On Archiving Linked and Open Data

Axel Polleres
(with a lot of input by Javier Fernandez & Jürgen Umbrich ;-)

30.05.2016 - 2nd Workshop on Managing the Evolution and Preservation of the Data Web, MEPDaW@ESWC16
What we will talk about...

- Monitoring Evolution and Archiving ... why is it important? Some examples...
- General Challenges of Archiving the Web of Data
- Some challenges more in-depth
- Discussion... (hopefully 🙂 )
Why evolution matters

(Creationists: please ignore this slide...)

- Monitoring evolution is relevant
Evolution matters

- Changes tell us “something”
- Uncertain information
- Validity of the information

Donald Trump: Difference between revisions
Evolution matters

- Evolution may reveal actions vs. consequences
  - E.g. CityDataPipeline, [http://citydata.wu.ac.at/](http://citydata.wu.ac.at/)
    - Collecting, Integrating and Predicting Open City Data
Preservation matters

- Web archives: Common Crawl, Internet Memory, Internet Archive, ...

[Image of Internet Archive Wayback Machine with a link to The New York Times]

http://www.nytimes.com

Saved 36,943 times between November 12, 1996 and May 25, 2016.

PLEASE DONATE TODAY. Your generosity preserves knowledge for future generations. Thank you.
Time-based access matters

- The Memento protocol
  
  RFC 7089

Follow your nose
(HTTP content negotiation with datetime)

Batch discovery
(list of URIs of Mementos of the Original Resource)

But...
Challenges

- Poor **granularity** ("some" snapshots)
- Aggregated data, only, rather than raw data access
  - (e.g. in Google trends)
- Few work on archiving the **Web of Data**, or on integrating archives
- What is the right **query language**?
  - basic retrieval features (get version at timestamp $t$)
  - when did a certain information disappear?
  - when was it changed?
  - structured queries?
- Scalability problems

Is it easier/better for RDF/Linked Data?
Most semantic Web/Linked Data tools are focused on this “static view” but do not consider versioning/evolution.
RDF Archiving. Example

RDF Graph $V_1$

```xml
ex:C1 ex:hasProfessor ex:P1.
ex:S1 ex:study ex:C1.
ex:S2 ex:study ex:C1.
```

RDF Graph $V_2$

```xml
ex:C1 ex:hasProfessor ex:P1.
ex:S1 ex:study ex:C1.
ex:S2 ex:study ex:C1.
ex:S3 ex:study ex:C1.
```

RDF Graph $V_3$

```xml
ex:C1 ex:hasProfessor ex:P1.
ex:C1 ex:hasProfessor ex:P2.
ex:C1 ex:hasProfessor ex:S2.
ex:S1 ex:study ex:C1.
ex:S3 ex:study ex:C1.
```
One of the first real LOD use cases: The Dynamic Linked Data Observatory (evolving Linked Data since 2012)

Weekly dumps of crawl snapshots...


The Dynamic Linked Data Observatory

The Dynamic Linked Data Observatory is a framework to monitor Linked Data over an extended period of time. The core goal of our work is to collect frequent, continuous snapshots of a subset of the Web of Data that is interesting for further study and experimentation, with an aim to capture raw data about the dynamics of Linked Data. The resulting corpora will be made openly and continuously available to the Linked Data research community.

**Databases**

- **Datasets**
  - Good news! We started with the weekly crawls.

**Publications**

- Tobias Käfer, Ahmed Abdelrahman, Jürgen Umbrich, Patrick O’Byrne, Aidan Hogan.

**Team & Contact**

- Tobias Käfer (KIT, Karlsruhe)
- Jürgen Umbrich (DERI, Galway)
- Aidan Hogan (DERI, Galway)
- Patrick O’Byrne (NUI, Galway, Ireland)
- Ahmed Abdelrahman (Dublin, DERI, Galway, Ireland)
- Google group

**Results**

- Update: The newest crawl dumps should be available for download roughly one day after they got crawled.
Research challenges on evolving structured interlinked data

- How can we *represent archives* of continuously evolving linked datasets? (efficiency vs. compact representation)
- How can we *minimize the redundant information* of archives? (e.g. duplicates in snapshots)
- How can we improve *completeness* of archiving?
- How can emerging retrieval demands in archiving be satisfied?
  - *e.g.* time-traversing and traceability? Avoiding bottlenecks?
- How can certain *time-specific queries* over archives be answered?
  - Can we re-use existing technologies (e.g. SPARQL or temporal extensions)?
  - *What is the right query language for such queries?*
  - *e.g.* knowing if a dataset has changed, and how, in a certain time period?
General archiving challenges

- **The synchronisation problem**
  - how can we monitor changes?

