Data Provenance: What next?

Citation for published version:
Buneman, P & Tan, W-C 2019, 'Data Provenance: What next?', ACM SIGMOD Record, vol. 47, no. 3, pp. 5-16. https://doi.org/10.1145/3316416.3316418

Digital Object Identifier (DOI):
10.1145/3316416.3316418

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
ACM SIGMOD Record

Data Provenance: What next?

Peter Buneman
University of Edinburgh
opb@inf.ed.ad.uk

Wang-Chiew Tan
Megagon Labs
wangchiew@megagon.ai

ABSTRACT

Research into data provenance has been active for almost twenty years. What has it delivered and where will it go next? What practical impact has it had and what might it have? We provide speculative answers to these questions which may be somewhat biased by our initial motivation for studying the topic: the need for provenance information in curated databases. Such databases involve extensive human interaction with data; and we argue that the need continues in other forms of human interaction such as those that take place in social media.

1. INTRODUCTION

The purpose of this paper is neither to define provenance nor to provide a survey of the relevant research; there are numerous contributions to the literature that do this [19, 18, 25, 45, 49, 71, 28]. What we hope to do here is to draw out new strands of research and to indicate what we can do practically on the basis of what we now know about provenance. A good starting point is to state two generally held but conflicting observations: first that the more provenance information one can collect the better; second that it is impossible in practice to record all relevant provenance information.

Before narrowing our discussion to data provenance, let us look at these two observations. Imagining the impossible, suppose we could record all the provenance associated with some process or artefact (digital or otherwise). In what would be a massive amount of provenance data, would we be able to answer simple questions such as where some data was copied from or whether a process invoked a particular piece of software? Such questions may involve the querying of huge data sets and complex code. So simply recording total provenance, even if it were possible, still requires complex analysis. It requires us to extract simple explanations from a massive and complex structure of data and code. What are those explanations?

Being more realistic, in practice, we only have resources to record a limited amount of provenance information. So what do we record? We may – as is the case with physical artefacts – have some standard attributes (ownership, location etc.) but for computational processes and data can we predict what will be asked of provenance? Again we have to understand what kinds of explanation we are likely to want. Is there any minimal requirement on what we should record or how we should record it?

For most purposes, what we should record is application dependent. For example, if an application is targeting to answer the provenance of a sales figure reported in a company earnings report, then the data provenance that consists of the source data and the program or query that was used to generate the report are likely to be sufficient. However, sometimes, intermediate sales results from specific regions are combined with other data sources or results from other regions to generate the final report. In this case, to provide a comprehensive understanding of the sales figure in the company earnings report, it may also be necessary to track the programs that were used to generate the intermediate results.

In yet another type of application, it is important that the results are repeatable and reproducible. This is true of experiments in chemistry and physics where it is not only crucial that one can obtain the same results by re-running the experiments but also by running it at other locations. Software repeatability and reproducibility have also become an important topic. To enable software reproducibility, it is typically necessary to document the hardware, the version of operating system and software libraries used, in addition to the program and data used to execute the experiment.

Insofar as data provenance is separable from other forms of provenance [8, 26] we focus on provenance that has to do with data: databases, data sets, file systems etc. In the Background section that follows, we summarise some of the important research contributions to data provenance, the motivation behind the research and the practical applications of it. In Section 3, we then look at possible applications of provenance in other areas of computer science.

2. BACKGROUND

As often happens, the first paper that addressed provenance [78] in databases had to be "rediscovered" several years after it was written. This paper introduced a form of tagging or annotation to describe the source of elements of a relational database, a form of *where*-provenance. Then in the later 1990s under various names the study started in earnest. In [79] a method based on inverse functions was used to visualize the *lineage* of data in scientific programming; and in [21, 22], in the context of data warehouses, an operational definition was given of what tuples in some source data "contributed to" a tuple in the output of a relational query – perhaps a form of *why*-provenance.

The authors' interest in the topic was sparked by their collaboration with biologists [44] involved in the Human Genome Project who were building *curated* databases of molecular sequence data. While a curated database resembles a data warehouse in the integration of existing databases, it also involves the manual correction and augmentation of the source data, and it cannot simply be characterized as a data warehouse or view. The biologists complained that they were losing track of where their data had come from. Now biologists are, by training, quite meticulous in keeping a record of what they have done – in this case what queries they have made or what manual additions or corrections were made, so in some sense the provenance of some small element of data – a number or a tuple – was available. However extracting the information they needed from a complex workflow of updates and queries on other databases was proving difficult. What they appeared to need was a simple explanation e.g.: "this number was entered by ... on ..." (where-provenance); or "this tuple was formed by joining tuple t_1 from R_1 to tuple t_2 from R_2" (how-provenance); or "this tuple is in the result because some other tuple was in the input" (why-provenance).

The example in Figure 1 illustrates the types of provenance described above. Consider a `Friend` relation, a `Profile` relation, and a query that joins the two relations to find pairs of friends with identical occupations (shown below).

```
select f.name1, f.name2
from   Friend f, Profile p1, Profile p2
where  f.name1 = p1.name and
       f.name2 = p2.name and
       p1.occupation = p2.occupation
```

The value "Carl" in the result is derived from the value "Carl" in the `Friend` relation. Hence, if there were an annotation on who entered that information and when, this information can propagate to the result according to where-provenance. The figure also illustrates that the how-provenance of the output tuple is the result of joining three tuples (Carl, Bob), (Bob, 30, analyst), and (Carl, 50, analyst) from the input. The why-provenance of the output consists of the same three source tuples. We will discuss the finer differences between the latter two types of provenance in Section 2.1. However, it is important to note that to fully explain why the output tuple exists, one must also account for the query. That is, these three tuples satisfy all the equality condition in the where clause of the query.

