DATASET VERSIONING THROUGH CHANGELOG ANNOTATION

By

Benno Lee

Prepared for:

Peter Fox, Thesis Advisor

Jim Hendler, Advisor

Deborah MacGuiness, Member

Beth Plale, Member

Rensselaer Polytechnic Institute
Troy, New York

November 2016
(For Graduation December 2017)

© Copyright 2016
by
Benno Lee
All Rights Reserved

CONTENTS

LIST OF TABLES v

LIST OF FIGURES vi

ACKNOWLEDGMENT vii

ABSTRACT viii

1. INTRODUCTION 1

1.1 Data Set Proliferation 2

1.1.1 Explain why we choose the data types that we did 2

1.1.2 Digital Libraries 3

1.2 Unifying multiple systems 4

1.2.1 Grid 4

1.2.2 Heterogeneous systems 5

1.2.3 Transition to data driven systems 5

1.2.4 How much does data grow? 5

1.3 Data Quality/Provenance 5

1.3.1 Changelogs 7

1.4 Provenance Distance 7

1.5 Data Versioning Operations 8

1.5.1 Types of Change 10

1.6 Conclusion - Thesis Statement 11

1.7 RDFa 11

2. PREVIOUS WORK 12

2.1 Spreadsheets 12

2.2 Database Systems 14

2.3 Ontologies 16

3. CONCEPTUAL MODEL 17

3.1 ADDITION 17

3.2 INVALIDATION 18

3.3 MODIFICATION 18

3.4 MULTIPLE LINKED VERSIONS 19

4. VERSIONING SPREADSHEETS 20
4.1 Provenance Analysis 20
4.2 Versioning Comparison 20
5. DATABASE VERSIONING 23
6. ONTOLOGY VERSIONING 24
7. CONCLUSION 25
REFERENCES 26

LIST OF TABLES

LIST OF FIGURES

3.1 Model of the relationships between Versions 1 and 2 when adding an Attribute 2 to Version 2 as a result of Change A 18

3.2 Model of the relationships between Versions 1 and 2 when invalidating Attribute 1 from Version 1 as a result of Change I 19

3.3 Model of the relationships between Versions 1 and 2 when modifying Attribute 1 from Version 1 as a result of Change M, resulting in Attribute 2 from Version 2 19

4.1 Provenance graph for the entry CAM001 entry of the Noble Gas Database. Other than the labels, the structure of each of the data objects is very much the same. 21

ACKNOWLEDGMENT

ABSTRACT

Science is constantly changing and the data that drives it changes along with it. Understanding the behavior of data as it changes to correct errors and to add value, improves data quality and eases data integration. Current methods to track the evolution of data primarily focus on the collection of data provenance, but lacks the fidelity to evaluate the extent of a change. The conceptual model proposed in this document addresses this challenge by leveraging semantic technologies to automatically embed versioning information into changelog artifacts generated when changes are made to a dataset. A method to calculate change distance then follows using the resulting semantic graph by attaching weights to different change types. Initial applications begin with spreadsheets then databases with the ultimate goal in applying the model to online ontologies.

CHAPTER 1
INTRODUCTION

John C. Maxwell once said, "Change is inevitable. Growth is optional." While this inspirational quote refers to the human character, it also holds true for scientific datasets. With changing technology, data collected by researchers grew at an astounding rate. NASA's Atmospheric Science Data Center reported a growth from hosting around five million files to twenty million files between 2001 and 2004 [1]. The ATLAS project at CERN reports that it generates on the order of four thousand new datasets per day from experimental tests alone[2]. This explosion of data marks a change in the perspective of research from a software driven approach, where programs are used to confirm results using data, to a data driven approach, where vast quantities of data is integrated to demonstrate trends and produce results. Instead of simply using computers to assist in performing a complex computation, researchers now use broad data collections to inform advances in science. This means that data quality needs to be ensured in order to extract valid conclusions from large data sets as humans can no longer manually curate the data mining process. However, the need for quality assurance means that a portion of the files generated do not contribute to data set growth since data has not been added, simply modified. Many NASA datasets have required re-processing of their data, either to improve data quality or to correct for errors [3]. Data traceability now becomes particularly important to identify sources that contribute to improved data quality. It creates a need to understand not only that a data set has changed, but to also understand how much a data set has changed. Data versioning is the method of tracking the changes performed on a data set and determining the extent to which it has changed. In this document, data versioning is approached using technology provided by semantic technologies and applying them to artifacts currently generated by scientific data sets.

1.1 Data Set Proliferation

1.1.1 Explain why we choose the data types that we did

Data has existed long before computers, populating the storage space of filing cabinets and data closets the first transistor radio. It then comes as no surprise that libraries and library sciences provide the early methods of data management. The challenges and goals that face physical libraries remain valid even as data collection migrates to electronic alternatives. Digital storage and the Internet has opened new opportunities and methods to administer book data by separating logical representations and physical representations [1]. However, the migration has not been without its problems. Early citations used stagnant Uniform Resource Locators (URL) to refer to online documents, but this would lead to a condition known as link rot where moving the document would invalidate the URL [4]. This eventually led to the development of Persistent URLs (PURL) which also succumbed to link rot, and this eventually led to the distributed Digital Object Identifier (DOI) system used to track documents today. The DOI network provides a robust system to track documents, but when tracking data, it faces difficulty following the rate of change with some more volatile data sets. Distribution organizations assign a DOI whenever a new edition of a document becomes available, and due to the publication process, documents change very rarely so a new DOIs are rarely necessary. However, data sets are products and thus succumb to the iterative process of error correction and growth. Data collection often continues on after initial publication. DOI distributors treat new files like new sections to a paper and changes to files as edits so a new identifier must be issued to the data set.

For similar reasons, treating data as documents produces problems when applying technologies from software management [5][6]. Structure provides the most significant distinguisher between data and software since a data set with a removed file remains usable but a software project would break. The function of code comes from its content, but the function of data comes from its ability to store and organize data. This should not be confused with data formats which impose structure onto data in much the same way programming languages provides a medium to express actions. However, exporting data in different formats is currently easier than

exporting code into different languages. Data sets do not represent a single object, unlike a software project[7]. Data sets are compact representations of all possible subsets of the data set, which are also datasets. For this reason, the structures of data sets and software becomes incompatible and software versioning technologies are insufficient to capture this nuance.

