

User-centric Annotation Management for Biological Data

Qinglan Li Alexandros Labrinidis Panos K. Chrysanthis

Advanced Data Management Technologies Laboratory
Department of Computer Science, University of Pittsburgh
Pittsburgh, PA 15260, USA
{qinglan, labrinid, panos}@cs.pitt.edu

Abstract. Annotations play an increasingly crucial role in scientific exploration and discovery, as the amount of data and the level of collaboration among scientists increases. Although all such systems are implemented to take user input (i.e., the annotations themselves), very few systems are user-centric, taking into account user preferences on how annotations should propagate and be applied over data. In this paper, we propose to treat annotations as first-class citizens for biological data management by presenting a *user-centric, view-based annotation framework*, called ViP. Under the ViP framework we consider user preferences over the time semantics of annotations (by supporting future annotations) and over the network semantics of annotations (by supporting both implicitly-defined and explicitly-defined annotation propagation paths). In addition to novel functionality, we describe a novel caching technique which enables ViP to outperform the state of the art. We also propose to demonstrate our prototype implementation of the ViP framework. As part of the demo, we propose on the one hand to highlight the user-interface/functionality of our system and on the other hand to visualize the server/behind-the-scenes aspect.

1 Introduction

We are witnessing an accelerated pace of discovery and innovation in science research. This is true across all sciences, from gene sequencing and drug discovery to weather modeling and the exploration of the Universe. Without a doubt, data management is playing a pivotal role in scientific exploration nowadays. In addition to efficiently managing the tsunami of experimental data generated, data management also facilitates effective collaboration among scientists, by recording *data provenance* and by supporting *annotations* [3, 2]. Data provenance essentially keeps track of where the data is coming from (and what transformations it has been through), whereas annotations enable users to record additional information about the data stored (and propagate this information to all “related” data items).

There are a lot of projects that deal with annotation propagation and management, for example, DBNotes [2], Mondrian [5], ULDB [1], bdbms [4], and MMS [7]. Our interest in this research area came from our participation in the Center for Modeling

Pulmonary Immunity (CMPI)¹. Our group is responsible for the design and development of the *data sharing platform* (DataXS), where experimental data, analysis, and models will be shared among project participants. In such a diverse setting, the ability to record annotations and propagate them to all related data items and interested parties is crucial to the success of the project.

As part of the design process and during the implementation of our first prototype, we were able to identify two distinct *usage patterns* which are not handled by the current state of the art. This led to the development of the ViP annotation framework [6], whose main contribution is to give users the ability to specify *when, to what/how, and for whom* a user's annotations will be visible and/or propagated (in addition to specifying the actual annotations). Towards this, the ViP framework utilizes *views* both as a specification mechanism and as a user-interface mechanism.

Support for user-centric time semantics for annotations. For example, assume that we have a microarray scanner which was mis-calibrated on a certain day. When this is first discovered, we want to be able to annotate all experimental data in the system accordingly, but also do this for all data that would fall in this category, but are entered in the system later. Since users have different understanding/explanations of why/how certain biological process unfold, it is possible that they want to personalize the time setting of such annotations (i.e., whether they would applied just now, or also in the future). Most current systems do not support annotations that are also valid in the future (Table 1). The only exception is MMS [7], which always supports future time semantics (i.e., without giving the user the option to choose). We refer to this feature as “*user-centric time semantics*”.

Support for propagation of annotations in user-defined ways. For example, provide the ability to “link” related data items together, so that an annotation on one of them would be visible to the other one and vice-versa. Most annotation-enabled systems propagate annotations along data provenance paths. In other words, annotations are propagated over existing implicit annotation propagation paths between source data and derived data (i.e., driven by the database schema and data transformations). Although this can happen over multiple derivation levels, it fails to capture relationships between data items that do not share a common “ancestry” in the database. As we have witnessed from our involvement in the CMPI project, this can happen often in biological databases. To address this, ViP builds explicit paths for annotation propagation. It also allows users to protect data privacy on these paths, that is, each user can have his or her own paths to propagate annotations. Since these paths can form a network, we refer to this feature as “*user-centric network semantics*”. Although existing systems support implicit annotation propagation paths, none except for our proposal supports explicit, user-defined annotation propagation paths (Table 1).