What we should again emphasize is that the purpose of data provenance is to extract relatively simple explanations for the existence of some piece of data from some complex workflow of data manipulation. In this sense it has a similar purpose to *program slicing* which seeks to provide an explanation for a part of the output of some complex program to a small part of the input – an explanation that is much simpler than the program itself.

Given that provenance is about explanation of some part of a complex process, it is natural to ask whether there is a unified language or model for describing provenance. PROV is a W3C recommendation for a model or ontology in which one can describe provenance [60, 58]. The intention is to produce a general model for any kind of provenance such as that associated with artefacts or some general computational process. At its core, PROV can be used to describe causal relationships between entities and activities, and in doing this can naturally describe the evaluation of a workflow. Because of this the term "workflow provenance" has sometimes been used to distinguish the ambit of PROV from that of data provenance. Worse, the terms "fine-grained" and "coarse-grained" have been used for this distinction. We do not believe these distinctions to be helpful. While it is straightforward to use PROV to describe some basic aspects of data provenance, we do not do so in this paper because it does not add much to the formalisms that have been found useful in the context of databases. Conversely, there is no reason why the formalisms developed for "fine-grained" data manipulation cannot be used in a larger context as we shall see in Section 3.1.

2.1 Annotation and provenance

From the beginning it was recognized that provenance should be expressed as a form of annotation. This was precisely the purpose of the Polygen model [78]: to annotate data elements with their provenance. However, there is a much more fundamental connection between the two topics, which again shows up in curated databases. Much of curated data is about *annotation* of existing data structures. Sometimes this annotation is expressed in the primary tables in a relational database, but sometimes important information about the currency or validity of some data is held in an auxiliary table or –

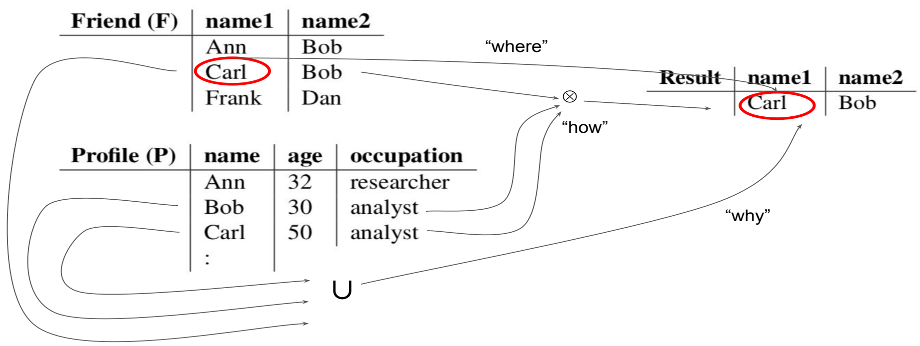


Figure 1: An illustration of where, how, and why-provenance.

in the case of semistructured data – some additional subtrees in a hierarchy or some additional edges in a graph representation. In fact, annotation data is semistructured by nature and often lives in some kind of auxiliary database. Queries over the "core" data often do not recognize this annotation, and this is one of the main sources of misleading or dirty data in both data warehouses and curated databases.

The basic question is then how do annotations propagate through database queries? This is a question that is closely related to data provenance and one that has driven much of the most interesting research on data provenance since its inception.

Annotation. The Polygen model [78] inspired the subsequent system DBNotes [6, 20] and other following work (e.g., [35, 10]). For each relational algebra operator, DBNotes provided a rule to propagate annotations based on where data is copied from. These rules are sensitive to the way the query is formulated: even though two queries are equivalent in the normal sense of always producing the same result the way the rules propagate annotations through the two queries may differ. Another propagation scheme that is agnostic to the way equivalent queries are formulated was also proposed to propagate the same annotations to the result.

That provenance may be sensitive to query formulation is seen in [10] which discusses update languages and uses a propagation scheme that is an extension of that in DBNotes. From a theoretical perspective, relational update languages, such as the update fragment of SQL, are often regarded as uninteresting because they are no more expressive than query languages. Consider the action of an SQL update: it replaces a version of the database with a new version. If we think of the old version as the input and the new version of the output, then that transformation from input to output can be expressed as a query in relational algebra. For example, Figure 2.1 shows a simple update query and an equivalent – in the sense that it produces the same output – query that doesn't involve updates. The backwards arrows show where all components of the table, values tuples and the table itself, come from. While the two queries produce the same answer, the provenance is different. The first update query only affects the where-provenance of the cell that the number "5" belongs to in the output. All the other components of the result table "come from" the corresponding component of the input table. On the other hand the more complicated query not only creates a new value 5, but a new tuple containing that value and a new table. In the figure the components that are created by the query are outlined in dotted red; the components that are copied are outlined on black.

