Incorporating Data Citation in a Biomedical Repository: An Implementation Use Case

Snehil Gupta, BSc[1], Connie Zabarovskaya, MBA[1], Brian Romine, BA[1], Daniel A. Vianello, MSEE[1], Cynthia Hudson Vitale, MLIS[1], Leslie D. McIntosh, PhD, MPH[1]
[1]Washington University in St. Louis, USA

Abstract

Research data is a dynamic and evolving entity and the ability to cite such data depends on recreating the same datasets utilized in the original research. Despite the availability of several existing technologies, most data repositories lack the necessary setup to recreate a point-in-time snapshot of data, let alone long-term sustainability of dynamic data without restoring an entire database. Through this project, we adopted a subset of the Research Data Alliance data citation working group recommendations to establish a robust informatics system supporting dynamic data and its use for reproducible research within our evolving clinical data repository. We implemented key recommendations: data versioning, times-stamping, query storing, query time-stamping, query PID, and data citation in one data repository, implemented entirely at the database level, and were able to successfully reproduce a previous dataset as it existed at a specific point-in-time using only the PID as provided in a citation.

Introduction

Clinical health recommendations are highly dependent upon rigorous research and quality data[1-2]. Progressing the scientific method and improving health outcomes, practices and processes for true reproducibility extends beyond the methods section of a journal article and into the availability and sharing of specific time-stamped data , executed database queries, research protocols, software code, versioning information, datasets, metadata, and more[3]. Given the protected and sensitive nature of much biomedical research, sharing in this domain is complicated, yet still necessary to treat and solve real-world health issues.

The success of clinical care and trials is heavily dependent upon the validity and verifiability of previous research[4], but few tools and protocols for accessing or citing biomedical data (or metadata) have been integrated into existing biomedical informatics technology systems. Though there are many components and software systems that comprise the electronic biomedical informatics infrastructure, electronic patient health record (EHR) systems are one of the most heavily used by investigators. Research outcomes by investigators from Washington University in St. Louis (WU) using EHR data have included a variety of research claims for potential clinical and policy recommendations, with an average of 115 research studies per year. The EHR is also a complex data system given the fact that medical record data is large, evolving, and highly variable.

Problem Addressed

Given the data complexities and the potential impact research in this domain can have, the citation of queries made against EHR data is essential to verify claims, support data sharing, and improve the reproducibility of biomedical research. Though tools exist such as Informatics for Integrating Biology and the Bedside[5] (i2b2) with many features and functions to facilitate the reuse and discoverability of EHR datasets, the significant issue of citing the evolving data needs to be addressed.

The Research Data Alliance[6] (RDA), an international group of individuals working towards facilitating data-driven research, had a working group on Data Citation[7] (RDA-DC) to address the aforementioned challenge by identifying the key requirements for enabling research-reproducibility within an evolving database (Table 1). They offer recommendations enabling researchers and data centers to identify and cite data used in experiments and studies and support a dynamic, query-centric view of datasets. These recommendations equip data repositories with the ability to precisely identify and retrieve the exact data previously accessed – supporting reproducible processes for sharing and reuse of data.

Table 1: Research Data Alliance (RDA) working group on data citation recommendations for citing data in evolving datasets*

	Concept	Description
R1	**Data Versioning**	Apply versioning to ensure retrieval of earlier states of datasets.
R2	**Time-stamping**	Ensure that operations on data are timestamped.
R3	**Query Store**	Store the queries used to select data and associated metadata.
R4	Query Uniqueness	Rewrite the query to a normalized form so that identical queries can be detected.
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 Time-stamping**	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).
R8	**Query PID**	Assign a new persistent identifier (PID) or reuse prior PID to the query.
R9	**Store Query**	Store query and metadata in the query store.
R10	**Citation Text**	Provide a recommended citation text and the PID to the user.
R11	Landing Page	Make the PIDs resolve to a human readable landing page.

