Annotation and More Annotation: Some Problems Posed by (and to) Val Tannen

Peter Buneman
University of Edinburgh, UK

Stijn Vansummeren
UHasselt, Data Science Institute, Belgium

Abstract

Among the many research accomplishments of Val Tannen, his work on provenance and semirings is probably the most widely known. In this paper, we discuss questions that arise when applying this general framework to the setting of curated databases, and in particular the setting where we can have multiple annotations on the same data, as well as annotations on annotations.

2012 ACM Subject Classification Information systems → Database design and models; Theory of computation → Database theory; Theory of computation → Data provenance

Keywords and phrases Annotation, provenance, semiring, curated data

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.4

Funding *Stijn Vansummeren*: Supported by the Bijzonder Onderzoeksfonds (BOF) of Hasselt University (Belgium) under Grant No. BOF20ZAP02.

Acknowledgements Val Tannen of course!

1 Background

By rights, Val Tannen should be a co-author of this short discussion because most of the ideas arose from ongoing conversations with him about the need to put several annotations on the same structure. There are various reasons for wanting to do this, the most important being that it is common in practice to find structures with multiple annotations, and the distinction between annotation and data is not always clear.

What we want to do in this paper is first to compare provenance and annotation: provenance being something that describes, or is intrinsic to, the formation of data; annotation being something that is superimposed on data after its formation. Second, to look at the commonplace practice of having multiple annotations on a structure. How do we represent annotations on annotations, and how well does this fit with the elegant theory of semirings [8] for which Val Tannen is responsible?

Provenance and annotation have been studied together by the database community for 25 years or more [2, 6, 16, 17]. They are obviously connected: provenance is a form of annotation, and provenance may tell us how annotations should propagate through queries. Green, Karvounarakis and Tannen's original work [8] on semirings referred to them as "provenance semirings", other researchers [10, 12] have used the term "annotation semirings". What we claim is that provenance and annotation are also fundamentally different and it is worth examining these differences.

1.1 Provenance

While there is an unending sequence of attempts to define, characterize, formalize and standardize provenance, all of them agree that provenance concerns the properties of the process by which something has been formed and used. Provenance of any kind is hence an

account of the history of an object. As such we do not expect to modify it. Indeed, there are efforts, both in databases [14] , and more generally [15], to ensure that one does not "rewrite history". One might argue that, ideally, everything should carry its provenance, and that provenance – whatever definition one chooses – is an unalterable, intrinsic, property of an object. This does not mean that there has to be a unique model of provenance; but if one has two definitions of provenance there has to be a notion of consistency and a method to combine the two. For instance, in [9] there is a hierarchy of semirings that shows how one semiring may be "more informative" with respect to provenance than another.

1.2 Annotation

Whereas the database community started to worry seriously about provenance and annotation at the start of the millennium, the practice of annotating data was widespread in curated databases at least ten years earlier. In stark contrast to provenance, annotations are regarded as being added to, or superimposed upon, *existing* data. For instance, the widely used Swissprot/Uniprot [1] database is regarded as a *secondary* database built on top of underlying sequence data (the *primary data*). Similarly, geospatial databases may be regarded as annotations on a terrestrial coordinate system.

Informally, annotation has the following properties: annotations are placed on data *after* it has been created; annotations do not "influence" the data they annotate; and if the underlying data changes, annotations may become invalid. Of course, without a proper representation of the notions of time, influence or update in a database, we cannot expect to formulate these conditions. But, in practice they are understood and follow naturally from keeping annotated data in a separate database together with a copy or view of the underlying primary data.

The curators and users of curated databases typically know which parts of the database are imported from the primary data and which fields are added as annotation. The distinction between annotation and the primary data is hence conveyed by some elementary form of provenance although this provenance information is not necessarily explicitly represented in the database.

2 Provenance and Simple Annotation

As a practical illustration of provenance and annotation in a curated database, Table 1 shows the relational schema of the object table of the GtoPdb [13] database. This is a base table and several other tables, all of which describe various substances, implicitly inherit and extend this schema. In this table, the columns object_id, name, abbreviation, and systematic_name form the primary data. The columns last_modified and old_object_id record some rudimentary provenance information. All other columns are annotation, either commenting on the primary data, or (like in_gctp) recording whether the primary data should be exported in a particular view on the database.

What is remarkable about this schema is first that most of it is annotation and second, as we shall see, that most of the annotation fields can usefully be given a semiring structure. It is also interesting that last_modified, a field that we have associated with provenance also has a simple semiring structure.