The techniques employed by these technologies, however, can remain applicable to data sets and are often necessary when communicating change data to users. Version producers often refer to versions using numbers in the dot-decimal style. While the values often signify the Major-minor numbers associated with the version, the names remain meaningless and can arbitrary assignment such as Ubuntu released numbered by Year-month values [8]. The arbitrary nature of the numbers often entails referring to versions by English nicknames instead. Such a regular method of naming release versions also means that determining the magnitude of change between two releases becomes impossible. Numbering the version this way, however, does allow computers and readers to quickly parse the version name and discern that a change has occurred, but little value exists beyond that [9]. The technique of distributed and federated employed by GIT does provide significant value to modern methods of versioning data [10]. As data workflows and data set dependencies grow, their volatility also expands, meaning that they become more likely to generate new versions. The federated approach available in the GIT environment allows developers to establish change dams that collect modifications and releasing the data at regular intervals,reducing the changes to a manageable flow.

1.1.2 Digital Libraries

Data as documents vs Data of documents.

Data spreadsheets resemble books because they have a clear hierarchical structure. Each spreadsheet is a full unit by itself. The Excel files themselves are known as workbooks. XML documents also have a clear hierarchy. Due to the rapid growth of new technologies, the need for proper data versioning extends beyond major agencies and even applies to smaller projects [11] [12] [13]. Indeed, version control systems can provide contexts for modifications when combined with error

detection systems, providing a more complete picture of the system's behavior [14].

Early work in the field of digital data versioning begins in the library sciences. As the computing field grows, many data publications are now being provided in digital formats as well as print. The challenges of tracking a digital library are often the same as those faced in data repository management [15]. Not only do librarians often have to simultaneously work with several versions of the same material, but they also must pay attention to the structure of their collection as well as its content. They have also been used to track and cite datasets, but unlike documents, datasets have a tendancy to change after publication. As a result, new versions of data would get a new DOI to prevent confusion, and this unnecessarily consumes the available handles and the time of institutions assigning handles [4]. Other identifier systems have also had problems with maintaining healthy links to data as it changes and moves, but systems using newer technology have been able to achieve success in localized networks. Built on XML and web service technology, the Mellon Fedora project linked together various disparate digital library collections at the University of Virginia [16]. Digital publications not only include books and journals, but also grew to include web pages and wikis, requiring new scalable methods to track changes on these publications [17].

1.2 Unifying multiple systems

1.2.1 Grid

In the early 2000s, versionings systems were being prepared for the future of grid technologies. The grid provided a unique environment that had to handle a variety of inputs, and therefore, different input data could run on distinct sets of grid services. This meant that different versions of the same data could be generated by differing services on the same grid [18]. A versioning environment would also have to be general enough to accommodate disparate input types. CERN grid for the Compact Muon Solenoid experiment separates the physical and logical storage of files, allowing multiple users to refer to the same file without needing to copy the file across the grid [19]. Versioning policy could not exist in this case if a clear replication policy did not exist since changes to the underlying file can have

repercussions with its replications across the grid. The versioning method relies on predictable replication in order to reliably communicate change to grid users.

1.2.2 Heterogeneous systems

While innovative, the grid was neglected in favor of networked heterogeneous systems. The proliferation of mobile computational devices such as laptops and smart phones created a supply of small data repositories that were too compact to warrant the use of grid technologies. Baker and Yarmey describe a method of networking together small, volatile datasets with larger, versioned data distribution centers for environmental data [20]. Heterogeneous systems encounter challenges with data integration resulting from non-uniform data interfaces. Autonomous solutions to understanding change grows in importance as evidence grows that speedy and relevant changes play a significant role in successful system function [21]. The key often seems to be using XML to provide an common language propagating change across heterogenous systems. The need to propagate changes becomes more apparent as autonomous units grow and share data [22]. Semantic technology's growth now provides a new method of propagating data change that can not only be universally consumed in formats like XML but also encodes meaning to each of these changes.

1.2.3 Transition to data driven systems

1.2.4 How much does data grow?

1.3 Data Quality/Provenance

Provenance describes the sequence of actions and the individuals involved in generating a particular artifact. In the case of art, provenance would include a history of previous owners to ensure the validity of the work, in short, to ensure its quality.

In the early days of computing, hardware constraints restricted the size of data to very manageable quantities. However, recent advances in storage methods has produced devices, even on a personal level, with the capability to store data that far exceeds the size of the programs that utilize them. It is then no surprise that the

development of versioning management tools for software However, the proliferation of powerful and mobile computing devices has allowed researchers to generate data faster than it can be tracked [23]. Data processing is automated so data quality needs to be ensured since a human will not be able to determine data quality for each data entry individually.

The information that details the activities and agents involved in generating a data entity is known as provenance. Studying the structure of a data's provenance has provided insight into behavior and quality by revealing the qualities of the contributing data sources and code sources [24] [25]. Provenance management differs from organization to organization, but most generally agree the core factors of scripts, calibrations, and inputs form the core components of lineage tracking [26] [2]. Software revision management tools such as Git and SVN also keep track of provenance information when logging new commits to a project, using branches and merges [7]. Provenance is a powerful mechanism to follow data quality changes or error introduction . The information also gives data warehouses a mechanism to evolve and adjust their ability to provide better data [27]. It is, however, important to distinguish between the contributions of provenance to change tracking and the contributions of versioning. Provenance allows for the identification of change localities; it signals where changes in data have occurred. However, provenance does not elaborate on the extent of the alterations made, not should it. The priority of provenance tracking is understanding where and how a data object was generated, not it's relationship to other data products [23]. In the field of semantic technologies, the W3C recommendation, PROV provides a data model to encode provenance information into searchable graphs. It has played a significant contribution in maintaining the quality and reproducibility of datasets and reporting in the National Climate Assessment [28, 29, 30]. PROV expresses relationships between versions with the wasRevisionOf or alternateOf property. The properties do not allow for more elaboration as to what changes were made to transform from one version into its alternate. Versioning provides the context for a change by providing the comparison between data objects.

1.3.1 Changelogs

Changelogs, sometimes called patch notes, are artifacts resulting from the versioning process often found in major software projects. They detail the changes made within the system and some will have explanations on the motivations behind changes [31]. These logs provide a great source of value to developers as they can be used to give insight to the health of a software project [32] [33]. Changelogs also allow developers to link bugs and errors with their corrections in later versions [34]. Despite these contributions, changelogs are rarely found along with data. Possible reasons for this omission may include the size of data as it may become difficult to scalably communicate data change (Data exceeds ability to track reference) or that the ease of data acquisition and storage inclines towards simply using the latest version. As a result, new dataset versions are often accompanied by usage documents, but documentation on version transitions are omitted. This creates a impediment to making data forward and backwards compatible, costing data consumers both time and money.

