.RDF Stream Processing Model

RDP Stream Processing Community Group Note

<http://www.w3.org/community/rsp/>

**Authors**

Please add your name and affiliation to the list below (in alphabetical order of surname)

* [Jean-Paul Calbimonte](http://jeanpi.org/), EPFL, Switzerland
* [Alasdair J G Gray](http://www.alasdairjggray.co.uk), Heriot-Watt University, UK
* [Monika Solanki](https://w3id.org/people/msolanki), Aston University, UK

**Table of Contents**

[Introduction](#h.vogkf2191io3)

[Requirements](#h.sw7dq131xe50)

[Functional Requirements](#h.9xr2422mdmm2)

[Non-Functional Requirements](#h.zaj7vgung9j3)

[Time Model](#h.lsdv4gfqodl2)

[Stream Data Model](#h.soulxnxwsmaf)

[Serialisation](#h.rrc24n2dc0t8)

[Query Language](#h.l8xfskibjbja)

[Operators](#h.lkgpsahf7ua5)

[Time Operators](#h.ys408yc37iam)

[Windowing Operators: Streams to Bag-of-Triples](#h.81jpfzxqdhew)

[Inverse Window Operators: Bag of Triples to Streams](#h.z8jolamdimfd)

[Stored Data Windows](#h.iltglficaxc2)

[Semantics](#h.i0ku3cocqrqc)

[References](#h.j9q7p1ggix29)

# Introduction

Set up the context (briefly)

Goal: define common core model for RDF stream processors that is extensible.

(Not going for a one-size fits all approach)

In concrete: an RDF stream extensions (stream data model) and SPARQL stream extensions (query language) that operate over the RDF stream model.

# Requirements

*(Stated at a high level, not the nitty gritty of all the use cases. Enough to allow a reader to understand what the systems that implement this model are capable of doing.)*

<http://www.w3.org/community/rsp/wiki/Use_cases>

Not entirely copy them here, but abstract them as requirements.

How the use cases map to the requirements. An idea may be to identify requirements and number them so that we can relate the use cases to the requirement numbers. In this document or a different one?

## Functional Requirements

Need to be able to process a stream of data without first storing it.

## Non-Functional Requirements

# Time Model

Time abstractions, notion of time and ordering

(We will avoid specifying mechanism for dealing with delayed and out of order triples. These are left for the implementations to decide for themselves.)

Time as point-in-time

Time as an interval (point in time is a special case of this)

# Stream Data Model

Extend RDF model:

Add Interval/ time annotations to RDF graphs.

Resulting in time annotated RDF graphs

semi-informal:

A graph G defined as a set of triples:

G = {( subj, pred, obj)}

A stream is a sequence (G, τ ) where τ ∈ T the set of all intervals (timestamps).

The content of the stream can be seen then as a sequence of quads (s, p, o, G) where G is the context/graph of the triple.

RDF streams without an explicit timestamp/interval can be processed, if a system timestamp/interval is assigned to them as they arrive to the system.

Refs: <http://www.w3.org/community/rsp/wiki/RDF_Stream_Model>

### Serialisation

How can the model be serialised?

(The Stream Data Model is defined independently of the various serialisations.)

# Query Language

Based on SPARQL, defined to pose queries to a dataset, composed of (potentially)

- RDF streams (one or more)

- RDF graphs (one or more)

Existing SPARQL 1.1 operators are included (semantics for RDF streams need to be defined) Other operators required, detailed below.

## Operators

New or additional operators.

We can define a tentative syntax, even for pedagogic reasons, but in every case explain the semantics of the introduced operators.

### Time Operators

These operators are important as we are dealing (potentially) with intervals.

Allen’s Operators ([1983](#id.9krvnzex4ke7)) and derivations

* Before: X before Y if end time of X is earlier than start time of Y
* Equal: X equal Y if start and end times are the same
* After: X after Y if start time of X is later than the end time of Y
* During: X during Y if start time of X is later than the start time of Y and the end time of X is eariler than the end time of Y
* Contains: Inverse of during
* Overlaps: X overlaps Y if the start time of X is earlier than the start time of Y, the end time of X is earlier than the end time of Y, and the end time of X is later than the start time of Y
* Overlapped by: Inverse of Overlaps
* Meets: X meets Y if the end time of X is the start time of Y
* Met by: Inverse of Meets
* Starts: X starts Y if the start time of X and Y are the same
* Started by: Inverse of Starts
* Finishes: X finishes Y if the end time of X and Y are the same
* Finished by: Inverse of Finishes

### Windowing Operators: Streams to Bag-of-Triples

Making finite something that is infinite. The window term may be changed (working name for the time being).

Distinction between the window and the window operator.

Time-based (sliding) windows

Graph-based? number of graphs (akin to tuple based windows)

Jumping/sliding/tumbling/historic: analyze if these are only special cases of previous ones.

### Inverse Window Operators: Bag of Triples to Streams

When the output of a query is also a stream

Produce streams back, could feed a new query

RSTREAM, DSTREAM, ISTREAM

### Stored Data Windows

How ofter do you refresh stored data?

Is stored data a different type from the output type of a windowing operator?

## Semantics

Semantics of the query language and its operators

Abstract of any serialisations or implementations

# References

Allen, J.F., 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), pp.832–843. Available at: http://dl.acm.org/citation.cfm?id=358434 [Accessed June 6, 2014].