As we noted above, provenance appears to be a form of annotation. This is in particular true for the last_modified and old_object_id columns in Table 1. Conversely, if as in [2, 9], one wants to study the propagation of annotations through queries, one should understand provenance. If, for example, we want to know whether a tuple in the result of a query should

Table 1 Schema of the GtoPdb [13] object table.

Column	Type
object_id	integer
name	varchar(1000)
abbreviation	varchar(100)
systematic_name	varchar(100)
old_object_id	integer
last_modified	date
comments	text
structural_info_comments	text
annotation_status	integer
only_iuphar	boolean
grac_comments	text
only_grac	boolean
no_contributor_list	boolean
quaternary_structure_comments	text
in_cgtp	boolean
in_gtip	boolean
gtip_comment	text
in_gtmp	boolean
gtmp_comment	text
cite_id	varchar(20)

be included in a view specified by a field in_gctp, then we should be able to use provenance to determine the value of that field. In fact one can use semirings and the semiring semantics of relational algebra to describe the propagation of such fields. If there is only a single annotation column C, and if the values in this column can be ascribed a semiring structure $\mathcal{K}$, such as is for example the case for the in_cgtp column in Table 1 where $\mathcal{K}$ is the Boolean semiring, then we can simply propagate the annotation C by means of the semiring semantics introduced in [8].

▶ Remark. Before continuing our discussion, it is worth noting that we have to be careful in choosing the semiring $\mathcal{K}$ we intend to work in. Indeed, the formal definition of $\mathcal{K}$-relations proposed in [8] interprets all tuples that are annotated with the semiring's additive unit 0 as being absent from the database, i.e., *non-existent*. This need not be the semantics that one wants. For example, in GtoPdb database and assuming that in_cgtp is the only annotation attribute, a value of 0 in in_cgtp does not mean that the tuple is non-existent or should be deleted. Rather, it means that the tuple should not be exported to the gctp view. This difference can be resolved by moving to a semiring other than the Boolean semiring. This alternate semiring has pairs (b_1, b_2) of booleans as elements; where $b_1 = 0$ implies that $b_2 = 0$ and where $b_1 = 0$ indicates that the tuple is really not present. Otherwise, $b_1 = 1$ indicates tuple presence and b_2 indicates whether the tuple is exported to in_gctp. The semiring operations are defined pointwise in the obvious manner. For reasons of parsimony we will ignore this issue and continue to work with the Boolean semirings for fields such as in_gctp.

The GtoPdb object table (Table 1) illustrates the common practice in curated databases of having multiple annotations; and this leads to the following question, central to this paper.

> *How does one* collectively *propagate this multitude of annotations? In particular, can we always cast this as a form of semiring provenance and, if so, what is the semiring structure required? Moreover, how may we implement such propagation?*

$I_1 =$

A	B	...	X	Y	Z
a_1	b_1	...	1	1	0
a_2	b_2	...	0	1	0
a_3	b_3	...	0	0	0
a_4	b_4	...	1	1	1

$I_2 =$

A	B	...	$\mathcal{A}$
a_1	b_1	...	$\{X, Y\}$
a_2	b_2	...	$\{Y\}$
a_3	b_3	...	$\emptyset$
a_4	b_4	...	$\{X, Y, Z\}$

$I_3 =$

A	B	...	$\mathcal{A}'$
a_1	b_1	...	X
a_1	b_1	...	Y
a_2	b_2	...	Y
a_4	b_4	...	X
a_4	b_4	...	Y
a_4	b_4	...	Z

together with the base instance I and $\pi_{AB...} I_3 \subseteq I$

Figure 1 Three ways of annotating with sets.

To give some insights into this question, let us restrict our attention, at least for the purpose of this section, to the setting where there are multiple annotations, but all of these annotations are of the same form. We will return to distinct forms in Section 3.

To start our discussion, we remark that the GtoPdb object table exhibits the following two forms of annotation that commonly arise in curated databases. The first of these we shall call *believers*. This is a catch-all term we will use for people or agents that "approve" of a tuple. For example it is common for curators of a database to review (and possibly correct) parts of a database. In another scenario a database may "export" a number of smaller databases or views, and tuples may be tagged with the set of views that they should be included in. In particular, the in_gctp, in_gtip, and in_gtmp columns in Table 1 represent such tags. Note that while may view a single believer annotation as single Boolean value, the set of multiple believer annotations taken together is a *set-valued* annotation.