***Bold** items are ones included in this paper.

The overarching aim for this project was to determine what, if any, RDA-DC recommendations could be implemented within our biomedical data repository. Through this effort, we aimed to establish a robust informatics system supporting evolving data and its use for research and reproducibility of research processes within our clinical data repository.

Methods

As described in the following sections, to complete this pilot project we: assessed the CBMI data infrastructure; conducted a gap analysis of the infrastructure with the RDA-DC recommendations; defined our local requirements; and evaluated our approach for the implementation.

Infrastructure

The WU Center for Biomedical Informatics (CBMI) supports a number of research data resources as part of the core of a learning health system (LHS). The CMBI Research Data Core (RDC) is at the heart of the LHS and is a confederation of data resources, comprised of multiple database instances used to collect and store data from patient EHRs generated during routine patient care, and a variety of web applications that interact with one or more of these databases. The databases conform to varying data models and are built primarily using PostgreSQL database management system.

As depicted in Figure 1, we currently have an identified data repository. All downstream databases are limited datasets, and along with the identified data repository, are or will be updated through a nightly incremental extract-transform-load (ETL) process.

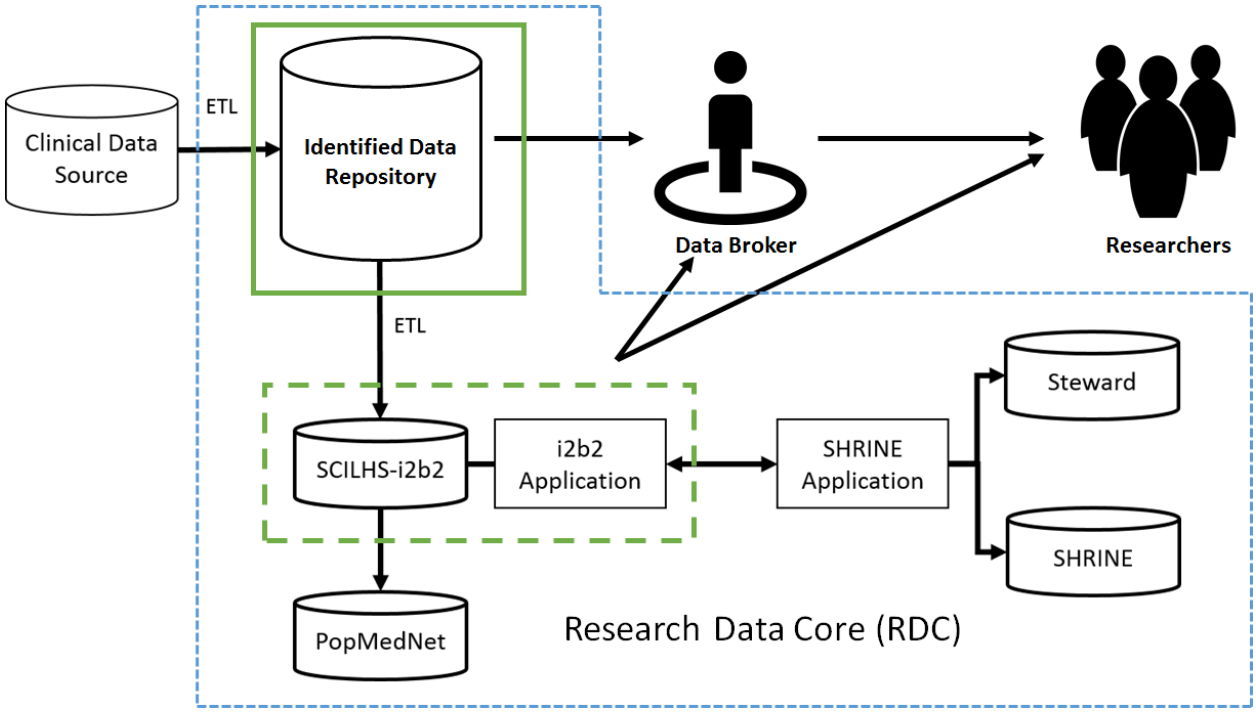

Figure 1: A simplified diagram of the data flow from the clinical data source to researchers. The highlighted portions represent the focus (green solid line) and future expansions (green dotted line) of this project.