The interesting observation is that if we take provenance into account, that is the query or update is a function that not only produces a result but also produces *where*-provenance associated with the values and tuples in a table, update languages become *more* expressive than query languages. Moreover [10] provides a completeness result: if the where-provenance can be expressed in (nested) relational algebra, then there is an update query in which the same where-provenance is implicit.

Semiring provenance The seminal work of [40] describes a formalism of data provenance that captures and extends previous formalisms such as why-provenance of [14] and lineage described in the Trio system [5].

A *commutative semiring* is a quintuple $(K, \mathbb{0}, \mathbb{1}, \oplus, \otimes)$. Here, K is a set of elements containing the distinguished elements $\mathbb{0}$ and $\mathbb{1}$, $\oplus$ and $\otimes$ are two binary operators that are both commutative and associative and $\mathbb{0}$ and $\mathbb{1}$ are the identities of $\oplus$ and $\otimes$ respectively. In addition, $\otimes$ is distributive over $\oplus$ and $\mathbb{0} \otimes t = t \otimes \mathbb{0} = \mathbb{0}$.

We assume that every tuple in the source database has a tuple identifier, and $I \subseteq K$ is the set of all such source tuple identifiers. The provenance of an element in an output table is expressed as a polynomial, an expression built up from $I, \mathbb{0}, \mathbb{1}, \oplus$ and $\otimes$. The provenance of an output tuple for each relational operator (select, project, cross product, union, rename) is obtained from the provenance polynomial of each input tuple. The simplest case is selection in which the provenance of an out-

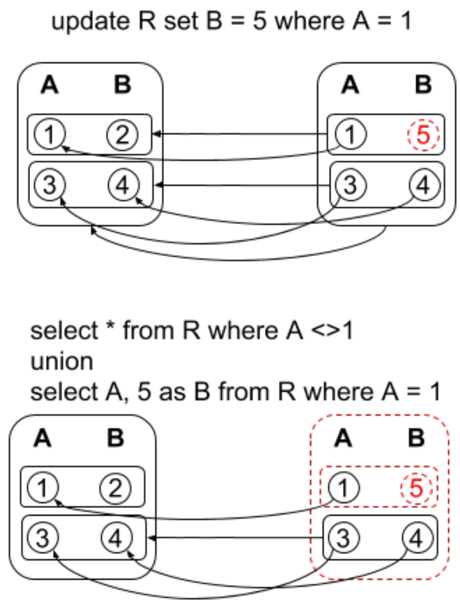


Figure 2: How updates affect provenance

put tuple is the same as the provenance of the (unique) corresponding input tuple. For join, suppose that $t_1 \in R_1$ and $t_2 \in R_2$ combine to produce $t \in R_1 \times R_2$. If e_1, e_2 are the provenance polynomials of t_1, t_2 then the polynomial for t is the polynomial $e_1 \otimes e_2$. For union, if $t \in R_1$ has provenance e_1 and the same tuple $t \in R_2$ has provenance e_2 then the provenance of t in $R_1 \cup R_2$ is the polynomial $e_1 \oplus e_2$. For a tuple t in the output of a projection, the provenance is the polynomial $e_1 \oplus \ldots \oplus e_n$ where $e_1, \ldots, e_n$ are the polynomials of the tuples in the input that "project onto" t. The polynomials attached to the tuples in the output of a query are built up inductively by these rules and others described in [40]. We can think of the polynomial as a description of *how* each tuple was constructed – by "joining" ($\otimes$) and "merging" ($\oplus$) other tuples.

The example below shows a query in SQL over the `Friend` relation of Figure 1. The query finds all people who share a friend with someone. In some sense the query is trivial because everyone shares a friend with themselves, however the provenance is interesting.

Query:

```
select f1.name1
from   Friend f1, Friend f2
where  f1.name2 = f2.name2
```

Assume that the tuples (Ann, Bob), (Carl, Bob), and (Frank, Dan) are annotated with i_1, i_2, and respectively, i_3. The result of the query is shown below alongside with annotations of the corresponding provenance polynomials and why-provenance.

name1	*provenance*	*why-provenance*
Ann	$i_1 \otimes i_1 \oplus i_1 \otimes i_2$	$\{\{i_1\}, \{i_1, i_2\}\}$
Carl	$i_2 \otimes i_2 \oplus i_1 \otimes i_2$	$\{\{i_2\}, \{i_1, i_2\}\}$
Frank	$i_3 \otimes i_3$	$\{\{i_3\}\}$

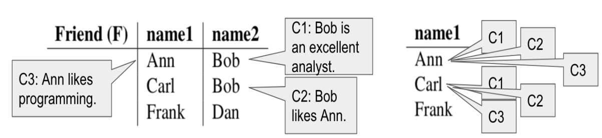
For example, the provenance polynomial for Ann is $i_1 \otimes i_1 \oplus i_1 \otimes i_2$ showing that i_1 and i_1 itself is one way of deriving the output tuple and another uses i_1 and i_2.

The remarkable property of these polynomials is that they unify many other generalizations of relational algebra such as bag semantics, C-tables and probabilistic databases. For bag semantics simply assign the "identifier" 1 to each tuple in the input and use the semiring $(\mathbb{N}, 0, 1, +, \times)$. The evaluation of the polynomial attached to a tuple gives the multiplicity of that tuple.

