Simple Transaction-Time
ESWC 2016

James Anderson <james@dydra.com>
Arto Bendiken <arto@dydra.com>

@dydradata @lomoramic @bendiken
<http://dydra.com/>

Overview

- Temporal RDF processing : alternative ...
  - Data models
  - Processing models
  - Expression
- Exemplified by
  - Archival : Diachron
  - Stream : C-SPARQL, CQELS, SPARQLStream, EP-SPARQL, INSTANS, R&WBase, STRABON
- Dydra : in terms of transaction time
  - capabilities
  - benefits
- Goal : raise three questions

Three Questions

- How to align levels?
  - interfaces and protocols
  - schematic, conceptual
  - physical, concrete representation
  - is `CLOCK_MONOTONIC` pervasive
- Why monolithic expressions?
  - dataset
  - projection
- Why homogeneous temporality?
  - static, archival
  - dynamic, stream

Precedent: TSQL

- TSQL[1][2]
  - Data : $(O+ \times T)$
    - concerns the temporal properties of *tuples*
  - Expression

```
CREATE TABLE Employee(
Name VARCHAR(30),
Manager VARCHAR(30) VALIDTIME REFERENCES Employee (Name), Dept
VARCHAR(20)) AS VALIDTIME PERIOD(DATE)

ALTER TABLE Employee ADD TRANSACTIONTIME

TRANSACTIONTIME SELECT Dept, COUNT(*) FROM Employee
GROUP BY Dept
HAVING COUNT(*) > 25
```

[3]

[1] : <https://www.cs.arizona.edu/~rts/initiatives/tsql2/tsqldesignapproach.pdf>

[2] : <http://timecenter.cs.aau.dk/TimeCenterPublications/TR-1.pdf>

[3] : <http://timecenter.cs.aau.dk/TimeCenterPublications/TR-8.pdf>

Precedent: Temporal RDF

- Temporal RDF[1]
  - Data : $((S \times P \times O) \times T)$
  - Processing
  - Expression

```
TriplePattern ::= (Subject Predicate Object ) : [T]
```

In scenarios where changes are frequent and only affecting a few elements of the document, creating a new physical version of the graph each time an update occurs may lead to large overheads when processing temporal queries that span multiple versions.[1]

[2]

[1] : <http://users.dcc.uchile.cl/~cgutierr/papers/temporalRDF.pdf>

Precedent: R&W Base

- R&W Base[1]
  - Data : $((S \times P \times O) \times T)$
  - Processing
    - SPARQL rewriting
    - delegation to endpoint
  - Expression

```
Datatset ::= FROM VersionIri
```

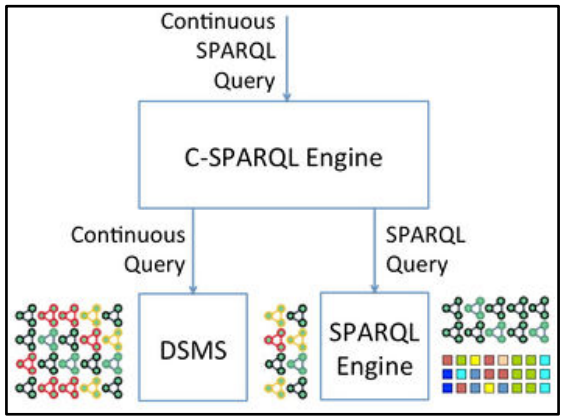
[2]

[1] : <http://events.linkeddata.org/ldow2013/papers/ldow2013-paper-01.pdf>

Precedent: C-SPARQL

- C-SPARQL[1]
  - Data : $((S \times P \times O) \times T)$
  - Processing
    - DSMS constitutes snapshot
    - SPARQL processor interprets snapshot as dataset
  - Expression

```
Datatset ::= FROM Iri Window ?
Window ::= ( RANGE Number TimeUnit ( ( STEP Number TimeUnit ) | TUMBLING ) )
         | ( TRIPLES Number )
```



[2]

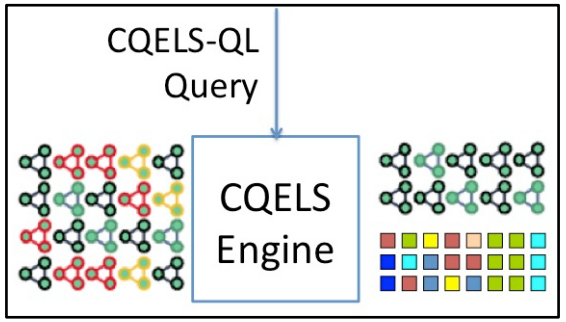
[1] : <http://www09.sigmod.org/sigmod/record/issues/1003/p20.article.barbieri.pdf>

[2] : <https://www.w3.org/community/rsp/wiki/images/thumb/c/c4/C-sparql-engine.jpg/800px-C-sparql-engine.jpg>

Precedent: CQELS

- CQELS[1]
  - Data : $((S \times P \times O) \times T)^* + ((S \times P \times O)^* \times T)$
  - Processing
    - Windowing constitutes snapshot dataset
    - Relational SPARQL processor interprets
    - Streaming re-generates a stream
  - Expression

```
GraphPatternNotTriples ::= ... | StreamGraphPattern
StreamGraphPattern ::= ( RANGE [ Window ] VarOrIRIReference { TriplesTemplate }
Window ::= ( RANGE Number TimeUnit ( SLIDE Number TimeUnit ) ? )
         | ( TRIPLES Number ) | NOW | ALL
```



[2]

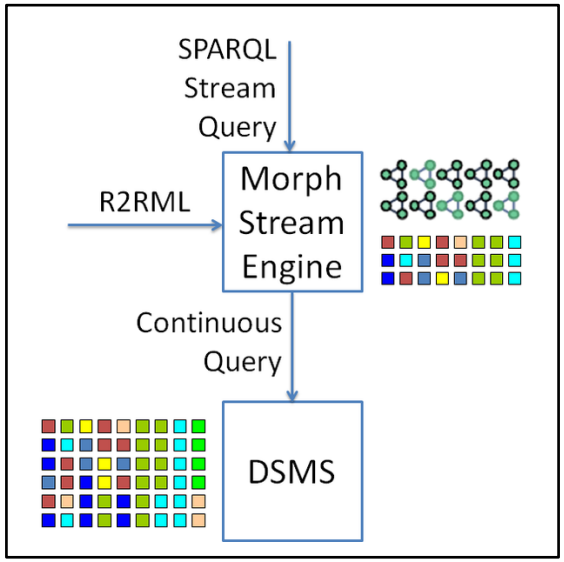
[1] : <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.361.1369>

