Semantic Management of Streaming Data

Alejandro Rodríguez, Robert McGrath, Yong Liu, and James Myers

National Center for Supercomputing Applications
University of Illinois
Urbana-Champaign, IL
{alejandr,mcgrath,yongliu,jimmyers}@ncsa.uiuc.edu

Abstract

One of the fundamental challenges facing the unprecedented data deluge produced by the sensor networks is how to manage time-series streaming data so that they can be reasoning-ready and provenance-aware. Semantic web technology shows great promise but lacks adequate support for the notion of time. We present a system for the representation, indexing and querying of time-series data, especially streaming data, using the semantic web approach. This system incorporates a special RDF vocabulary and a semantic interpretation for time relationships. The resulting framework, which we refer to as Time-Annotated RDF, provides basic functionality for the representation and querying of time-related data. The capabilities of Time-Annotated RDF were implemented as a suite of Java APIs on top of Tupelo, a semantic content management middleware, to provide transparent integration among heterogeneous data, as present in streams and other data sources, and their metadata. We show how this system supports commonly used time-related queries using Time-Annotated SPARQL introduced in this paper as well as an analysis of the TA-RDF data model. Such prototype system has already seen successful usage in a virtual sensor project where near-real-time radar data streams need to be fetched, indexed, processed and re-published as new virtual sensor streams.

Keywords: Semantic Web, RDF, Query Processing, Sensor Networks, Data Models, Temporal Databases, Time Series Observations

1 Introduction

In this paper we present a semantic stream manager aimed at the problem of integrating streaming data into RDF and semantic-web enabled infrastructure and applications. We introduce TA-RDF, a formal model to semantically represent streams and a software implementation to efficiently access those streams.

This has enabled integration of stream data with other data and metadata, including documents, imagery, workflows, provenance, and annotations. The TA-RDF model remains translatable/compatible with regular RDF by means of an special RDF vocabulary, which allows the use of existing RDF tools and technologies.

The necessity for such model becomes evident when observing the rapid development and deployment of various scales of sensor networks for various observation purposes, which have resulted in an unprecedented data deluge. One of the major challenges is to support various temporal relationship queries. In addition, sensor data streams publishing and re-publishing after certain processing require provenance support for validation and verification [16]. It would be advantageous to manage streaming data and their provenance (meta data about the causal relationship, history linkage etc.) coherently.

Semantic web technology has emerged to be a potential tool that might serve such purpose. The Resource Description Framework (RDF) was introduced by the W3C as a model and language to represent semi-structured metadata in the semantic web. In this model, information is represented via subject-predicate-object *assertions*, or triples, that indicate the value of user-defined properties for given *resources*. This essentially gives RDF an expressive power equivalent to the binary existential-conjunctive subset of first order logic. Although this might be sufficient for many applications, RDF omits all considerations of time and time-varying information from its model.

Thus, it is not surprising that this limitation becomes noticeable in sensor network data management, where time-varying data is at the core. In this paper, we present a semantic stream manager, both a model to semantically represent streams and a software implementation to efficiently access those streams, aimed at the problem of integrating this type of streaming data into RDF with the purpose of exploiting it in semantic-web applications and applications that benefit the management of provenance of data through semantic annotation. This will effectively enable the realization of managing streaming data and provenance coherently. Note that these data usually have three main characteristics:

- They are updated frequently, possibly at a high throughput.
- They can be modeled as a time-changing property or set of properties for given resources. For instance, the air temperature of a certain location.
- The values, or instances, of these properties can be serialized. That is, they can be ordered according to some timestamp taken from a discrete, totally ordered domain.

In practice it is sometimes the case that several of these properties are related to the same resources and share timestamps as sensors are adjusted to acquire data at the same regular intervals. For example, the temperature and humidity of a location can be modeled as properties of the same resource, measured at the same regular intervals. Our system is specifically designed to handle these characteristics by facilitating the representation of streams in RDF, but it is general enough to incorporate other time-related data, e.g. time-ordered events.

One typical requirement of streaming data management systems is the processing of continuous queries [1], i.e., queries which answer is itself a stream, often produced in real time as an aggregating computation performed over a moving window of data on one or more streams. As our focus is in the representation, storage and retrieval of time-series information, and consider the summarization and continuous processing of streams to be at a higher layer in the application stack, we do not consider this type of queries and instead focus on one-time queries, or queries which (conceptually) produce an answer immediately that depends only on the data currently in the system. For example: *What was the average temperature in Chicago during December 2008?*

The remainder of the paper is organized as follows. Section 2 gives a brief review of related work. Section 3 states a motivating example. Section 4 and 5 develop Time-Annotated RDF (TA-RDF), a formal extension to RDF. Section 6 develops TA-SPARQL, extending SPARQL. Section 7 discusses the prototype implementation. Section 8 concludes.

2 Related Work

Recent years have seen an increase in the interest to bring temporal semantics into the Semantic Web. One of the earlier works is that by Buraga et al. [4], which introduces an XML-based language for the use of Interval Temporal Calculus to express temporal relations between web-sites. Gutierrez et al. [6] provides a complete syntax and semantics, along with a query language specification, for temporal RDF, an RDF extension where every statement is timestamped with the point in time in which it is semantically valid. This work was later extended by a different group [15] with indexing and query processing algorithms specifically suited to temporal RDF.