These polynomials also capture why-provenance with the semiring $(\text{Why}(K), \emptyset, \{\emptyset\}, \cup, \uplus)$, where $x \uplus y$ denotes the pairwise union of all sets in the two collections x and y. The evaluation of the provenance polynomial will give rise to the set of sets shown on the rightmost column above. Indeed, if we interpret each tuple identifier as a set of a singleton set, then the provenance polynomial of Ann $i_1 \otimes i_1 \oplus i_1 \otimes i_2$ is $\{\{i_1\}\} \otimes \{\{i_1\}\} \oplus \{\{i_1\}\} \otimes \{\{i_2\}\}$ which is $\{\{i_1\}\} \oplus \{\{i_1, i_2\}\}$ and gives rise to the why-provenance $\{\{i_1\}, \{i_1, i_2\}\}$.

Observe that the why-provenance describes what tuples in the source are *sufficient* for deriving the output according to the query. Indeed, either i_1 alone or both i_1 and i_2 are sufficient for generating the output tuple Ann according to the query. It is easy to see that the why-provenance can be derived from the provenance polynomial but not the other way round; the provenance polynomial is more informative.

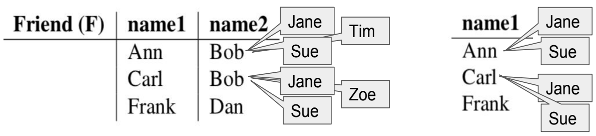
Semirings for propagating comments or beliefs can also be derived from the semiring framework. For example, the semiring $(\text{Lin}(K), \perp, \emptyset, \cup, \sqcup)$ which captures the lineage described in [22] can also be used to model how comments should propagate. Intuitively, the element $\perp$ denotes no lineage while $\emptyset$ denotes empty lineage, and $\sqcup$ is the usual union operator $\cup$ except that $\perp \sqcup X = X \sqcup \perp = \perp$.



The figure above exemplifies the "comments" semiring. The first source tuple (Ann, Bob) has two comments $C1$ and $C3$ and the second source tuple (Carl, Bob) has a single comment $C2$. Each of the first two tuples has all three comments in the result.

On the other hand, the belief of an output tuple can be captured with the following semiring $(\text{Belief}(K), \perp, \emptyset, \cup, \cap)$

which takes the intersection of the beliefs of the source tuples on a relational join.



Hence, {Jane, Sue} are the only remaining believers after the relational join operation.

Today, several database systems have been developed to support the propagation and querying of provenance such as Perm [37], LogicBlox, and Orchestra [39]. More recent implementations such as [2] provides a provenance-aware middleware implementation which can be used with different database back-ends and also supports *provenance for transactions*. Provenance support has also been implemented outside database systems. For example, in network provenance [82, 81], provenance is maintained and queryable at Internet-scale for diagnosing network errors in a distributed setting.

2.2 Provenance, repeatability, versioning

The ability to reproduce an experiment is essential to the credibility of the results of that experiment. The same is true for any kind of computational analysis or workflow that has been used to derive some data: the analysis must be repeatable. Whatever is needed to ensure repeatability is often regarded as provenance. The ability to record, reproduce, and query some computational process underlies "system-level" provenance [62], the provenance "challenge" [61] and at least one view of data citation [66]. Now almost all such analyses use some kind of external data source – this is obvious in the case of data citation, where the data source is the source being cited. The problem in all these cases is that the data source, and even its structure, is likely to evolve over time.

In curated databases we see a similar problem. When external data is incorporated, it is common to provide a link to source data as part of the provenance. While this requirement seems rather straightforward, there are at least two caveats to ensure a proper "implementation" that meets this requirement. First, the link should be a stable reference to the correct version of the database even if the database evolves. Most curated databases have a link which serves as a citation to its entire database. Web pages follow a similar organization where its URL refers to the latest version of the web page. When the database changes, the new database replaces the old database and hence, the link, which now refers to the new database, is no longer a valid reference for the previous database. The second issue is that the link is typically a coarse-grain approximation to a specific part of the database where the reference is typically intended for. While the HTML structure of web pages can be exploited to pinpoint to specific portions of the website, it is less obvious how specific portions of a database can be precisely referenced.

Data Versioning To ensure proper citation, some curated databases simply keep all past versions of the data. The onus is on the user to cite the correct (portions of the) version and to answer queries over multiple versions of data. For example, longitudinal queries such as "*what are all the changes in the last five versions?*", or "*when was this entry made?*" would be difficult to answer without going through each of the relevant database versions at least once.

Another approach, which is more economical on storage, stores only the changes (or deltas) between consecutive versions. However, the need to go through every relevant version for certain types of longitudinal queries such as "*return all versions where a particular entry exists*" is still unavoidable.

The archiving method of [13] strikes a balance between the two approaches described above; it keeps all database versions intact and economically by "merging", to the extent possible, different database versions together. Conceptually, every version is assumed to be in a hierarchical format such as in a JSON file format or XML. Every node has an associated set of intervals which captures the versions by which the node exists (the fat node method of persistent data structures [32]). Furthermore, if, as frequently happens, a node's interval set is identical to that of its parent one can save storage by taking the lack of an interval set to indicate that the interval set should be inherited. For biological databases such as those described in [13], it was observed that the dominant change is the addition of a node in the hierarchy, and that node modifications are relatively infrequent. This allows significant space savings, and a year's history of a database typically requires only a small percentage overhead in storage.