- **The appraisal problem**
  - how can we assess the quality of a dataset? (and does archiving help with that?)

- **The archiving & query problem**
  - how can we efficiently archive and perform time-based retrieval queries of a dataset?
The synchronization problem
how can we monitor changes?
Pull changes (crawl) vs. Push changes (notify)

- Observations:
  - Some services that publish or are mapped to RDF change **regularly**, but we don’t know the frequency upfront!
  - Some services mapped to RDF **announce/archive their changes already**, so they already keep an archive...
1- An adaptive archiver (recall: DIACHRON WS 2015)

Data Monitor Framework

- URI type
- scheduler
  - crawl schedule
  - crawl metadata
- downloader
  - politeness queue
  - meta
  - content
- links
- data
  - YYYY/MM/DD/HH/domain

**adaptive scheduler**
- check if URL was crawled,
- compare content with previous crawl(s),
- adapt schedule
Experiment: Rescheduling

- Evaluating strategies to compute next crawl time for URLs to accurately capture content change
Test setup

- Revision history of 2660 wikipedia-articles
  - Wiki-changes do not follow a typical Poisson distribution
- Several heuristics; e.g.:
  - e.g., increase the crawl frequency,...”
  - “if you observe several changes in a row”
  - “if the probability is high that the content changes after we observed a change in the last snapshot” → Markov models
  - “If the document changed more often than 50% in the last 10 days .... “
We observed the typical trade-off between recall and precision... Strategies based on Markov models seems to provide best and most stable trade-off
Experiment: Improve completeness - Crawl time estimation

- **Aim:** Estimate the overall crawl time and needed number of threads for a set of URLs from different domains

- **Heuristic**
  1. Estimate the crawl time per domain
     - Average download time, domain delay etc...
  2. First-fit bin-packing algorithms to determine overall crawl time
### Results

**Table 3. Results of crawl time estimation**

<table>
<thead>
<tr>
<th>Actual crawl time</th>
<th>Overestimate (crawls)</th>
<th>Underestimate (crawls)</th>
<th>total crawls</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t &lt; 30\text{min}$</td>
<td>78% (202)</td>
<td>-16% (154)</td>
<td>356</td>
</tr>
<tr>
<td>$30\text{min} &lt; t &lt; 60\text{min}$</td>
<td>134% (25)</td>
<td>-18% (46)</td>
<td>71</td>
</tr>
<tr>
<td>$60\text{min} &lt; t &lt; 120\text{min}$</td>
<td>638% (19)</td>
<td>-11% (51)</td>
<td>70</td>
</tr>
<tr>
<td>$120\text{min} &lt; t &lt; 180\text{min}$</td>
<td>1.3% (15)</td>
<td>-1% (215)</td>
<td>230</td>
</tr>
<tr>
<td>$180\text{min} &lt; t &lt; 240\text{min}$</td>
<td>- (0)</td>
<td>-36% (4)</td>
<td>4</td>
</tr>
<tr>
<td>$t &gt; 240\text{min}$</td>
<td>- (0)</td>
<td>-44% (28)</td>
<td>28</td>
</tr>
</tbody>
</table>

- Large over estimation (is acceptable)
- Small underestimation (e.g. 10mins for 60mins crawl)
Pull changes (crawl) vs. Push changes (notify)

Observations:

- Some services that publish or are mapped to RDF change regularly, but we don’t know the frequency upfront!

- Some services mapped to RDF announce/archive their changes already, so they already keep an archive...
2- “Recreate” the versions from sources (SEMANTiCS 2015 demo...)

- If raw historical data on changes is available...
- Aim: Fine grained access to previous versions, re-applying X2RDF transformations on the original source.
- Example: **DBpedia Wayback machine**
  - Re-apply mappings on the Wikipedia revision history.

http://data.wu.ac.at/wayback/
2– “Recreate” the versions

- How can one represent revisions while respecting DBpedia?
  - a) quads → `<dbpediaSubject> <pred> <obj> <Revision>` .
  - b) proprietary triples → `<ownSubject/Revision> <pred> <obj>` .