[2] : <https://www.w3.org/community/rsp/wiki/images/7/7f/Cqels-engine.jpg>

Precedent: SPARQL Stream

- SPARQL Stream[1]
  - Data : $((S \times P \times O) \times T)$
  - Processing
    - transform and delegate to DSMS
  - Expression

```
Datatset ::= FROM STREAM Iri Window ?
Window ::= FROM NOW ( - Number TimeUnit )?
           TO NOW ( - Number TimeUnit )?
           ( STEP Number TimeUnit ) ?
```



[2]

[1] : <http://oa.upm.es/5121/1/Enabling_Ontology-based_Access_to_Streaming_Data_Sources.pdf>

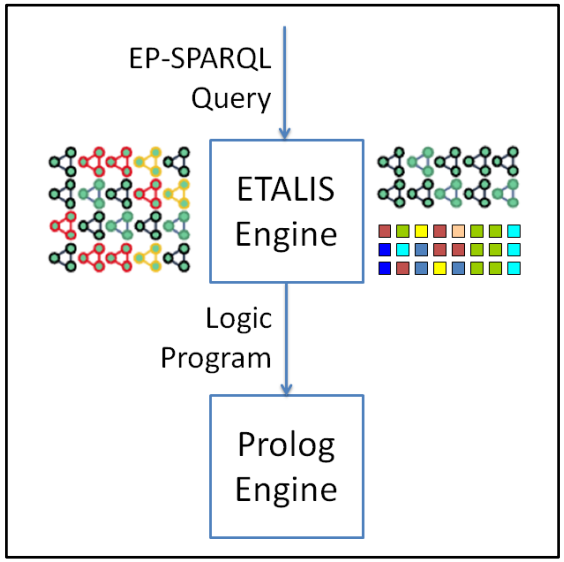
[2] : <https://www.w3.org/community/rsp/wiki/File:Morph-streams.png>

Precedent: EP-SPARQL

- EP-SPARQL[1]
  - Data : $((S \times P \times O) \times T \times T)$
  - Processing :
    - transform and delegate to Prolog
  - Expression

```
~~Datatset ::=~~
GraphPatternNotTriples ::= ... | SeqGraphPattern
      | EqualsGraphPattern
      | OptionalSeqGraphPattern
      | OptionalEqualsGraphPattern

NullOperator ::=  ... | getDURATION
      | getENDTIME
      | getSTARTTIME
```



[2]

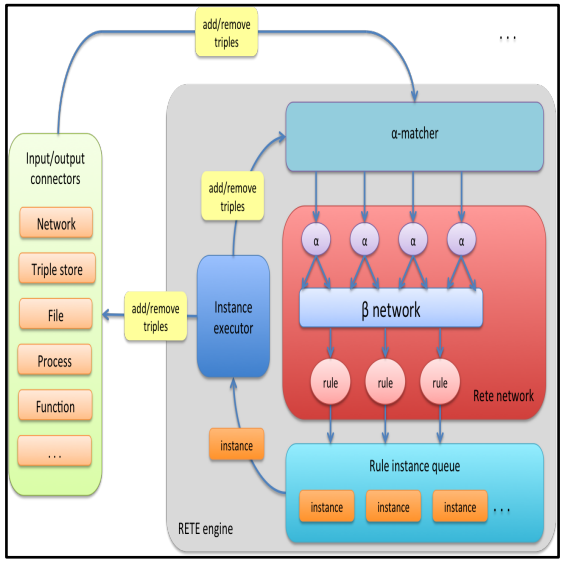
[1] : <http://www.wwwconference.org/proceedings/www2011/proceedings/p635.pdf>

[2] : <https://www.w3.org/community/rsp/wiki/images/c/cb/Etalis.png>

Precedent: INSTANS

- INSTANS[1]
  - Data : $(S \times P \times O)$
  - Processing :
    - Rete network propagation
  - Expression

```
Dataset ::=
Verb ::=  ... | :hasWIndow
```



[2]

[1] : <http://iswc2012.semanticweb.org/sites/default/files/paper_22.pdf>

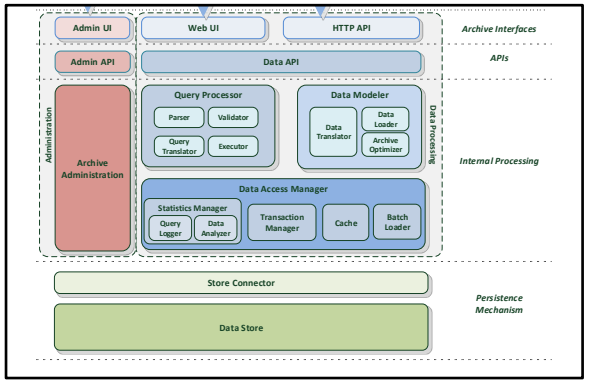
[2] : <http://iswc2012.semanticweb.org/sites/default/files/paper_22.pdf>

Precedent: DIACHRON

- Diachron[1]
  - Data : $(S \times P \times O)$
  - Processing :
    - SPARQL evaluation of DIACHRON query
  - Expression

```
SourceClause ::= ... | ( FROM DATASET URI ( AT VERSION URI )? )
  | ( FROM CHANGES URI ( BEFORE VERSION URI )? )
  | ( AFTER VERSION URI )
  | ( BETWEEN VERSIONS URI URI )

SourcePattern ::= ... | (DATASET URI ( AT VERSION URI )? )
  | ( CHANGES URI ( BEFORE VERSION URI )? )
  | ( AFTER VERSION URI )
  | ( BETWEEN VERSIONS URI URI )
```



[2]

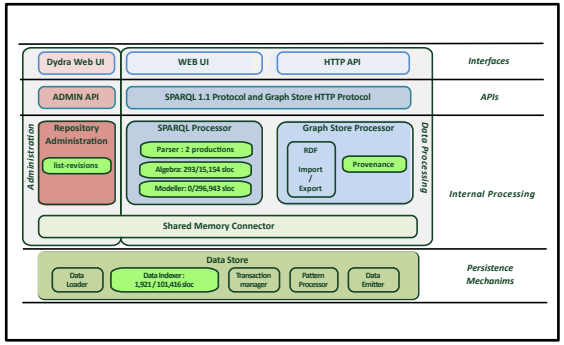
[1] : <http://arxiv.org/pdf/1504.01891.pdf>