The main challenge with the archiving strategy is that it is not obvious how to match and merge nodes of a version into nodes of an existing database archive. In [13], a critical assumption is that there are keys for nodes in a hierarchical structure [12]. The keys are paths of labels or values and identify nodes in a version. Hence, they also help identify which nodes in the database archive to match and merge into. If a node in the version does not exist in the database archive, then it is a node that is new to the version and will be created as a new node in the database archive with a new interval. Conversely, if a node in the database archive has no corresponding node in the database version, then that node no longer exists and its interval of versions is terminated accordingly. Otherwise, the node is merged into the node in the database archive and its interval of versions is extended,

Versions of data

Day 1

name	gender	age	primary interest
Anna	F	35	swimming
Bob	M	28	weight lifting

Day 24

name	gender	age	primary interest
Anna	F	35	swimming
Bob	M	28	weight lifting
Carol	F	20	kickboxing

Day 40

name	gender	age	primary interest
Anna	F	35	swimming
Carol	F	20	kickboxing

Day 44

name	gender	age	primary interest
Anna	F	35	swimming
Carol	F	20	weight lifting
Dan	M	38	cardio

Storing only the changes

Day 1

name	gender	age	primary interest
Anna	F	35	swimming
Bob	M	28	weight lifting

Day 24

name	gender	age	primary interest
Carol	F	20	kickboxing

Day 40

name	gender	age	primary interest
Bob	M	28	weight lifting

Day 45

name	gender	age	primary interest
Carol	F	20	***weight lifting***
Dan	M	38	cardio

Archiving the data

Day 1

[1-today]

	name	gender	age	primary interest
	Anna	F	35	swimming
	Bob	M	28	weight lifting

Day 24

[1-today]

	name	gender	age	primary interest
	Anna	F	35	swimming
	Bob	M	28	weight lifting
[24-today]	Carol	F	20	kickboxing

Day 40

[1-today]

	name	gender	age	primary interest
	Anna	F	35	swimming
[1-39]	Bob	M	28	weight lifting
[24-today]	Carol	F	20	kickboxing

Day 45

[1-today]

	name	gender	age	primary interest
	Anna	F	35	swimming
[1-39]	Bob	M	28	weight lifting
[24-today]	Carol	F	20	[24-44] Kickboxing [45-today] weight lifting
[45-today]	Dan	M	38	cardio

Figure 3: Three approaches to keeping all versions of data. Added tuples are shown in green, deleted tuples in red and modified values in orange.

denoting that the node continues to exist in the archive.

The assumption of a hierarchical key structure is reasonable for many curated databases and for scientific data formats [13]. Moreover the same technique can be applied to relational databases either by casting relations in a hierarchical format or performing archiving in the database engine by adding an interval to each tuple of each column of the relation schema to model the interval of versions. Figure 3 shows the three approaches in the relational context.

More recent work has directly tackled the problem of versioning relational databases [46, 57]. For example, [57] is a version-oriented storage engine designed from scratch to support versioning while [46] adds a versioning module on top of a relational database system. The latter architecture allows one to continue to exploit the advanced querying capabilities provided by a relational database system while adding efficient versioning capability to the system.

There is also a large body of work on temporal databases, which also support versioning as a special case. See [50] for a summary. In most versioning work, the notion of when a tuple has changed coincides with when the change is recorded in the database. Bi-temporal databases distinguishes these two types of time; transaction time and valid time. *Transaction time* denotes the time at which updates are applied to the database (hence, they can only "increase") which may be different from when the tuple is actually valid in the real world (valid time). In temporal databases, much of the effort is dedicated to managing and querying [52] these two notions of time efficiently.

Most versioning work and temporal databases has focused on recording data changes and there are relatively little that directly tackles the problem of managing both data and schema changes [68, 59, 23]. When the schema changes, can we easily query which data has changed (or not) across different versions? Can we effectively answer longitudinal queries across the versions? Can we seamlessly answer and even visualize the provenance of data that may consist of tuples from different versions, which may in turn be the result of another query on a database and so on?

3. WHAT IS NEXT?

So far, we have described, and asked questions about, existing work on data provenance that was largely motivated by curated databases. Next, we look at potential applications of provenance in data citation and in other areas of computer science, such as machine learning, social media, blockchain technology and privacy.

3.1 Provenance and data citation

Because so much knowledge is now disseminated through some form of database, there has been an increasing demand [34, 67] for these databases to be properly cited for the same reasons that we use citations for conventional publications. There is a problem in that data of interest is usually extracted from the database by some form of query. What citation should one associate with the query or with the results of the query? There have been two general approaches to this. One is to treat citation and provenance as synonymous. To this end [66] have developed a system that carefully records what one might call the complete provenance associated with the evaluation of a database query. In particular they want to guarantee that the evaluation of the query is reproducible at a later stage. Critical to their approach is some form of database archiving of the kind we described in the previous section.

In contrast [11] have taken citation to mean the extraction of "snippets" of information, such as authorship, title, date etc. that one sees in a conventional citation. In fact [24] has a specification of the snippets that are appropriate for data. The problem is particularly interesting for curated databases which closely resemble, and often replace, conventional publications. In curated databases, there may be hundreds of "authors" who have contributed data. How does one extract the authors appropriate to the result of a specific query? [11] propose that by associating views with (groups of) authors, one can solve this problem using the well studied techniques of rewriting through views [30, 43, 54]. Conventional citations are, of course, a rather weak form of provenance, but techniques from the study of data provenance are nevertheless useful. [27] gives an interesting application of semiring provenance to generate and combine appropriate citations from views.