On the specific topic of streaming data, Bolles et al. [3] represents a stream as sequence of RDF graphs, similar to other works [10], and presents a SPARQL extension for streams of RDF. This approach reduces the problem of querying a stream to querying over a user-specified (sliding) window on the stream, but has the disadvantage of ignoring the richer time semantics of timestamped but non-simultaneous events. Other work [8, 13] has focused on the ontological representation of time series sensor data, but has not included a model that allows the efficient evaluation of time-related queries.

Our work introduces a new semantic and indexing strategy for time in RDF conceived to minimize impact on current RDF storage and query engines, and tailored to streaming applications where actual, arbitrary data is stored outside of the scope of RDF (and indicated in RDF metadata simply as URIs), and these data change rapidly over time while metadata annotations might change at a different, lower pace. Our approach allows the seamless integration of both types of data into a single indexing system.

Location	Sensor	Units	Reading	Timestamp
Chicago O'Hare	Temperature	Celsius	20	2008-06-15
Chicago O'Hare	Temperature	Celsius	25	2008-06-16
		⋮		
Chicago O'Hare	Temperature	Celsius	25	2008-07-10
Chicago O'Hare	Daily Rainfall Accumulation	mm	0	2008-06-14
Chicago O'Hare	Daily Rainfall Accumulation	mm	15	2008-06-16
		⋮		
Chicago O'Hare	Daily Rainfall Accumulation	mm	5	2008-08-12

Location	Timestamp	Sensor 1	Reading 1	Sensor 2	Reading 2
Champaign	2008-06-15	Temp. (C)	30	Rainfall Acumm. (mm)	15
Champaign	2008-06-16	Temp. (C)	25	Rainfall Acumm. (mm)	0
			⋮		
Champaign	2008-07-16	Temp. (C)	25	Rainfall Acumm. (mm)	10

Figure 1: Example time-series data

3 Motivating Example

Consider the example on Figure 1. It shows a sample of time-series data from several sensors[1]. Typical queries that could arise over such information could be:

1. What was the total amount of rainfall accumulation in Chicago in January 2009?
2. What was the average temperature in the month of December 2008?
3. What was the temperature at Chicago at sunrise July 20th 2008?
4. When was the temperature at Champaign IL over 80 degrees?
5. What is the current temperature at O'Hare airport?
6. How many heavy rains occurred within a week of a heat wave?

These queries cover the three main categories of temporal queries identified by Yuan et al. [17], being those range queries (1,2), point queries (3,4,5), and time-relationship queries (6). Although it is possible to readily represent this information in RDF, the nature of RDF (where data are separated from their semantics) implies that the special meaning of all time markers as an ordering domain for related resources will be obscured from an underlying RDF management system and query engine, making the queries above especially expensive to perform.

One way to ease this problem is by incorporating into the representation an indication that certain resources are related by virtue of being different instances

[1]Chicago and Champaign are considered here simply as named concepts. The problem of modeling geo-spatial semantics is out of the scope of this paper.

over time of the same entities, and that these instances are to be differentiated (and indexed) by their ordered time markers. In our system, this is achieved by means of a special RDF vocabulary, the data stream vocabulary (dsv), and associated semantics, that allows a query engine to correctly interpret the time markers as indexing keys.

However, the use of this vocabulary introduces an extra overhead in storage by increasing the number of triples that need to be employed for representing the information. For this reason, we introduce an RDF extension, dubbed Time-Annotated RDF, where the additional semantic restrictions of the data stream vocabulary are implicitly represented without the requirement for extra triples.

4 Time-Annotated RDF

As reference and way of comparison with our model, we first briefly present some RDF basic concepts (found in larger detail elsewhere [7, 9]). RDF represents subsets of known information about a domain in RDF *graphs*, which are collections of triples. A *triple* is a tuple of the form $< Subject, Predicate, Object >$ and asserts a known property of a resource or a relationship between two resources, where *Subject* is a URI reference or a blank node, *Predicate* is a URI reference, and *Object* is a URI reference, blank node or literal. URI references name resources that can be directly identified by their URI. Additionally, RDF allows the use of *blank nodes*. Syntactically, a blank node is a resource identified by a local identifier instead of URI reference. Semantically, a blank node is an existential variable that allows to express information about a resource without identifying it.

Time-annotated RDF (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 $resource \in URI\_References \cup Blank\_Nodes \cup Literal$ and $time \in T = \mathcal{T} \cup \{Nil\}$. The time domain $\mathcal{T}$ is a discrete, totally ordered infinite set intended to represent discrete time and from now on 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.

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.

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 $t \in T$. For simplicity we restrict the model to point timestamps on this paper. This definition of streams accommodates for representing sequential or time-series data with arbitrary metadata and payload data, frames sampled at different or irregular intervals, missing data points, etc.