- Operations?
  - Get revisions meta-data for one resource (by revisionID or timestamp)
  - Get “materialised” versions of a resource (by revisionID or timestamp)
  - Get difference between two revisions
2- “Recreate” the versions

- More complex operations/queries? **Open challenge**
  - a) On-demand? Query rewriting, similar to RDB2RDF
  - b) Batch: Fetch the desired information, then store and query it
We are (obviously) not the only ones looking into this...

However:
Only one HDT per “irregular” dbpedia dump
The appraisal problem

How can we assess the quality of a dataset?
Data Quality issues:

- Missing
- Outdated data
- Wrong data
- Ambiguous Data
- Wrong meta-data
- Data source offline/not reachable

→ Archiving & looking at the history of datasets helps!
Open Data Portals

CKAN ... http://ckan.org/

- almost „de facto“ standard for Open Data Portals
- facilitates search, metadata (publisher, format, publication date, license, etc.) for datasets

- http://datahub.io/
- http://data.gv.at/

- machine-processable? ... ... partially
OPEN DATA PORTAL WATCH
... a first step.

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

- Periodically monitoring a list of Open Data Portals
  - 90 CKAN powered Open Data Portals
- Quality assessment
- Evolution tracking
  - Meta data
  - Data
Open Data Portal Watch

Brief overview of 89 Open Data CKAN portals

Sort by: [Domain] [Country] [Datasets] [Resources]
Filter: Filter by portal or country

annuario.comune.fi.it
Italy
- 358 DATASETS
- 1363 RESOURCES

catalogue.datalocale.fr
France
- 303 DATASETS
- 751 RESOURCES

dados.gov.br
Brazil
- 501 DATASETS
- 4344 RESOURCES

data.buenosaires.gov.ar
Argentina
- 123 DATASETS
- 626 RESOURCES

data.edostate.gov.ng
Nigeria
- 164 DATASETS
- 207 RESOURCES

data.glasgow.gov.uk
United Kingdom (common practice)
- 384 DATASETS
- 1943 RESOURCES

data.gov.uk
United Kingdom (common practice)
- 360 DATASETS
- 506 RESOURCES

data.gov.sk
Slovakia
- 216 DATASETS
- 556 RESOURCES

ckan.data.graz.gv.at
Austria
- 151 DATASETS
- 341 RESOURCES

data.kk.ca
Denmark
government
- 102 DATASETS
- 346 RESOURCES

data.lexingtonky.gov
government
- 93 DATASETS
- 186 RESOURCES

data.new.gov.au
Australia
- 311 DATASETS
- 458 RESOURCES

data.ohouston.org
non-commercial
- 227 DATASETS
- 361 RESOURCES

data.ottawa.ca
Canada
- 119 DATASETS
- 493 RESOURCES

data.cityofsantacruz.com
commercial
- 52 DATASETS
- 72 RESOURCES

dados.recife.pe.gov.br
Brazil
- 43 DATASETS
- 318 RESOURCES
# Quality Dimensions

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrievability</td>
<td>The extent to which meta data and resources can be retrieved.</td>
</tr>
<tr>
<td>Usage</td>
<td>The extent to which available meta data keys are used to describe a dataset.</td>
</tr>
<tr>
<td>Completeness</td>
<td>The extent to which the used meta data keys are non empty.</td>
</tr>
<tr>
<td>Accuracy</td>
<td>The extent to which certain meta data values accurately describe the resources.</td>
</tr>
<tr>
<td>Openness</td>
<td>The extent to which licenses and file formats conform to the open definition.</td>
</tr>
<tr>
<td>Contactability</td>
<td>The extent to which the data publisher provide contact information.</td>
</tr>
</tbody>
</table>

Objective measures which can be automatically computed in a scalable way.
Portal Overview

Open Data Portal Watch

Portal: GovData | Datenportal für Deutschland - GovData

Available Snapshots

Snapshot: Sun Feb 22 2015 23:52:47 GMT+0100 (CET)