3.2 Provenance and machine learning

Machine learning and artificial intelligence have become an indispensable part in our daily lives. Machine learning methods are commonly used to automate everyday decision making in all aspects of our lives; from predicting email spams [42] to predicting crop yields [76], loan application, autonomous driving [17], disease identification and recommendation of medical treatments [51]. Even if machine learning models perform very well in practice, it is natural to question why a certain decision or prediction has been made, especially when decisions are critical. Explanations of a model's output can help build further trust in the system's performance and understand the foundations by which a decision has been made.

In machine learning research, the problem of deriving explanations of machine learning models is called *interpretability*. Somewhat ironically, there is less consensus on what the exact interpretation of *interpretability* [31, 64] should be. However, the reason for the lack of consensus should not be surprising. Like the situation in provenance, different users have different requirements of interpretability. For example, the requirements for interpretability so that a programmer can debug the model is quite different from interpretability of the predication of a crop yield. In the latter, one may only need to explain that it is because the estimated rainfall is high/low but in the former, one may need to understand how many rounds of simulation have been applied, the parameters and software modules used.

While some models lend themselves well to some form of interpretability (e.g., generalized additive models [15]), other models, especially neural networks, are opaque. An approach to overcome the opaque nature of neural networks is to learn another less opaque model based on the predictions of the original model.

The goals of data provenance and interpretability are clearly similar. Both seek to find explanations, at different levels of granularity, for the output of a program or a process. A major difference is that in database provenance, the program and process that have been considered by researchers are typically not opaque as in machine learning models.

A promising area of cross-fertilization between provenance and interpretability is the following: Instead of learning models that are interpretable based on the predictions of the original model, one can learn rules or program (in some language) that can approximate a machine learning model or special cases of it. The problem of deriving rules from the model predictions is closely related to the problem of reverse engineering queries, which is to derive the specification from known behaviors such as known input and output mappings (e.g., [7, 75, 48] to name some recent work). These rules can be further abstracted to provide human friendly explanations for the model [74]. Interestingly, the process of reverse engineering often involves developing a machine learning model to learn a query for the given input and output data, which itself may require explanations.

3.3 Provenance and social media

Social media, such as Facebook, Instagram, Twitter etc., are an effective vehicle for disseminating news at scale. They provide an easy platform for users to continuously communicate and network with one another. The continuity and scale are critical characteristics that set it apart from traditional forms of communications such as phones, television, or newspapers. Unfortunately, its effectiveness for disseminating information has also been exploited for disseminating fake news and fake claims.

There has been substantial interest lately in how to detect fake news articles or fake claims (e.g., see [1, 4, 16, 47, 80, 77]), and having adequate provenance is seen as an essential part of this process. We discuss some potential directions for further work and argue that building a mechanism for understanding the provenance of news obtained through the social network is an important part of determining fake news or fake claims.

As with data provenance, the provenance of a piece of information found in an article or statement in social media should explain why that information is there and how it was created. One method of achieving this is to ensure that provenance is disseminated along with news propagation. We should also discredit news without mechanisms for authenticating its provenance. When an article is first created, it should include information such as the authorship and attribution to sources. The social network software responsible for disseminating the news should add the identity of the receiver into the chain of provenance information. Furthermore, there should be tamper-proof mechanisms built into the software to prevent the identity from being modified.

If provenance information may not be immediately available from an article, can we infer the provenance with social media network? For example, [72] identifies the source of rumor when all recipients are known (*rumor-centrality*). In [53], the *effectors* are determined under the independent cascade propagation model and in [65], the NetSleuth approach [65] estimates the sources under the assumption of the Susceptible Infected information propagation model. As shown in [33, 41], some provenance attributes can also be recovered from various social media websites and can lead to better knowledge of the sources.

A promising area for further research is to incorporate provenance into the *fact checking* problem. Fact checking originated from the data journalism community and refers to the problem of determining whether or not factual claims in media content are true. Today, there are websites[1] dedicated to analyzing and reasoning about facts. Google also supports an API for reviewing claims[2]. Note that whether a fact is true or not is actually independent of its provenance. However, since a trusted source tends to produce articles that are free of wrong facts, a property for judging whether a claimed fact is true or not can be based on the trustworthiness of the sources. In turn, this requires knowledge of the provenance attributes of these sources. Can we use provenance to as a reliable signal for determining whether a fact is true or not? Some recent work has begun to incorporate such information in determining the truth of news/facts [69]. Another promising direction is to incorporate trust and reputation management into social media. Can we maintain a reputation rating for different sources based on their history of the authenticity of news articles and correct facts that are wrongly reported and shared. In turn, these reputation ratings can be used as another signal for fact checking and checking for fake news [29]. Regardless of the method used to determine sources of fake news or fake claims, it is crucial that provenance about the sources can be obtained or inferred. It is also critical to create standards to institute a minimum set of attributes that should be provided before an author can publish or responsibly propagate an article on any social media platform.

