A Model for Data Curation and Transformation, with Provenance

David W. Archer, advised by Lois M. L. Delcambre

Department of Computer Science, Portland State University
Portland, OR 97207 USA
{darcher, lmd @cs.pdx.edu}

Motivation

One non-traditional style of data integration that has seen significant recent interest involves joining fine grain data (e.g., individual data values) from multiple sources to entities they describe, and then querying and manipulating resulting datasets. A typical example from the intelligence community: "Persons of interest", perhaps adversary bomb-makers, are profiled and tracked by integrating small amounts of data from military patrol logs, intelligence databases, cell phone logs, and surveillance records. Such data may be gathered by multiple analysts over extended periods, and may be queried and manipulated in various ways during and after integration to yield new materialized views. Data values may be inserted more than once, for example to document additional data sources. Data may be deleted, and then later re-inserted. Multiple data values for an attribute may be retained either temporarily or permanently. Analysts also commonly want to record expressions of confidence or doubt in data.

We call this a *data curation setting*, and observe that the usefulness of information in this setting depends on both the trustworthiness of the original data, and the decisions made in its curation. Thus detailed *provenance* of data: where it came from, and how, when, in what order, and by whom it was manipulated and queried, is at least as important in our setting as the data. We focus on four needs in data curation settings unaddressed in the current literature:

- Representation of sets of entities with **multi-valued** (that is, bag-valued) attributes sharing a common schema. We call this *simple non-first normal form relational (SNF2)* data
- Representation of provenance that may include a **combination** of query, DML, and DDL operations (e.g., a data value derived by a query and then updated by DML). We call this *multi-provenance*
- Ability to easily access, manipulate, and query the combination of SNF2 data and its multi-provenance using analogs of familiar relational operators
- Operators to express confirmation of or doubt about data

Intended Contributions

In this work, we will contribute a new conceptual model that supports:

- DDL, DML, and query of SNF2 data, including multiple creation of values
- Multi-provenance at the dataset, entity, schema, and attribute value levels

- Operators for and provenance of user expressions of confirmation and doubt about data
- Access to provenance via queries without requiring the user to understand its underlying implementation, nor write complex, recursive queries; and access to provenance via relational views of data and graphical views of provenance

We will also contribute a logical model that

- Faithfully supports our conceptual model
- Economically represents data, its provenance, and its relational structure
- Provides p-time bounded query execution

Conceptual Model

In our model, a *database slice* consists of a finite set of *datasets* of SNF2 data. The top of Figure 1 shows an example dataset (dataset D) with one entity and two attributes, along with other datasets, in a database slice. A *dataloaf* is a totally ordered set of database slices, along with a set of *external sources*. We refer to datasets, entities, attributes, attribute values, and external sources collectively as *components*. The *initial* database slice in a dataloaf is created when a DDL operation is performed to create its first dataset. An operator in our conceptual model operates on the most recent, or *current*, database slice in a dataloaf, and induces a new *result* database slice. The result database slice is a copy of the current database slice, with components modified or augmented as prescribed by the operation performed. An external source represents a data source outside a dataloaf. An example dataloaf, with four database slices and one external source, is shown at the center of Figure 1.

Operators induce *provenance* links from components in the current database slice to components of similar type derived from them in the result database slice (e.g., from a dataset to another dataset), or from external sources to components in the result database slice. Provenance links have attributes, including a *type* (name of operator or query, or *default,* or *renew*). When a result database slice is created from a source database slice, a *default* provenance link is induced from each component in the source database slice to each corresponding (i.e., copied, but unmodified) component in the result database slice. An *operator* provenance link is induced from each component in a source database slice (or an external source) to its modified copy (in the case of DDL or DML operations), or to components newly derived from it (in the case of a query or external data insertion) in the result database slice. We discuss *renew* provenance links below.

The *data definition language* in our model includes operators for creation of datasets, their attributes, and external sources, as well as deletion of datasets and attributes. The *data manipulation language* includes operators for insertion, deletion, and copying of whole datasets, individual entities, and individual attribute values, along with expressions of confirmation and doubt in data values. The *query language* provides Select, Project, Join, and Union operators, and includes predicates for selecting data based on its provenance. The dataloaf in Figure 1 shows an example where the most recent two operators applied were the insertion of a complete dataset, Dataset B, from an external source X, followed by a Join of datasets A and B, resulting in dataset D.

When components are deleted, they are retained and tagged as *expired*. Expired components are copied into result database slices, and connected by *default* provenance

links to their corresponding current database slice components, though they are not available for use by operators. Re-insertion via DML of a deleted attribute value results in a *renew* provenance link connecting the expired attribute value in a source database slice to the re-inserted value in a result database slice.

