RDF Archiving and RDF Streaming: Two Sides of the Same Coin?

Javier D. Fernández, Axel Polleres

Stream Reasoning Workshop 2016 - Berlin
08/12/2016
RDF evolution at Scale

Update rate
- second
- minute
- hour
- day
- week
- month
- year

Virtual/Augmented Reality

Internet of Things

Number of sources

DBpedia

Dynamic Linked Data Observatory

BTC
Publication, Exchange and Consumption of large RDF datasets

- Most RDF formats (N3, XML, Turtle) are **text** serializations, designed for “human” readability (not for machines)
  - Verbose = High costs to write/exchange/parse
  - A basic offline search = (decompress)+ index the file + search

- **Binary RDF (HDT)**
  - Highly **compact** serialization of RDF
  - Allows fast RDF retrieval in compressed space (without prior decompression)
    - Includes internal **indexes** to solve basic queries once it is loaded in **main memory**
    - Very **fast on basic queries** (triple patterns), x 1.5 faster than Virtuoso, RDF3x.
    - Complex queries (joins) on the same scale of current solutions (Virtuoso, RDF3x).

[DBpedia](http://dbpedia.org) NT + gzip 5 GB | HDT 6.6 GB | HDT + gzip 2.7 GB
---
[431 M.triples~ 63 GB](http://rdfhdt.org)
RDF evolution at Scale

Update rate
- second
- minute
- hour
- day
- week
- month
- year

Virtual/Augmented Reality

Internet of Things streaming?

Dynamic Linked Data Observatory

DBpedia

BTC

Number of sources
1. Learn patterns from the stream
2. Sender sends the ID of the pattern and the data that differ from the pattern

- Remains efficient in performance (similar to DEFLATE)
  - Time overheads are relatively low and can be assumed in many scenarios.
- Operations on the compressed information
  - E.g. Discard all info except predicate `ex:CelsiusValue`
RDF evolution at Scale

Update rate

- second
- minute
- hour
- day
- week
- month
- year

Virtual/Augmented Reality

Internet of Things

Versions?

Dynamic Linked Data Observatory

DBpedia

BTC

Number of sources

10^0 10^1 10^2 10^3 10^4 10^5 10^6
Preservation/Archiving in the last few years

Research projects

Managing the Evolution and Preservation of the Data Web (FP7)

Preserving Linked Data (FP7)

The Dynamic Linked Data Observatory

http://swse.deri.org/dyldo/

http://data.wu.ac.at/wayback/

Archives

Tools

Benchmarking?
BEAR: Evaluating Query and Storage Strategies for RDF Archives

1) Guidelines on benchmarking archives of semantic data
   - How can one define the corpus?
   - How can one design benchmark queries? Which queries?

BEAR: concrete extensible benchmark
   - Data: Crawl from Linked Data Observatory
   - Basic queries: Materialize, get Version...
   - Initial evaluation on archiving policies

RDF Archiving. Archiving policies

a) Independent Copies/Snapshots (IC)

b) Change-based approach (CB)

c) Timestamp-based approach (TB)
RDF Archiving. Archiving policies

a) Independent Copies/Snapshots (IC)

b) Change-based approach (CB)
c) Timestamp-based approach (TB)
Benchmarking: Define the queries

- Instantiation of archive queries in AnQL [1]
  - \( \text{Mat}(Q, V1) \)
    - version materialization
  - \( \text{Diff}(Q, V1, V2) \)
  - \( \text{Ver}(Q) \)
  - \( \text{join}(Q1, vi, Q2, vj) \)
  - \( \text{Change}(Q) \)

Benchmarking: Define the queries

- Instantiation of archive queries in AnQL

- \( \text{Mat}(Q, V1) \)
- \( \text{Diff}(Q, V1, V2) \)
  - delta materialization
- \( \text{Ver}(Q) \)
- \( \text{join}(Q1, vi, Q2, vj) \)
- \( \text{Change}(Q) \)

```sql
SELECT * WHERE {
  { {Q :[v1]} MINUS {Q :[v2]} } BIND (v1 AS ?V ) } UNION
  { {Q :[v2]} MINUS {Q :[v1]} } BIND (v2 AS ?V ) }
```
Benchmarking: Define the queries