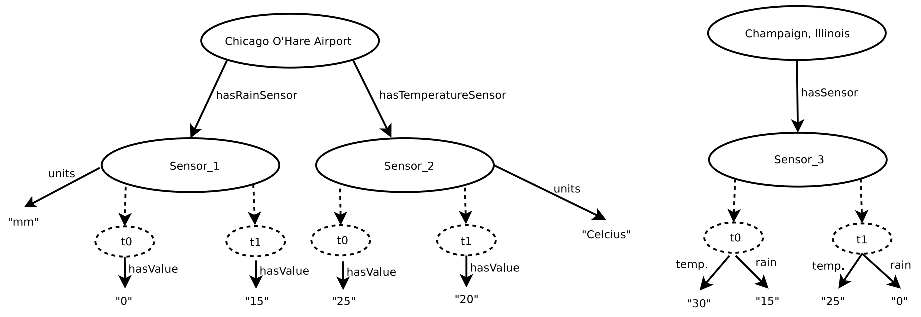

Figure 2: Example of TA-RDF data

The inspiration for time-annotating resources instead of triples comes from the fact 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.

5 Model Theory

5.1 TA-RDF Interpretation

The model theory for TA-RDF extends directly of that one for RDF, as defined in RDF Semantics [7]. This extension consists of modifying interpretations to consider a time-annotated resource as a new resource, and triples as being composed of these time-annotated resources. An RDF interpretation I of an RDF graph is a function that maps URI references, literals and blank nodes in V, the vocabulary of the graph, to objects in a set of resources IR. It also maps triples in $V \times V \times V$ to $\{true, false\}$.

We extend an interpretation as follows; if T is the time domain, then the interpretation of $r[t]$ where r is a URI reference or literal and $t \in T$, is $I(r[t]) = IS(< r, t >)$, with IS being a mapping from $(URI \cup L) \times T$ to IR that forms part of the interpretation I. Mapping of blank nodes to URI references are extended in the obvious way to map to $(URI \cup L) \times T$ instead. All other aspects of an interpretation remain as in an RDF interpretation, except for an additional semantic restriction to indicate that Nil represents all points in time for a given resource. That restriction is formalized in the following definitions.

Definition 1 (Coverage of resources)**.** Given two time-annotated resources $r_1[t_1]$ and $r_2[t_2]$, we say $r_1[t_1]$ **covers** $r_2[t_2]$ ($r_1[t_1] \succeq r_2[t_2]$) iff $I(r_1) = I(r_2)$ and either $t_1 = t_2$ or $t_1 = Nil$.

Definition 2 (Coverage of triples)**.** Given two time-annotated triples $triple_1 =< s_1[t_1^s], p_1[t_1^p], o_1[t_1^o] >$ and $triple_2 =< s_2[t_2^s], p_2[t_2^p], o_2[t_2^o] >$, then $triple_1$ **covers** $triples_2$ ($triple_1 \succeq triples_2$) iff $s_1[t_1^s] \succeq s_2[t_2^s] \wedge p_1[t_1^p] \succeq p_2[t_2^p] \wedge o_1[t_1^o] \succeq o_2[t_2^o]$.

Table 1: Data Stream Vocabulary

Name	Type	Description
dsv:belongsTo	Property	Indicates a resource is a frame in a stream
dsv:hasTimestamp	Property	Points toward the timestamp of the frame
dsv:Nil	Resource	The optional Nil timestamp

With these definitions, we impose over a TA-RDF interpretation the condition that for any two triples $triple_1, triples_2$, it holds that

$$(I(triple_1) = TRUE \wedge triple_1 \succeq triple_2) \Rightarrow I(triple_2) = TRUE$$

5.2 Compatibility with RDF

It is possible to translate between TA-RDF and RDF by explicit use of the data stream vocabulary, presented at Table 1, as long as the additional semantic restrictions are preserved. Semantically, two sets of restrictions are needed:

1. A property of a stream holds for all its frames:

$$\begin{array}{ll} (i) & I(<\mathbf{s}, p, o>) \wedge I(<r, \text{dsv:belongsTo}, \mathbf{s}>) \Rightarrow I(<r, p, o>) \\ (ii) & I(<r, p, o>) \wedge I(<r, \text{dsv:belongsTo}, \mathbf{s}>) \wedge \\ & \qquad I(<r, \text{dsv:hasTimestamp}, \text{dsv:Nil}>) \Rightarrow I(<\mathbf{s}, p, o>) \end{array}$$

Analogous for $<s, \mathbf{s}, o>$ and $<s, p, \mathbf{s}>$

2. All frames have a timestamp and are unique, in the sense that no frame belongs to more than one stream, no frame has more than one timestamp and no two frames of the same stream has the same timestamp:

$$\begin{array}{ll} (i) & I(<r_0, dsv : belongsTo, s_0>) \Rightarrow \exists t \in T[I(<r_0, dsv : hasTimestamp, t>)] \\ (ii) & (I(<r_0, dsv : belongsTo, s_0>) \wedge I(<r_0, dsv : belongsTo, s_1)) \\ & \Rightarrow I(s_0) = I(s_1) \\ (iii) & (I(<r_0, dsv : hasTimestamp, t_0>) \wedge I(<r_0, dsv : hasTimestamp, t_1>)) \\ & \Rightarrow t_0 = t_1 \\ (iv) & (I(<r_0, dsv : hasTimestamp, t>) \wedge I(<r_1, dsv : hasTimestamp, t>) \wedge \\ & \quad I(<r_0, dsv : belongsTo, s>) \wedge I(<r_1, dsv : belongsTo, s)) \\ & \quad \Rightarrow I(r_0) = I(r_1) \end{array}$$