Gap Analysis

We performed a gap analysis comparing multiple data locations (i.e., i2b2, i2b2i-SCILHS, PopMedNet) with the RDA recommendations to understand the deficiencies in our existing RDC infrastructure as compared with the RDA-DC recommendations. The results are summarized in Table 2.

Table 2: Gap Analysis Summary

Database	Data Versioning (R1)	Data Timestamp (R2)	Query Store (R3/R9)	Query Timestamp (R7)	Query PID (R8)	Citation Text (R10)
Identified Data Repository	Yes (default)	Yes (default)	No	No	No	No
i2b2 (Local)	No	No	Yes (i2b2 default)	Yes (i2b2 default)	Yes (i2b2 default)	No
i2b2 (SCILHS)	No	No	Yes (i2b2 default)	Yes (i2b2 default)	Yes (i2b2 default)	No
PopMedNet	No	No	No	No	No	No

Requirements

Using the information gathered through the gap analysis, we identified key requirements for research reproducibility implementation in our local environment (WU RDC). These include following RDA recommendations: Data Versioning (R1), Data Timestamping (R2), Query Store (R3/R9), Query Timestamping (R7), Query PID (R8) and Query Citation (R10). Additionally, we identified the following infrastructure and design requirements for long-term sustainability:

1. The implementation must be able to handle large volume of data efficiently and relatively fast, because versioning requires storing the history of changes to data, as well as scanning and updating large amounts of data.
2. Software tools used to enable data versioning or query auditing must have regular updates and maintenance support from community or reliable commercial source. Otherwise, in-house developers should understand the software tools used, so the local team can provide support and updates, if no other source is available.
3. It must be relatively easy to implement to allow for easy maintenance and upkeep.
4. It must be relatively easy to use for data brokers, as they will need to run queries and access query audit tables without requiring the involvement of a database administrator for this task.
5. Must be compatible with PostgreSQL, the primary database management system (DBMS) used for EHR data repositories in RDC.

Approach

We considered several methods for fulfilling this project's technical requirements while making our implementation fully compliant with the RDA recommendations. After assessing other tools, we decided to work entirely at the database level rather than the application level and thus employed the changes below rather than adopting an existing tool.

Data Versioning and Time-stamping (R1/R2)

We evaluated the existing versioning and time-stamping methods in the identified data repository. The original method stored all data versions in the same table; the older versions of data were labeled with inactive status as opposed to active status for current version of data. During incremental data loads, each update to a record would de-activate the existing record and insert the new record as active. This method is cumbersome and inefficient, because storing all versions of data in the same table increased the overhead during scanning of the table both for ETL processes and for data broker queries over the data. The original design also does not allow for efficient data reproducibility for data brokers.

As of PostgreSQL 9.5, there is not yet a default implementation of the temporal features of the SQL:2011 standard. To address this constraint, we considered several extensions currently available for PostgreSQL to provide more functionality than the existing design such as: pgMemento, TableLog, Table_Version, and temporal_tables. More information about each PostgreSQL extension is available in the appendix (https://osf.io/bzvst/). After reviewing the capacity of each of the extensions to satisfy our technical requirements, we ultimately settled on temporal_tables. The temporal_tables extension allows for storing historical versions of records in a separate history table. Full tuples are stored along with their respective validity intervals, which allows for simple querying but increases the size footprint of the database. Interval ranges and the moving of data from a given table to its paired history table is all performed by triggers, which simplifies the ETL processes.