A second and ubiquitous form of annotation are *comments*. Again, we may view a single comment as a single string. Multiple comments are commonplace, however, and in practice they are represented by concatenating strings in such a way that the reader can separate them. Hence, a single comment annotation typically encodes a *set* of comment values. Columns like structural_info_comments in Table 1 are clearly *comments* annotations.

Consider the setting where there is only one of these forms of annotation – believers or comments – on the primary data. Figure 1 shows three possible methods of representing such annotated tables. The first, which applies only to believers, adds a new Boolean column for each curator or view to the primary database table. This method is only practical if we may assume that the set of possible curators or views is known and small. If this is not the case, and if one has the luxury of working systems that support nested tables [4] one can add a column that simply contains the set of curators (respectively, views or comments) as shown by I_2. Otherwise (and also common practice) one may add an auxiliary table – with the appropriate foreign key constraint – that specifies a binary relation between the primary data and the believers respectively comments. This third method I_3 is often used in practice when curators or contributors details are also kept in the database.

The first two representation methods I_1 and I_2 naturally lend them to a semiring-based semantics for carrying the annotations through queries. For both believers and comments these semirings are semirings of sets, straightforward but different. In the case of believers

and representation method I_1, the added Boolean columns can be viewed as boolean vectors of a fixed dimension, which clearly form a semiring. If as in I_2 we are using a set-valued column then we want the lattice of subsets of believers as the appropriate semiring. For example, if tuples r, t_1 and t_2 are respectively believed by sets R, T_1 and T_2 then the set of people who believe in the tuple with provenance polynomial $r_1(t_1 + t_2)$ is $R \cap (T_1 \cup T_2)$: we are using the semiring of subsets of some set – a distributive lattice.

On the other hand, if R, T_1 and T_2 are sets of comments, we would expect to see the tuple with polynomial $r_1(t_1 + t_2)$ annotated with $R \cup T_1 \cup T_2$. Here we are using a commutative idempotent monoid – a degenerate semiring in which the two binary operations coincide and the empty set is the multiplicative unit. This is also the semiring used in lineage [6].

By contrast, representation method I_3, while ubiquitous in practice, does not immediately seem to lend itself to a semiring-based semantics. The reason is that the values in the annotation column $\mathcal{A}'$ in the auxiliary table (being the individual believers or comments) does not necessarily have a natural semiring interpretation. For: which individual believer corresponds to the product or sum of two individual believers?

We find it interesting to remark, however, that we may instead view representation method I_3 as a means to *implement* the annotation propagation semantics of method I_2 under the above-described distributive lattice and lineage semirings. Whereas a straightforward implementation of I_2 requires systems that support nested tables (as the annotation column contains a set), method I_3 is able to implement this using ordinary flat relational database management systems. In particular, when we view I_2 as a nested relation then I_3 together with the base instance I is a particular form of a so-called *shredded representation* of I_2 [5,7,11]. It is possible to simulate full nested relational algebra (NRA) by means of ordinary flat relational algebra (RA) on shredded representations [5,7], something that also Val has worked on [11]. Hence, we may indeed view I_3 and particular relational algebra queries on I_3, as a way to simulate semiring-based querying on on I_2.

In full generality, however, shredding requires that RA is endowed with the ability to invent new identifiers. This is required to simulate the creation of new (deeply) nested sets. In the particular case of annotation-propagation that we are interested in here and the specific case where the only NRA operations that we want to do on set-based attributes in I_2 are intersection and union (our semiring operations), it seems that one does not require the ability to invent new identifiers. Indeed, because there is a functional dependency from the non-annotation attributes in I_2 to the set-based annotation attribute $\mathcal{A}$, we may always "identify" a set (including newly created ones) simply by means of the tuple of non-annotation attributes. Moreover, if we only intend to propagate the annotation on I_2 through *positive* relational algebra queries, then it seems that the simulating shredded queries on I_2 will also be positive. A full verification of these conjectures has not appeared in the literature, however, to the best of our knowledge.

A connected interesting and possibly open question is the following: even if from a theoretical point of view we can always "identify" nested sets by means of the tuple of non-annotated values in method I_3, we almost certainly do not want to do this in practice because such tuples can be very wide. Indeed, in the simulating shredded queries we will often have to join using the identifier columns as join fields, and thus the fewer join fields the better. Hence the question: when is it possible to use only a subset of the non-attribute columns as set-identifier keys in the shredded representation? How does this work if we have primary key information on the primary data?

3 Multiple annotations

Beyond illustrating some distinctions between annotation and provenance, the GtoPdb example shows that multiple annotations on a tuple are commonplace. Moreover, where semirings may be associated with annotations, different semirings serve different purposes.

