A Provenance Model for Manually Curated Data

Peter Buneman[1], Adriane Chapman[2], James Cheney[1],
and Stijn Vansummeren[3,⋆]

[1] University of Edinburgh
[2] University of Michigan, Ann Arbor
[3] Hasselt University and Transnational University of Limburg, Belgium

Abstract. Many curated databases are constructed by scientists integrating various existing data sources "by hand", that is, by manually entering or copying data from other sources. Capturing provenance in such an environment is a challenging problem, requiring a good model of the process of curation. Existing models of provenance focus on queries/views in databases or computations on the Grid, not updates of databases or Web sites. In this paper we motivate and present a simple model of provenance for manually curated databases and discuss ongoing and future work.

1 Introduction

Many scientific databases[1] are constructed by manual effort of scientists acting as *curators*. Curators use a wide variety of sources to select, organize, classify and annotate existing data into a database on some topic. Such databases are now supplanting printed "reference manuals" as standard sources of raw scientific data. Although they are most widespread in bioinformatics, scientific databases in many disciplines involve a degree of manual curation.

Curation typically involves "manually" reading journal articles or browsing remote databases to find relevant new information. Data gleaned from journal abstracts or copied from other databases is entered directly by the curator using a Web form or custom interface (such as a MS Access application). This manual process, in which the scientist's brain is in the inner loop, is what distinguishes curation from related activities such as data integration. Curated data is generally of higher quality, but is correspondingly more expensive to produce.

The perceived value of curated databases rests on their provenance: that is, the fact that they have been constructed by well-informed individuals who have exercised scientific judgment in assimilating data sources. However, the volatile nature of electronic media makes it difficult to trust blanket assertions about the provenance of such databases. Instead, it is widely believed that explicit evidence about the provenance of such databases must be recorded in order to

⋆ Postdoctoral researcher of the Fund for Scientific Research - Flanders.

[1] In this paper the term "database" should be interpreted broadly to include data stored in flat files, XML documents, Web sites, etc., not just RDBMSs.

L. Moreau and I. Foster (Eds.): IPAW 2006, LNCS 4145, pp. 162–170, 2006.

preserve the scientific record and assess the scientific value of such databases (and any results derived from them). This *provenance information* should, at least, indicate the origin, context, intermediate source(s) and modification history of the data. Besides its intrinsic value as part of the scientific record, provenance can be used to detect duplicate copies of information and to help automate "data cleaning" tasks such as propagating corrections to source databases and tracking down the sources of discrepancies.

While it is very easy to build up a database by searching for, copying and pasting data, maintaining provenance information adequate for scientific applications requires additional effort, which adds to the already-high cost of curated data. Curators usually attempt to add links to the original publications or source databases, but in practice, provenance records are often absent, incomplete or ad hoc, often despite curators' best efforts. Also, manually-managed provenance records are at higher risk of human error or falsification. Since a great deal of information relevant to provenance is available to the systems involved, we believe that it is important to develop techniques for data curation which automate provenance management as much as possible.

Previous approaches to provenance management [1,6,7,9,11,18] typically have focused on situations in which all of the interactions with data take place in a single controlled environment (e.g. an operating system, file system, or database), or in which new data is only constructed from existing data using nondestructive mechanisms such as database views or scientific workflows. Both assumptions are violated in the case of manually curated databases. Tracking the provenance of data that moves among databases or Grid resources is challenging because there is no one system that can capture all of the actions involved; instead, many systems must cooperate in order to maintain the chain of provenance. Also, curated databases are updated in-place with local copies of source data rather than constructed as views of source databases.

The purpose of this paper is to describe the challenges involved in managing provenance for manually curated databases, and to summarize our current approach to them. Section 2 defines the problem and describes the constraints which we believe must be met by a practical solution. In Section 3, we propose a *copy-paste model* for describing user actions in assimilating external data sources into curated database records. We define provenance as a relation between versions of a database describing how each part of the output was derived from data in earlier versions or external sources. Section 4 discusses ongoing and future work on implementing and extending the copy-paste model, including a proof-of-concept implementation, approaches to tracking user actions, extensions to the copy-paste model, and data citation.

2 The Problem

Biological databases such as UniProt [16] and the Nuclear Protein Database [8] are manually curated; that is, they are constructed (at least in part) by curators modifying individual records directly rather than by an automatic view

or data extraction process. To the extent that extraction, cleaning and integration operations involve user interaction, data warehouses [7] could also be considered to be manually curated. For the moment we focus on tracking provenance for fine-grained, manual updates, rather than queries, views, or "bulk" updates.