We implemented the following design using temporal_tables:

1) Each live data table that needs versioning is given an additional column named ‘sys_period’ which holds a timestamp range value identifying the start timestamp, end timestamp, and boundary condition of when the record is or was valid. The name ‘sys_period’ is chosen to be consistent with the temporal_tables documentation.
2) For each live data table, a corresponding ‘history’ table is created to hold historical data. These history tables have the same structure as the live data tables but without the same primary keys or unique constraints, and they are used for storing only historical versions of the data, which are not valid at the moment. For consistency, all history tables are given the ‘hist_’ prefix then the same name as the table being versioned.
3) The trigger provided by the temporal_tables extension is added to each live data table to be called before insert, update, or delete transactions. Before an update or delete transaction, the trigger inserts a copy of the existing row from the live data table into the history table with the timestamp range modified to show the current date as the upper bound of the validity interval.
4) Views are created for each table, named with the suffix ‘_with_history’. These will allow a user to query all live and historical records in a given table or tables as they are now or as they were at any given point in time.

5) A new limited access schema, named 'databroker' is created with views pointing to each '_with_history' view. These views are defined to automatically limit the results to a certain timestamp. This avoids two possible user errors: either in neglecting to add the required WHERE clause, or by specifying an incorrect timestamp. For example: for a new query in these views results would be limited automatically to a timestamp value as of when the query is run (i.e., 'now()' or 'current_timestamp()'). Similarly, for a query that is re-run, results would be limited automatically to the timestamp that the initial query ran. As a result, the data broker can query these views without having to specify time ranges for when the data is or was valid.

Query Store, Time-stamping and PID (R3/R7/R8/R9)

Because query store, time-stamping or PID features did not exist in the identified data repository, we needed a method of logging queries that have been run by data brokers and make them uniquely identifiable as well. We implemented a function named 'databroker.query_tracking()', which serves three purposes: i) To ensure the currently executing query has been logged; ii) To return a date and time value to use in querying data; and iii) To return a pre-formatted citation text. In this process, a unique query_id is assigned to each new query, stored in the database, and is included in the citation text (see also the next section). This function is never called directly by an end user but instead is included in the WHERE clause of the definition of views setup for use by data brokers.

The function will attempt to add the user if it is a new user/session, add the query if it is a new query, and check to see if the user had requested for the data to be re-run as of a specific date. If a re-run of data was requested, then the returned timestamp will match the original transaction timestamp. If a re-run of the query was not explicitly requested then the query will be treated as a new query and returned value will be the current timestamp at the time of running. A more detailed explanation is included in the appendix[8].

Query Citations (R10)

Query citation was another feature that did not exist in the identified data repository, so it has been implemented as a functionality within data broker views. Whenever a new query is run using these views, the data broker will get two values returned: i) the dataset for the query; and ii) a pre-formatted citation line to provide to researchers. The citation is designed to both be easy to collect and to use in publications as well as allowing the format to be consistent across projects. The citation text includes the unique persistent identifier (PID) of the query and is the only item needed to rerun a query with guaranteed data replicability at a later time. This feature was added to facilitate citing data in scientific communication by providing the pre-formatted citation, requiring only 'copy and paste', thus potentially reducing the possibility that data will be used without citation. It also can provide sufficient information for a future researcher to request the same dataset.

Data Reproducibility Workflow

Beyond the 'databroker.query_tracking()' function listed above, two other helper functions were created for use by the data brokers. In order to reproduce data, returned as part of a query run in the past, the researcher will provide the query PID to the data broker. The data broker will be able to re-run the query by executing the 'databroker.rerun_historical_query()' function. This function will take two arguments: a desired name for an output cursor and the query PID provided by the researcher. This is described in more detail in the appendix. The final function, 'databroker.has_my_data_changed()', takes the same two arguments as the 'databroker.rerun_historical_query()' function, and will run the identified query twice, once as of the current date and time and again as of the original query date and time. It will then compare the results of both queries looking for any differences. The result of that comparison (boolean true or false) is provided to the data broker. If the result is true, a temporary table with the data differences is also created. In all cases, the data broker need only to use any existing database query and administration tool, such as pgAdmin3, that can run queries and display system messages.