Figure 3 shows an example translation of the TA-RDF graph in Figure 2. The equivalence between TA-RDF and RDF extended with the data stream vocabulary is attained through the translation process, that maps between RDF graphs and TA-RDF graphs, and is formalized in the following definitions:

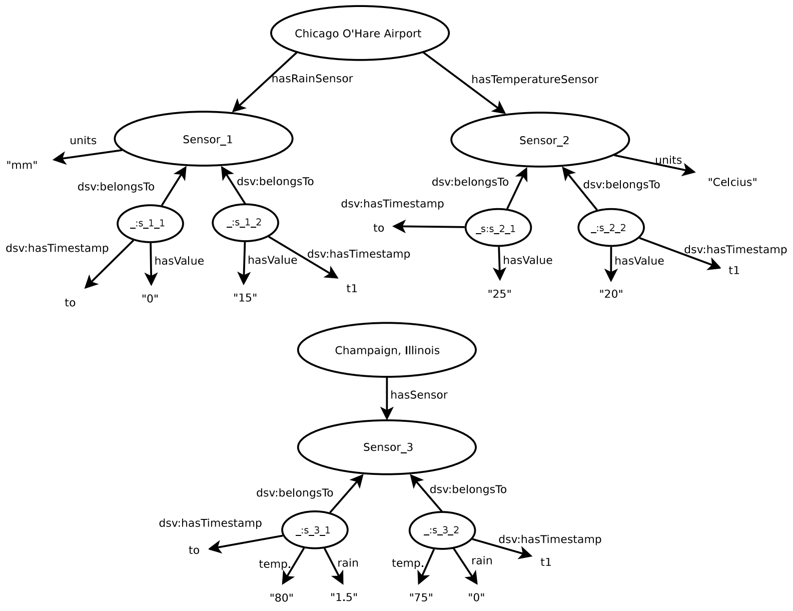

Figure 3: RDF graph of the translation of the TA-RDF data.

Definition 3 (Syntactic Translation)**.** Let G^{TA} be a TA-RDF graph, and G an RDF graph, then G is the **translation** of G^{TA} (written as $\mathrm{G}^{TA} \rightsquigarrow G$) iff

$$
\begin{aligned}
&< s[t_s], p[t_p], o[t_o] > \in \mathrm{G}^{TA} \Leftrightarrow \exists r_s, r_p, r_o \\
&\quad [(< r_s, \text{dsv:belongsTo}, s > \in G \wedge < r_s, \text{dsv:hasTimestamp}, t_s > \in G \wedge r_s \in B) \vee \\
&\qquad (t_s = \text{dsv:Nil} \wedge r_s = s)] \wedge \\
&\quad [(r_p < r_p, \text{dsv:belongsTo}, p > \in G \wedge < r_p, \text{dsv:hasTimestamp}, t_p > \in G \wedge r_p \in B) \vee \\
&\qquad (t_p = \text{dsv:Nil} \wedge r_p = p)] \wedge \\
&\quad [(r_o < r_o, \text{dsv:belongsTo}, o > \in G \wedge < r_o, \text{dsv:hasTimestamp}, t_o > \in G \wedge r_o \in B) \vee \\
&\qquad (t_o = \text{dsv:Nil} \wedge r_o = o)] \wedge \\
&\quad < r_s, r_p, r_o > \in G
\end{aligned}
$$

Where B is the set of blank nodes. Notice that defining translation as a relation, rather than a function, allows triples such as $< r[Nil], \_, \_ >$ (omitting the required translations for predicate and object), to be represented in RDF in two different ways: $< r, \_, \_ >$ and $\{< r', \_, \_ >, < r', dsv : belongsTo, r >, < r', dsv : hasTimestamp, dsv : Nil >\}$.

Definition 4 (Semantic Translation)**.** Let I be an RDF interpretation and I^{TA} a TA-RDF interpretation. Then I is the **translation** of I^{TA} (written as $\mathrm{I}^{TA} \rightsquigarrow I$) iff

$$
\begin{aligned}
&\mathrm{I}^{TA}(< s[t_s], p[t_p], o[t_o] >) \Leftrightarrow \exists r_s, r_p, r_o \\
&\quad [(I(< r_s, \text{dsv:belongsTo}, s >) \wedge I(< r_s, \text{dsv:hasTimestamp}, t_s >)) \\
&\quad\quad \vee (I(t_s) = I(\text{dsv:Nil}) \wedge I(r_s) = I(s))] \wedge \\
&\quad [(I(< r_p, \text{dsv:belongsTo}, p >) \wedge I(< r_p, \text{dsv:hasTimestamp}, t_p >)) \\
&\quad\quad \vee (I(t_s) = I(\text{dsv:Nil}) \wedge I(r_p) = I(p))] \wedge \\
&\quad [(I(< r_o, \text{dsv:belongsTo}, o >) \wedge I(< r_o, \text{dsv:hasTimestamp}, t_o >)) \\
&\quad\quad \vee (I(t_o) = I(\text{dsv:Nil}) \wedge I(r_o) = I(o))] \wedge \\
&\quad\quad I(< r_s, r_p, r_o >)
\end{aligned}
$$