- QUALITY
- SIZE
- OPENNESS
- RETRIEVABILITY
- CONTACTABILITY
ODP Evolution
## Changes between the first and last snapshots

### Dataset Changes

#### 70 Portals with Dataset Changes

- Avg. increase by 87.05% for 60 portals
- Avg. decrease by -54.16% for 10 portals

**Show [10 1] entries**

<table>
<thead>
<tr>
<th>PORTAL</th>
<th>FROM</th>
<th>TO</th>
<th>CHANGE</th>
<th>CHANGE PERCENTAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>data.sa.gov.au</td>
<td>484</td>
<td>5721</td>
<td>5237</td>
<td>1082.02%</td>
</tr>
<tr>
<td>datos.codeandomexico.org</td>
<td>94</td>
<td>715</td>
<td>621</td>
<td>660.64%</td>
</tr>
<tr>
<td>data.opendataportal.at</td>
<td>46</td>
<td>323</td>
<td>277</td>
<td>602.17%</td>
</tr>
<tr>
<td>(2014-07-17) → (2015-03-16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>annuario.comune.fi.it</td>
<td>50</td>
<td>351</td>
<td>301</td>
<td>602.00%</td>
</tr>
<tr>
<td>(2014-08-07) → (2015-03-15)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>udct-data.aigid.jp</td>
<td>431</td>
<td>2110</td>
<td>1679</td>
<td>389.56%</td>
</tr>
<tr>
<td>(2014-08-07) → (2015-03-16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>catalogo.datos.gob.mx</td>
<td>111</td>
<td>360</td>
<td>249</td>
<td>224.32%</td>
</tr>
<tr>
<td>(2014-08-08) → (2015-03-15)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Dumps

OPEN DATA PORTAL WATCH provides an archive of Open Data portal crawls (weekly snapshots/dynamic crawling framework):

<table>
<thead>
<tr>
<th>Open Data Portal Watch Dumps</th>
<th>Open Data Portal Watch Dumps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Parent Directory</td>
<td>Parent Directory</td>
</tr>
<tr>
<td>africacopendata.org/</td>
<td>2014-07-17.gz</td>
</tr>
<tr>
<td>annuario.comune.fi.it/</td>
<td>2014-07-25.gz</td>
</tr>
<tr>
<td>bermuda.io/</td>
<td>2014-08-05.gz</td>
</tr>
<tr>
<td>catalog.data.gov/</td>
<td>2014-08-12.gz</td>
</tr>
<tr>
<td>catalog.data.ug/</td>
<td>2014-08-27.gz</td>
</tr>
<tr>
<td>catalogo.datos.gob.mx/</td>
<td>2014-09-01.gz</td>
</tr>
<tr>
<td>catalogodatos.gub.uy/</td>
<td>2014-09-07.gz</td>
</tr>
<tr>
<td>Last modified</td>
<td>Last modified</td>
</tr>
<tr>
<td>05-Feb-2015 15:28</td>
<td>05-Feb-2015 15:13</td>
</tr>
<tr>
<td>Size</td>
<td>Size</td>
</tr>
<tr>
<td>-</td>
<td>2.2M</td>
</tr>
<tr>
<td>-</td>
<td>2.2M</td>
</tr>
<tr>
<td>-</td>
<td>2.2M</td>
</tr>
<tr>
<td>-</td>
<td>2.2M</td>
</tr>
<tr>
<td>-</td>
<td>2.2M</td>
</tr>
<tr>
<td>-</td>
<td>2.2M</td>
</tr>
<tr>
<td>-</td>
<td>2.2M</td>
</tr>
<tr>
<td>-</td>
<td>2.2M</td>
</tr>
</tbody>
</table>
Quality assessment & evolution of Open Data portals

Jürgen Umbrich*, Sebastian Neumaier*, Axel Polleres*
Vienna University of Economics and Business, Vienna, Austria
Email: *firstname.lastname@wu.ac.at

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

- Key findings:
  - Significantly varying quality across portals
  - Rapid growth for some portals
  - Huge variety and range of datasets
  - Open Data Portal search is a big problem
  - Time: many datasets only provide current, but no historical data

Best paper award at IEEE OBD-2015 😊
Historical vs. current-only data (monotonic changes vs. non-monotonic changes)

- Weather data (every 15min) from 21 Austrian weather stations...
- vs.
- Population per gender and age in Vienna districts

Updated every 15min, only current data

Updated annually, historical data since 2011

Connection to Challenge 1 (Synchronization): Adequate meta-data could help us to steer crawling, and more efficient storage, we are experimenting with this...
Now: How do data quality and archiving connect?