Changelogs are also difficult to consume as they are often built to be read only by humans. The transition between different versions of large datasets is then left largely up to the human user's ability to understand and process the modifications mentioned within the change log. In the absence of any sort of change notes, it is up to the consumer to be able to locate the points of differences between the versions they are working among. Therefore, providing an machine consumable changelog would accelerate and assist in navigating through dataset changes and error corrections.

1.4 Provenance Distance

Data workflows have become complex and sized in such a way that even subtle changes may have meaningful effects [35]. One approach is following the provenance of the new data and identifying potential differences between how the data has become generated. Since provenance graphs have both directed edges and labeled nodes, there are a few methods of measuring similarity available beyond basic isomorphism [36] [37] [38] . This similarity measure is known as the provenance distance

between the two data objects. A concern that arises with this approach is that, as previously stated, provenance only provides a context for versioning. Consider an example where there are versions 1 and 2 of a data object, both compiled from a reference source. Since there are two versions, it is understood that the versions are different so the comparison extends to their reference sources. If both versions were compiled from the same source, it can only be concluded that the compilation methods were different. There also exist methods that compare workflows based on quality criteria that leverages provenance to bound quality of service [39]. In the case of the NASA datasets, those methods should not have changed and no further insight has been found. If the reference sources have been changed, then it is expected that there would be a new version or data object and no new information has been found. This reasoning motivates the development of a versioning method that utilizes changelogs and RDFa to determine versioning distance.

1.5 Data Versioning Operations

Architecture has a principle that says form follows function, but, for data, form equals function. As a result, data has as many different forms as it has functions. Biological experiments often use data within cyclical data workflows where outputs are immediately fed back into new experiments [12]. Even though the goal of the experiment is the final data set, all the intermediary data sets provide significant value in reaching the goal. Libraries store data about their collections in large databases where both old and new versions of literature need to be maintained [15]. Some data exist in such a highly constrained environment that it must be managed at near the hardware level [40]. The challenge no longer becomes generating data, but instead, fitting the data into a format that users find useful and can consume.

The challenge of data versioning systems is to provide a unifying environment that can handle the plethora of forms and functions of its data. At its core, versioning systems only need to concern themselves with three operations: addition, deletion, and modification. Most literature surveys do not realize the significance of this commonality as this means that versioning methods can be described by delineating how each operation is approached by a system [12] [11]. Data addition

generally constitutes the least complicated versioning operation because it interacts the least with pre-existing data. However, new data does share context with pre-existing data and provides a method of measuring data set growth. Since data sets no longer have to be used in its entirety and can be freely subsetted, a data set's complexity increases significantly with its growth. Every new file added to a data set doubles the number of available subsets.

Data deletion, however, has a more philosophical difference between systems. From the perspective of a versioning specialist, data should never be deleted since knowing why data was excluded is as important as knowing why data was included. The software versioning manager GIT uses a method of compressing older data to conserve space without deleting the data [7]. Pragmatically, this is not always possible due, generally, to the physical constraints of storage space. In high energy physics, observational data often cannot be re-collected due to cost, and as a result, poor quality data cannot be re-processed or replaced [25]. The decision in this document is to use the term invalidation when referring to data removal operations as it implies that whether permanently deleted or not, there exists a more valid alternative.

Data modification encompasses the most involved data versioning operation. As a result, it often comprises a majority of the description of a data versioning service. In truth, data modification can be summarized as the invalidation of an instance of a data object - which can be a file, a record, or anything in a data set - followed by the addition of a new instance of that data object. However, this kind of operation is used so often to fix errors and update data sets that it is considered a unique operation. Modification owes its complexity to interacting with both pre-existing data from the invalidation stage and new data from the addition stage. However, this compound relationship fully contextualizes the relationship the operation has in relating the old data and the new data. In some cases, this only provides forward or backwards references between data versions, but having both gives users context for data's current state and update to new data [41].

Due to the ubiquity of the data addition, invalidation, and modification operations in versioning systems, the conceptual versioning model presented in Chapter

3 centers around capturing the relationships established by each of the operations. While other functions exist commonly in versioning systems such as object locking to prevent simultaneous conflicting changes, viewing to see the version an object belongs to, and branching to allow distributed modifications, these functions comprise the space of utility operations that support the three core processes.

1.5.1 Types of Change

Focus on change types because focusing on the values has comparably less value. When working with large data sets, individual changes are less important than changes made to the set abstractly.

Change is the fundamental characteristic in the study of versioning. Continuing the comparison with software, the measure of change in applications is primarily functional. Data, on the other hand, has not only function but also structure. While swapping the ordering of columns in a spreadsheet may not effect the usability of the data, it could cause issues with codes that use the data. This highlights the idea that many different forms of change exist with differing degrees of intensity. In this document, change is viewed as having three general categories: scientific, technical, and lexical. The most impactful change, scientific, denotes changes that have modified the fundamental science used to generate the dataset. Changes in algorithms or sampling methods generally fall under scientific modifications as they effect the base assumptions and possible error within the dataset. Technical impacts do not change the underlying science of the data, but impose a large enough change as to warrant notice. Structure alteration and unit conversions count as technical changes since the dataset now needs to be consumed differently but remains valid for use. Lexical changes belie the transformations that can best be described as corrections. Filling in previously missing values or fix erroneous values may be lexical changes.

The exact category that a particular change falls into can be controversial. The decision to change concentration units from parts per million to milligrams per milliliter poses a Technical change for a data producer. However, for a data consumer, the change may be viewed as a Scientific change as it invalidates the methods they had previously used. This conflict in view illustrates the data consumer-

producer dynamic. In general, data producers are in control of the methods of versioning, but data consumers determine the classification of a data change. Reference 19 demonstrates the viewpoint as producers tend to use versioning systems to ensure data quality of service through audits and recovery tools. Meanwhile, a consumer will analyze the historical changes and determine the impact this may have to their data use. As a result, this means that data versioning systems must communicate a dynamic view of the changes in a system contextualized by the user of that data.