Lemma 1. *All TA-RDF graphs have a translation. All valid RDF interpretations and TA-RDF interpretations have a translation.*

The following theoretical results establish the translation process to be unique up to simple entailment, reversible, and consistent between semantic translation and syntactic translation. Proofs are omitted from this paper due to space constraints.

The first theorem establishes that the translation from TA-RDF to RDF is unique up to simple entailment.

Theorem 1. *Let* G^{TA} *be a TA-RDF graph and* G_1, G_2 *RDF graphs. Then*

$$(\mathrm{G}^{TA} \rightsquigarrow G_1 \wedge \mathrm{G}^{TA} \rightsquigarrow G_2) \Rightarrow (G_1 \models G_2 \wedge G_2 \models G_1)$$

The translation from RDF to TA-RDF is also unique as shown by the following theorem. Remember that in TA-RDF as in RDF, equality of graphs ignores the renaming of blank nodes.

Theorem 2. *Let* $\mathrm{G}_1^{TA}, \mathrm{G}_2^{TA}$ *be TA-RDF graphs and* G_1 *an RDF graph. Then*

$$(\mathrm{G}_1^{TA} \rightsquigarrow G_1 \wedge \mathrm{G}_2^{TA} \rightsquigarrow G_1) \Rightarrow (\mathrm{G}_1^{TA} = \mathrm{G}_2^{TA})$$

Finally, the two following results guarantee that translated graphs are semantically equivalent.

Theorem 3. *Let* $\mathrm{G}^{TA}, \mathrm{I}^{TA}$ *be a TA-RDF graph and a TA-RDF interpretation, respectively, and* G, I *be an RDF graph and an RDF interpretation, such that* $\mathrm{G}^{TA} \rightsquigarrow G$ *and* $\mathrm{I}^{TA} \rightsquigarrow I$*. Then* I^{TA} *satisfies* G^{TA} *iff* I *satisfies* G*.*

Corollary 1. *Let* G_1, G_2 *be RDF graphs and* $\mathrm{G}_1^{TA}, \mathrm{G}_2^{TA}$ *TA-RDF graphs, such that* $\mathrm{G}_1^{TA} \rightsquigarrow G_1$ *and* $\mathrm{G}_2^{TA} \rightsquigarrow G_2$*, then*

$$G_1 \models G_2 \Leftrightarrow \mathrm{G}_1^{TA} \models \mathrm{G}_2^{TA}$$

The translation process for graphs requires the differentiation of resources representing frames, and to guarantee the reversibility of the translation new semantics restrictions have been imposed over RDF for the TA-RDF vocabulary.

However, in practical applications, constructing the translation of a TA-RDF graph could create “new”, globally unused URI references for each frame, eliminating the need for the additional semantic rules in the vocabulary regarding uniqueness of frames, which increases the compatibility with existent RDF storage and query systems.

6 Time-Annotated SPARQL

Along with TA-RDF, we present a query language based on SPARQL [14], a W3C recommendation for querying RDF. Like TA-RDF, Time-annotated SPARQL (TA-SPARQL) is designed to stay as similar as possible to its pre-existing 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. We present the language through several examples before listing all the constructs that are new in TA-SPARQL vs SPARQL. Using the example of Section 3, we write in TA-SPARQL some of the queries introduced in that section:

What was the total amount of rain fell on Chicago in January 2009?

```
SELECT sum(?rain)
WHERE {
  <urn:OHARE> <urn:hasRainSensor>
      ?x["2009-01-01Z-06:00"^^xsd:date ..
         "2009-01-31Z-06:00"^^xsd:date] .
  ?x          <urn:hasReading>          ?rain .
}
```

What was the temperature on Chicago at sunrise of July 20th 2008?

```
SELECT ?temperature
WHERE {
  <urn:OHARE> <urn:hasTemperatureSensor>
      ?x["2008-07-20T05:34:00Z-06:00"^^xsd:dateTime] .
  ?x          <urn:hasReading>          ?temperature .
}
```

When was the temperature at Champaign, IL over 80 degrees?

```
 SELECT ?x.t
 WHERE {
   <urn:Champaign> <urn:hasSensor>            ?x[*] .
   ?x              <urn:hasTemperatureReading> ?y .
   FILTER (?y>=80)
 }
```

What is the current temperature at O’Hare?

```
SELECT ?temperature
WHERE {
  <urn:OHARE> <urn:hasTemperatureSensor> ?x[LAST] .
  ?x          <urn:hasReading>          ?temperature .
}
```

How many heavy rains occurred within a week of a heat wave?

```
SELECT count(?y)
WHERE {
  {
    <urn:Champaign> <urn:hasSensor>             ?x[*] .
    ?x              <urn:hasTemperatureReading> ?temp .
    FILTER(?temp > 35)
  }
  { <urn:Champaign> <urn:hasSensor>       ?y[*] .
    ?y              <urn:hasRainReading>  ?rain .
    FILTER (?rain > 25 && abs(?x.t - ?y.t) < 7*86400000)
  }
}
```

Notice that times are converted to UTC in milliseconds for comparisons and numeric operations (hence 7 days $\rightarrow$ 7 days*86400000 ms/day).