We define the *provenance management problem* for manually curated databases as follows. Given a definition of the complete and correct history of a database as it evolves over time, the goal is to store sufficient provenance information to be able to answer queries about the history given only the provenance information and the final database state(s). Note that we do not assume an absolute definition of history; instead, the appropriate form of historical information depends on the application.

This provenance management problem may seem already solved; for example, file systems, database triggers, version control systems, and Wikis provide adequate provenance management (in the form of creation/modification timestamps, user activity logs, change logs, etc.) for their application areas. However, all of these systems rely on strong assumptions: there is a single system that monitors all access to the data, and the data is stored in a single format. When information crosses boundaries between systems, such provenance information usually becomes invalid, and there is no way to say that data in one system comes from another system (possibly with a different data model). Similarly, solutions using Grid technology [12] and customized e-notebook or workbench software [15] require a substantial level of coordination among databases and applications. While such tools are likely to be beneficial, it will take time for them to become widely adopted among scientists. Recently, Muniswamy-Reddy et al. [3,13] have developed a "provenance-aware storage system" (PASS), which tracks provenance at the file system level. While certainly relevant (especially for scientific data stored using files), PASS addresses provenance issues orthogonal to those arising in other forms of databases.

In contrast to common data integration situations in business, there is no central authority that controls all of the scientific databases relevant to a given area. Thus, no single system can monitor all scientific databases or mandate changes or standard practices. In addition, not all databases are being actively maintained, and others may be resistant to change. Even motivated database curators may lack the resources to modify their own databases. Conversely, databases often change independently and have widely varying record-keeping practices, use a wide variety of data models (ranging from RDBMS to flat files to XML to filesystems), and their curators employ a wide variety of independent tools (Web browsers, editors, database access applications, etc.)

Under these constraints, we believe that a practical solution must have the following characteristics:

- **decentralized**: deployable one database at a time, without requiring cooperation among all databases at once
- **data model-independent**: should work for data stored in flat file, relational, XML, file system, Web site, etc. models

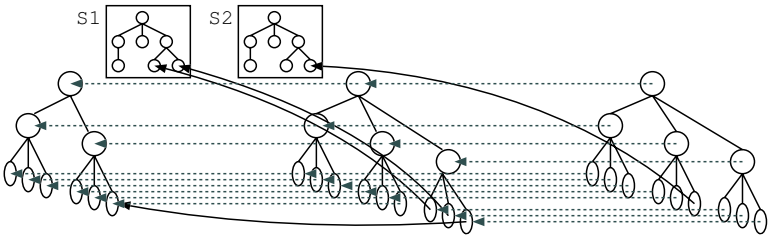

Fig. 1. A two-step provenance record

- **usable with minimum changes to existing curatorial practice**: ideally, provenance tracking is invisible to the user
- **useful without significant changes to existing database systems**: since it is impractical to impose global standards across databases
- **robust in the face of changes to the databases**: since we cannot stop other databases from changing anyway
- **scales gracefully** to situations in which many databases cooperate to maintain provenance chains.

3 The Copy-Paste Model

In this paper, we focus on the problem of managing provenance for a single database as data is copied from external sources or modified within the target database. This is a common situation, and we believe our approach meets most of the criteria above; some issues, such as robustness in the face of changes to other databases, are not yet handled.

As remarked by Groth et al. [10], it is important to think carefully about what provenance information to retain, based on what information will be useful evidence to later observers. In our setting, we aim only to track the internal relationships among a sequence of versions of the target database and fixed source databases. Thus, we define the "complete and correct history" in the above problem statement as follows. We take an abstract view of databases as maps from data locations (or *citations*) to data values. For the purposes of this paper, we consider only a "flat" data model, in which a database is simply a set of key-value pairs. Many other refinements are possible, for example using line numbers to locate data in text files, XPath or XPointer expressions to address data in XML, or using key information to locate data in relational databases. A more realistic instance of our model that uses paths to address relational or XML data can be found in [5].