[2] : <http://arxiv.org/pdf/1504.01891.pdf>

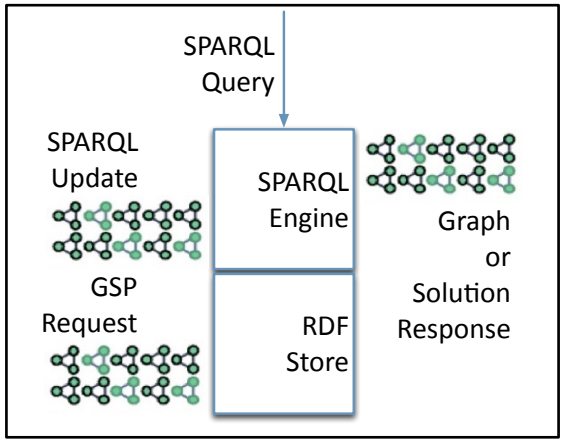
Alternatively: Dydra

- Dydra[1]
  - Data : $((S \times P \times O \times G)^* \times R)$
  - Processing :
    - SPARQL, semantics analogous to named graphs
  - Expression:

```
GraphPatternNotTriples ::= ... | RevisionGraphPattern
RevisionGraphPattern ::= REVISION ( Var | RevisionRef | String ) GroupGraphPattern
RevisionRef ::=  ( R Integer ? / ) ? RevisionRelation [3]
RevisionRelation ::= ( Revision ( / Revision )? ) | ( / Revision ) | XPathDuration )
                     ( / ( XPathDuration | Integer ) )?
Revision ::=  UUID | /HEAD(~[0-9]+)?/ | XPathDateTime
NullOperator ::= ... | THEN
```



[2]

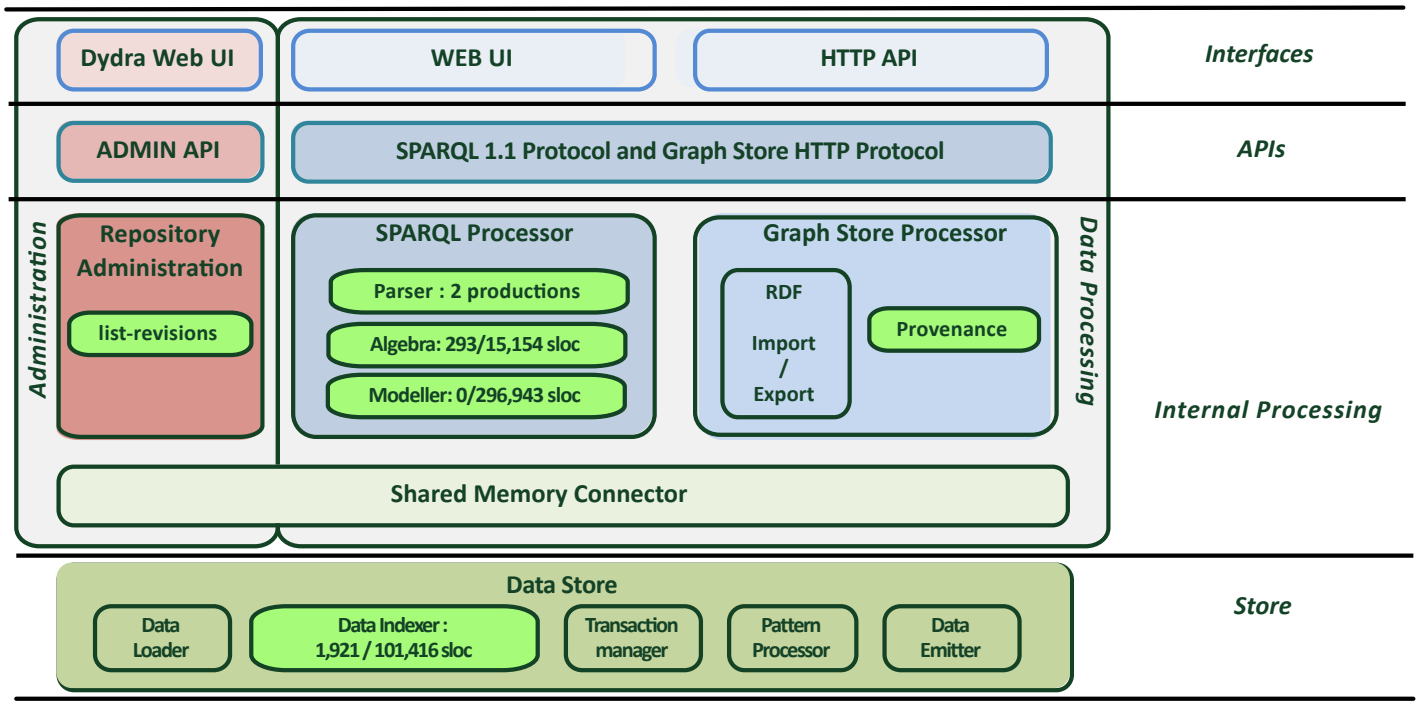


[1] : <http://dydra.com>

[2] : <http://arxiv.org/pdf/1504.01891.pdf> (sampled)

