vaisman[65] provided TXPath, a temporal extension of XPath for querying bitemporal XML data. In Ref. [66], Brahmia et al. proposed a temporal extension of the JSON Path language (JSONPath)[70] to support querying transaction-time JSON data in a τJSchema (temporal JSON Schema) framework[71].

Recently, Ref. [3] studied temporal querying of streaming data which are in general considered as temporal big data. However, neither ontologies have been used in data querying, nor JSON format has been used to represent and store streaming data. On the contrary, while considering a temporal relational environment, the authors extend the temporal algebra TA[72], which is equipped with six primitive temporal operators (selection, projection, Cartesian product, union, difference, and grouping), with non-temporal operators to support both sequenced and non-sequenced temporal queries on streaming tables[73]. A streaming table, which is a special kind of a temporal relational table, stores streaming data for a user-defined period called historical period; outside their historical periods, data are vacuumed[18].

6 Conclusion

In this paper, we have proposed, for the τJOWL framework[29], a TSQL2-like temporal ontology query language, named τSQWRL. τSQWRL has been defined as a valid-time extension to the SQWRL query language. It augments SQWRL with several temporal functions. The τSQWRL language allows a τJOWL base administrator to query, in a user-friendly manner, temporal OWL 2 ontologies that are automatically constructed from temporal JSON big data (e.g., generated by IoHT sensor networks). In sum, the present paper deals with querying of temporal ontologies, corresponding to temporal big data, by focusing on the data side while assuming a separation of concerns: (1) the τJOWL approach to manage the temporal and multi-version ontology, to be used as a temporal and multi-version schema for such big data, and (2) a temporal database provided with a TSQL2-like query language to manage the time-varying and evolving big data generated by an IoHT infrastructure. For these reasons, the proposed language fills a gap in the state-of-the-art of temporal OWL 2 ontology instances querying, and completes the τJOWL framework with an appropriate temporal query language. As far as the examples of τSQWRL queries provided in Section 3.4 are concerned, they illustrate the use of the proposed language. In particular, they are conceived to give an idea on how a temporal ontology query language could be helpful when expressing (in a compact way) temporal complex queries in order to fulfill non-trivial users' requirements, in an IoHT context.

In the near future, we plan to finish the development and testing of the τSQWRL-Processor tool that supports the τSQWRL language; we intend to experimentally evaluate both the usability and the performances of this tool. The usability evaluation will be based on the assessment of the feedbacks that will be received from the large set of users who will be asked to use and test the τSQWRL-Processor tool during some reasonable period. The performance evaluation will be possibly based on the adoption of a readily available IoHT benchmark. If such a benchmark could not be found, a synthetic temporal knowledge base will be constructed, through the use of the τJOWL framework, and a set of τSQWRL queries expressed on such a knowledge base will be defined. Then, performance figures will be collected (i.e., by recording execution times) from the τSQWRL-Processor for the execution of such queries. In order to evaluate the performance scalability with respect to the growth of the multi-version ontology (as mentioned in goal (G5) of Section 3.2), queries will be executed on several successive versions of the knowledge base, such that each new version has a different size and introduces new atoms, with respect to the previous one. Moreover, we will also deal with other important aspects of the τSQWRL language, that

is its completeness and expressiveness (as mentioned in goal G6 of Section 3.2), using theoretical inspection and user feedbacks. Additionally, since τSQWRL currently supports only valid-time, we intend to extend it to also include transaction time[8], so that it can be used for querying also transaction-time and bitemporal OWL 2 ontology instances. Last but not least, optimization of τSQWRL queries also deserves a special attention and, thus, it will be taken into account in our future work. As a matter of fact, we plan to develop advanced cost-based optimization algorithms for the "Query Optimizer" module (introduced in Section 4) that allows to produce, for each τSQWRL query, an efficient query execution plan taking into account some important performance measures like Central Processing Unit (CPU) usage, memory usage, and query response time. Such an execution plan will be a sequence of operators that can be directly executed by the query execution engine. The design of such operators is currently under way.

References

[1] M. H. Böhlen, A. Dignös, J. Gamper, and C. S. Jensen, Database technology for processing temporal data (invited paper), in *Proc. 25th Int. Symp. on Temporal Representation and Reasoning*, Dagstuhl, Germany, 2018, pp. 2:1–2:7.