We further assume that changes to the database can be modeled using transactions comprising simple "copy-paste" updates. Updates (denoted by U) are sequences of atomic editing operations: insertion $\mathsf{ins}\ l\ v$ which creates a new location l that is mapped to data value v, deletion $\mathsf{del}\ l$ which deletes location l (and its corresponding data value); and copy-paste $l \leftarrow l'$ which updates the value at

location l to the value mapped by l'. We write $I \xrightarrow{U} O$ to for a triple (I, U, O) such that applying update U on input database I yields output database O. The *one-step provenance relation* $\mathcal{P}(I \xrightarrow{U} O)$ is a subset of $dom(I) \times dom(O)$, defined as follows:

$$\begin{aligned}
\mathcal{P}(I \xrightarrow{l \leftarrow l'} O) &= \{(m,m) \mid m \in dom(I), m \neq l\} \cup \{(l', l)\} \\
\mathcal{P}(I \xrightarrow{\mathsf{ins}\ l\ v} O) &= \{(m,m) \mid m \in dom(I)\} \cup \{(\mathtt{NULL}, l)\} \\
\mathcal{P}(I \xrightarrow{\mathsf{del}\ l} O) &= \{(m,m) \mid m \in dom(I) - l\} \cup \{(l, \mathtt{NULL})\} \\
\mathcal{P}(I \xrightarrow{U;U'} O) &= \bigcup \{r \otimes s \mid r \in \mathcal{P}(I \xrightarrow{U} J), s \in \mathcal{P}(J \xrightarrow{U'} O)\}
\end{aligned}$$

where $r \otimes s$ is defined as follows:

$$\begin{aligned}
(l,m) \otimes (m,n) &= \{(l,n)\} && (l, m, n \neq \mathtt{NULL}) \\
(l,m) \otimes (m,n) &= \{(\mathtt{NULL}, l)\} && (l = \mathtt{NULL} \text{ or } m = \mathtt{NULL}) \\
(l,m) \otimes (m,n) &= \{(l, \mathtt{NULL})\} && (m = \mathtt{NULL} \text{ or } n = \mathtt{NULL}) \\
r \otimes s &= \emptyset && \text{otherwise}
\end{aligned}$$

Given a sequence of database instances and updates $I_0 \xrightarrow{U_1} \cdots \xrightarrow{U_n} I_n$, we define the multi-step provenance relation $Prov(Tid, Loc, Loc)$ as

$$\{1\} \times \mathcal{P}(I_0 \xrightarrow{U_1} I_1) \cup \cdots \cup \{n\} \times \mathcal{P}(I_{n-1} \xrightarrow{U_n} I_n).$$

Intuitively, $\mathcal{P}(I \xrightarrow{U} O)$ (and consequently, $Prov$) describe how data is copied, inserted, or deleted during a transaction (or sequence of transactions). Figure 1 depicts an example two-step provenance relation on a database with tree-structured locations. Here, the "boxed" trees S_1, S_2 are external sources, whereas the other trees denote the original database, the database after the first update, and the database after the second update, respectively. The solid and dashed lines describe the $Prov$ relation; dashed lines indicate provenance links for *unchanged* data. Such links can always be omitted from the $Prov$ relation to save space.

Using the $Prov$ relation, we can define a number of interesting basic provenance queries giving information about the creation time of a location, the deletion time, where a location was copied from, and whether a location has been modified:

$$\begin{aligned}
&Inserted(t,l) \leftarrow Prov(t, \mathtt{NULL}, l). && Deleted(t,l) \leftarrow Prov(t, l, \mathtt{NULL}). \\
&Copied(t,l,m) \leftarrow Prov(t,l,m), l \neq m. && Unchanged(t,l) \leftarrow Prov(t,l,l).
\end{aligned}$$

The $Inserted(t,l), Copied(t,l',l), Unchanged(t,l)$ queries informally say that the data at location l at the end of transaction t was inserted during t, copied from the data at l at the beginning of t, or unchanged during t, respectively; similarly, $Deleted(t,l)$ says that the data at l was deleted during t.

Transaction identifiers can be used to index a table $Trans(\underline{Tid}, Uid, Time, \ldots)$ containing additional metadata about transactions. The following recursive query $Q(l, tid, uid, t)$ defines the relation "the data at l at the end of transaction tid was originally inserted by user uid at time t":

$$Q(l, tid, uid, t) \leftarrow Inserted(tid, l), Trans(tid, uid, t).$$
$$Q(l, tid, uid, t) \leftarrow Copied(tid, l, m), Q(m, tid - 1, uid, t).$$
$$Q(l, tid, uid, t) \leftarrow Unchanged(tid, l), Q(l, tid - 1, uid, t).$$

Many interesting provenance queries also refer to the raw data. A simple example is to return the original source (that is, the external link, or `NULL` if none) alongside each result. Another example is using provenance to filter out data from known unreliable sources, or to group or aggregate the data by source. Querying the data and its provenance "side-by-side" involves additional processing, since the data and provenance may be stored in separate databases. However, existing techniques for multidatabase querying and managing annotations can be used for this purpose [1,17].