1.6 Conclusion - Thesis Statement

1.7 RDFa

The modifications to implement a machine readable changelog should still remain in a form viewable by a human user. The subjectivity of versioning information means that domain scientists should still be able to interpret changes as they relate to their application. Resource Description Framework in Attributes (RDFa) allows web developers encode information XHTML and HTML documents using standardized ontologies. The developer would use the attributes in RDFa to augment the existing HTML tags to provide extra meaning that a web crawler can extract. The displayed text would remain unchanged signifying that if a changelog could be produced in HTML, then data producers can embed machine readable versioning information into a changelog using RDFa while still leaving it human readable as well.

CHAPTER 2
PREVIOUS WORK

2.1 Spreadsheets

In this project, spreadsheets were chosen for study as they resemble text-like data objects while still maintaining a level of complexities. Though not as well encapsulated as other data format types such as the Hierarchical Data Format (HDF) or Network Common Data Form (NetCDF), spreadsheets provide many helpful tools that scientist favor for quick data storage and distribution over comma separated values (CSV). There also exists other document-like formats that are not discussed in this paper such as eXtensible Markup Language (XML). The initial work was done with the "Noble gas isotope abundances in terrestrial fluids" workbook (Noble Gas) [42]. The "Paragenetic Mode for Copper Minerals" workbook (Copper Data) was used to give better insight into data changes due to collaboration with the data set's author [43].

The Noble Gas data set was initially published on June 11, 2013 and then released a second version on March 8, 2015. Many significant changes were made to the data set between the two versions, which makes this data set particularly challenging to version. The physical structure of the data set changed from eight separate Excel spreadsheets to a single spreadsheet. The second version also trimmed 195 columns to 54 columns in the second release. In addition, many new locations were surveyed and added to the second release. Documentation accompanied the data set explaining different components of the spreadsheet and its usage, but it included no versioning information. This lack of versioning or transitioning information indicates a focus on data usage rather than data maturation, which is not a particularly bad approach. It makes logical sense to simply download the latest data set when it becomes available and not worry about the format of the invalidated data set. This approach convenient for new users of the data as the cost to consumer the new version of the data is the same cost they would have spent to acquire the data in the first place. However, users of the old data are disproportionately effected by

the change in versions since old code and workflows may need to be updated to accommodate the changes in addition to the cost of consuming the architecture of the old data set. In this case, users would need to read the documentation to understand whether 182 from the June data set is still available in the March data and, if it is, in which column it resides in the March spreadsheet. This brings to light the additional concern for the Noble Gas data that the documentation is not easily machine consumable, meaning that all mapping activities will need to be performed manually. Not only is this approach time consuming, but it also does not scale well into larger data sets.

The Copper data set was acquired during the process of a workshop to generate new methods of visualizing mineralogy data, initially on June 8, 2016. The process entailed trying various orders and organization for the data and results in various new versions of the data that depend on varying filtering requirements, acquired on August 21, 2016. Unlike the Noble Gas data set, the Copper Data had no accompanying documentation, since the primary consumers of the data at the workshop were also mineralogy experts. However, this data set had more stable characteristics including physical and logical structure. Only two columns were removed from the transition to the second version, but sixteen new columns were added to the data collection. It also demonstrates a change in orientation with respect to data usage since the previous data set was designed to be distributed for general usage and discovery. In this case, the structure and organization of the data within the set was driven for a specific purpose in the development of more expressive visualizations. As a result, versioning information is driven by developmental needs instead of the other way around with versioning information bridging the gap between software migrations.

The data files from both data sets can easily be tracked using standard version management services such as GIT or SVN. Likewise, there exist comparison tools like Spreadsheet Compare from Microsoft Corporation that can generate diff-like outputs for each of the data sets. In conjunction with commit logs, the comparison outputs provide a basic versioning methodology that describes the data set's evolution. However, these applications rely on human attention and interaction

to operate, and with larger data sets, proper documentation becomes difficult to maintain. With the Copper Data, the demand for new versions of the spreadsheets exceeded the time necessary to document version history as a result of rapid product evolution during the workshop. In consequence, the process to manually commit and annotate changed data impairs the natural progression of scientific development.

2.2 Database Systems

Databases remain the most relied upon technology for storing and searching large quantities of data rapidly. While the dynamic combination of tables means that data bases remain flexible enough to represent complex objects, it also means that they represent a much more complicated case for attribution. Since tables may be combined in different ways to answer complex queries, indexes do not remain constant across requests to the database. The approaches to database versioning typically focus on ensuring the reproducibility of queries to the database. This can often be difficult as with spreadsheets since changes to the content or structure can result in different solutions from the database for the same query even using time stamps. For example, consider the query to select all columns of row A from a database on March 1st, then the database undergoes a schema change to add a new column to the table on April 1st. A subsequent request for all columns of row A would include the new column which does not represent the response on March 1st. In addition, even if the data is timestamped, the time signature is associated with the row and not the schema, meaning that the query may still return row A with the new column with a NULL value, depending on the distribution. The query, not the data, would need to be modified to exclude the new column.

This presents as a challenge because unlike data files and spreadsheets, databases are generally not instanced. Databases often store massive quantities of data and replication of that data to archive snapshots or distribution frequently proves too costly to be feasible. Instead, interaction with the database occurs from a centralized source through transactions. Various methods have been studied to manage changes within these systems focusing primarily on schema versioning, emphasizing data's structural component [44]. This provides a method to enact a transactional rollback

on the database to execute queries in an environment reminiscent of the original execution. The framework of the resulting database environment can become quite complicated as a result of the complexity of the tables representing intricate data objects [45]. This results from the need to manage the time instances of realization, storage, and validity. The datum becomes realized at collection, then stored upon entry into the database, and finally valid until the present or new data replaces it. More recently, new methods have been developed to adjust to the enormous quantities of data populating modern databases, focusing on query citation rather than data citation [46] [47]. Citation by query avoids the complexities involved with referencing data that can grow and move. However, this method relies on the existence of a versioning system for data. This method also recognizes that modifying queries to operate on the current state of the database may often be easier than rolling back transactions or schema to reproduce the results of a query [48]. As a result, to versioning a database system may be more feasible as data size increases by applying methods to the query results and not to the data.

The RRUFF Database is "an integrated database of Raman spectra, X-ray diffraction and chemistry data for minerals" [49]. It features a web accessible change log using the transactional log generated by the database software[1]. As the records in specific tables change, the log reports these changes, supplying persistent access to the modifications made to the RRUFF data. The approach to this alteration information highlights the always on-line approach to moder databases where changes to the data do not constitute a new database. The log demonstrates strong versioning characteristics with not only a breakdown of the change components, but also a commentary on the motivation for the difference. In addition, its HTML structure allows automated web crawlers to systematically consume the version information. With the integration of web ontologies, the change log would also be intelligible to automated agents.