[1] https://www.factcheck.org, https://www.truthvalue.org

[2] https://developers.google.com/search/docs/data-types/factcheck

3.4 Provenance and blockchain technology

Blockchain technology, or more generally, Distributed Ledger Technology (DLT) has been developed to keep a distributed immutable ledger of financial transactions. The ledger can be seen as a provenance record of, say, bitcoins; and it is therefore entirely unsurprising that DLT could be used to record provenance in other settings. There is some commercial interest in using DLT to record supply side provenance – for example the farm from which a lamb chop originated [56, 73], and there have been suggestions that it could be used for valued artefacts [70]. Superficially this kind of provenance looks rather like where-provenance for digital artefacts. Indeed there is at least one system [55] that has been developed to record data provenance at the level of file systems. The system-level provenance [62] operations on files such as read, write, share and modify are recorded using DLT.

Whether the cost of current DLT justifies its use for these applications or whether there are sufficient financial incentives to maintain a distributed ledger for the provenance of artefacts are questions well beyond the scope of this paper. However there is one interesting observation regarding data provenance. DLT was developed [63] in part to prevent "double spending": the same coin cannot be given to two parties, and a similar constraint holds for the provenance of artefacts. In nearly all forms of *data* provenance, it is understood that data gets copied, thus we do not need this constraint. Whether this will allow us to to develop simpler or less costly distributed ledgers for data provenance is an open question.

3.5 Provenance and privacy

On the face of it, provenance negates privacy. Gaining knowledge of where some piece of clinical data has come from is exactly what techniques such as differential privacy are designed to prevent. This contradiction itself poses some interesting questions because there are many situations in which we want both provenance and privacy. Imagine, for example that we have some clinical patient records provided by a hospital H and a research group R that wants to analyze some of the data in those records. H writes programs to export anonymized data to R and R writes some analysis programs. H and R interact, and both H and R keep provenance associated with their activities perhaps for repeatability as described in Section 2.2. In what sense have they kept enough provenance to describe the combined interaction?

This raises some interesting issues with provenance models. In what sense can we *compose* the provenance descriptions of two interacting activities. In the simple world of database queries, composition is a natural requirement and is usually satisfied. The provenance of the composition of two queries can be easily derived from the provenance of each of those queries. However, it is not clear how in, for example, PROV [60] one might glue together two provenance graphs of interacting activities, and whether this would be a satisfactory model of the combined activity. In our example of medical records, supposed R discovered some anomaly that indicated that H had a patient at risk. Would one have enough information to identify that patient? Also, suppose that neither R nor H wanted to reveal their individual provenance data, could some secure multi-party computation algorithm be used to identify the patient?

4. CONCLUSIONS

We have attempted to describe some areas in which data provenance is finding applications and is opening up new lines of research. There is no doubt that the theory of provenance, annotation in relational databases, and versioning will continue to develop and will be developed for other data models. Some examples of recent work in these areas include [36], where semirings are extended to capture the semantics of SPARQL queries (with OPTIONAL) on annotated RDF data and [38] where semirings are extended to deal with negation.

However the developments that will have the most impact will, we believe, stem from the public understanding of provenance. For example, we have seen how provenance can be understood and exploited in the social media, but there are even simpler situations in which one could develop useful applications of provenance. Consider the apparently innocuous copy and paste operations and how much provenance has been lost in their use. It would surely be a relatively simple matter to instrument these operations to carry some kind of provenance token that is generated for the source data (document, spreadsheet etc.) and for this to be carried across, along with the data being copied into a provenance repository associated with the target. In experimental environments for curated databases, such a mechanism has already been shown to be workable [9] and not at all costly in resources.

Today, the prevalence of open data [3] makes it even more compelling for data providers and consumers alike to instrument such provenance-aware generation and copy-paste mechanisms. Just as we prefer to read documents with proper authorship and from trusted sources, shouldn't we place higher value on documents that contain provenance or are generated by editors that are provenance-aware? Isn't it time to instrument good "provenance manners" to practice for the mass market by enabling documents to generate provenance tokens and editors to be provenance-aware?

5. REFERENCES

[1] H. Allcott and M. Gentzkow. Social media and fake news in the 2016 election. *J. of Economic Perspectives*, 31(2):211–236, 2017.

[2] B. Arab, S. Feng, B. Glavic, S. Lee, X. Niu, and Q. Zeng. GProM - a swiss army knife for your provenance needs. *IEEE Data Eng. Bull.*, 41(1):51–62, 2018.

[3] J. Attard, F. Orlandi, S. Scerri, and S. Auer. A systematic review of open government data initiatives. *Government Information Quarterly*, 32(4):399–418, 2015.

[4] G. Barbier, Z. Feng, P. Gundecha, and H. Liu. *Provenance Data in Social Media*. Morgan & Claypool Publishers, 2013.

[5] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. Uldbs: Databases with uncertainty and lineage. In *VLDB*, pages 953–964, 2006.

[6] D. Bhagwat, L. Chiticariu, W. C. Tan, and G. Vijayvargiya. An annotation management system for relational databases. *VLDB J.*, 14(4):373–396, 2005.

[7] A. Bonifati, R. Ciucanu, and S. Staworko. Learning join queries from user examples. *TODS*, 40(4):24:1–24:38, 2016.

[8] S. Bowers. Scientific workflow, provenance, and data modeling challenges and approaches. *J. Data Semantics*, 1(1):19–30, 2012.

[9] P. Buneman, A. Chapman, and J. Cheney. Provenance management in curated databases. In *SIGMOD*, pages 539–550. ACM, 2006.