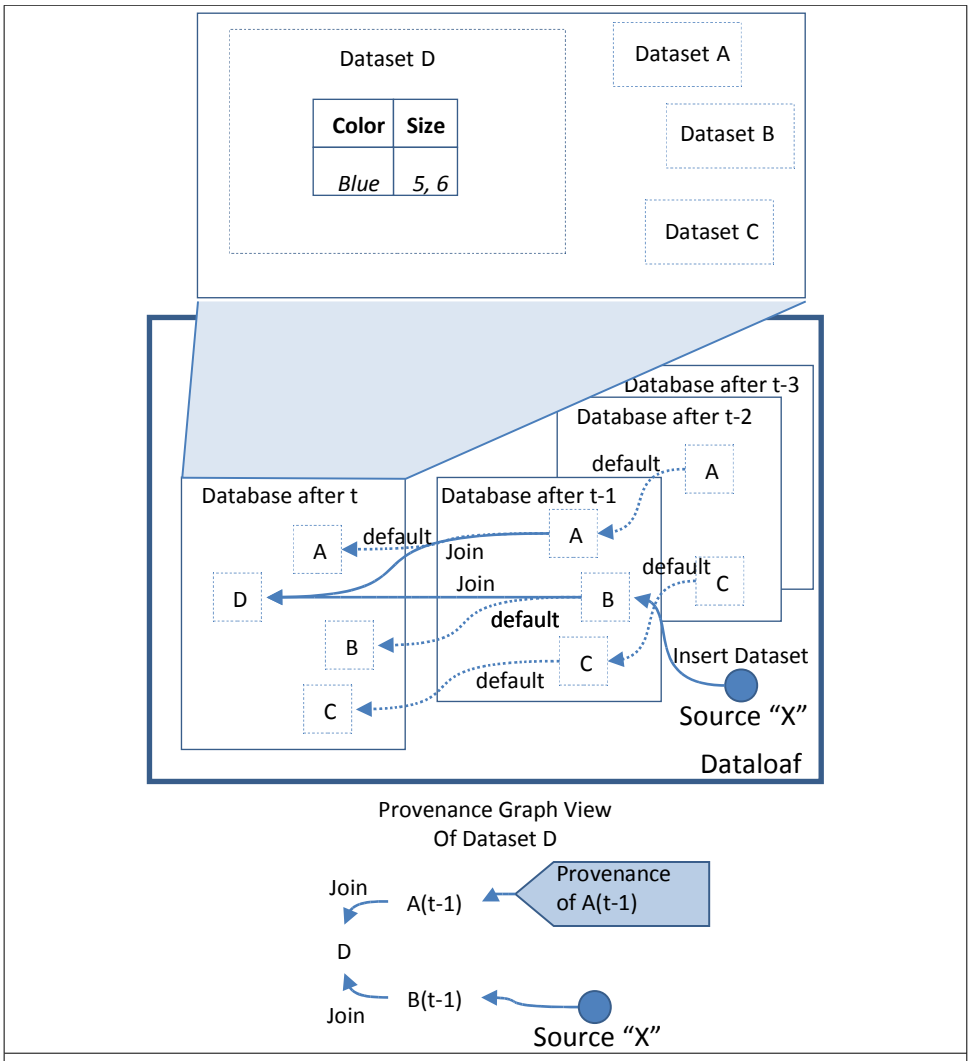

Figure 1. Examples using our conceptual model. Provenance links are shown only at the dataset level. Time progresses from right to left in the dataloaf shown.

We define a *provenance graph view* that includes both data and provenance of a selected component. The graph consists of a set of directed edges representing provenance links, and a set of vertices representing components. The set of edges is the subset of the provenance links in the dataloaf that connect to the selected component either directly or via a path of provenance links. The set of vertices is the set of

components in such paths, including external sources. Figure 1 shows the provenance graph for dataset D at the bottom.

Logical Model

Our conceptual view of databases allows a user to visualize each database slice as a step in time, with all components either newly derived, or "copied forward" from identical predecessor components. In our logical model, we eliminate the redundancy inherent in "copying forward" unchanged components during each application of an operator. We represent the structure and provenance of all components in all datasets with a single directed graph called a *DPGraph*. Components are typed vertices in a DPGraph. Edges in a DPGraph are either *structure edges* or *action edges*. Structure edges represent the relational structure of data. Action edges denote the derivation of one component from another by an operator, or an update to a component by an operator. Our logical model does not include the notion of *default* provenance links. In our logical model, the current database is comprised of a set of *current vertices,* each of which represents the most recent update to a component. Thus the current database may consist of vertices derived at different times, yet it includes the most up-to-date derivation of each component. Operators in our logical model are the same as those in our conceptual model.