The semantics of TA-SPARQL queries are informally described in Table 2 by providing translations from TA-SPARQL queries (or segments of queries) to equivalent versions in SPARQL, assuming the SPARQL queries are performed over the translation of the TA-RDF graph. The translations in do not represent the algebraic or physical operations required to evaluate the queries, they just show a semantically equivalent form of the query in order to describe its meaning. TA-SPARQL also provides a simple form of aggregation for functions SUM, AVG, MIN, MAX and COUNT implicitly grouped by all the projected variables not present in an aggregation function, meaning $SUM(?y)$ is the summation of variable ?y for all results in the query for which the remaining variables have identical values.

Table 2: Translation from TA-SPARQL to SPARQL

TA-SPARQL	SPARQL
Specifying resource in a triple by timestamp	
$\{ \quad \ldots$ $resource[t_0]\ \ predicate\ \ object\ \ .$ $\ldots \quad \}$	$\{ \quad \ldots$ $?F\ \ predicate\ \ object\ \ .$ $?F\ \ dso:belongsTo\ \ resource\ \ .$ $?F\ \ dso:hasTimestamp\ \ ?F\_T\ \ .$ $\ldots$ $FILTER(?F\_T = t_0) \quad \}$
Specifying resource in a triple by range	
$\{ \quad \ldots$ $resource[t_0..t_1]\ \ predicate\ \ object\ \ .$ $\ldots \quad \}$	$\{ \quad \ldots$ $?F\ \ predicate\ \ object\ \ .$ $?F\ \ dso:belongsTo\ \ resource\ \ .$ $?F\ \ dso:hasTimestamp\ \ ?F\_T\ \ .$ $\vdots$ $FILTER(?F\_T >= t_0\&\&?F\_T <= t_1) \quad \}$
Specifying resources with a timestamp in any stream	
$\{ \quad \ldots$ $?x[t_0]\ \ predicate\ \ object\ \ .$ $\vdots \quad \}$	$\{ \quad \vdots$ $?F\ \ predicate\ \ object\ \ .$ $?F\ \ dso:belongsTo\ \ ?x\ \ .$ $?F\ \ dso:hasTimestamp\ \ ?F\_T\ \ .$ $\ldots$ $FILTER(?F\_T >= t_0\&\&?F\_T <= t_1) \quad \}$ Replace all other occurrences of $?x$ by $?F$

Continued on Next Page...

TA-SPARQL	SPARQL
Most recent frame in a stream	
	$\{ \quad \ldots$
	$?F\ \ predicate\ \ object\ \ .$
	$?F\ \ dso: belongsTo\ \ resource\ \ .$
$\{ \quad \ldots$	$?F\ \ dso: hasTimestamp\ \ ?F\_T\ \ .$
$resource[LAST]\ \ predicate\ \ object\ \ .$	$OPTIONAL\{$
$\ldots \quad \}$	$?F\_2\ \ dso: belongsTo\ \ resource\ \ .$
Similarly for $resource[FIRST]$	$?F\_2\ \ dso: hasTimestamp\ \ ?F\_2\_T\ \ .$
	$FILTER(F\_T < \|F\_2\_T)$
	$\}$
	$\ldots$
	$FILTER(!BOUND(F\_2\_T)) \quad \}$
All frames in a stream	
	$\{ \quad \ldots$
	$?F\ \ predicate\ \ object\ \ .$
$\{ \quad \ldots$	$?F\ \ dso: belongsTo\ \ resource\ \ .$
$resource[*]\ \ predicate\ \ object\ \ .$	$\}UNION\{$
$\ldots \quad \}$	$resource\ \ predicate\ \ object\ \ .$
	$\}$
	$\ldots \quad \}$
Referring to the timestamp of a frame	
$\{ \quad \ldots$	$\{ \quad \ldots$
$resource[*]- >?x\ \ predicate\ \ object\ \ .$	$?x\ \ predicate\ \ object\ \ .$
	$?x\ \ dso: belongsTo\ \ resource\ \ .$
$\vdots$	$?x\ \ dso: hasTimestamp\ \ ?x\_T\ \ .$
	$\ldots$
$FILTER(?x.t >?z) \quad \}$	$FILTER(?x\_T >=?z) \quad \}$
Aggregation	
$SELECT\ ?x\ ?y\ SUM(?z)$	
$WHERE \ldots$	Results grouped by (?x,?y), no SPARQL equivalent.
Similarly for $MIN, MAX, AVG, COUNT$	

7 Example Implementation

We developed an implementation of TA-RDF and TA-SPARQL based on Tupelo semantic middleware[2]. Tupelo is an extensible semantic content repository framework for the storage and retrieval of RDF data that transparently interfaces RDF stores such as Sesame or Mulgara. The Semantic Data Stream Manager (SDSM) is built on top of Tupelo to provide an interface and broker between Tupelo and streams. Figure 4 shows the general architecture of the system.

