Approaches to Making Data Citeable
Recommendations of the RDA Working Group

Andreas Rauber,
Ari Asmi, Dieter van Uytvanck
Stefan Pröll
Outline

- Challenges addressed by the WG
- Recommendation of the RDA Working Group
- Benefits & Pilots
  - Links to Cross-WG and further activities
- Summary
Citing (identifying) data may seem easy
- from providing a URL in a footnote
- via providing a reference in the bibliography section
- to assigning a PID (DOI, ARK, …) to dataset in a repository

What’s the problem?
Citation of Dynamic Data

- Citable datasets have to be static
  - Fixed set of data, no changes: no corrections to errors, no new data being added
- But: (research) data is **dynamic**
  - Adding new data, correcting errors, enhancing data quality, …
  - Changes sometimes highly dynamic, at irregular intervals
- Current approaches
  - Identifying entire data stream, without any versioning
  - Using “accessed at” date
  - “Artificial” versioning by identifying batches of data (e.g. annual), aggregating changes into releases (time-delayed!)
- Would like to cite precisely the **data as it existed at certain point in time**, without delaying release of new data
Granularity of Data Citation

- What about the granularity of data to be cited?
  - Databases collect enormous amounts of data over time
  - Researchers use specific subsets of data
  - Need to identify precisely the subset used

- Current approaches
  - Storing a copy of subset as used in study -> scalability
  - Citing entire dataset, providing textual description of subset -> imprecise (ambiguity)
  - Storing list of record identifiers in subset -> scalability, not for arbitrary subsets (e.g. when not entire record selected)

- Would like to be able to cite precisely the subset of (dynamic) data used in a study
Data Citation – Requirements

- Dynamic data
  - corrections, additions, …

- Arbitrary subsets of data (granularity)
  - rows/columns, time sequences, …
  - from single number to the entire set

- Stable across technology changes
  - e.g. migration to new database

- Machine-actionable
  - not just machine-readable,
    definitely not just human-readable and interpretable

- Scalable to very large / highly dynamic datasets
  - but should also work for small and/or static datasets
- Research Data Alliance
- WG on **Data Citation**: Making Dynamic Data Citeable
- WG officially endorsed in March 2014
  - Concentrating on the problems of large, dynamic (changing) datasets
  - Focus!
    Not: PID systems, metadata, citation string, attribution, …
  - Liaise with other WGs and initiatives on data citation (CODATA, DataCite, Force11, …)

- https://rd-alliance.org/working-groups/data-citation-wg.html
Outline

- Challenges addressed by the WG
- Recommendation of the RDA Working Group
- Benefits & Pilots
  - Links to Cross-WG and further activities
- Summary
Making Dynamic Data Citeable

Data Citation: Data + Means-of-access

- Data → time-stamped & versioned (aka history)

Researcher creates working-set via some interface:

- Access → assign PID to QUERY, enhanced with
  - Time-stamping for re-execution against versioned DB
  - Re-writing for normalization, unique-sort, mapping to history
  - Hashing result-set: verifying identity/correctness

leading to landing page

S. Pröll, A. Rauber. Scalable Data Citation in Dynamic Large Databases: Model and Reference Implementation. In IEEE Intl. Conf. on Big Data 2013 (IEEE BigData2013), 2013
Data Citation – Deployment

- Researcher uses workbench to identify subset of data
- Upon executing selection („download“) user gets
  - Data (package, access API, …)
  - PID (e.g. DOI) (Query is time-stamped and stored)
  - Hash value computed over the data for local storage
  - Recommended citation text (e.g. BibTeX)
- PID resolves to landing page
  - Provides detailed metadata, link to parent data set, subset,…
  - Option to retrieve original data OR current version OR changes
- Upon activating PID associated with a data citation
  - Query is re-executed against time-stamped and versioned DB
  - Results as above are returned
Data Citation – Deployment

Note: query string provides excellent provenance information on the data set!