[3] : <https://en.wikipedia.org/wiki/ISO_8601>

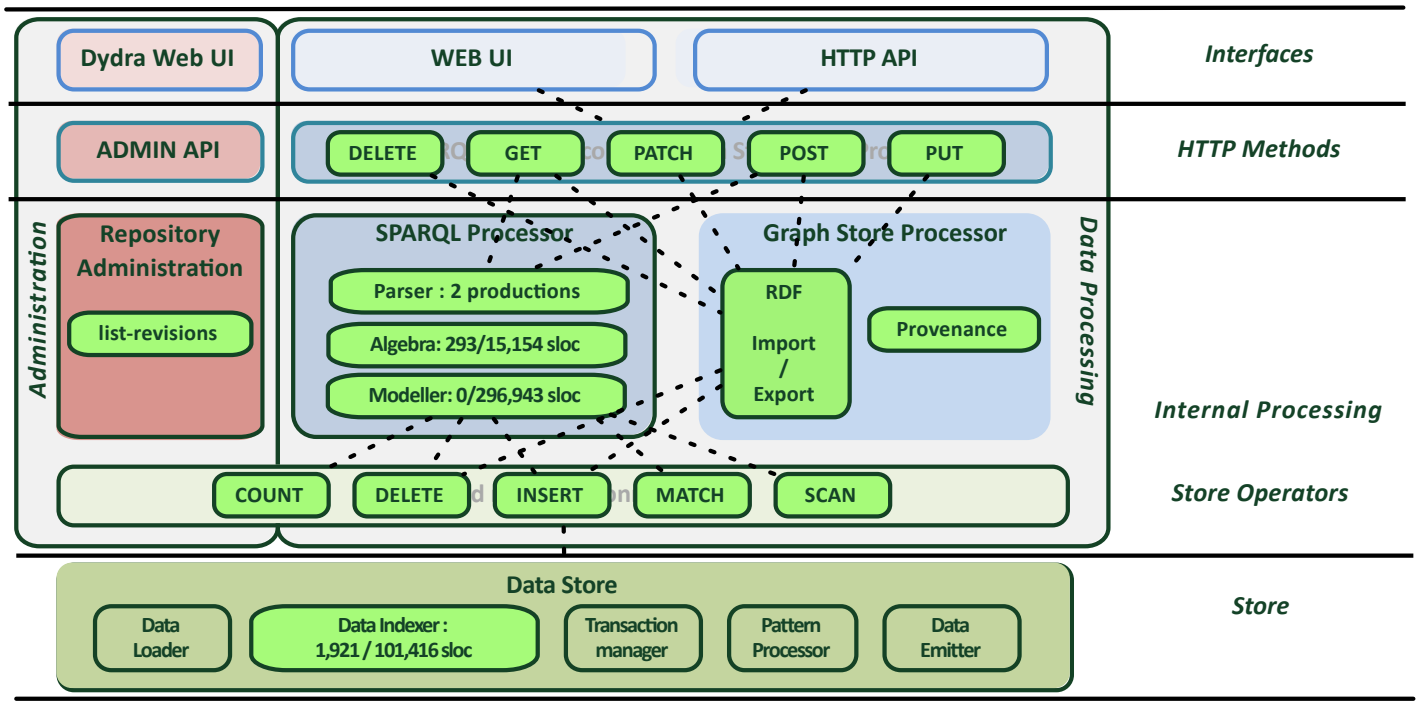
Dydra: Architecture



[1] : <http://dydra.com>

[2] : <> sampled

Dydra: Data-Flow



[1] : <http://dydra.com>

[2] : <> sampled

By Analogy to Graph Provenance

- Named graphs originated as provenance designators
- Evolved to permit assertions about statement sets
- Temporal coincidence is analogous to physical or conceptual
  - but neither equivalent nor autonomous
- Add a `REVISION` clause to the SPARQL -- RDF API

```
RevisionGraphPattern ::= REVISION ( Var | RevisionRef | String ) GroupGraphPattern
```

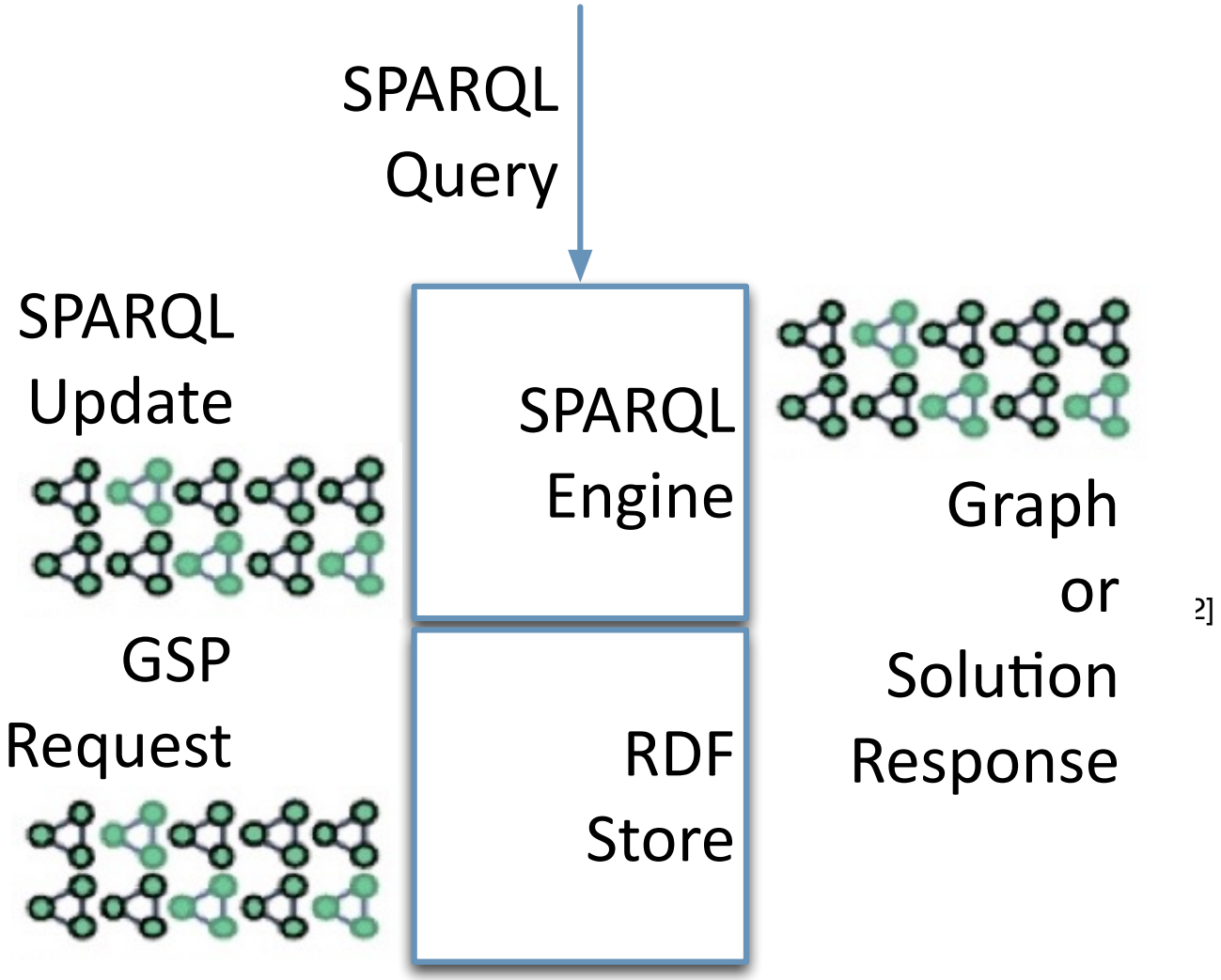

Dydra : Revision Designators

- Elementary
  - absolute: *UUID* : `01234567-0123-0123-012345678901`
  - relative : */HEAD(~[0-9]+)?/* : `HEAD~1, HEAD~1..HEAD`
  - temporal : *XPathDateTime* : `20150303T180000Z`
- Compound
  - interval : *ISOInterval* : `20150303T180000Z/PT10M`
  - stream : *ISORepeatingInterval* : `R/20150303T180000Z/PT10M/PT01M`

Dydra : Revision Clauses

- Constant
  - absolute: `REVISION` *`UUID`* `{ ... }`
  - relative : `REVISION /HEAD(~[0-9]+)?/ { ... }`
  - temporal : `REVISION` *`XPathDateTime`* `{ ... }`
- Variable
  - SIP : `{ ?subject :revision ?revision } ... REVISION ?revision { ... }`
  - Protocol : `HTTP://...?REVISION=`*`UUID`* `... REVISION ?revision { ... }`
- Intra-Repository : `REVISION ?revision { ?subject a ?class }`
- Extra-Repository : `SERVICE ?revisionEndpoint { ?subject a ?class }`