- Instantiation of archive queries in AnQL

- \( \text{Mat}(Q, V1) \)
- \( \text{Diff}(Q, V1, V2) \)
- \( \text{Ver}(Q) \)
  - results of \( Q \) annotated with the version
  - join\((Q1, vi, Q2, vj)\)
- \( \text{Change}(Q) \)

```sql
SELECT * WHERE { Q : ?V }
```
Benchmarking: Define the queries

- Instantiation of archive queries in AnQL
  - \( \text{Mat}(Q,V1) \)
  - \( \text{Diff}(Q,V1,V2) \)
  - \( \text{Ver}(Q) \)
  - \( \text{join}(Q1,v1,Q2,v2) \)
  - \( \text{Change}(Q) \)

```sql
SELECT * WHERE { {Q : [v1]} {Q : [v2]} }
```
Benchmarking: Define the queries

- Instantiation of archive queries in AnQL
  - $\text{Mat}(Q, V_1)$
  - $\text{Diff}(Q, V_1, V_2)$
  - $\text{Ver}(Q)$
  - $\text{join}(Q_1, vi, Q_2, vj)$
  - $\text{Change}(Q)$
    - Returns consecutive versions in which Diff of a query is not null

Open question remains: What is the right query syntax for archive queries?

```sql
SELECT ?V1 ?V2 WHERE
{ {{Q : ?V1} MINUS {Q : ?V2}} UNION
  {{Q : ?V2} MINUS {Q : ?V1}}
  FILTER( abs(?V1 - ?V2) = 1 ) }
```
RDF evolution at Scale

ANDREAS HARTH - STREAM REASONING IN MIXED REALITY APPLICATIONS, STREAM REASONING WORKSHOP 2015

Update rate
second
minute
hour
day
week
month
year

Virtual/Augmented Reality
Internet of Things

Number of sources

10^0 10^1 10^2 10^3 10^4 10^5 10^6

DBpedia
BTC

Dynamic Linked Data Observatory
ERI

DBpedia
BTC

Internet
of
Things

Virtual/
Augmented
Reality
So far, so good. But...

- **Use Case**
  - Querying historical and streaming data
  - ZAMG
    - How to archive historical data efficiently?
      - 24/7 measurements, updated each 15 minutes
    - How to efficiently serve versioning&streaming queries across time?

- **Is RDF Archiving** equal to **RDF Streaming**?
  - Is P equal to NP?
An RDF stream is a sequence of RDF graphs, including associated metadata... such as:

- **production time**: when the data element was produced.
- **receiving time**: when the data element was made available to the RSP engine.
- **start time, end time**: when the data element was valid, started, and ended.

An RDF stream should be defined as a potentially unbounded sequence of time-annotated elements.

```prolog
:g1 {axel :isIn RedRoom. :darko :isIn RedRoom} {g1 prov:generatedAtTime t1}
:g2 {axel :isIn GreenRoom. :darko :isIn RedRoom} {g2 prov:generatedAtTime t2}
:g3 {axel :isIn GreenRoom.} {g3 prov:generatedAtTime t3}
```
A version-annotated triple is an RDF triple \((s, p, o)\) with a label \(i \in \mathbb{N}\) representing the version in which this triple holds, denoted by the notation \((s, p, o) : [i]\).

An RDF version of an RDF archive \(A\) at snapshot \(i\) is the RDF graph \(A(i) = \{(s, p, o) | (s, p, o) : [i] \in A\}\).

```r
:g1 {:axel :isIn :RedRoom. :darko :isIn :RedRoom} {:g1 owl:versionInfo "1.0"
:g2 {:axel :isIn :GreenRoom. :darko :isIn :RedRoom} {:g2 owl:versionInfo "2.0" ;
   dct:isVersionOf g1 .}
:g3 {:axel :isIn :GreenRoom.} {:g3 owl:versionInfo "3.0" ; dct:isVersionOf g2 .}
```
The demon is in the details

Streaming

\[
g_1 \{ :\text{axel :isIn :RedRoom}. :\text{darko :isIn :RedRoom} \} \quad \{ g_1 \ \text{prov:generatedAtTime} \ t_1 \}
\]

\[
g_2 \{ :\text{axel :isIn :GreenRoom}. :\text{darko :isIn :RedRoom} \} \quad \{ g_2 \ \text{prov:generatedAtTime} \ t_2 \}
\]

\[
g_3 \{ :\text{axel :isIn :GreenRoom}. \} \quad \{ g_3 \ \text{prov:generatedAtTime} \ t_3 \}
\]

\[
( :\text{darko :isIn :RedRoom} ) \text{ is still valid in } t_3
\]