[2] W. Lu, Z. Zhao, X. Wang, H. Li, Z. Zhang, Z. Shui, S. Ye, A. Pan, and X. Du, A lightweight and efficient temporal database management system in TDSQL, *Proc. VLDB Endow.*, vol. 12, no. 12, pp. 2035–2046, 2019.

[3] F. Grandi, F. Mandreoli, R. Martoglia, and W. Penzo, Unleashing the power of querying streaming data in a temporal database world: A relational algebra approach, *Inf. Syst.*, vol. 103, p. 101872, 2022.

[4] Z. Brahmia, F. Grandi, and R. Bouaziz, Temporal Blockchains for intelligent transportation management and autonomous vehicles support in the internet of vehicles, in *Modelling and Simulation of Fast-Moving Ad-Hoc Networks* (*FANETs and VANETs*), T. S. Pradeep Kumar and M. Alamelu, eds. Hershey, PA, USA: IGI Global, 2023, pp. 155–189.

[5] S. Ketu and P. K. Mishra, Internet of healthcare things: A contemporary survey, *J. Netw. Comput. Appl.*, vol. 192, p. 103179, 2021.

[6] F. Grandi, Temporal databases, in *Encyclopedia of Information Science and Technology*, 3rd ed, M. Khosrow-Pour, ed. Hershey, PA, USA: Idea Group Reference, 2015, pp. 1914–1922.

[7] C. S. Jensen and R. T. Snodgrass, Temporal database, in *Encyclopedia of Database Systems*, 2nd ed, L. Liu and M. T. Özsu, eds. New York, NY, USA: Springer, 2018, pp. 3945–3949.

[8] C. S. Jensen and R. T. Snodgrass, Transaction time, in *Encyclopedia of Database Systems*, 2nd ed, L. Liu and M. T. Özsu, eds. New York, NY, USA: Springer, 2018, pp. 4200–4201.

[9] C. S. Jensen and R. T. Snodgrass, Valid time, in *Encyclopedia of Database Systems*, 2nd ed, L. Liu and M. T. Özsu, eds. New York, NY, USA: Springer, 2018, pp. 4359–4360.

[10] N. Guarino, *Formal Ontology in Information Systems*. Amsterdam, The Netherlands: IOS Press, 1998.

[11] F. Grandi, Multi-temporal RDF ontology versioning, in *Proc. 3rd Int. Workshop on Ontology Dynamics* (*IWOD*), Washington, DC, USA, 2009, pp. 1–10.

[12] F. Grandi and M. R. Scalas, The valid ontology: A simple OWL temporal versioning framework, in *Proc. 3rd Int. Conf. on Advances in Semantic Processing*, Sliema, Malta, 2009, pp. 98–102.

[13] V. Milea, F. Frasincar, and U. Kaymak, tOWL: A temporal web ontology language, *IEEE Trans. Syst. Man Cybern. B Cybern.*, vol. 42, no. 1, pp. 268–281, 2012.

[14] C. S. Jensen and R. T. Snodgrass, Temporal data models, in *Encyclopedia of Database Systems*, 2nd ed, L. Liu and M. T. Özsu, eds. New York, NY, USA: Springer, 2018, pp. 3940–3945.

[15] M. Klein and D. Fensel, Ontology versioning on the Semantic Web, in *Proc. 1st Int. Conf. Semantic Web Working*, Stanford, CA, USA, 2001, pp. 75–91.

[16] N. F. Noy and M. A. Musen, Ontology versioning in an ontology management framework, *IEEE Intell. Syst.*, vol. 19, no. 4, pp. 6–13, 2004.

[17] C. A. Weltyand R. Fikes, A reusable ontology for fluents in OWL, in *Proc. 4th Int. Conf. on Formal Ontology in Information Systems*, Baltimore, MD, USA, 2006, pp. 226–236.

[18] R. T. Snodgrass, *The TSQL2 Temporal Query Language*. Boston, MA, USA: Kluwer Academic Publishers, 1995.

[19] X. Jin, B. W. Wah, X. Cheng, and Y. Wang, Significance and challenges of big data research, *Big Data Res.*, vol. 2, no. 2, pp. 59–64, 2015.

[20] A. Ali, J. Qadir, R. ur Rasool, A. Sathiaseelan, A. Zwitter, and J. Crowcroft, Big data for development: Applications and techniques, *Big Data Anal.*, vol. 1, no. 2, pp. 2:1–2:24, 2016.