¹ The Center is a joint effort between the University of Pittsburgh, Carnegie Mellon University, and the University of Michigan, bringing together experimentalists and modelers to study pulmonary immunity in response to three bio-defense pathogens (the influenza A virus, Mycobacterium tuberculosis, which causes TB, and Francisella tularensis, the bacterium responsible for tularemia).

Standard Features	DBNotes[2]	Mondrian[5]	ULDB[1]	bdbms[4]	MMS[7]	ViP[6]
Annotation	Yes	Yes	Confidence	Yes	Yes	Yes
Provenance	Yes	Yes	Lineage	Yes	Yes	Yes
<i>Time Semantics:</i>						
· Implicitly-defined	No	No	No	No	Yes	Yes
· Explicitly-defined	No	No	No	No	No	Yes
<i>Network Semantics:</i>						
· Implicitly-defined	Limited	Limited	Limited	Limited	Yes	Yes
· Explicitly-defined	No	No	No	No	No	Yes
Propagation Type	Eager	On-demand	On-demand	Eager	On-demand	Hybrid
Annotation Storage	Naive	Naive	x-relations	Anno. table	q-type	A-table
Scalability	Small	Medium	Medium	Medium	Large	Large
Query	pSQL	Color algebra	TriQL	A-SQL	Predicate	ViP-SQL

Table 1. Standard Annotation Management Features Comparison

User-centric Features	DBNotes[2]	Mondrian[5]	ULDB[1]	bdbms[4]	MMS[7]	ViP[6]
<i>Time Semantics:</i>						
· Valid Time	No	No	No	No	No	Yes
<i>Network Semantics:</i>						
· Propagation Method	Yes	No	No	Limited	No	Yes
<i>Access Control:</i>						
· Annotations	No	No	No	Limited	No	Yes
· Annotation Views	No	No	No	No	No	Yes
· Annotation Paths	No	No	No	No	No	Yes

Table 2. User Centric Annotation Management Features Comparison

Contributions: This research project has both theoretical and practical contributions as follows:

- based on our experience from a real system implementation, we propose new annotation propagation methods, suitable for biological data,
- we propose user-centric features that enable users to personalize annotation propagation, and
- we propose to use views as a user-interface and also as the formal mechanism to optimize the implementation of the new annotation propagation features.

Roadmap In the next section we briefly describe the main aspects of the ViP framework. In Section 3 we present the implementation of our prototype. In Section 4 we provide an overview of the different components of our prototype system along with the demonstration highlights. We conclude in Section 5 and acknowledge in Section 6.

2 The ViP Framework

To the best of our knowledge, ViP brings user-centric features in many aspects that are not considered in most related works as shown in Table 2.

2.1 User-centric Time Semantics

User-centric Time Semantics During our involvement in the CMPI project, we observed that *experimental data was almost always entered in the database in an order different than the one it was generated*. In fact, even data about the same experiment could be entered at completely different times, since more than one lab were involved in generating the data (for example, one lab would generate the luminex data whereas a different lab would produce microarray data for the same tissues). Looking at annotations, this means that if one wanted to annotate data from a particular experiment with an observation about the tissues, it would **not** be enough to do this once, as additional experimental data may be added into the database later (which would not automatically “inherit” the annotation).

To address this, we proposed the concept of *valid time* for annotations and for annotation paths (which we describe next). This is specified by the user for each annotation and works in tandem with using *database views* to describe the annotation targets. Using views allows us to declaratively describe the data to be annotated instead of simply enumerating them. Combined together, we can set, for example, the valid time for an annotation to be $[now, \infty)$ which means that the annotation will be applied to matching items now and also in the future.

When we consider the time dimension of annotation propagation, we can easily distinguish four different cases:

1. *now* only (e.g., mark all the data that have been processed until today),
2. *now + future* (where an annotation is propagated to data items currently in the database, and also to those that are added in the database in the future, e.g., the typical calibration experiment mentioned above),
3. *future* only (e.g., all the files until today have been fixed, but all files submitted in the future should be marked accordingly),
4. *future interval* only (e.g., for one week after the Daylight Saving Time show an annotation that reminds scientists to make sure they have accounted for Daylight Saving Time in experiment settings). It also be able to start from now.