Dydra: RSP


SPARQL
Query
SPARQL
Update
GSP
Request
SPARQL
Engine
RDF
Store
Graph
or
Solution
Response

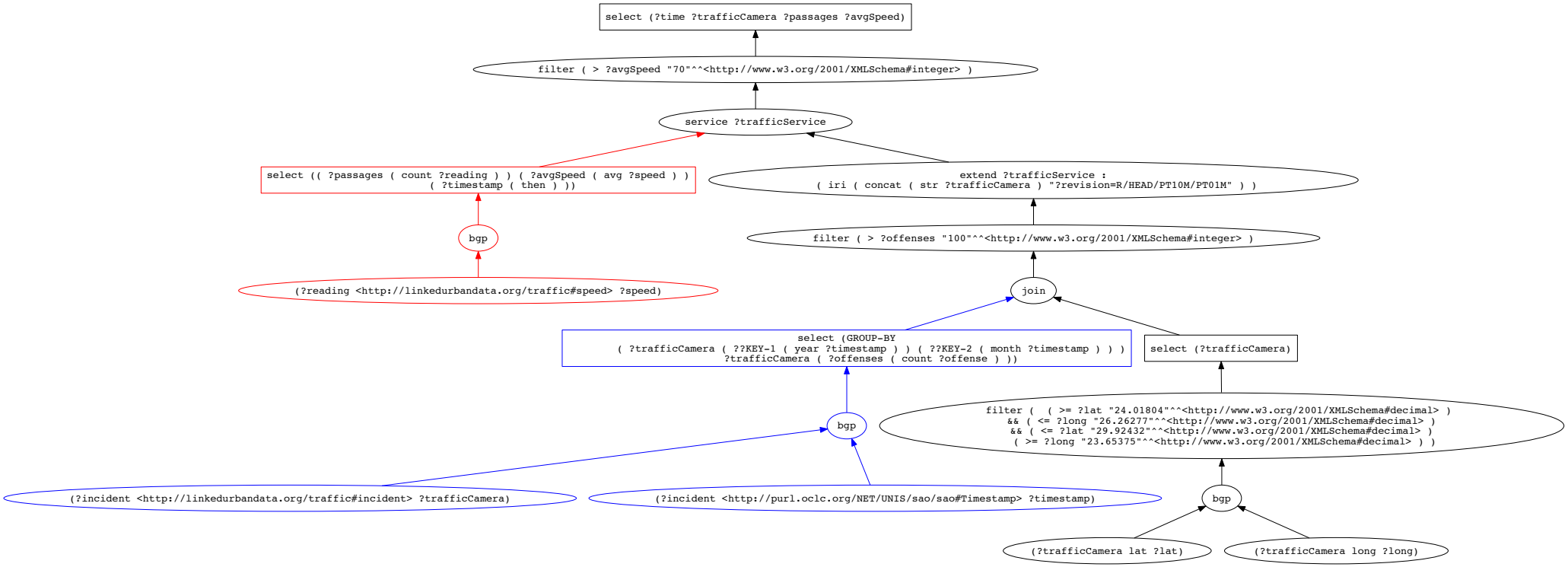
```
PREFIX c: <http://linkedurbandata.org/city#>
PREFIX t: <http://linkedurbandata.org/traffic#>

select  ?time   ?trafficCamera ?passages ?avgSpeed where {
# select tollgates in some region (static base data)
{{ select ?trafficCamera where {
  ?trafficCamera <http://www.w3.org/2003/01/geo/wgs84_pos#lat> ?lat .
  ?trafficCamera <http://www.w3.org/2003/01/geo/wgs84_pos#long> ?long .
  filter ( (?lat >= 24.01804) && (?lat <= 29.92432)
        && (?long >= 23.65375) && (?long <= 26.26277) )
  } }
# reduce that set to those with some activity level
{ select ?trafficCamera (count(?offense) as ?offenses)
  where {
   ?incident t:incident ?trafficCamera .
   ?incident <http://purl.oclc.org/NET/UNIS/sao/sao#Timestamp> ?
timestamp .
  } group by ?trafficCamera (year(?timestamp)) (month(?timestamp))

}
  filter (?offenses > 100) }
# look for incidents at that location
bind (iri(concat(str(?trafficCamera), '?revision=R/HEAD/PT10M/PT01M'))
as ?trafficService)
service ?trafficService {
  select (count(?reading) as ?passages)
         (avg(?speed) as ?avgSpeed)
         (then() as ?timestamp)
  where { ?reading t:speed ?speed .}
}
  filter (?avgSpeed > 70)
}
```


```
(select
   (service ?trafficService
            (select
             (bgp (triple ?reading <http://linkedurbandata.org/traffic#speed> ?speed))
             (:HAVING (> ?avgSpeed 70) (?passages (count ?reading))
              (?avgSpeed (avg ?speed)) (?timestamp (then))))
            (extend
             (join
              (select
               (bgp
                 (triple ?incident <http://linkedurbandata.org/traffic#incident> ?trafficCamera)
                 (triple ?incident <http://purl.oclc.org/NET/UNIS/sao/sao#Timestamp> ?timestamp))
               (:HAVING (> ?offenses 100) :GROUP-BY
                (?trafficCamera (??KEY-1 (year ?timestamp)) (??KEY-2 (month ?timestamp)))
                ?trafficCamera (?offenses (count ?offense))))
              (select
               (filter
                (bgp (triple ?trafficCamera <http://www.w3.org/2003/01/geo/wgs84_pos#lat> ?lat)
                     (triple ?trafficCamera <http://www.w3.org/2003/01/geo/wgs84_pos#long> ?long))
                (&& (>= ?lat 30623/1275)
                    (&& (<= ?long 39079/1488)
                        (&& (<= ?lat 35191/1176) (>= ?long 18923/800)))))
                (?trafficCamera)))
             ?trafficService
             (iri
              (concat (str ?trafficCamera) "?revision=R/HEAD/PT10M/PT01M")))
            :QUERY-TEXT
            "SELECT ( COUNT ( ?reading ) AS ?passages ) ( AVG ( ?speed ) AS ?avgSpeed ) ( THEN ( )
AS ?timestamp ) WHERE { ?reading <http://linkedurbandata.org/traffic#speed> ?speed . } HAVING ( ?
avgSpeed > 70 ) ")
   (?trafficCamera ?time ?passages ?avgSpeed))
```