Results

From the gap analysis (Table 1), we identified that a few RDA recommendations were already fulfilled by some of the data repositories in RDC while others were not. The features involved in data versioning and query auditing lacked in the majority of the systems and needed to be developed in a manner that renders easy querying by data brokers and researchers in any of the systems. With this knowledge, we drew the initial scope of work. We had

several repositories to work with and in which to implement the requirements. Noting that each database would need to be approached individually, we broke down the scope of implementation into phases. In this phase, we worked only with identified data repository of RDC which does not have any application interface and is expected to be directly accessed by local data brokers for providing data - either aggregate, identified, limited, or de-identified datasets as required and approved for various research projects.

During the design phase we discerned that existing data versioning and timestamping implementation in the identified data repository was cumbersome and inefficient. The original design also did not allow for efficient data reproducibility for data brokers. This led us to evaluate a number of other methods (pgMemento, TableLog, Table_Version, and temporal_tables) for fulfilling these requirements, ultimately deciding to use the temporal_tables extension in PostgreSQL. The ease of access on the part of our data brokers was key to this decision.

We implemented the RDA recommendations of Data Versioning (R1), Data Time-stamping (R2), Query Store (R3/R9), Query Time-stamping (R7), Query PID (R8) and Query Citation (R10) in the identified data repository following the approach outlined earlier in this paper.

We ran validation test cases and successfully demonstrated the reproduction of a dataset from a specific point-in-time in an evolving data environment. Validation was performed as follows:

1. A snapshot of identified data repository was instantiated where all reproducibility requirements of this phase were implemented
2. A sample research query SQL was executed in this repository snapshot
3. Some of the data presented in query results was manually altered to simulate a dynamic data environment
4. The above sample research query SQL was executed again, this time using the PID generated from initial run passed to the 'databroker.rerun_historical_query()' function described above
5. Two query results were compared

In both query execution instances, the exact dataset was produced, attesting to a successful pilot research data reproducibility workflow. Details of validation test cases can be found in the appendix.

An additional feature was also developed in the process to allow data brokers to check for any records that changed since the first time a query was run, by using the function 'databroker.has_my_data_changed()'. This function can identify differences in data by re-running the query for the timestamp it was originally run, and comparing the results to the results of running the same query using the current timestamp (Figure 2).

IFF $(A \cup B) - (A \cap B) \neq \{\varnothing\}$
THEN $C = (A \cup B) - (A \cap B)$

Figure 2. Formula for finding changes in data between historical and current version. Historical dataset is 'A', the current dataset is 'B', and 'C' is the differences between the two data sets showing only the changes in data.

As a result of implementing requirements originating from the RDA recommendations, the data brokers can now support EHR data citation and reproducibility for WU researchers utilizing the identified data repository of RDC as a data source.

Discussion

By providing better mechanisms for citing evolving data, the rigor of research, meta-analysis, and reproducibility of EHR data will be more feasible, verifiable, and efficient for investigators. Currently we have implemented and tested the enhancements in a partial snapshot of our biomedical research repository by querying data and rerunning the same queries using only its PID to deliver the same results as the original query after controlled updates were made.

Further testing will involve testing queries against changes derived from our incremental load ETLs which, as of the the date of this paper submission, are undergoing simultaneous testing and development efforts. This implementation was met with a favorable response by existing WU CBMI data brokers. A specific feature that make this implementation particularly easy to train are the DDL changes are transparent to the data brokers. Hence, the data brokers do not need to know the specific database features such as the sys_period column, nor the tables

with historical records, nor how to specify any temporal SQL syntax; all of that is hidden within the definition of the views queries by the data brokers. Instead, they need to learn only about two new functions that allow them to re-run queries and look for changes to re-run a query.