- **Idea:** if we know how data changed, we could assess “bogus” changes...
  - Look at **time series:** Detect outliers over historical data
  - Example: wikipedia change history!
    - Wrong/disputed data changes often!

**e.g. obvious Idea:**
Once data is semantically integrated in an archive, it is easy to include time-series analysis to filter out implausible/inconsistent data automatically...
Now: How do data quality and archiving connect?

- An idea only, so far: if we know how data changed, we could assess “bogus” changes...
- Look at time series: Detect outliers over historical data
- Example: Wikipedia change history!
- Assumption: Wrong/disputed data changes often!
Finally: The *archiving problem*

Now, how can we efficiently archive and perform time-based retrieval queries of a dataset?
RDF Archiving. Archiving policies

a) Independent Copies/Snapshots (IC)

b) Change-based approach (CB)

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

- Structured query languages managing time.
  - Temporal databases (T-Quel, TSQL2)
    - Overlapping, meeting, before, equal, during, finish
  - RDF/Linked Data
    - SPARQL extensions
      - T-SPARQL, SPARQL-ST
      - AnQL
    - DIACHRON Query Language
      - SPARQL with specific constructors such as DATASET (similar to a named graph), VERSION, or CHANGES
BEAR: Benchmarking the Efficiency of RDF Archives

- Blueprint on benchmarking archives of semantic data
  - How can one define the corpus?
  - How can one design benchmark queries? Which queries?
- BEAR: concrete basic benchmark
  - Data: Crawl from Linked Data Observatory
  - Basic queries: Materialize, get Version...
  - Initial evaluation on archiving policies
**BEAR: Benchmarking the Efficiency of RDF Archiving**

- **Blueprint on benchmarking archives of semantic data**
  - How can one define the corpus?
  - How can one design benchmark queries? Which queries?
- **BEAR: concrete basic benchmark**
  - Data: Crawl from Linked Data Observatory
  - Basic queries: Materialize, get Version...
  - Initial evaluation on archiving policies
**BEAR: Benchmarking the Efficiency of RDF Archiving**

- **Define the corpus**
  - Number of versions / size

  **Definition 1 (RDF Archive).** A version-annotated triple is an RDF triple \((s, p, o)\) with a label \(i \in N\) representing the version in which this triple holds, denoted by the notation \((s, p, o) : [i]\). An RDF archive graph \(A\) is a set of version-annotated triples.

  - Data dynamicity
    - Version change ratio
    - Version data growth
  - Data static core
  - Total triples (version-oblivioues)
  - RDF vocabulary
    - Per version / evolution
**BEAR: Benchmarking the Efficiency of RDF Archiving**

- Design of benchmark queries
  - Cardinality / Selectivity + dynamicity
  - Archive-driven C/S/D
  - Version-driven C/S/D
  - Basic temporal retrieval features of queries
    - Version/Delta Materialization ($V_i$)
    - Version($Q$): in which version $Q$ is not empty
    - Change($V_i, V_j$): true if delta!=$null$
    - Join($Q_1, V_i, Q_2, V_j$)
    - Change($Q$): Returns versions in which $\text{Diff}(Q, V_i, V_{i-1}) \neq \emptyset$
Instantiation of basic archive queries, e.g. in AnQL [1]


Mat(Q,V)
Diff(Q,V1,V2)
Ver(Q)
join(Q1,vi,Q2,vj)
Change(Q)

SELECT ?V1 ?V2 WHERE
{ {P :?V1 } MINUS {P :?V2} } UNION
{ {P :?V2 } MINUS {P :?V1} }
FILTER( abs(?V1-?V2) = 1 ) }

Open question remains: What is the right query syntax for archive queries?
BEAR: Benchmarking the Efficiency of RDF Archiving