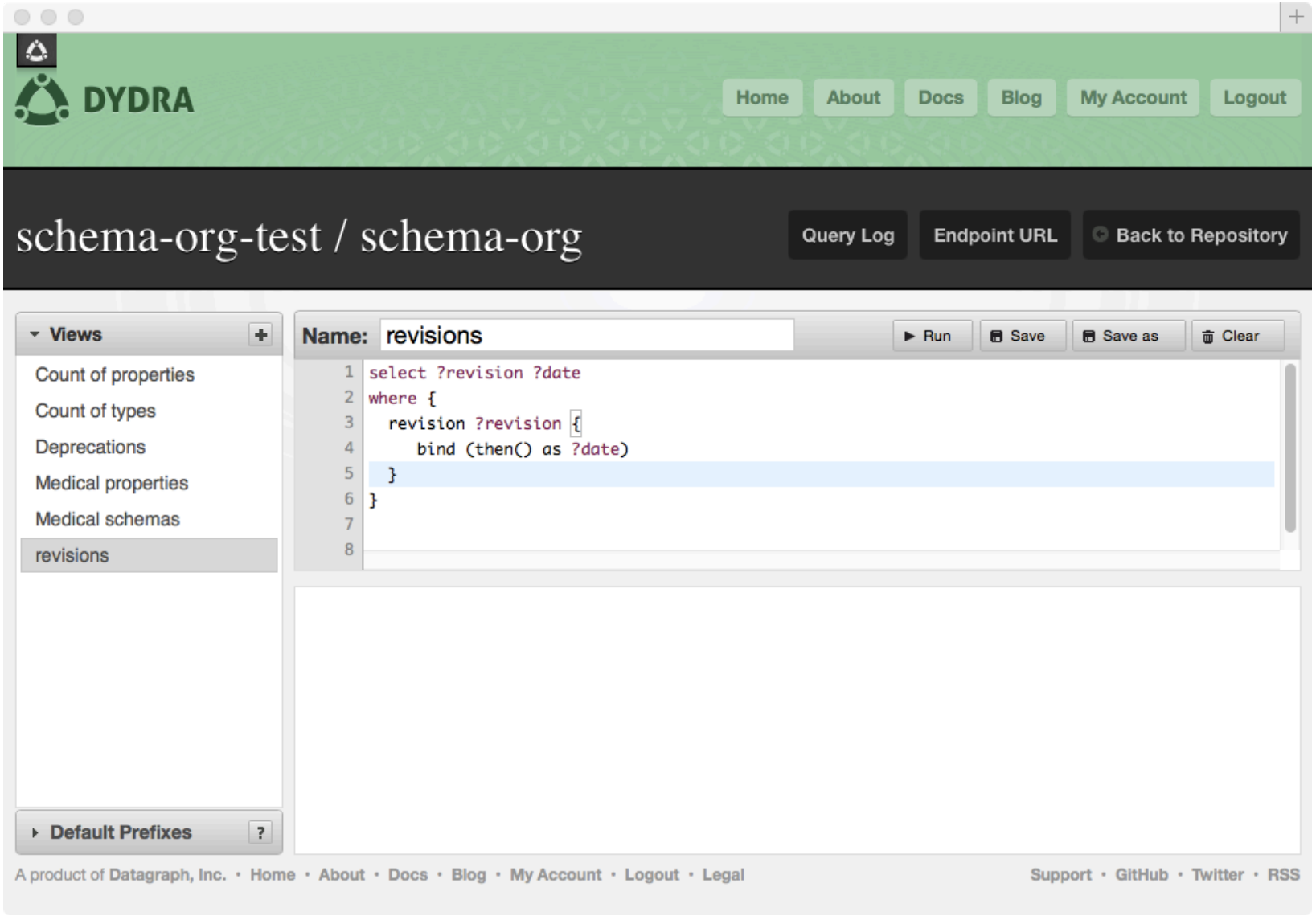

Dydra : Processing Model



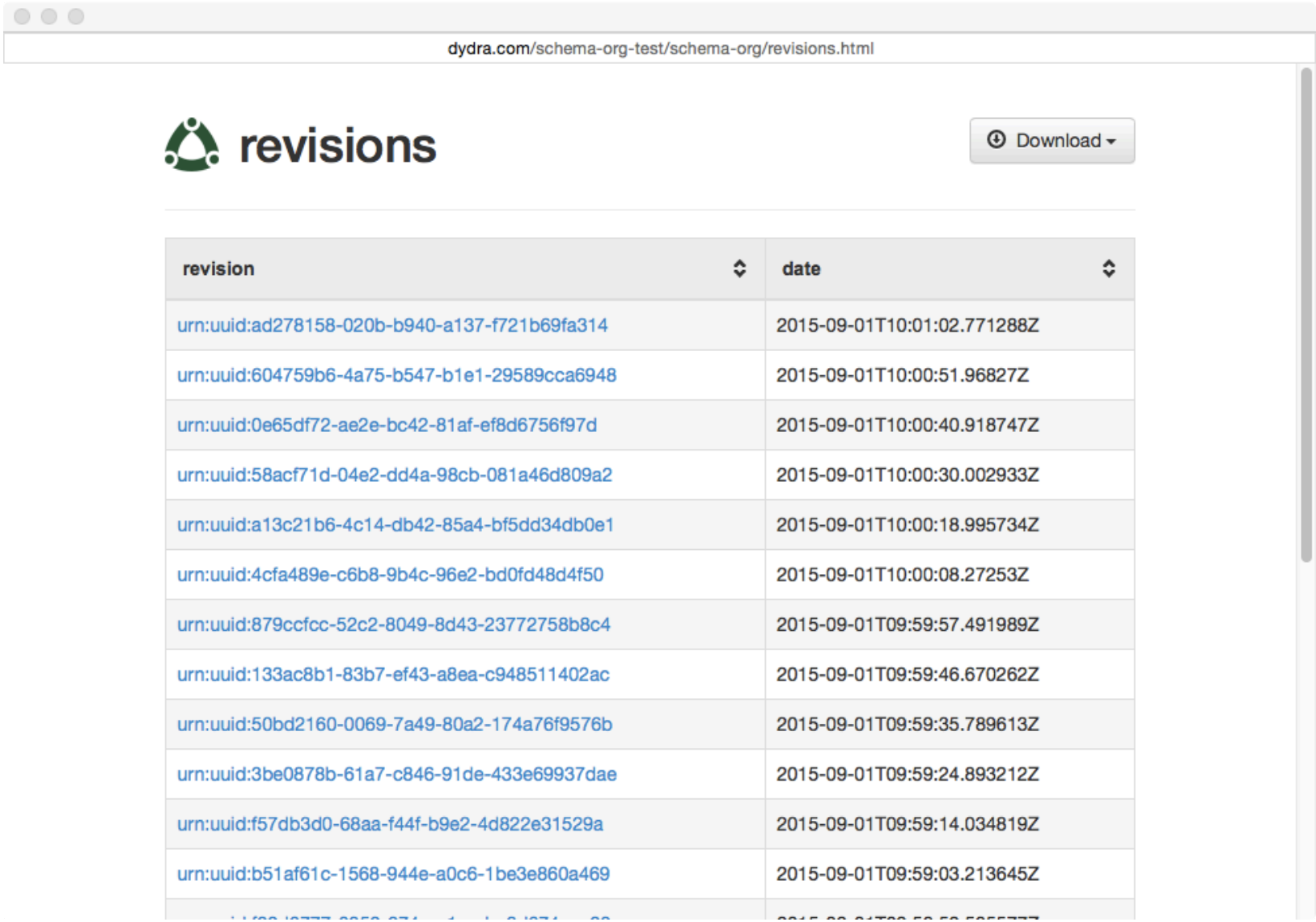
Dydra: Examples

- http://dydra.com/schema-org-test/schema-org
  - http://dydra.com/schema-org-test/provenance
- http://dydra.com/schema/efo
  - http://dydra.com/schema/efo-provenance