The SDSM incorporates the semantics of TA-RDF on top of preexisting RDF storage managers to allow for indexing of streams and time related data, in particular the system is especially optimized to answer range and point queries as those exemplified by queries 1 through 5 in Section 3. This is achieved by indexing all time annotated resources by their timestamp and the stream to which they belong. The time annotation index implicitly stores all the triples associated with the data stream vocabulary, so there is no extra overhead in the amount of triples stored by Tupelo.

The SDSM defines a Java API for writing and querying stream data. An application writes data to the SDSM, which extracts time stamps and other metadata, and writes the data to the stream. The time stamps are indexed by the SDSM, and other metadata written to Tupelo. Tupelo manages the data

[2]Tupelo. http://tupeloproject.ncsa.uiuc.edu

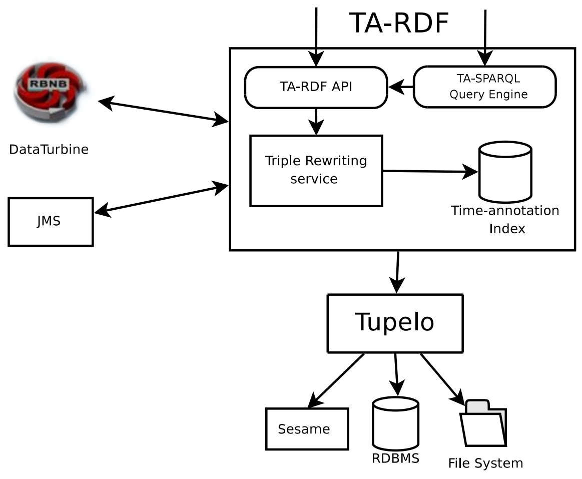

Figure 4: General architecture of the Semantic Data Stream Manager.

as a blobs, identified by unique URIs. TA-SPARQL queries are sent to the SDSM, which resolves them. Other queries are sent to Tupelo. Data blobs corresponding to frames of arbitrary data in a stream are grouped together in pages in a user-defined granularity that can be set independently for any group of incoming frames. Missing and out-of-sequence frames can still be handled and indexed correctly, although the performance degrades as frames out of sequence increase the segmentation of the pages.

Tupelo allows the integration of data and metadata by handling not only triples but also resources as blobs of data that can be independently stored and retrieved. Time-annotating and indexing resources through TA-RDF on top of Tupelo provides a mechanism to handle time-varying data that possess little or no time-varying metadata, while still allowing the extensive use of metadata when needed.

Given the real-time nature of streaming data in many real world applications, the data stream manager integrates high throughput streaming systems like Open Source DataTurbine [5] and publish/subscribe systems like Java Messaging System (JMS). DataTurbine is a high performance streaming system that allows consumers and producers to process data at different speeds without data loss. DataTurbine also abstracts the exact nature of the possibly heterogeneous data sources. SDSM allows fetching and posting data transparently through these systems with the purpose of interfacing with a variety of real-time sources. To read from DataTurbine, an application opens a connection via the SDSM, and issues TA-SPARQL queries. The SDSM and Tupelo fetch the correct data via DataTurbine. To post data to DataTurbine, the application writes data to the SDSM, which creates the appropriate RDF metadata and posts the data to DataTurbine.

Similarly, JMS is Java standard for messaging systems that include facilities for event-based communication through the publish/subscribe paradigm. This interface allows client applications of SDSM to continuously fetch updates on streams. Notice that this does not provide a general continuous query systems as arbitrary TA-SPAQRL queries are not allowed through this interface.

In addition, Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) standards have seen great adoption among many sensor network communities. We provide a web interface that returns the supported temporal queries in SWE Observation and Measurement XML-compliant format, which allows interoperability with other SWE-compliant tools.

The TA-SPAQRL implementation is a proof of concept implementation that does not incorporate query optimization. Therefore we do not provide experimental results on the performance of query evaluation. However, Tupelo provides programmatic access to RDF (and TA-RDF) data, which makes the implementation completely functional.

This system can easily be integrated into workflow managers as an input/output mechanism for data streams and time series data, for readily access to these data independently of the possibly heterogeneous data sources and linking to their corresponding metadata. We have incorporated the system into Cyberintegrator [2], a scientific workflow engine, highly extensible through the use of user-provided workflow tasks, that maintains provenance of all data employed by extensive use of semantic annotations in RDF.

An early version of the SDSM has already been deployed and is currently in use in the context of the NCSA digital watersheds [11, 12] project, which aims to process NEXRAD radar data in almost real time, and make it available through streams of virtual sensors which present data obtained by processing the original streams, all the while maintaining provenance of this processes. It has also being integrated into other semantic technologies developed at NCSA, such as a semantic data center in a Cybercollaboratory to allow the exploration of data streams. These applications require performing simple range queries over millions of frames in time frames acceptable for web applications.