The cases above present the four *valid time* usages in User-centric Time Semantics of ViP.

2.2 User-centric Network Semantics

The second usage pattern that we observed during our involvement in the CMPI project was that *there exist many relationships, or paths, between data items that cannot be inferred by the existing database schema*. Such links materialize because, for example, tissues from multiple, unrelated experiments are processed together, in a single assay (for example, on a single plate that needs to be filled up to minimize costs).

To address this, ViP enables users to specify explicit *annotation paths*, linking data items together. Annotations should be propagated along these paths, reaching “related” data items, as specified by users. Since these paths are essentially forming a network, we refer to this feature as “*user-centric network semantics*”.

ViP enables users to specify the propagation method. In DBNotes [2], users can specify *custom* propagation scheme to bind the source and target tuples while there is a join operation, so that the annotations that are associated to the source tuples will be propagated to the target tuples. ViP provides a stronger and more complex scheme, that is, we propose to empower users to specify *explicit paths* between data items, thus establishing additional annotation propagation paths. Such explicit paths are defined using views as follows:

- given a source view, V_s and a destination view, V_d
- an explicit annotation propagation path $V_s \rightarrow V_d$ is defined, such that any annotation that is added in a member of V_s must be propagated to all members of V_d .

3 Implementation Highlights

3.1 Implementation Using Views

We propose to use the concept of *database views* as the building block to implement the technologies mentioned above. Database views can be used to describe (at a high-level) the results of a database query. For example, instead of attaching a comment about mis-calibration to individual files (and miss files that are added in the future), using views enables the system to record this annotation in a single location (the view) and to also associate this annotation with files matching the view definition in the future.

Views enable us to build a specialized cache for storing annotations (and may also storing annotation propagation paths), thus improving performance.

3.2 User-centric Access Control

We advocate that scientific annotation must have a strong user-centric component. First of all, much of the data is not public, so appropriate access controls need to be in place for the raw data, and the annotations on them. Secondly, even for public data, the annotations are often private, since they reflect additional analysis that is not ready to be made available to all. Thirdly, in many cases, even the way that raw data are associated (i.e., by specifying explicit paths for annotation propagation) corresponds to private information that should not be made public. Given all these reasons, the ViP framework includes a strong user-centric access control module.

Some systems consider the *access control* on the data level, or even on the update authorization part [4]. Instead, we propose to fully support this feature in a broader domain, both on annotations and on annotation paths. Individual users have different annotation views and paths. We support arbitrary user hierarchies. This is different than traditional access control, since access control on annotation views (given user-centric time semantics) and on annotation paths (given user-centric network semantics) essentially means who can “execute” the annotation propagation mechanism and not on the data itself.

Name	Start Date	Species	Type	Treatment	Treatment Strain	Lab	Time Points	Delete
Mouse Invitro Influenza A	03/11/2008	Mouse	Invitro	Influenza A	H5-PR8	Ross	Time Points	Delete
Monkey Invitro Influenza A	02/22/2008	Monkey	Invitro	Influenza A	H5-Indo	Ross	Time Points	Delete
Monkey Invitro Influenza A	02/21/2008	Monkey	Invitro	Influenza A	Fujian	Ross	Time Points	Delete
Mouse Invivo Influenza A (E10K Sublethal PR8)	02/01/2008	Mouse	Invivo	Influenza A	A/PR	Ross	Time Points	Delete
Mouse Invitro Mycobacterium Tuberculosis	01/10/2008	Mouse	Invitro	Mycobacterium Tuberculosis	Erdman	ADMT	Time Points	Delete
Mouse Invitro Uninfected	09/06/2007	Mouse	Invitro	Uninfected	PBS	Morel	Time Points	Delete

Fig. 1. Annotation Views

4 Prototype Highlights

4.1 User Interface

The ViP framework relies heavily upon the concept of *database views* to declaratively describe annotations and annotation paths². Clearly, users are not expected to provide view definitions in SQL. In our ViP framework, a user can easily specify filtering conditions to locate certain data items. This functionality enables users to specify views using a point and click interface (Figure 1); these views can be trivially used to support user-centric time and network semantics.