[21] A. Davoudian and M. Liu, Big data systems: A software engineering perspective, *ACM Comput. Surv.*, vol. 53, no. 5, p. 110, 2021.

[22] A. K. Sandhu, Big data with cloud computing: Discussions and challenges, *Big Data Mining and Analytics*, vol. 5, no. 1, pp. 32–40, 2022.

[23] IETF, The JavaScript Object Notation (JSON) Data Interchange Format, Internet Standards Track document, https://tools.ietf.org/html/rfc8259, 2022.

[24] L. Fegaras, Incremental query processing on big data streams, *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 11, pp. 2998–3012, 2016.

[25] C. W. Tsai, C. F. Lai, H. C. Chao, and A. V. Vasilakos, Big data analytics: A survey, *J. Big Data*, vol. 2, pp. 21: 1–21: 32, 2015.

[26] P. Ceravolo, A. Azzini, M. Angelini, T. Catarci, P. Cudré-Mauroux, E. Damiani, A. Mazak, M. Van Keulen, M. Jarrar, G. Santucci, et al., Big data semantics, *J. Data Semant.*, vol. 7, no. 2, pp. 65–85, 2018.

[27] R. M. Keller, S. Ranjan, M. Y. Wei, and M. M. Eshow, Semantic representation and scale-up of integrated air traffic management data, in *Proc. Int. Workshop on Semantic Big Data*, San Francisco, CA, USA, 2016, pp. 4: 1–4: 6.

[28] M. V. Nural, M. E. Cotterell, H. Peng, R. Xie, P. Ma, and J. A. Miller, Automated predictive big data analytics using ontology based semantics, *Int. J. Big Data*, vol. 2, no. 2, pp. 43–56, 2015.

[29] Z. Brahmia, F. Grandi, and R. Bouaziz, τJOWL: A systematic approach to build and evolve a temporal OWL 2 ontology based on temporal JSON big data, *Big Data Mining and Analytics*, vol. 5, no. 4, pp. 271–281, 2022.

[30] W3C, OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation, http://www.w3.org/TR/owl2-primer/, 2012.

[31] O. Etzioni, K. Golden, and D. S. Weld, Sound and efficient closed-world reasoning for planning, *Artif. Intell.*, vol. 89, nos. 1&2, pp. 113–148, 1997.

[32] R. Fikes, P. Hayes, and I. Horrocks, OWL-QL—a language for deductive query answering on the Semantic Web, *J. Web Semant.*, vol. 2, no. 1, pp. 19–29, 2004.

[33] W3C, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission, https://www.w3.org/Submission/SWRL/, 2004.

[34] M. J. O'Connor and A. K. Das, SQWRL: A query language for OWL, in *Proc. 6^{th} Int. Workshop on OWL: Experiences and Directions*, Chantilly, VA, USA, 2009, pp. 208–215.

[35] W3C, OWL 2 QL, in *OWL 2 Web Ontology Language Profiles* (*Second Edition*), https://www.w3.org/TR/owl2-profiles/#OWL_2_QL, 2012.

[36] M. J. O'Connor and A. K. Das, A lightweight model for representing and reasoning with temporal information in biomedical ontologies, in *Proc. 3^{rd} Int. Conf. on Health Informatics*, Valencia, Spain, 2010, pp. 90–97.

[37] M. J. O'Connor and A. K. Das, A method for representing and querying temporal information in OWL, in *Proc. 3^{rd} Int. Joint Conf. on Biomedical Engineering Systems and Technologies*, Valencia, Spain, 2010, pp. 97–110.

[38] SWRLAPITemporal functionality of the SWRLAPI Protégé project on GitHub, https://github.com/protegeproject/swrlapi/wiki/SWRLAPITemporal, 2022.

[39] A. Zekri, Z. Brahmia, F. Grandi, and R. Bouaziz, τOWL: A framework for managing temporal semantic web documents, in *Proc. 8^{th} Int. Conf. on Advances in Semantic Processing*, Rome, Italy, 2014, pp. 33–41.

[40] S. Batsakis, K. Stravoskoufos, and E. G. M. Petrakis, Temporal reasoning for supporting temporal queries in OWL 2.0, in *Proc. 15th Int. Conf. on Knowledge-Based and Intelligent Information and Engineering Systems*, Kaiserslautern, Germany, 2011, pp. 558–567.