[1] `http://rruff.info/index.php/r=rruff_log_display`

2.3 Ontologies

The web also provides a new venue for computational versioning in addition to offering a new means to communicate change. Machine readable vocabularies are maintained in large data collections known as ontologies. As the language improves and the vocabulary refines concepts, ontology versioning becomes a vital component in its growth. Previous work has emphasized the importance of making data not only backwards compatible to provide comparisons, but also forward compatible such that applications can seamlessly migrate from older concepts to newer ones [41].

PROV is a W3C recommendation that deliniates a method to express data provenance with semantic technologies [50]. Using the model of relating activities, agents, and entities, data managers can express the origins of their datasets. However, when an entity is revised, the PROV data model can only express the relationship as a revision or that the new dataset was derived from the original. This leaves

CHAPTER 3
CONCEPTUAL MODEL

The conceptual model used within this thesis is built around the expression of three core versioning operations: addition, invalidation, and modification. These three activities can be represented by interacting with three types of concepts: versions, attributes, and changes. Versions represent the data entities being compared. These could be two different editions of a book or versions of software. It is important to understand that a version is an abstraction as it can be represented by multiple physical files. In the sections that follow, operations will only consider the interaction between two versions and will be explained later in the chapter. Versions then contain attributes representing a quantity being modified. Specifically for tabular data, attributes would correspond to an identifier that refers to particular rows or columns within the data. Attributes of the two versions are then connected by a change. This link functions as a very general concept which can be subclassed into more informative types such as unit changes, improving the expressiveness of the model beyond PROV's revisionOf concept.

3.1 ADDITION

When a change adds a new attribute to a version, it only needs to refer to version two and its corresponding attribute. The reasoning should be fairly obvious as the attribute never existed in version one, and therefore, there is nothing to refer to and no need to form a relationship between the change and version one. However, by linking the addition change to version one, we address a difficulty with comparing provenance graphs. When two data objects have identical structures, it is difficult to determine what time the objects were added to the dataset and which version they belong to. As a result, determining the compatability of the two objects becomes difficult. The change contributions to the dataset evolution appears naturally using this construction. The resulting model can be seen in Figure 3.1. Some relationships are specifically left out, such as that between Change A and Version 2, to not confuse

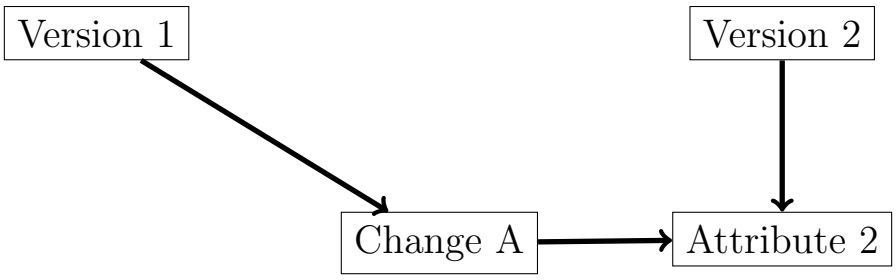

Figure 3.1: Model of the relationships between Versions 1 and 2 when adding an Attribute 2 to Version 2 as a result of Change A

identification of other types of changes. The relationship between Change A and Version 2 can still be implied from Attribute 2.

3.2 INVALIDATION

The Invalidation operation corresponds to the delete concept found in other applications. The choice of invalidation over delete results from the policy that, in versioning, data should never be deleted. In practicality, this may not be particularly feasible due to space limitations and relative validity. In either case, the change invalidates an attribute in version one, resulting in version two. Unlike the Addition operation, Invalidation forms a clear relationship between both versions, which can be seen in Figure 3.2. Notice again that since Attribute 1 no longer exists in Version 2, there is no corresponding Attribute 2 to refer to.

From Figure 3.1, we can see the confusion that could result from requiring explicit relationships between versions and changes in both the Addition and Invalidation operations. Linking Change A to Version 2 would create a duplicate connection and provides a mechanism to identify when items specifically enter or leave a version.

3.3 MODIFICATION

The final operation is Modification, and it maps a change from one attribute from version one to its corresponding attribute in version two. The particular type of change in this case is purposely left out in order to allow data producers to subclass and customize the resulting graph to properly reflect the versioning that they desire.

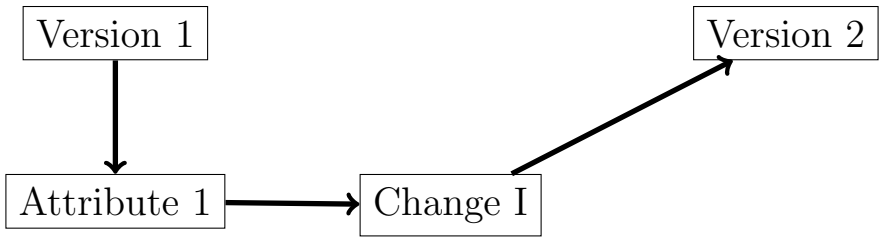

Figure 3.2: Model of the relationships between Versions 1 and 2 when invalidating Attribute 1 from Version 1 as a result of Change I

3.4 MULTIPLE LINKED VERSIONS

Using the construction outlined in the previous three sections, many changes can be compiled together into a graph in a changelog. After all additions, invalidations, and modifications have been compiled into a single graph, a complete mapping from version one to version two may be developed. The orientation of the relationships in the graph allows a flow to be created from attributes in version one to corresponding attributes in version two. Taking version two and performing the same graph construction to a version three results in not only a flow from version two to version three, but also from version one to version three. As a result, the flow can be used to construct a mapping from version one to version three or any future version.