If only one database tracks provenance, then the chain of provenance can only be followed to the point where data was copied from an external source; only "local" provenance queries can be answered. We can only answer "global" queries if all the databases involved record provenance. Maintaining consistent and valid provenance when the distributed databases are being updated asynchronously is an interesting area for further research.

4 Ongoing and Future Work

4.1 An implementation

We have implemented a "copy-paste database", CPDB, that tracks the provenance of data copied from external sources to the target database [5]. CPDB permits the user to copy source data from external sites or databases into the target database, and modify the data to fit the target database's structure. The user's actions are intercepted and provenance information is recorded in a *provenance store*. CPDB uses paths as locations to address data stored in either XML or relational form. CPDB addresses a number of implementation issues that were left implicit in the discussion in Section 3. We now discuss these issues further.

How Can We Address and Update Heterogeneous Data? The approach we take to the first issue is to require that the citable data in every source database is published as a "fully-keyed" XML view, and the updatable data in the target database is presented as an *updatable* XML view. Thus, paths can be used to select data from sources and locate data in copy-paste updates to the target database. We believe these are reasonable requirements since XML publishing and view update for legacy RDBMSs are widely recognized as important research problems and have already received attention [2]. Moreover, many commercial relational databases already provide some capability for publishing their data as XML. In addition, any Web page can be interpreted as an XML document, addressed by its URL and an XPath or XPointer describing the cited data.

This approach does *not* require that any of the source or target databases represent data internally as XML. Any underlying data model for which path addresses make sense can be used. Also, the databases need not expose all of their data. Instead, it is up to the databases' administrators how much data to expose for copying or updating. The data in many scientific databases consists of a "catalog" relation that contains all the raw data, together with supporting cross-reference tables. It is only this catalog that would typically need to be made available by a source database.

Proxies or client-side scripting could also be used to add more useful citations to non-cooperating database websites without changing the curator's natural behavior. However, a chain is only as strong as its weakest link, and the provenance data derived from this method can only contain information about the database's website, not any of the underlying database values. This is related to the problem of *data citation* discussed below.

How Can We Capture the User's Actions in an Unobtrusive Way? The current implementation of CPDB provides a Web interface that enables the user to import data from a source database and paste it into the target. However, this interface is relatively clumsy compared to what curators normally do, namely browsing to databases' Web sites, copying data from the relevant pages, and pasting it into a target database entry form. For an approach to automatic provenance tracking to be successful, it has to make life easier for the curators, not harder.

Fortunately, it appears to be possible to instrument Web browsers so that the user's browsing, copying, pasting, and form submission actions are recorded as provenance records. We are currently investigating whether existing techniques for Web browser activity logging being developed in the human-computer interaction community [14] can be adapted to record rich provenance information unobtrusively.

How Can the Resulting Provenance Records be Stored and Queried Efficiently? The *Prov* relation given in the previous section is potentially very verbose since it stores links between unchanged data in subsequent versions. (These are shown by the dotted lines in Figures 1). Thus, the number of links between two versions of a database is proportional to the size of the data, not the difference between the versions. While these links are needed to answer provenance queries, they do not need to be stored explicitly. In CPDB, we store only "essential" provenance links, that is, links having to do with copies, inserts, or deletes. CPDB also exploits optimization opportunities offered by the tree structure of path addresses. The full *Prov* relation can then be defined as a view of the reduced form. The size of the reduced form of *Prov* is proportional only to the size of the update operation, not the size of the database.

This reduces the amount of storage space needed for the provenance of manual updates to an acceptable level. We also performed experiments that show that the performance of queries to the core provenance relation is better than for

more explicit forms of *Prov*. The reason for this is that it is often faster to calculate provenance links "on the fly" than to fetch them from storage.

4.2 Future Work

Database Archiving and Citation. The ability to locate data precisely in the source database is fundamental to the copy-paste model we developed in Section 3. It is also important if the curator has not directly copied the data, but has interpreted it and wants to provide a citation to that portion of the database. However, in addition to location information, citations carry other useful data such as authorship, title, etc. It is becoming increasingly important to make curated databases citable and to describe how a database or a part of it should be cited [4]. We are currently considering techniques for augmenting database archiving and citation with support for provenance tracking and querying.

Extending Our Model of Provenance. The copy-paste language introduced in Section 3 suffices to model the behavior of a human curator who inserts individual items into the curated database. Ideally, however, we would also like to track the provenance of data constructed by automatic processes (e.g., a database query, a workflow, a scientific algorithm, ...). We are currently investigating extensions of our approach to provenance to full query/update languages for relational, complex object, and XML data, including aggregation and "fusion" operations such as summation, joins or unions. Among the research challenges here are finding space-efficient, compact representations for such provenance information and providing appropriate high-level query language operations for querying more complex provenance expressions.