[41] P. F. Patel-Schneider and I. Horrocks, A comparison of two modelling paradigms in the Semantic Web, *J. Web Semant.*, vol. 5, no. 4, pp. 240–250, 2007.

[42] W3C, RDF/XML syntax specification (revised), W3C recommendation, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/, 2004.

[43] W3C, OWL 2 Web Ontology Language document overview (second edition), W3C recommendation, http://www.w3.org/TR/owl2-overview/, 2012.

[44] C. S. Jensen and R. T. Snodgrass, Temporal query languages, in *Encyclopedia of Database Systems*, 2nd ed, L. Liu and M. T. Özsu, eds. New York, NY, USA: Springer, 2018, pp. 4023–4028.

[45] I. Seylan, E. Franconi, and J. De Bruijn, Effective query rewriting with ontologies over DBoxes, in *Proc. 21^{st} Int. Joint Conf. on Artificial Intelligence*, Pasadena, CA, USA, 2009, pp. 923–929.

[46] J. F. Allen, Maintaining knowledge about temporal intervals, *Commun. ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[47] C. Zaniolo, S. Ceri, C. Faloutsos, R. T. Snodgrass, V. S. Subrahmanian, and R. Zicari, *Advanced Database Systems*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997.

[48] K. Torp, Temporal strata, in *Encyclopedia of Database Systems*, 2nd ed, L. Liu and M. T. Özsu, eds. New York, NY, USA: Springer, 2018, pp. 4035–4040.

[49] E. Pitoura, Query optimization, in *Encyclopedia of Database Systems*, 2nd ed, L. Liu and M. T. Özsu, eds. New York, NY, USA: Springer, 2018, pp. 3008–3009.

[50] M. H. Böhlen, Temporal coalescing, in *Encyclopedia of Database Systems*, 2nd ed, L. Liu and M. T. Özsu, eds. New York, NY, USA: Springer, 2018, pp. 3917–3921.

[51] K. Kulkarni and J. E. Michels, Temporal features in SQL: 2011, *SIGMOD Rec.*, vol. 41, no. 3, pp. 34–43, 2012.

[52] J. Chomicki, D. Toman, and M. H. Böhlen, Querying ATSQL databases with temporal logic, *ACM Trans. Database Syst.*, vol. 26, no. 2, pp. 145–178, 2001.

[53] F. Grandi, Introducing an annotated bibliography on temporal and evolution aspects in the semantic web, *SIGMOD Rec.*, vol. 41, no. 4, pp. 18–21, 2012.

[54] E. Baratis, E. G. M. Petrakis, S. Batsakis, N. Maris, and N. Papadakis, TOQL: Temporal ontology querying language, in *Proc. 11^{th} Int. Symp. on Advances in Spatial and Temporal Databases*, Aalborg, Denmark, 2009, pp. 338–354.

[55] F. Grandi, T-SPARQL: A TSQL2-like temporal query language for RDF, in *Local Proc. 14^{th} East-European Conf. on Advances in Databases and Information Systems*, Novi Sad, Serbia, 2010, pp. 21–30.

[56] The SWRLAPI Protégé project, https://archive.is/GtlM5, 2022.

[57] The SWRL temporal ontology, https://github.com/protegeproject/swrlapi/wiki/SWRLTemporalOntology, 2014.

[58] The SWRLAPI temporal built-in library, https://github.com/protegeproject/swrlapi/wiki/SWRLTemporalBuiltInLibrary, 2014.

[59] M. O'Connor, The Semantic Web rule language, http://protege.stanford.edu/conference/2009/slides/SWRL2009ProtegeConference.pdf, 2022.

[60] K. Stravoskoufos, E. G. M. Petrakis, N. Mainas, S. Batsakis, and V. Samoladas, SOWL QL: Querying spatio-temporal ontologies in OWL, *J. Data Semant.*, vol. 5, no. 4, pp. 249–269, 2016.

[61] L. Zhu, N. Li, and L. Bai, Algebraic operations on spatiotemporal data based on RDF, *ISPRS Int. J. Geo-Inf.*, vol. 9, no. 2, pp. 80:1–80: 16, 2020.

[62] N. Maris, A reasoner for querying temporal ontologies, master dissertation, Dept. Electron. Comput. Eng., Tech.

Univ. Crete, Crete, Greece, 2009.