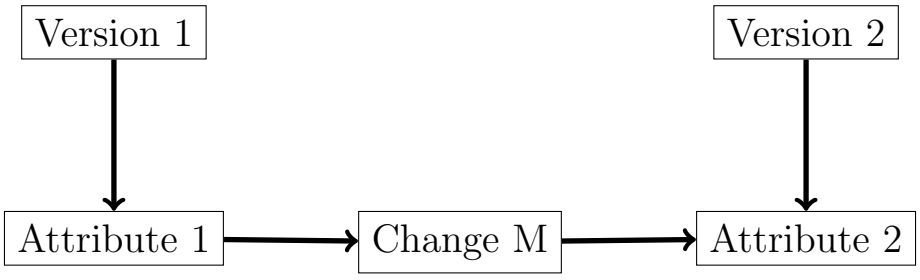

Figure 3.3: Model of the relationships between Versions 1 and 2 when modifying Attribute 1 from Version 1 as a result of Change M, resulting in Attribute 2 from Version 2

CHAPTER 4
VERSIONING SPREADSHEETS

Two datasets were initially studied. The "Noble gas isotopes in hydrocarbon gases, oils and related ground waters" database, These are the most challenging

The Paragenetic Mode for Copper Minerals database produced two versions, one at a workshop on June 8, 2016 and another at a following workshop on August 21, 2016. These both take the form of Excel spreadsheets, which has the benefit of having strict row and column numbering. This allows unique identifiers to be used when referring to individual pieces of data and providing a level of abstraction. The structural changes made to the Copper Dataset resemble those found in the Noble Gas Dataset.

4.1 Provenance Analysis

The first approach to determining the provenance distance for the datasets began with the Noble Gas Dataset. The dataset provides a set of references from which the values were extracted and compiled into the dataset. As a result, a simple provenance mapping was constructed, using PROV, from each reference to its corresponding row in the spreadsheet. After this was done for each version of the dataset, we can generate graphs to compare objects from the two versions like the one found in Figure 4.1. We can tell from the labels that different Activities were used to compile the data entry, but the structure of the graph does not provide any information as to how extensive this change was for this version. Here we can see how the wasRevisionOf relationship would break down in determining the magnitude of change between the two versions. The relationship must, therefore, be expanded in order to provide the desired data necessary to make an evaluation.

4.2 Versioning Comparison

The initial challenge for both of these datasets is producing an appropriate mapping between their previous and latest versions. Through the second compila-

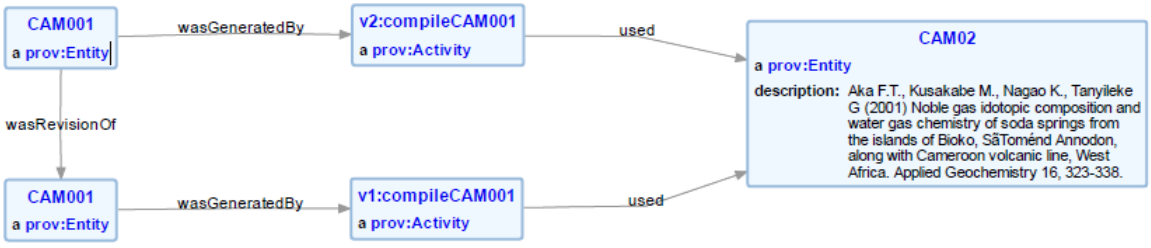

Figure 4.1: Provenance graph for the entry CAM001 entry of the Noble Gas Database. Other than the labels, the structure of each of the data objects is very much the same.

tion process, many columns were removed from the dataset or moved around, and as a result, the identifiers associated with the Attribute in version two did not necessarily correspond to their identifier in version one. The column headings in the Noble Gas Dataset is stored in the spreadsheet as data and makes it difficult for a general system to automate mapping columns between the two versions since any number of rows may contain metadata information. Additionally, while the second release of the Noble Gas Dataset did have a document detailing the organization of the columns contained within the dataset, it did not have any information to map entries from the old dataset to the new dataset. This limits the ability to map and update changes to human actors. Immediately apparent is that any columns that have a mapping from version one to version two means that these columns (Attributes from the model) will only undergo Modification operations since there exists an associated Attribute in both the previous and current version. The conclusion then follows that all remaining Attributes (including both columns and rows) belonging to version one were Invalidated and those belonging to version two were Added. Using this conclusion, we can pre-calculate the Added and Invalidated Attributes and separate any report into changes grouped by operator.

Once a mapping exists between the two versions, a comparison was performed to determine whether an Attribute of the data object changed. In this case, a simple equality operator was used to determine if anything was modified. In practice, more advanced or complicated methods can be used to determine equality. Since, all Additions and Invalidations have already been predetermined, we can output each Modification as we see them. As is common in versioning, the changes

were outputted to a changelog document formatted using HTML. The idea here is that by providing the changelog in HTML, the changes can be made available online and therefore accessible by data consumers. Another benefit of providing a changelog in HTML is that the document can be additionally enriched by RDFa. The changelog document becomes no longer restricted to human consumption, but allows autonomous agents to more intelligently interact with dynamic data systems. The conceptual model detailed in Chapter 3 becomes encoded into the changelog and the graph resulting from the log can be extracted automatically.

The resulting graph provides a structure upon which a flow may be calculated. This becomes an alternative method to determine the provenance distance between two datasets with a much higher fidelity. As mentioned previously, the Change concept is meant to be sub-classed to provide more freedom to represent the particular change bridging the two versions. For example, the He Count entry for the Noble Gas Database changed its units from parts per million to cc STP of given gas specie per cc STP of the total gas. This would be better qualified as a unit change and would be associated with a certain weight in contribution to the total change to the dataset.

CHAPTER 5
DATABASE VERSIONING

The framework for communicating change information resembles the spreadsheet context. However, the main difference is the method to refer to the Attributes as they are mapped across versions. Database tables do not have strict identifiers for rows and columns as spreadsheets do. Row identification relies on indexed keys to uniquely identify entries, but these entries can also be arranged and presented in different orderings depending on the queries used to view the database. From Reference 24, elements within the database do not have to be digitally organized in the same way that it is physically presented.

Databases are also meant to be kept online for extended periods of time. Spreadsheets must be entirely republished in order for a change to the data to be made public. Changes in a database are therefore more sparse in between each version. In order to make changelogs more comprehensive, they can describe changes by day or other increment of time or describe the latest particular subset of time.

Once the changes within the database are encoded into the versioning conceptual model, publishing the changes can follow the methods detailed in Chapter 4

CHAPTER 6
ONTOLOGY VERSIONING

The Global Change Master Directory is a repository to find Earth science data from NASA. They employ a strict set of keywords to tag and label datasets in order to make them more searchable. Version 1.0.0 of GCMD Keywords was published on April 24, 1995, and as of the time of writing, the most recent version of the keywords is 8.4. As can be seen, the naming scheme of the versions changed since the first publication of the keywords. In the initial scheme, each part of the decimal system represented a different level of the GCMD Keyword hierarchy. When a change occured to a concept in a level of the hierarchy, the associated version number increments.