- Researcher uses workbench to identify subset of data
- Upon executing selection ("download") user gets
  - Data (package, access API, …)
  - PID (e.g. DOI) (Query is time-stamped and stored)
  - Hash value computed over the data for local storage
  - Recommended citation text (e.g. BibTeX)

- PID resolves to landing page
  - Provides detailed metadata, link to parent data set, subset,…
  - Option to retrieve original data OR current version OR changes

- Upon activating PID associated with a data citation
  - Query is re-executed against time-stamped and versioned DB
  - Results as above are returned
Data Citation – Deployment

- Researcher uses workbench to identify subset of data
- Upon executing selection ("download"); user gets
  - Data (package, access API, …)
  - PID (e.g. DOI) (Query is time-stamped and stored)
  - Hash value computed over the data for local storage
  - Recommended citation text (e.g. BibTeX)
- PID resolves to landing page
  - Provides detailed metadata, link to parent data set, subset,…
  - Option to retrieve original data OR current version OR changes
- Upon activating PID associated with a data citation
  - Query is re-executed against time-stamped and versioned DB
  - Results as above are returned

Note: query string provides excellent provenance information on the data set!

This is an important advantage over traditional approaches relying on, e.g. storing a list of identifiers/DB dump!!!
2-page flyer, more extensive doc to follow

14 Recommendations

Grouped into 4 phases:
- Preparing data and query store
- Persistently identifying specific data sets
- Upon request of a PID
- Upon modifications to the data infrastructure

History
- First presented March 30 2015
- Major revision after workshop April 20/21
- Series of webinars (next: June 24, 18:00 CEST)
A) Preparing the Data and the Query Store

- **R1 – Data Versioning**: Apply versioning to ensure earlier states of data sets the data can be retrieved

- **R2 – Timestamping**: Ensure that operations on data are timestamped, i.e. any additions, deletions are marked with a timestamp

- **R3 – Query Store**: Provide means to store the queries used to select data and associated metadata
A) Preparing the Data and the Query Store

- **R1 – Data Versioning:** Apply versioning to ensure earlier states of data sets the data can be retrieved

- **R2 – Timestamping:** Ensure that operations on data are timestamped, i.e. any additions, deletions are marked with a timestamp

- **R3 – Query Store:** Provide means to store the queries used to select data and associated metadata

**Note:**
- R1 & R2 are already pretty much standard in many (RDBMS-) research databases
- Different ways to implement
- A bit more challenging for some data types (XML, LOD, …)
B) Persistently Identify Specific Data sets (1/2)

*When a data set should be persisted:*

- **R4 – Query Uniqueness:** Re-write the query to a normalised form so that identical queries can be detected. Compute a checksum of the normalized query to efficiently detect identical queries.

- **R5 – Stable Sorting:** Ensure an unambiguous sorting of the records in the data set.

- **R6 – Result Set Verification:** Compute a checksum of the query result set to enable verification of the correctness of a result upon re-execution.

- **R7 – Query Timestamping:** Assign a timestamp to the query based on the last update to the entire database (or the last update to the selection of data affected by the query or the query execution time). This allows retrieving the data as it existed at query time.
Data Citation – Recommendations

B) Persistently Identify Specific Data sets (2/2)

When a data set should be persisted:

- **R8 – Query PID:** Assign a new PID to the query if either the query is new or if the result set returned from an earlier identical query is different due to changes in the data. Otherwise, return the existing PID.

- **R9 – Store Query:** Store query and metadata (e.g. PID, original and normalised query, query & result set checksum, timestamp, superset PID, data set description and other) in the query store.

- **R10 – Citation Text:** Provide a recommended citation text and the PID to the user.
C) Upon Request of a PID

- **R11 – Landing Page:** Make the PIDs resolve to a human readable landing page of the data set that provides metadata including a link to the superset (PID of the data source) and citation text snippet.

- **R12 – Machine Actionability:** Make the landing page machine-actionable, allowing to retrieve the data set by re-executing the timestamped query.
D) Upon Modifications to the Data Infrastructure

- **R13 – Technology Migration:** When data is migrated to a new representation (e.g. new database system, a new schema or a completely different technology), migrate also the queries and associated checksums.