In addition, we are generalizing the copy-paste model to study the provenance behavior of arbitrary computations based on *rewriting machines*, a variant of Turing machines. In Turing machines, there is at best an implicit correlation between input and output data. Without knowing the intent of the designer of a machine, it is impossible to give a correct provenance semantics for it. In rewriting machines, the basic machine operations are *rewritings*, which generalize copy-paste operations and have a natural, unambiguous provenance semantics.

5 Conclusions

Tracking the provenance of data in manually curated databases is challenging, primarily because the most obvious approaches require unrealistic levels of homogeneity, cooperation and coordination among databases. While the long-term solution may require changes to common scientific practice, there are steps that can be taken in the short term and at the local level to improve record-keeping in current practices. We have developed and implemented a model for recording the provenance of data that flows into a single curated database and discussed incremental extensions to this model that provide further improvements. Many substantial challenges remain.

References

1. D. Bhagwat, L. Chiticariu, W. C. Tan, and G. Vijayvargiya. An annotation management system for relational databases. In *Proc. of the Intl. Conf. on Very Large Data Bases (VLDB)*, pages 900–911. Morgan Kaufmann, 2004.
2. V. P. Braganholo, S. B. Davidson, and C. A. Heuser. From XML view updates to relational view updates: old solutions to a new problem. In *VLDB 2004*, pages 276–287, 2004.
3. U. Braun, S. Garfinkel, D. A. Holland, K.-K. Muniswamy-Reddy, and M. I. Seltzer. Issues in automatic provenance collection. In *Proceedings of the 2006 International Provenance and Annotation Workshop (IPAW 2006)*, 2006. This proceedings.
4. P. Buneman. How to cite curated databases and how to make them citable. In *SSDBM*, 2006. To appear.
5. P. Buneman, A. P. Chapman, and J. Cheney. Provenance management in curated databases. In *SIGMOD*, 2006. To appear.
6. P. Buneman, S. Khanna, and W.-C. Tan. Why and Where: A characterization of data provenance. In *ICDT*, pages 316–330, 2001.
7. Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. In *Proceedings of the 27th VLDB Conference, Roma, Italy*, pages 41–58, 2001.
8. G. Dellaire, R. Farrall, and W. A. Bickmore. The nuclear protein database (NPD): sub-nuclear localisation and functional annotation of the nuclear proteome. *Nucleic Acids Research*, 31(1):328–330, 2003.
9. I. Foster, J. Vockler, M. Eilde, and Y. Zhao. Chimera: A virtual data system for representing, querying, and automating data derivation. In *International Conference on Scientific and Statistical Database Management*, pages 1–10, July 2002.
10. P. Groth, S. Miles, , and S. Munroe. Principles of high quality documentation for provenance: A philosophical discussion. In *Proceedings of the 2006 International Provenance and Annotation Workshop (IPAW 2006)*, 2006. This proceedings.
11. P. Groth, S. Miles, W. Fang, S. C. Wong, K.-P. Zauner, and L. Moreau. Recording and using provenance in a protein compressibility experiment. In *HPDC*, 2005.
12. P. T. Groth, M. Luck, and L. Moreau. A protocol for recording provenance in service-oriented grids. In T. Higashino, editor, *OPODIS*, volume 3544 of *Lecture Notes in Computer Science*, pages 124–139. Springer, 2004.
13. K. Muniswamy-Reddy, D. Holland, U. Braun, and M. Seltzer. Provenance-aware storage systems. In *Proceedings of the 2006 USENIX Annual Technical Conference,*, Boston, MA, June 2006. To appear.
14. N. Roussel, A. Tabard, and C. Letondal. All you need is log. WWW2006 Workshop on Logging Traces of Web Activity: The Mechanics of Data Collection, May 2006. Manuscript available at `http://torch.cs.dal.ca/~www2006/roussel-www2006-MechanicsDataCollection.pdf`.
15. R. D. Stevens, A. J. Robinson, and C. A. Goble. *myGrid*: personalised bioinformatics on the information grid. *Bioinformatics*, 2003.
16. UniProt. http://www.ebi.ac.uk/uniprot/.
17. Y. R. Wang and S. E. Madnick. A polygen model for heterogeneous database systems: The source tagging perspective. In D. McLeod, R. Sacks-Davis, and H.-J. Schek, editors, *16th International Conference on Very Large Data Bases, August 13-16, 1990, Brisbane, Queensland, Australia, Proceedings*, pages 519–538. Morgan Kaufmann, 1990.
18. J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, pages 262–276, 2005.