Since there exists an implementation of the GCMD Keywords in RDF, the URIs can be used as references for the Attributes in the concept model. As mentioned in Reference 8, in order for a data consumer to understand how to use a new version of an ontology, they need not only understand what concepts are new and what concepts are old, but also how to map the old ontology onto the new ontology. The mapping then informs the migration of Keyword labeling of datasets between versions.

CHAPTER 7
CONCLUSION

REFERENCES

[1] B. R. Barkstrom and J. J. Bates, "Digital library issues arising from earth science data," 2006.

[2] M. Branco, D. Cameron, B. Gaidioz, V. Garonne, B. Koblitz, M. Lassnig, R. Rocha, P. Salgado, and T. Wenaus, "Managing atlas data on a petabyte-scale with dq2," *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062017, 2008. [Online]. Available: http://stacks.iop.org/1742-6596/119/i=6/a=062017

[3] B. Barkstrom, *Earth Science Data Management Handbook: Users and User Access*. CRC Press, April 2014, vol. 1. [Online]. Available: https://books.google.com/books?id=pI3rTgEACAAJ

[4] S. Lyons, "Persistent identification of electronic documents and the future of footnotes," *Law Library Journal*, vol. 97, pp. 681–694, 2005.

[5] W. F. Tichy, "Rcsa system for version control," *Software: Practice and Experience*, vol. 15, no. 7, pp. 637–654, 1985.

[6] S.-Y. Chien, V. J. Tsotras, and C. Zaniolo, "Version management of xml documents," in *Selected Papers from the Third International Workshop WebDB 2000 on The World Wide Web and Databases*. London, UK, UK: Springer-Verlag, 2001, pp. 184–200. [Online]. Available: http://dl.acm.org/citation.cfm?id=646544.696357

[7] S. Chacon, *Pro Git*, 1st ed. Berkely, CA, USA: Apress, 2009.

[8] "Common questions: Ubuntu release and version numbers," Canonical Ltd., accessed: December 12, 2016. [Online]. Available: https://help.ubuntu.com/community/CommonQuestions##Ubuntu%20Releases%20and%20Version%20Numbers

[9] J. Dijkstra, *On complex objects and versioning in complex environments*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 13–23. [Online]. Available: http://dx.doi.org/10.1007/BFb0024353

[10] P. Cederqvist, R. Pesch *et al.*, *Version management with CVS*. Network Theory Ltd., 2002.

[11] S. Burrows, "A review of electronic journal acquisition, management, and use in health sciences libraries," *Journal of the Medical Library Association*, vol. 94, no. 1, p. 67, 2006.

[12] B. Tagger, "A literature review for the problem of biological data versioning," Online, July 2005. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/btagger/LitReview.pdf

[13] A. Stuckenholz, "Component evolution and versioning state of the art," *SIGSOFT Softw. Eng. Notes*, vol. 30, no. 1, pp. 7–, Jan. 2005. [Online]. Available: http://doi.acm.org/10.1145/1039174.1039197

[14] M. Fischer, M. Pinzger, and H. Gall, "Populating a release history database from version control and bug tracking systems," in *Proceedings of the International Conference on Software Maintenance*, ser. ICSM '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 23–32. [Online]. Available: http://dl.acm.org/citation.cfm?id=942800.943568

[15] U. K. Wiil and D. L. Hicks, "Requirements for development of hypermedia technology for a digital library supporting scholarly work," in *Proceedings of the 2000 ACM Symposium on Applied Computing - Volume 2*, ser. SAC '00. New York, NY, USA: ACM, 2000, pp. 607–609. [Online]. Available: http://doi.acm.org/10.1145/338407.338517

[16] S. Payette and T. Staples, *The Mellon Fedora Project Digital Library Architecture Meets XML and Web Services*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 406–421. [Online]. Available: http://dx.doi.org/10.1007/3-540-45747-X_30

[17] K. Berberich, S. Bedathur, T. Neumann, and G. Weikum, "A time machine for text search," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '07. New York, NY, USA: ACM, 2007, pp. 519–526. [Online]. Available: http://doi.acm.org/10.1145/1277741.1277831

[18] J. Kovse and T. Härder, "V-grid-a versioning services framework for the grid," in *Berliner XML Tage*, 2003.

[19] K. Holtman, "CMS Data Grid System Overview and Requirements," CERN, Geneva, Tech. Rep. CMS-NOTE-2001-037, Jul 2001. [Online]. Available: http://cds.cern.ch/record/687353

[20] K. S. Baker and L. Yarmey, "Data stewardship: Environmental data curation and a web-of-repositories," *The International Journal of Data Curation*, vol. 4, no. 2, pp. 12–27, 2009.

[21] M. Bouzeghoub and V. Peralta, "A framework for analysis of data freshness," in *Proceedings of the 2004 International Workshop on Information Quality in Information Systems*, ser. IQIS '04. New York, NY, USA: ACM, 2004, pp. 59–67. [Online]. Available: http://doi.acm.org/10.1145/1012453.1012464

[22] R. Rantzau, C. Constantinescu, U. Heinkel, and H. Meinecke, "Champagne: Data change propagation for heterogeneous information systems," in *In: Proceedings of the International Conference on Very Large Databases (VLDB), Demonstration Paper, Hong Kong*, 2002.

[23] R. Bose and J. Frew, "Lineage retrieval for scientific data processing: A survey," *ACM Comput. Surv.*, vol. 37, no. 1, pp. 1–28, Mar. 2005. [Online]. Available: http://doi.acm.org/10.1145/1057977.1057978

[24] D. Dai, Y. Chen, D. Kimpe, and R. Ross, "Provenance-based object storage prediction scheme for scientific big data applications," in *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE, 2014, pp. 271–280.

[25] R. Cavanaugh, G. Graham, and M. Wilde, "Satisfying the tax collector: Using data provenance as a way to audit data analyses in high energy physics," in *Workshop on Data Lineage and Provenance*, Oct. 2002.