Sub-graphs of a DPGraph can be defined to represent useful abstractions. One kind of sub-graph represents the current contents of a dataset, with its schema (attributes) and instance (entities and data values). We call these *current dataset graphs,* and note that they directly model datasets in our conceptual model. A *provenance graph* is a sub-graph of the DPGraph that represents the provenance of a component. A provenance graph is rooted at the component whose provenance it displays, and includes all action edges and component vertices that comprise the history of the component. Provenance graphs directly model the *provenance graph view* from our conceptual model.

Related Work

The *non-first normal form relational model* [1] includes DML, relational operators and selection predicates [2] for attributes with non-atomic values, along with operators (nest, unnest) to translate to and from first normal form relations. Nest and unnest are not needed in our setting, but we adopt this model's Join and Select operators.

The *temporal relational model* [3] addresses multi-valued data where only a single value is applicable during a given time period providing both a kind of non-first normal form data, and a kind of "when" provenance. Our model is different from the temporal model, because of the need to represent multiple values valid simultaneously.

Traditional data integration is realized either by constructing *common data storage,* e.g. using an ETL approach, or by providing *uniform data access*, e.g. a federated database [4]. These approaches support no provenance, although they may record *workflow* logs. Our work does not adopt traditional data integration workflow logs. Instead, we track provenance of data items individually.

A *lazy* provenance system by Cui and Widom [5] computes provenance of a query result after the fact by computing a reverse query that finds the set of tuples that produced the result. By constructing provenance links during operator execution, we avoid the reverse query approach in our work.

Eager provenance models support provenance at the attribute value [6], tuple [7,8], or sub-tree [9,10] level. None support provenance at all levels (relation, attribute, tuple, and data value), making it difficult to query provenance comprehensively. None support SNF2 data. Eager models store provenance as relational data in additional attributes [6,7,8], or in adjunct relations [9,10]. Querying such data requires that a user understand the details of the provenance schema, and write queries to explicitly trace provenance of components one "generation" at a time. Often this requires the use of recursive query techniques and complex joins. Some eager models do not support query operators [10], while others do not support DDL or DML [6,8,9]. Only one model from the literature tracks both DML and query operators [7], but none support multi-provenance, and none support expressions of confidence or doubt in data. Only one model supports provenance for deleted components [7]. Our work adopts and augments both the per-value and per-tuple provenance approaches. Our model adopts and augments the "DML + query" provenance approach used in Trio, as well as leveraging the Trio approach of retaining deleted components.

Conclusion and Future Work

This paper presents motivations and expected contributions of our research. At the workshop, we plan to discuss the current state of our research and highlight areas of the work where we seek insight. Beyond the scope of the current work, we envision a query language for limiting aspects of, or the range of, data provenance visible in derived datasets, as a way of providing provenance information on a "need-to-know" basis. We also envision *forensic* mechanisms that would allow a user to revisit former states of a database and track provenance forward in time to more recent versions. No current literature on provenance models discusses these concepts.

Abbreviated References

[1] Haskin, R., Lorie, R. "On Extending the Functions of a Relational Database System," In Proc. of the 1982 ACM SIGMOD International Conference, 1982.

[2] Jaeschke, G., Schek, H. "Remarks on the Algebra of Non First Normal Form Relations", In Proc. of the 1st ACM SIGMOD Symposium on Principles of Database Systems, 1982.

[3] Ben-Zvi, J. *The Time Relational Model*. PhD thesis. University of Calif., Los Angeles. 1982.

[4] Sheth, A., Larson, J. "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Comput. Surv.* 22, 3, 1990.

[5] Cui, Y., Widom, J. "Lineage Tracing for General Data Warehouse Transformations," The VLDB Journal 12(1), 2003.

[6] Bhagwat, D., Chiticariu, L., Tan, W., Vijayvargiya, G. "An Annotation Management System for Relational Databases," In Proc. of the 30th Int'l. Conference on Very Large Data Bases, 2004.

[7] Benjelloun, O., Das Sarma, A., Halevy, A., Theobald, M., Widom, J. 2008. "Databases with uncertainty and lineage," The VLDB Journal 17(2), 2008.

[8] Green, T., Karvounarakis, G., Tannen, V. "Provenance Semirings," In Proc. of the 26th ACM SIGMOD-SIGACT Symposium on Principles of Database Systems, 2007.

[9] Buneman, P., Khanna, S., Tan, W. "Why and Where: A Characterization of Data Provenance," In Proc. of the 8th Int'l. Conference on Database Theory, 2001.

[10] Buneman, P., Chapman, A., Cheney, J., Vansummeren, S. "A Provenance Model for Manually Curated Data," In Proc. of the Int'l. Provenance and Annotation Workshop, 2006.