- **R14 – Migration Verification:** Verify successful query migration should, ensuring that queries can be re-executed correctly.
Outline

- Challenges addressed by the WG
- Recommendation of the RDA Working Group
- Benefits & Pilots
  - Links to Cross-WG and further activities
- Summary
Benefits

- Retrieval of precise subset with minimal storage overhead
- Subset as cited or as it is now (including e.g. corrections)
- Query provides provenance information
- Checksums support verification
- Same principles applicable across all settings
  - Small and large data
  - Static and dynamic data
  - Different data representations (RDBMS, CSV, XML, LOD, …)
- Would work also for more sophisticated/general transformations on data
WG Pilots

- Pilot workshops and implementations by
  - Various EU projects (TIMBUS, SCAPE, …)
  - NERC (UK Natural Environment Research Council Data Centres)
  - ESIP (Earth Science Information Partners)
  - CLARIN (XML, Field Linguistics Transcriptions)
  - Virtual Atomic and Molecular Data Centre

- Prototype solutions for
  - SQL, CSV, XML (partially)
  - LOD/RDF, triple-store DBs in the queue
  - Distributed data
Outline

- Challenges addressed by the WG
- Recommendation of the RDA Working Group
- Benefits & Pilots
- Links to Cross-WG and further activities
- Summary
Next Steps / Cross-WG

For “our” WG

- Policy guidelines:
  - How important is query uniqueness?
  - How to manage need to keep the query/subset private for some time?

- Beyond select/project
  - Can we also handle more complex transformations?
  - Boundary to storing entire processes? Link to research objects?

- Impact and specific challenges
  - Distributed data
  - Support for data schema migration
  - Analysis of overhead, trade-off’s, issues with hash computation, …
Next Steps / Cross-WG

Cross-WG

- Recommendation on machine-actionability:
  - How to encode landing page to achieve this consistently?

- Query Store:
  - What information to include?
  - How to represent this information in a consistent encoding?

- Citation text snippet:
  - What to recommend? How to cite?

- Domain-specific aspects
  - Further pilots, feedback, …
  - Establishing it as part of the infrastructure
  - And anything other WGs might need from us…
Join RDA and Working Group

If you are interested in joining the discussion, contributing a pilot, wish to establish a data citation solution, …

- Register for the RDA WG on Data Citation:
  - Website: https://rd-alliance.org/working-groups/data-citation-wg.html
  - Mailinglist: https://rd-alliance.org/node/141/archive-post-mailinglist
Thank you!

https://rd-alliance.org/working-groups/data-citation-wg.html
Dynamic Data Citation for SQL Data
LNEC, MSD Implementation
SQL Prototype Implementation

- LNEC Laboratory of Civil Engineering, Portugal
- Monitoring dams and bridges
- 31 manual sensor instruments
- 25 automatic sensor instruments
- Web portal
  - Select sensor data
  - Define timespans
- Report generation
  - Analysis processes
  - LaTeX
  - publish PDF report
SQL Prototype Implementation

- Million Song Dataset
- Largest benchmark collection in Music Retrieval
- Original set provided by Echonest
- No audio, only several sets of features
  (16 – 1440 measurements/features per song)
- Harvested, additional features and metadata
  extracted and offered by several groups
  e.g. [http://www.ifstuwien.ac.at/mir/msd/download.html](http://www.ifstuwien.ac.at/mir/msd/download.html)
- Dynamics because of metadata errors, extraction errors
- Research groups select subsets by genre, audio length, audio quality,…
SQL Time-Stamping and Versioning

- **Integrated**
  - Extend original tables by temporal metadata
  - Expand primary key by record-version column

- **Hybrid**
  - Utilize history table for deleted record versions with metadata
  - Original table reflects latest version only

- **Separated**
  - Utilizes full history table
  - Also inserts reflected in history table

- **Solution to be adopted depends on trade-off**
  - Storage Demand
  - Query Complexity
  - Software adaption
Add query store containing:
- PID of the query
- Original query
- Re-written query + query string hash
- Timestamp (as used in re-written query)
- Hash-key of query result
- Metadata useful for citation / landing page (creator, institution, rights, ...)
- PID of parent dataset (or using fragment identifiers for query)
SQL Query Re-Writing