Dydra: Query @Revision



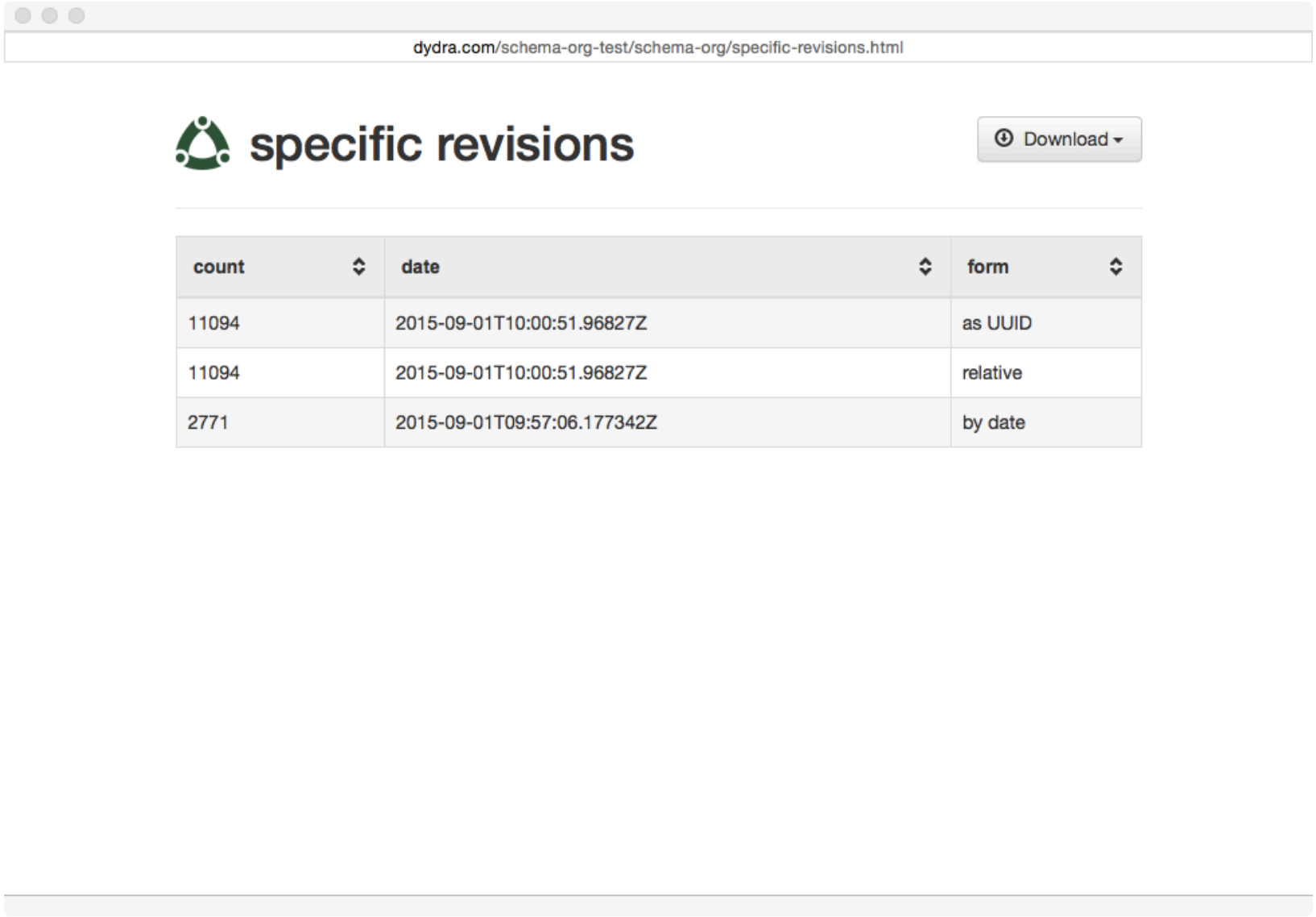
Dydra: View @Revision



Dydra: Query @Revision

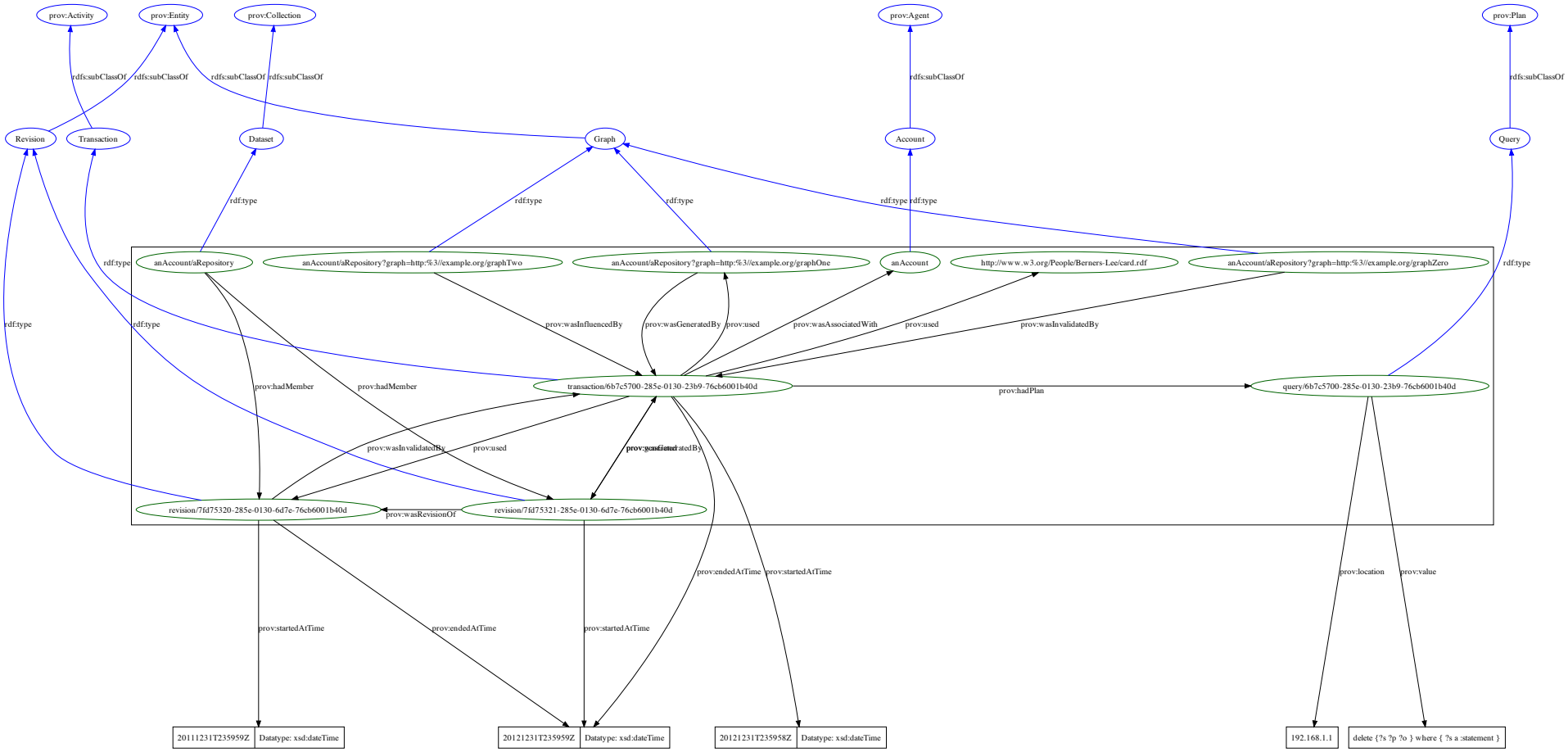
```
select *
where {
  { revision <urn:uuid:604759b6-4a75-b547-b1e1-29589cca6948> {
      { select (count(*) as ?count) where {?s ?p ?o} }
      bind (then() as ?date)
      bind ("as UUID" as ?form)
    }
  } union
  { revision "HEAD~1" {
      { select (count(*) as ?count) where {?s ?p ?o} }
      bind (then() as ?date)
      bind ("relative" as ?form)
    }
  } union
  { revision "2015-09-01T09:57:07" {
      { select (count(*) as ?count) where {?s ?p ?o} }
      bind (then() as ?date)
      bind ("by date" as ?form)
    }
  }
}
```


Dydra: View @Revision



count	date	form
11094	2015-09-01T10:00:51.96827Z	as UUID
11094	2015-09-01T10:00:51.96827Z	relative
2771	2015-09-01T09:57:06.177342Z	by date

Provenance Recording



Dydra: as SPARQL endpoint

```
select  ?versionId ?title ?term ?comment
where {
  { service <http://localhost/schema-org-test/provenance> {
      select ?location ?versionId ?endpoint ?title
      where {
        ?revision <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>  <urn:dydra:Revision> .
        { graph ?revision {
            ?revision <http://www.w3.org/ns/sparql-service-description#endpoint> ?endpoint .
            ?revision <http://purl.org/dc/elements/1.1/title> ?title .
            ?revision <http://usefulinc.com/ns/doap#revision> ?versionId .
          }
        }
      }
    }
  }
  service ?endpoint {
    ?term <http://www.w3.org/2000/01/rdf-schema#comment> ?comment .
    filter (regex(?comment, '.*deprecat.*'))
  }
}
```


Storage Model: Evolution

1. Next, "change based" @statement id