Versioning

\[
g_1 \{ :\text{axel :isIn :RedRoom}. :\text{darko :isIn :RedRoom} \} \quad \{ g_1 \ \text{owl:versionInfo} \ "1.0" \}
\]

\[
g_2 \{ :\text{axel :isIn :GreenRoom}. :\text{darko :isIn :RedRoom} \} \quad \{ g_2 \ \text{owl:versionInfo} \ "2.0" ; \text{dct:isVersionOf} \ g_1 . \}
\]

\[
g_3 \{ :\text{axel :isIn :GreenRoom}. \} \quad \{ g_3 \ \text{owl:versionInfo} \ "3.0" ; \text{dct:isVersionOf} \ g_2 . \}
\]

We “cannot” assume ( :darko :isIn :RedRoom ) by version 3.0
The demon is in the details

+ validity intervals

: g1 { :axel :isIn :RedRoom. :darko :isIn :RedRoom } { :g1 :atInterval t1/t1+10s }

: g2 { :axel :isIn :GreenRoom. :darko :isIn :RedRoom } { :g2 :atInterval t2/t2+10s }

: g3 { :axel :isIn :GreenRoom. } { :g3 :atInterval t3/t3+10s }

We “cannot” assume (:darko :isIn :RedRoom) by t3

We “cannot” assume (:darko :isIn :RedRoom) by version 3.0

*t3>t2+10s*
Many RDF Stream Query Languages (RSP-QL, CQELS, C-SPARQL, SPARQLStream, etc.) with different features and operators.

- RSPs should support defining **one or more time windows** over an RDF stream.

Query: Who is where in the last minute? in **CQELS**:

```
SELECT ?room ?person
WHERE {
  STREAM ex:social [RANGE 1m]
  { ?person :isIn ?room }
}
```
Some SPARQL-temporal extensions (T-SPARQL, SPARQL-ST) + DIACHRON Query Language) + AnQL

Query: Who is where? in DIACHRON:

```
SELECT ?version ?room ?person
WHERE {
  DATASET <g> AT VERSION ?version { 
    ?person :isIn ?room }
}
```

Query: Who is where? in AnQL:

```
SELECT ?version ?room ?person
WHERE {
  ?person :isIn ?room : ?version
}
```

- We cannot specify a “last minute” unless it is explicitly determined in the version
- We cannot assume an input stream but a “materialised” dataset
In contrast, archiving query languages can be translated to SPARQL and benefit from streaming constructions.

Query: get differences in **AnQL**:  

```sparql
SELECT * WHERE {
  { ?person :isIn ?room :[v1] }
  MINUS
  { ?person :isIn ?room :[v2] }
  BIND (v1 AS ?V )
}
```

Query: get differences in **CQELS**:  

```sparql
SELECT ?room ?person
WHERE {
  STREAM ex:social [NOW TO NOW - 1 HOURS]
  { ?person :isIn ?room }
  STREAM ex:social [NOW - 1 HOUR TO NOW - 2 HOURS]
  { FILTER NOT EXISTS{
    ?person :isIn ?room }
  }
}
```
Conclusions/Agenda

- Representation of RDF versions and RDF streaming has a common root on **temporal RDF**
  - The similarities are bigger than the differences
    - Mostly different in **validity time vs. transaction/observation time**
    - But they are two independent communities so far (* RDF versioning is in an early stage *)
  - There is a need of **efficiently representing and retrieving archives of RDF streams**
    - Store: Following/adapting current archiving policies
    - Queries: Time-based queries on archiving are “simpler” than in streaming
      - But some operators (e.g. diff) could be efficiently integrated/supported by RSP engines.
    - RDF archives can benefit from vast research in RDF stream reasoning.
Thanks!

Dept. of Information Systems & Operations
Institute for Information Business
Welthandelsplatz 1, 1020 Vienna, Austria

DR. Javier D. Fernández
T +43-1-313 36-5241
F +43-1-313 36-739
jfernand@wu.ac.at
www.ai.wu.ac.at

📞😊 Big (Semantic) Data
📞😊 Versions
📞😊 Evolving Data
📞😊 Streaming
📞😊 Encryption
📞😊 Compression

rdfhdt.org

http://aic.ai.wu.ac.at/qadlod/