- blueprint on benchmarking archives of semantic data
  - How can one define the corpus?
  - How can one design benchmark queries? Which queries?
- BEAR: concrete basic benchmark
  - Data: Crawl from Linked Data Observatory
  - Basic queries: Materialize, get Version...
  - Initial evaluation of archiving policies (IC, CB, TB)
**BEAR: Benchmarking the Efficiency of RDF Archiving**

- **Corpus**

| versions | $|V_0|$ | $|V_{57}|$ | growth | $\bar{\delta}$ | $\delta^-$ | $\delta^+$ | $C_A$ | $O_A$ |
|----------|--------|--------|--------|---------------|-----------|-----------|-------|-------|
|          | 58     | 30m    | 66m    | 101%          | 31%       | 32%       | 27%   | 3.5m  | 376m |

(a) Number of statements

(b) Relative growth and dynamicity
BEAR: Benchmarking the Efficiency of RDF Archiving

- Queries and systems
  - We implemented and evaluate archiving systems on **Jena-TDB** and **HDT**, based on IC, CB and TB policies.
    - Confirm the initial premises of the archiving policies:
      - In space, IC is the worst, CB improves the space and TB increases the size as it has to index a new dimension
      - In time, CB is bad at getting a particular version because it has to reapply the changes … (but good e.g. for `Change(SELECT * {?S ?P ?O})`)
  - Serve as an initial baseline to compare archiving systems
  - More info: [https://github.com/webdata/BEAR](https://github.com/webdata/BEAR)
Finally, many open questions remain still!

Archiving and querying evolving semantic Web data

<table>
<thead>
<tr>
<th>Objective</th>
<th>Research Question</th>
</tr>
</thead>
</table>
| Representation     | ✤ minimize the redundant information  
                      ✤ respect the original modeling and provenance information             |
| Query language     | ✤ capture the expressiveness of emerging retrieval demands in archiving  
                      ✤ our base operations are meant to be an **extensible** starting point  
                      ✤ design a query language satisfying these requirements for evolving interlinked data |
| Indexing           | ✤ index archives at large scale (and keeping up with evolution rate – streaming vs. archiving) to process the queries efficiently |
| Query optimization | ✤ optimizing query resolution plans for archives  
                      ✤ enabling the integration of other sources (federated infrastructure)  
                      ✤ Query rewriting for querying archives of structured non-RDF sources? Open Data! |
| Application        | ✤ Is there a actual and urgent need in the community?  
                      ✤ We believe yes, but where’s the killer-app?                           |
Thanks!

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

Univ.Prof. Dr. Axel Polleres
T +43-1-31336/5297
axel.polleres@wu.ac.at
polleres.net

Big (Semantic) Data Versions
Evolving Data Streaming
Compression
Instantiation of archive queries in AnQL [1]


- $\text{Mat}(Q,V)$
- $\text{Diff}(Q,V_1,V_2)$
- $\text{Ver}(Q)$
- $\text{join}(Q_1,vi,Q_2,vj)$
- $\text{Change}(Q)$

```
SELECT * WHERE { Q :[v] }
```
Backup

- Instantiation of archive queries in AnQL [1]


- **Mat**(Q,V)
- **Diff**(Q,V1,V2)
- Ver(Q)
- join(Q1,vi,Q2,vj)
- Change(Q)

```
SELECT * WHERE {
  { {Q :[v1]} MINUS {Q :[v2]} } BIND (v1 AS ?V )
}
UNION
{ { {Q :[v2] } MINUS {Q :[v1]}} BIND (v2 AS ?V )
}
```
Instantiation of archive queries in AnQL [1]


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

```sql
SELECT * WHERE { P :?V }
```
Instantiation of archive queries in AnQL [1]


- Mat(Q,V)
- Diff(Q,V1,V2)
- Ver(Q)
- join(Q1,v1,Q2,v2)
- Change(Q)

```
SELECT * WHERE { {Q :[v1]} {Q :[v2]} }
```
• Instantiation of archive queries in AnQL [1]


- **Mat(Q,V)**
- **Diff(Q,V1,V2)**
- **Ver(Q)**
- **join(Q1,vi,Q2,vj)**
- **Change(Q)**

```sql
SELECT ?V1 ?V2 WHERE
{ {P :?V1 } MINUS {P :?V2} } UNION
{ {P :?V2 } MINUS {P :?V1} }
FILTER( abs(?V1-?V2) = 1 ) }
```