[26] B. R. Barkstrom, *Data Product Configuration Management and Versioning in Large-Scale Production of Satellite Scientific Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 118–133. [Online]. Available: http://dx.doi.org/10.1007/3-540-39195-9_9

[27] P. Vassiliadis, M. Bouzeghoub, and C. Quix, *Towards Quality-Oriented Data Warehouse Usage and Evolution*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 164–179. [Online]. Available: http://dx.doi.org/10.1007/3-540-48738-7_13

[28] C. Tilmes, P. Fox, X. Ma, D. L. McGuinness, A. P. Privette, A. Smith, A. Waple, S. Zednik, and J. G. Zheng, *Provenance Representation in the Global Change Information System (GCIS)*, ser. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, June 2012, vol. 7525, ch. Provenance and Annotation of Data and Processes, pp. 246–248. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34222-6_28

[29] X. Ma, J. G. Zheng, J. C. Goldstein, S. Zednik, L. Fu, B. Duggan, S. M. Aulenbach, P. West, C. Tilmes, and P. Fox, "Ontology engineering in provenance enablement for the national climate assessment," *Environmental Modelling & Software*, vol. 61, pp. 191 – 205, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364815214002254

[30] X. Ma, P. Fox, C. Tilmes, K. Jacobs, and A. Waple, "Capturing provenance of global change information," *Nature Clim. Change*, vol. 4, no. 6, pp. 409–413, Jun 2014, commentary. [Online]. Available: http://dx.doi.org/10.1038/nclimate2141

[31] A. Capiluppi, P. Lago, and M. Morisio, "Evidences in the evolution of os projects through changelog analyses," in *Taking Stock of the Bazaar:*

Proceedings of the 3rd Workshop on Open Source Software Engineering, J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani, Eds., May 2003, citation: Capiluppi, A., Lago, P., Morisio, M. (2003). ?Evidences in the evolution of OS projects through Changelog Analyses.? in Feller, P., Fitzgerald, B., Hissam, B. Lakhani, K. (eds.) Taking Stock of the Bazaar: Proceedings of the 3rd Workshop on Open Source Software Engineering ICSE?03 International Conference on Software Engineering Portland, Oregon May 3-11, 2003. pp.19-24.. [Online]. Available: http://roar.uel.ac.uk/1037/

[32] D. German, "Automating the measurement of open source projects," in *In Proceedings of the 3rd Workshop on Open Source Software Engineering*, 2003, pp. 63–67.

[33] K. Herzig and A. Zeller, "Mining cause-effect-chains from version histories," in *2011 IEEE 22nd International Symposium on Software Reliability Engineering*, Nov 2011, pp. 60–69.

[34] K. Chen, S. R. Schach, L. Yu, J. Offutt, and G. Z. Heller, "Open-source change logs," *Empirical Softw. Engg.*, vol. 9, no. 3, pp. 197–210, Sep. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:EMSE.0000027779.70556.d0

[35] C. Tilmes, Y. Yesha, and M. Halem, "Distinguishing provenance equivalence of earth science data," *Procedia Computer Science*, vol. 4, pp. 548 – 557, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050911001153

[36] B. Cao, Y. Li, and J. Yin, "Measuring similarity between gaphs based on the levenshtein distance," *Applied Mathematics & Information Sciences*, vol. 7, no. 1L, pp. 169–175, 2013.

[37] X. Gao, B. Xiao, D. Tao, and X. Li, "A survey of graph edit distance," *Pattern Analysis and Applications*, vol. 13, no. 1, pp. 113–129, 2010. [Online]. Available: http://dx.doi.org/10.1007/s10044-008-0141-y

[38] W. Goddard and H. C. Swart, "Distances between graphs under edge operations," *Discrete Math.*, vol. 161, no. 1-3, pp. 121–132, Dec. 1996. [Online]. Available: http://dx.doi.org/10.1016/0012-365X(95)00073-6

[39] Y. Ma, M. Shi, and J. Wei, "Cost and accuracy aware scientific workflow retrieval based on distance measure," *Information Sciences*, vol. 314, no. C, pp. 1–13, Sep. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2015.03.055

[40] M. D. Flouris, "Clotho: Transparent data versioning at the block i/o level," in *In Proceedings of the 12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies (MSST 2004)*, 2004, pp. 315–328.

[41] M. Klein and D. Fensel, "Ontology versioning on the semantic web," in *Stanford University*, 2001, pp. 75–91.

[42] B. Polyak, E. Prasolov, I. Tolstikhin, L. Yakovlev, A. Ioffe, O. Kikvadze, O. Vereina, and M. Vetrina, "Noble gas isotope abundances in terrestrial fluids," 2015. [Online]. Available: https://info.deepcarbon.net/vivo/display/n6225

[43] S. Morrison, R. Downs, J. Golden, A. Pires, P. Fox, X. Ma, S. Zednik, A. Eleish, A. Prabhu, D. Hummer, C. Liu, M. Meyer, J. Ralph, G. Hystad, and R. Hazen, "Exploiting mineral data: applications to the diversity, distribution, and social networks of copper mineral," in *AGU Fall Meeting*, 2016.

[44] J. F. Roddick, "A model for schema versioning in temporal database systems," *Australian Computer Science Communications*, vol. 18, pp. 446–452, 1996.

[45] P. Klahold, G. Schlageter, and W. Wilkes, "A general model for version management in databases," in *Proceedings of the 12th International Conference on Very Large Data Bases*, ser. VLDB '86. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1986, pp. 319–327. [Online]. Available: http://dl.acm.org/citation.cfm?id=645913.671314

[46] S. Pröll and A. Rauber, "Citable by design - A model for making data in dynamic environments citable," in *DATA 2013 - Proceedings of the 2nd International Conference on Data Technologies and Applications, Reykjavík, Iceland, 29 - 31 July, 2013*, 2013, pp. 206–210. [Online]. Available: http://dx.doi.org/10.5220/0004589102060210

[47] M. Helfert, C. Francalanci, and J. Filipe, Eds., *DATA 2013 - Proceedings of the 2nd International Conference on Data Technologies and Applications, Reykjavík, Iceland, 29 - 31 July, 2013*. SciTePress, 2013.

[48] S. Proell and A. Rauber, "Scalable data citation in dynamic large databases: Model and reference implementation," in *IEEE International Conference on Big Data 2013 (IEEE BigData 2013)*, 10 2013.

[49] B. Lafuente, R. T. Downs, H. Yang, and N. Stone, "1. the power of databases: The RRUFF project," in *Highlights in Mineralogical Crystallography*, T. Armbruster and R. M. Danisi, Eds. Walter de Gruyter GmbH, 2015, pp. 1–30. [Online]. Available: http://dx.doi.org/10.1515/9783110417104-003

[50] K. Belhajjame, H. Deus, D. Garijo, G. Klyne, P. Missier, S. Soiland-Reyes, and S. Zednik, *PROV Model Primer*, W3C Working Group, Apr. 2013, 30. [Online]. Available: https://www.w3.org/TR/prov-primer