   - efficient HEAD access
   - **inefficient urn:uuid? access**
   - space efficient
   - **inefficient mutation at scale**
2. Initially, "independent copies" @quad term id
   - efficient HEAD access
   - efficient urn:uuid? access
   - **space inefficient**
   - **inefficient mutation at scale**
3. Current, "time-slice based" (various)
   - O|Δ| HEAD access
   - O|Δ| urn:uuid? access
   - space efficient
   - efficient mutation at scale

Storage Model: Current

- Current, "time-slice based" (various)
  - OlΔl HEAD access
  - OlΔl urn:uuid? access
  - space efficient
  - efficient mutation at scale
- Analogous to T-SPARQL[1], but where T is anisotropic, with one dimension - transaction time, comprising UUID values and the others as application concerns.

[1] : Grandi : <http://www-db.deis.unibo.it/~fgrandi/papers/GraphQ10.pdf>

Results: Update Processing

- Time : efficient mutation at scale
  - effort proportional to mutation size (+ / -)
- Space
  - initial space requirements (initially sub-optimal)
  - subsequent compaction (2x)
- Detrimental practices
  - zero-sum non-monotonic updates
  - blank nodes

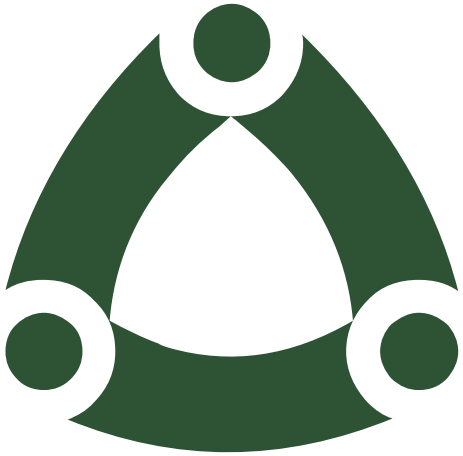
Query Processing Performance

- indexed revisions v/s materialized
- spurious b+tree

Space

ontology	**revisions**	**files** (MB)	**quads**		**store** (MB)	**bytes/quad**
			total	**HEAD**		
gist	16	2.9	47,273	849	38.4	851
schema.org	23	19.1	156,150	9,144	40.6	273
STW	7	94.2	771,375	108,967	240.0	311
efo	144	1,384.0	12,807,240	260,503	4,270	358
efo	@2.69	2.8	260,503	260,503	84.6	324

Thank you



DYDRA

James Anderson <james@dydra.com>
Arto Bendiken <arto@dydra.com>

@dydradata @lomoramic @bendiken
<http://dydra.com/>