[63] C. E. Dyreson, Observing transaction-time semantics with TTXPath, in *Proc. 2nd Int. Conf. on Web Information Systems Engineering*, Kyoto, Japan, 2001, pp. 193–202.

[64] D. Gao and R. T. Snodgrass, Temporal slicing in the evaluation of XML queries, in *Proc. 29th Int. Conf. on Very Large Data Bases*, Berlin, Germany, 2003, pp. 632–643.

[65] F. Rizzolo and A. A. Vaisman, Temporal XML: Modeling, indexing, and query processing, *VLDB J.*, vol. 17, no. 5, pp. 1179–1212, 2008.

[66] Z. Brahmia, F. Grandi, S. Brahmia, and R. Bouaziz, τJSONPath: A temporal extension of the JSONPath language for the τJSchema framework, in *Proc. 4th Int. Conf. on Artificial Intelligence and Smart Environments* (*ICAISE*), Errachidia, Morocco, https://bdsde.sciencesconf.org/, 2022.

[67] W3C, XML Path language (XPath) 3.0, W3C Recommendation, https://www.w3.org/TR/xpath-30/, 2014.

[68] W3C, XQuery 3.1: An XML query language, W3C recommendation, https://www.w3.org/TR/2017/REC-xquery-31-20170321/, 2017.

[69] F. Currim, S. Currim, C. Dyreson, and R. T. Snodgrass, A tale of two schemas: Creating a temporal XML schema from a snapshot schema with τXSchema, in *Proc. 9th Int. Conf. on Extending Database Technology*, Crete, Greece, 2004, pp. 348–365.

[70] IETF, JSONPath: Query expressions for JSON, internet-draft, https://datatracker.ietf.org/doc/draft-ietf-jsonpath-base/, 2022.

[71] S. Brahmia, Z. Brahmia, F. Grandi, and R. Bouaziz, τJSchema: A framework for managing temporal JSON-based NoSQL databases, in *Proc. 27th Int. Conf. on Database and Expert Systems Applications*, Porto, Portugal, 2016, pp. 167–181.

[72] A. Dignös, M. H. Böhlen, J. Gamper, and C. S. Jensen, Extending the kernel of a relational DBMS with comprehensive support for sequenced temporal queries, *ACM Trans. Database Syst.*, vol. 41, no. 4, pp. 26: 1–26: 46, 2016.

[73] L. Carafoli, F. Mandreoli, R. Martoglia, and W. Penzo, Streaming tables: Native support to streaming data in DBMSs, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 47, no. 10, pp. 2768–2782, 2017.

Zouhaier Brahmia received the BSc, MSc, and PhD degrees in computer science from University of Sfax, Tunisia in 2003, 2005, and 2011, respectively; he is currently an associate professor at the Department of Computer Science, Faculty of Economics and Management, University of Sfax. His research interests include temporal databases, database schema versioning, temporal, evolution, and versioning aspects in emerging (XML, NoSQL, etc.) databases, big data, Semantic Web ontologies, knowledge representation, IoT data management, and blockchains.

Rafik Bouaziz received the PhD degree in computer science from the University of Tunis El Manar, Tunis, Tunisia in 1991, and a Habilitation in computer science from the University of Sfax, Sfax, Tunisia in 2007. He was the director of the Economy, Management, and Computer Science Doctoral School, University of Sfax, between 2011 and 2014, and the president of the same university between 2014 and 2017; he is currently a full professor at the Department of Computer Science, Faculty of Economics and Management, University of Sfax. His research interests include temporal databases, real-time databases, information systems engineering, ontologies, and data warehousing and workflows.

Fabio Grandi received a Laurea degree cum Laude in electronics engineering from the University of Bologna, Italy in 1988, and the PhD degree in electronics engineering and computer science in 1994; from 1989 to 2012 he worked at the CSITE center of the Italian National Research Council in Bologna in the field of neural networks and temporal databases, initially supported by a CNR fellowship. In 1993 and 1994 he was an adjunct professor at the University of Ferrara, Italy. In 1994 he was appointed as a research associate at the University of Bologna. Since 1998 he has been an associate professor at the Department of Computer Science and Engineering, University of Bologna. He published more than 100 papers in scholarly journals and conference proceedings. He is a member of the TSQL2 Language Design Committee and the co-author of the book “The TSQL2 Temporal Query Language”. His scientific interests include temporal, evolution, and versioning aspects in data management, WWW and Semantic Web, knowledge representation, and storage structures and access cost models.