8 Conclusions

In this paper we have introduced a system for the storage, indexing and querying of time-varying data in the context of semantic web technologies, and accompanied it with a formal model theory that makes explicit the compatibility and differences with RDF, the standard for semantic data in the web. Although they are general enough for the representation of any time-varying data, both the model and the system are especially tailored for the management of streaming data, such as it is typically found in sensor networks. Combining data of arbitrary nature, be it text, numeric, still images, arbitrary binary data, etc., with metadata in a single, integrated system allows easy access to the data while leveraging semantic representations for the purpose of maintaining and providing provenance of said data, a necessary condition to ensure interoperability, particularly among scientific communities.

We have also provided a query language, TA-SPARQL, based on the standard query language for the Semantic Web, SPARQL, to facilitate expressing typical time-related queries in a natural way for users familiar with SPARQL. TA-SPARQL goes beyond other query languages for streams in RDF presented previously and based on sliding windows, and allows to exploit the richer temporal relationships representable by the semantic model.

The TA-RDF and TA-SPARQL have been implemented in a prototype service, integrated with the NCSA Tupelo semantic middleware. This has enabled integration of stream data with other data and metadata, including documents, imagery, workflows, provenance, and annotations.

9 Acknowledgments

This material is based upon work supported by the National Science Foundation (NSF) under Award No. BES-0414259, BES-0533513, and SCI-0525308 and the Office of Naval Research (ONR) under award No. N00014-04-1-0437. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of NSF and ONR.

References

[1] S. Babu and J. Widom. Continuous queries over data streams. *ACM Sigmod Record*, 30(3):109–120, 2001.

[2] P. Bajcsy, R. Kooper, L. Marini, B. Minsker, and J. Myers. CyberIntegrator: A meta-workflow system designed for solving complex scientific problems using heterogeneous tools. In *The Geoinformatics Conference*, pages 10–12, May 2006.

[3] A. Bolles, M. Grawunder, and J. Jacobi. Streaming SPARQL - Extending SPARQL to process data streams. *Lecture Notes in Computer Science*, 5021:448, 2008.

[4] S. Buraga and G. Ciobanu. A RDF-based model for expressing spatio-temporal relations between web sites. In *Proceedings of the Third International Conference on Web Information Systems Engineering 2002*, pages 355–361, 2002.

[5] T. Fountain, S. Tilak, P. Hubbard, P. Shin, and L. Freudinger. The Open Source DataTurbine Initiative: Streaming data middleware for environmental observing systems. In *International Symposium on Remote Sensing of Environment*, 2009.

[6] C. Gutierrez, C. Hurtado, and A. Vaisman. Introducing time into RDF. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):207–218, 2007.

[7] P. Hayes. RDF semantics. http://www.w3.org/TR/rdf-mt/, February 2004.

[8] C. A. Henson, H. Neuhaus, A. P. Sheth, K. Thirunarayan, and R. Buyya. An ontological representation of time series observations on the Semantic Sensor Web. In *1st International Workshop on the Semantic Sensor Web*, 2009.

[9] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. http://www.w3.org/TR/rdf-concepts/, February 2004.

[10] M. Lewis, D. Cameron, S. Xie, and B. Arpinar. ES3N: A semantic approach to data management in sensor networks. In *Semantic Sensor Networks Workshop*, 2006.

[11] Y. Liu, L. Marini, R. Kooper, A. Rodríguez, D. Hill, J. Myers, and B. Minsker. Virtual sensors in a Web 2.0 virtual watershed. In *IEEE Fourth International Conference on eScience*, pages 386–387, 2008.

[12] Y. Liu, X. Wu, D. Hill, A. Rodríguez, L. Marini, R. Kooper, J. Myers, and B. Minsker. A new framework for on-demand virtualization, repurposing and fusion of heterogeneous sensors. In *Sensor Web Enablement workshop 2009, The 2009 International Symposium on Collaborative Technologies and Systems*, 2009.

[13] M. Perry, F. Hakimpour, and A. Sheth. Analyzing theme, space, and time: an ontology-based approach. *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 147–154, 2006.

[14] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. http://www.w3.org/TR/rdf-sparql-query/, January 2008.

[15] A. Pugliese, O. Udrea, and V. Subrahmanian. Scaling RDF with time. In *17th International World Wide Web Conference*, pages 605–614, Beijing, China, 2008. ACM.

[16] S. Reddy, G. Chen, B. Fulkerson, S. Kim, U. Park, N. Yau, J. Cho, M. Hansen, and J. Heidemann. Sensor-Internet share and search: Enabling collaboration of citizen scientists. In *Proceedings of the ACM Workshop on Data Sharing and Interoperability on the World-wide Sensor Web*, pages 11–16, Cambridge, Mass, April 2007. ACM.

[17] M. Yuan and J. McIntosh. A typology of spatiotemporal information queries. *Mining Spatio-Temporal Information Systems*, pages 63–82, 2002.