In particular, we will showcase the following ways of adding annotations (i.e., defining *annotation views*):

- A set of conditions used for filtering results is used in its entirety (exact match); this is implemented in “Save View” tab functionality as well as annotation definition functionality in our system.
- If the set of conditions are not enough to adequately describe the set of data to be annotated, then we will allow the user to provide additional constraining predicates (typically a date range).
- To also support a simplified interface, we will also enable the user to just specify the list of data items to annotate (i.e., enumerate).

We will also showcase adding of explicit annotation paths, by providing the above *view definition* abilities as a two-step process (for specifying the from and the to “nodes” of the explicitly-defined annotation path).

Finally, we should be able to annotate specific data items directly (which could trigger annotation propagation across pre-established annotation paths).

² The MMS system [7] advocated the use of views for metadata management; our system is targeting annotations (i.e., a special case of metadata), but on the other hand is significantly extending their proposal with additional semantics.

4.2 Visualization

We believe it is absolutely crucial to be able to visualize how the ViP framework works in order to demonstrate it. Towards this, we will produce a server-side visual monitor that will display appropriate statistics for all data items in our system (e.g., number of queries, annotations, and query time). This would allow us to illustrate what is happening upon insertion/deletion of an annotation view or an annotation path or a data item. In addition to illustrating the semantics of the ViP framework, we would also use this server-side visualization to demonstrate the behavior of our caching technique and also the performance of the system (by maintaining appropriate timers).

4.3 Demonstration Scenarios

We will demonstrate many different scenarios with different sequence of operations in the ViP framework in order to highlight their behavior and their performance. Two characteristic examples are as follows.

Annotation Views

- Define an annotation view V (e.g., all experiments performed on June 15, 2008) with a specific annotation tag T and visualize it at the server
- Query an item that belongs to V ; it should have tag T
- Insert a new item D that should match V 's definition and visualize it
- Query item D ; it should also have "received" tag T

Annotation Paths

- Establish an annotation path from V_1 to V_2 (e.g., all experiments performed on June 15, 2008 should be linked to experiments performed on June 16, 2008 because all tissues were transferred) and visualize it at the server
- Query a data item that belongs in V_2 ; it should have no annotations
- Add annotation (e.g., tag TT) to a data item that belongs to V_1
- Query the same data item from V_2 ; it should have "received" tag TT

5 Conclusions

The proposed ViP framework provides a novel view-based annotation propagation scheme. It also brings user-centricity throughout annotation propagation, applied in time and network semantics. ViP utilizes views both as a specification mechanism and as a user-interface mechanism, and employs caching techniques for improved performance compared to the state of the art.

6 Acknowledgements

Research of the project is supported by NIH-NIAID grant NO1-AI50018. We would like to thank Chad Spensky for his help in designing the user interface.

References

1. O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. In *Proc. of the VLDB*, pages 953–964, 2006.
2. D. Bhagwat, L. Chiticariu, W.-C. Tan, and G. Vijayvargiya. An annotation management system for relational databases. In *Proc. of the VLDB*, pages 900–911, 2004.
3. P. Buneman, S. Khanna, K. Tajima, and W.-C. Tan. Archiving scientific data. *ACM Transaction Database System*, 29(1):2–42, 2004.
4. M. Y. Eltabakh, M. Ouzzani, and W. G. Aref. bdbms – a database management system for biological data. In *Proc. of the CIDR*, 2007.
5. F. Geerts, A. Kementsietsidis, and D. Milano. Mondrian: Annotating and querying databases through colors and blocks. In *Proc. of the ICDE*, pages 82–92, 2006.
6. Q. Li, A. Labrinidis, and P. K. Chrysanthis. ViP: a user-centric view-based annotation framework for scientific data. In *the 20th International Conference on Scientific and Statistical Database Management (SSDBM)*, July 2008.
7. D. Srivastava and Y. Velegrakis. Intensional associations between data and metadata. In *Proc. of ACM SIGMOD Conference*, pages 401–412, 2007.