That we have so many annotations on a single tuple in the GtoPdb example is partly due to database design. Take, for example, the view specified by in_gtmp=true. Since object_id is a key one could place the gtmp_comment field in another table with key object_id and the appropriate key inclusion. Multiple uses of this transformation would be confusing to say the least, and the database designers rightly kept all this information in the same place.

If we do keep this information in the same place, we have to allow that two annotations may differ on what tuples are mapped to a semiring zero, so that we have to allow the explicit appearance of zero in an annotation column. As remarked in Section 2, in the familiar representation of a singly-annotated relation as a $\mathcal{K}$-relation, the tuples annotated with a zero are not present.

In Section 1.2 we made the informal observation that if we changed the underlying data then an annotation could become invalid. In the case of a tuple annotation, upon what parts of a tuple does an annotation depend? In particular does one annotation depend on another? In Table 1 it is unlikely that a change to abbreviation could alter the validity of in_gtmp. However the gtmp_comment field is only going to be present if in_gtmp is true. We can informally define the *scope* of an annotation as that part of a tuple (both non-annotation columns and annotation columns) it annotates and say that two annotations are *independent* if neither is in the scope of the other.

For independent annotations there is an obvious method of treating them as a single annotation. For example, if we have a semiring $\mathcal{K}_1$ for believers and $\mathcal{K}_2$ for comments then there is an obvious product semiring defined on $\mathcal{K}_1 \times \mathcal{K}_2$ with all operations defined pointwise. We have seen this before: in the column-based representation of I_1 in figure 1, the annotation columns X and Y taken together can be regarded as the product of two Boolean-valued annotations.

A more interesting problem arises when the annotations are *not* independent. That is when one annotation depends on, or is within the scope of, another. This is also common in curated databases, and happens with both believer and comments annotations. For example, it is possible for one curator to verify or check the work of another, so "A verifies B's verification of ..." presupposes that B verified something. It can also happen when one view is a sub-view of another, and it is possible to provide comments on top of comments, or on top of verifications.[1]

For such correlated annotations, it is not immediate how we may view them as semiring elements. In fact, it is not *a priori* clear how representation methods I_1 and I_2 of Figure 1 extend to this setting. By contrast, for method I_3, the shredded representation of the annotation, the extension is straightforward: starting with schema R we want to add two annotation attributes $\mathcal{A}$ and $\mathcal{A}'$. We construct the instances I, $I_{\mathcal{A}}$ and $I_{\mathcal{A}\mathcal{A}'}$ over the schemas R, $R \cup \{\mathcal{A}\}$ and $R \cup \{\mathcal{A}, \mathcal{A}'\}$ that satisfy the inclusions $\pi_R I_{\mathcal{A}} \subseteq I$ and $\pi_{R\cup\{\mathcal{A}'\}} I_{\mathcal{A}\mathcal{A}'} \subseteq I_{\mathcal{A}}$. The following questions now arise: can we still view this as a shredded representation of generalized version of I_2? If so, can we view the annotation attribute of generalized I_2 as having a semiring structure? In discussions that we have had with Val on this topic, it seems that a semiring structure exists, but that the elements in the domain of this semiring must be themselves $\mathcal{K}$-relations. Full answers to these questions are still open.

[1] Somewhat confusingly "A believes that B believes ..." does not imply that "B believes ..." so we really cannot regard this as an annotation on an annotation.

The comment semiring can be treated similarly. The generalization of I_3 described above works equally well for comments on comments except that we are using the degenerate semiring with only union. What is interesting is that it is common to build up chains of comments of indefinite length. This can also be easily represented in the semiring framework. If C is the domain of comments and $C*$ is the set of sequences over C then the monoid we need is the prefix-closed subsets of $C*$, which is closed under union.

4 Closing and further questions

The foregoing account of annotation in curated databases is not just a theoretical exercise. As we have already mentioned, the base "object" table in GtoPdb [13] has six Boolean fields for views and six comment fields each of which pertains to one of those fields. Tens of other tables inherit from this table; moreover many other tables in the database have a comment field. Understanding which comments are relevant to a tuple in the output of a query is a non-trivial task in practice. Another interesting field that can be regarded as an annotation, but also part of the provenance is a "time of last modification" – which may be represented in some semiring.

As our discussion illustrates, it is not always clear how to cast the propagation of annotation as an application of querying under the semiring semantics. In particular, ensuring that multiple annotations can be suitably propagated seems to require us to form semirings of increasingly rich structure: semirings of sets, product semirings, and (in Section 3), semirings of $\mathcal{K}$-relations. Further research is required to complete this picture of multiple annotation propagation and the induced semiring structure.

