

tSPARQL: Using Quadstores for Temporal Querying of RDF



Tetherless World

http://tw.rpi.edu/

Gregory Todd Williams
willig4@cs.rpi.edu

Overview

There are many existing RDF vocabularies that allow temporal data to be encoded in RDF. However, directly querying this temporal data is difficult in SPARQL for two reasons:

- Attributes that semantically belong to an object are encoded as belonging to a temporal "condition".
- Multiple encodings of temporal data means multiple query paths are necessary to retrieve similar data that have been encoded differently.

This project proposes an extension to the SPARQL query language that allows consistent querying of temporal data.

Solution

By utilizing the quadstore ability of most RDF "triplestores", temporal information may be stored with each triple for querying. On import, triples are converted to temporal quads:

```
_:kanzaki a foaf:Person ; # QUAD: Subject, Predicate, Object, Time Interval
foaf:name "Masahide Kanzaki" ;
whois:stage [
  a whois:Stage ;
  whois:place "Tokyo, Japan" ;
  whois:since "1982" .
] ;
whois:stage [
  a whois:Stage ;
  whois:place "Mie, Japan" ;
  whois:born "1960" ;
  whois:until "1978" .
] .

_:kanzaki rdf:type foaf:Person _:Always .
_:kanzaki foaf:name "Masahide Kanzaki" _:Always .
_:kanzaki whois:place "Tokyo, Japan" _:Interval1 .
_:kanzaki whois:place "Mie, Japan" _:Interval2 .

_:Always rdf:type time:Interval ∅ .
_:Interval1 rdf:type time:Interval ∅ .
_:Interval1 time:begins "1982" ∅ .
_:Interval2 rdf:type time:Interval ∅ .
_:Interval2 time:begins "1960" ∅ .
_:Interval2 time:ends "1978" ∅ .
```

Two variants of a new SPARQL graph pattern ("TIME") are introduced for querying temporal data:

```
TIME ?interval { ... }
TIME [ a time:Interval ; ... ] { ... }
```

Similar to SPARQL's GRAPH pattern, TIME allows triple matching to be restricted to time intervals matching given criteria.

The query "Find all people who lived in Tokyo during 2007" now becomes:

```
SELECT ?name WHERE {
  TIME [ time:inside "2007"^^xsd:dateTime ] {
    [ a foaf:Person ; foaf:name ?name ;
      whois:place "Tokyo, Japan" . ]
  }
}
```

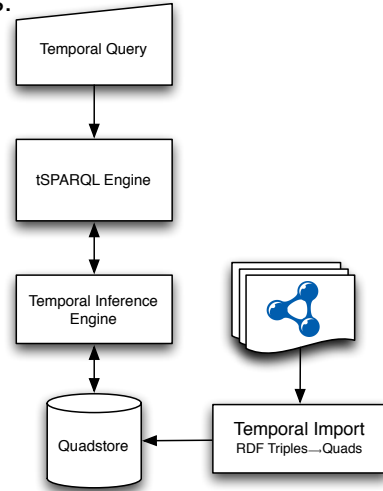
[time:inside "2007"^^xsd:dateTime] is inferred for any time interval that overlaps 2007 using the OWL-Time ontology³.

Components

Temporal Import turns RDF Triples into Temporal Quads.

Temporal Inference Engine generates temporal relationship such as: before, inside, overlaps, contains, during.

tSPARQL Engine executes temporal queries, using temporal quads to answer queries about temporal information. Temporal data is ignored for simple (non-temporal) triple pattern matching.



Example Data

Non-temporal data:

```
_:albert a foaf:Person;
foaf:name "Albert Einstein" .
```

Interval data (using the bio vocab¹):

```
_:pauline a foaf:Person;
bio:condition [ a bio:Condition;
time:begins "1858" ;
time:ends "1876" ;
foaf:name "Pauline Koch" .
];
bio:condition [ a bio:Condition;
time:begins "1876" ;
foaf:name "Pauline Einstein" .
] .
```

Interval data (using the whois vocab²):

```
_:kanzaki a foaf:Person;
foaf:name "Masahide Kanzaki";
whois:stage [ a whois:Stage;
whois:place "Tokyo, Japan";
whois:since "1982" .
];
whois:stage [
a whois:Stage;
whois:place "Mie, Japan";
whois:born "1960";
whois:until "1978" .
] .
```

[1] <http://jandavis.com/blog/2006/03/refactoring-bio-with-einstein-part-4-employment-and-families>
[2] <http://www.kanzaki.com/ns/whois>
[3] <http://www.isi.edu/~pani/OWL-Time.html>