- Adapt query to history table

```sql
SELECT results.track_id, results.artist, results.release
FROM MSD AS results 
JOIN ( 
  SELECT track_id, max(timestamp) AS latestTimestamp 
  FROM MSD 
  WHERE timestamp <= (SELECT @queryExecutionTimestamp) 
  AND (track_id NOT IN 
    (SELECT track_id FROM MSD AS deletedRecords 
     WHERE deletedRecords.status_mark = 'deleted' 
     AND (deletedRecords.timestamp < @queryExecutionTimestamp))
  
  GROUP BY track_id
) AS version ON results.track_id = version.track_id AND results.timestamp = version.latestTimestamp
WHERE results.tags = 'classic' AND results.duration > 120
ORDER BY results.track_id;
```
Dynamic Data Citation for CSV Data
Open Source Reference Implementation
Dynamic Data Citation for CSV Data

- Why CSV data? (not large, not very dynamic…)
  - Well understood and widely used
  - Simple and flexible
  - Most frequently requested during initial RDA meetings

- Goals:
  - Ensure cite-ability of CSV data
  - Enable subset citation
  - Support particularly small and large volume data
  - Support dynamically changing data

- 2 Options:
  - Versioning system (subversion/svn, git, …)
  - Migration to RDBMS
CSV Prototype: Basic Steps

- Upload interface
  - Upload CSV files
- Migrate CSV file into RDBMS
  - Generate table structure, identify primary key
  - Add metadata columns for versioning
  - Add indices
- Dynamic data
  - Update / delete existing records
  - Append new data
- Access interface
  - Track subset creation
  - Store queries
CSV Data Prototype

Data Citation Tool for CSV Data

This tool allows to upload, update and reference CSV subsets.

Upload CSV data

- Upload new data
- Update existing data
- View existing data
Upload a new CSV data file

Provide a name first, then upload the file.

Database schema: CITATION_DB
Table name: MillionsongDataset

Choose

msd1k.csv

Primary key: track_id

Select primary key.

Migrate into Database.
View existing data
CSV Data Prototype

The image shows a screenshot of a CSV data prototype. The prototype includes a table with columns labeled 'artist_name', 'duration', 'artist_familiarity', 'artist_hottness', and 'year'. The table contains entries for Joseph Locke and The Sun Harbor's Chorus, with corresponding values for each column. The interface also includes buttons for initializing query store, storing current selection, and finalizing the dataset.
CSV Data Prototype

SQL string

Suggested citation text:

Download area

Download CSV Subset

Download Latest Subset

Download Full DB

Download Diff CSV file
Progress update from VAMDC
Distributed Data Centre

Carlo Maria Zwölf
Virtual Atomic and Molecular Data Centre
carlo-maria.zwolf@obspm.fr
VAMDC

- Virtual Atomic and Molecular Data Centre
- Worldwide e-infrastructure federating 41 heterogeneous and interoperable Atomic and Molecular databases
- Nodes decide independently about growing rate, ingest system, corrections to apply to already stored data
- Data-node may use different technology for storing data (SQL, No-sql, ASCII files),
- All implement VAMDC access/query protocols
- Return results in standardized XML format (XSAMS)
- Access directly node-by-node or via VAMDC portal, which relays the user request to each node
Workshop prior to RDA P4

Issues identified
- Each data node could modify/delete/add data without tracing
- No support for reproducibility of past data extraction

Proposed Data Citation WG Solution:
- Considering the distributed architecture of the federated VAMDC infrastructure, it seemed very complex to apply the “Query Store” strategy
  - Should we need a QS on each node?
  - Should we need an additional QS on the central portal?
  - Since the portal acts as a relay between the user and the existing nodes, how can we coordinate the generation of PID for queries in this distributed context?
Status / Progress since RDA P4

- Versioning adopted prior to P4
- Central service registering user interactions with data
- At each client SW notifies tracing service that a given user is using, at a given time, that specific software for submitting a given query
- Will assign single identifier for each unique query centrally
- Query store initially private (confidentiality issues)
Further Pilots

- **NERC**: UK Natural Environment Research Council
  - ARGO buoy network: SeaDataNet
  - Butterfly monitoring, Ocean buoy network, National hydrological archive, ...
- **ESIP**: BCO-DMO
- XML Data in Field Linguistics (CLARIN, XBase)
- Further Pilots on XML, LOD, ...

- **Workshops**:
  - NERC Workshop, London, July 1/2 2014
  - ESIP Mtg in Washington, Jan 8 2015: Earth Science Data
  - Data Citation Workshop, Riva di Garda, April 20/21
  - Bilateral meetings with data centers