We close our discussion with two further topics.

We introduced the notion of the scope – the set of attributes in a tuple that is subject to some annotation. Do the rules for combining annotations still hold? For example, if a projection loses some or all the attributes in the scope of an annotation, do we want to keep that annotation?

We also defined scope informally in terms of update to a tuple. But we have no model for update, and this brings up the question of how annotation behaves under update. Provenance and update have been studied in [3], but does this tell us anything about annotation; and is there a larger picture that includes semirings?

Finally, although we have suggested that the scope of an annotation could be part of a tuple, why could it not be even broader. Could we annotate data values, tables, columns, as well as arbitrary views?

References

1 Amos Bairoch and Brigitte Boeckmann. The swiss-prot protein sequence data bank. *Nucleic acids research*, 19(Suppl):2247, 1991.

2 Deepavali Bhagwat, Laura Chiticariu, Wang-Chiew Tan, and Gaurav Vijayvargiya. An annotation management system for relational databases. *The VLDB Journal*, 14:373–396, 2005.

3 Peter Buneman, James Cheney, and Stijn Vansummeren. On the expressiveness of implicit provenance in query and update languages. *ACM Transactions on Database Systems (TODS)*, 33(4):1–47, 2008.

4 Peter Buneman, Shamim A. Naqvi, Val Tannen, and Limsoon Wong. Principles of programming with complex objects and collection types. *Theor. Comput. Sci.*, 149(1):3–48, 1995. doi:10.1016/0304-3975(95)00024-Q.

5 James Cheney, Sam Lindley, and Philip Wadler. Query shredding: efficient relational evaluation of queries over nested multisets. In Curtis E. Dyreson, Feifei Li, and M. Tamer Özsu, editors, *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 1027–1038. ACM, 2014. doi:10.1145/2588555.2612186.

6 Yingwei Cui and Jennifer Widom. Practical lineage tracing in data warehouses. In *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*, pages 367–378. IEEE, 2000.

7 Jan Van den Bussche. Simulation of the nested relational algebra by the flat relational algebra, with an application to the complexity of evaluating powerset algebra expressions. *Theor. Comput. Sci.*, 254(1-2):363–377, 2001. doi:10.1016/S0304-3975(99)00301-1.

8 Todd J Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 31–40, 2007.

9 Todd J Green and Val Tannen. The semiring framework for database provenance. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 93–99, 2017.

10 Grigoris Karvounarakis and Todd J Green. Semiring-annotated data: queries and provenance? *ACM SIGMOD Record*, 41(3):5–14, 2012.

11 Christoph Koch, Daniel Lupei, and Val Tannen. Incremental view maintenance for collection programming. In Tova Milo and Wang-Chiew Tan, editors, *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 75–90. ACM, 2016. doi:10.1145/2902251.2902286.

12 Egor V Kostylev, Juan L Reutter, and András Z Salamon. Classification of annotation semirings over containment of conjunctive queries. *ACM Transactions on Database Systems (TODS)*, 39(1):1–39, 2014.

13 Adam J Pawson, Joanna L Sharman, Helen E Benson, Elena Faccenda, Stephen PH Alexander, O Peter Buneman, Anthony P Davenport, John C McGrath, John A Peters, Christopher Southan, et al. The iuphar/bps guide to pharmacology: an expert-driven knowledgebase of drug targets and their ligands. *Nucleic acids research*, 42(D1):D1098–D1106, 2014.

14 Pingcheng Ruan, Gang Chen, Tien Tuan Anh Dinh, Qian Lin, Beng Chin Ooi, and Meihui Zhang. Fine-grained, secure and efficient data provenance on blockchain systems. *Proceedings of the VLDB Endowment*, 12(9):975–988, 2019.

15 Deepak K Tosh, Sachin Shetty, Xueping Liang, Charles Kamhoua, and Laurent Njilla. Consensus protocols for blockchain-based data provenance: Challenges and opportunities. In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 469–474. IEEE, 2017.

16 Y. Richard Wang and Stuart E. Madnick. A polygen model for heterogeneous database systems: The source tagging perspective. In *Proceedings of the 16th International Conference on Very Large Data Bases*, VLDB '90, pages 519–538, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

17 Allison Woodruff and Michael Stonebraker. Supporting fine-grained data lineage in a database visualization environment. In *Proceedings 13th International Conference on Data Engineering*, pages 91–102. IEEE, 1997.