Limitations

This work is limited due to only implementing it in one location (WU CBMI) with one type of database, PostgreSQL. This work was primarily completed with only two groups - the WU CBMI team and the RDA-DC team. Thus, there may be solutions we did not consider due to the limited number of groups involved in the work. Also, with other databases being more frequently employed (i.e., Oracle, MySQL), this limits the number of other institutions that could replicate this with the code only. However, this is offset by the descriptions of the workflow and implementation, which could be used at other institutions using their own implementation of triggers to move data into historical tables and their own views to handle temporal SQL syntax as implemented in their relational database management system.

Another limitation exists if the queries by the data brokers contain functions that calculate values based on the current date. For example, if a query included a function to calculate the age of a patient as of today, that function will be re-evaluated whenever the query is re-run. Training mitigates this limitation to some extent, but the limitation still exists in the current implementation.

Future Directions

On a broad scale, we will continue to better understand how to make biomedical informatics research reproducible, including: i) testing and expanding needed components to ensure a study using EHR data can be reproduced[9] (e.g., database citation, query access); ii) testing the replicability of past EHR studies within WU and other institutions; and, iii) defining needed workflows for biomedical data brokers for improved research reproducibility.

For the next phase of this research project, we will extend the implementation of RDA-DC requirements to each component within the RDC (e.g., i2b2, PopMedNet) as well as to turn on access to the new schema to the data brokers. The i2b2 database has an application interface through which only aggregate data can be obtained. While PopMedNet database is much like the identified data repository within the WU RDC, it does not have an application interface and is expected to be directly used by data brokers for providing either aggregate data or limited or de-identified dataset for approved research projects.

We would also like to explore scalability and performance improvement opportunities. Some of the recommendations that were not included in this phase may be considered next, such as storing the original and normalized query (R4), as well as generating and storing result set hashes (R6). We may also develop an automated method to generate a query PID from these systems that are globally unique within our environments as well as create a landing page (R11) for these to resolve to for all systems. While technically not complex to implement, we are establishing the guidelines to balance the needs for maximizing the amount of information publicly presented with the privacy of necessary components. Creating and sharing the query PID's will allow researchers, data brokers, and other stakeholders a method to access research components while minimizing the involvement of the original researchers.

Conclusion

While some would consider this parasitic research[10], implementing data citation into the biomedical pipeline can be seen as a method to reduce the bottleneck of reproducibility by eliminating the time constraint of communication. Through this study, we implemented the key recommendations of *data versioning, times-tamping and, query storing, query time-stamping, query PID, and data citation in one WU data repository*, and we were able to successfully demonstrate the ability to reproduce a dataset from a specific point-in-time *using only the PID as provided in a citation.*

References

1. Landis, S. C. et al. A call for transparent reporting to optimize the predictive value of preclinical research. Nature 490, 187–191 (2012).
2. Freedman, L. P. & Inglese, J. The increasing urgency for standards in basic biologic research. Cancer Res. 74, 4024–4029 (2014).
3. Khoury, M. J. et al. Transforming epidemiology for 21st century medicine and public health. Cancer Epidemiology Biomarkers & Prevention 22, 508–516 (2013).
4. Laine, C., Goodman, S. N., Griswold, M. E. & Sox, H. C. Reproducible research: moving toward research the public can really trust. Ann Intern Med 146, 450–453 (2007).
5. i2b2. https://www.i2b2.org/
6. Research data alliance https://rd-alliance.org/
7. Rauber, Andreas, Ari Asmi, Dieter van Uytvanck, and Stefan Proell. 'Data citation of evolving data-recommendations of the working group on data citation.' (2015).
8. Appendix: https://osf.io/bzvst/
9. https://github.com/CBMIWU/Research_Reproducibility
10. Longo, Dan L., and Jeffrey M. Drazen. 'Data sharing.' New England Journal of Medicine 374, no. 3 (2016): 276-277.