[10] P. Buneman, J. Cheney, and S. Vansummeren. On the expressiveness of implicit provenance in query and update languages. *TODS*, 33(4):28:1–28:47, 2008.

[11] P. Buneman, S. Davidson, and J. Frew. Why data citation is a computational problem. *Communications of the ACM*, 59(9):50–57, 2016.

[12] P. Buneman, S. B. Davidson, W. Fan, C. S. Hara, and W. C. Tan. Keys for XML. *Computer Networks*, 39(5):473–487, 2002.

[13] P. Buneman, S. Khanna, K. Tajima, and W. C. Tan. Archiving scientific data. *ACM TODS*, 29:2–42, 2004.

[14] P. Buneman, S. Khanna, and W.-C. Tan. Why and where: A characterization of data provenance. In *ICDT*, pages 316–330, 2001.

[15] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. Intelligible models for

healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *SIGKDD*, pages 1721–1730, 2015.
[16] S. Cazalens, J. Leblay, I. Manolescu, P. Lamarre, and X. Tannier. Computational fact-checking: a content management perspective. *PVLDB*, 11(12):2110–2113, 2018.
[17] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *ICCV*, ICCV, pages 2722–2730, 2015.
[18] J. Cheney, L. Chiticariu, and W. C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.
[19] J. Cheney, S. Chong, N. Foster, M. I. Seltzer, and S. Vansummeren. Provenance: a future history. In *SIGPLAN*, pages 957–964, 2009.
[20] L. Chiticariu, W. C. Tan, and G. Vijayvargiya. Dbnotes: a post-it system for relational databases based on provenance. In *SIGMOD*, pages 942–944, 2005.
[21] Y. Cui and J. Widom. Practical lineage tracing in data warehouses. In *ICDE*, pages 367–378, 2000.
[22] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM TODS*, 25(2):179–227, 2000.
[23] C. Curino, H. J. Moon, A. Deutsch, and C. Zaniolo. Update rewriting and integrity constraint maintenance in a schema evolution support system: PRISM++. *PVLDB*, 4(2):117–128, 2010.
[24] DataCite Metadata Schema for the Publication and Citation of Research Data. http://schema.datacite.org/meta/kernel-3/doc/DataCite-MetadataKernel_v3.1.pdf, October 2014.
[25] S. B. Davidson, S. C. Boulakia, A. Eyal, B. Ludäscher, T. M. McPhillips, S. Bowers, M. K. Anand, and J. Freire. Provenance in scientific workflow systems. *IEEE Data Eng. Bull.*, 30(4):44–50, 2007.
[26] S. B. Davidson, S. C. Boulakia, A. Eyal, B. Ludäscher, T. M. McPhillips, S. Bowers, M. K. Anand, and J. Freire. Provenance in scientific workflow systems. *IEEE Data Eng. Bull.*, 30(4):44–50, 2007.
[27] S. B. Davidson, D. Deutch, T. Milo, and G. Silvello. A model for fine-grained data citation. In *CIDR*, 2017.
[28] S. B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *SIGMOD*, pages 1345–1350. ACM, 2008.
[29] L. de Alfaro, M. D. Pierro, E. Tacchini, G. Ballarin, M. L. D. Vedova, and S. Moret. Reputation systems for news on twitter: A large-scale study. *CoRR*, abs/1802.08066, 2018.
[30] A. Deutsch, L. Popa, and V. Tannen. Query reformulation with constraints. *SIGMOD Record*, 35(1):65–73, 2006.
[31] F. Doshi-Velez and B. Kim. A roadmap for a rigorous science of interpretability. *CoRR*, 2017.
[32] J. R. Driscoll, N. Sarnak, D. D. Sleator, and R. E. Tarjan. Making data structures persistent. *JCSS*, 38(1):86–124, 1989.
[33] Z. Feng, P. Gundecha, and H. Liu. Recovering information recipients in social media via provenance. In *ASONAM*, pages 706–711, 2013.
[34] FORCE11. Data citation synthesis group: Joint declaration of data citation principles. https://www.force11.org/datacitation, 2014.
[35] F. Geerts, A. Kementsietsidis, and D. Milano. MONDRIAN: annotating and querying databases through colors and blocks. In *ICDE*, page 82, 2006.
[36] F. Geerts, T. Unger, G. Karvounarakis, I. Fundulaki, and V. Christophides. Algebraic structures for capturing the provenance of SPARQL queries. *JACM*, 63(1):7:1–7:63, 2016.
[37] B. Glavic, R. J. Miller, and G. Alonso. Using sql for efficient generation and querying of provenance information. *In search of elegance in the theory and practice of computation: a Festschrift in honour of Peter Buneman*, pages 291–320, 2013.
[38] E. Grädel and V. Tannen. Semiring provenance for first-order model checking. *CoRR*, abs/1712.01980, 2017.
[39] T. J. Green, G. Karvounarakis, Z. G. Ives, and V. Tannen. Provenance in ORCHESTRA. *IEEE Data Eng. Bull.*, 33(3):9–16, 2010.
[40] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.
[41] P. Gundecha, S. Ranganath, Z. Feng, and H. Liu. A tool for collecting provenance data in social media. In *SIGKDD*, pages 1462–1465, 2013.
[42] T. S. Guzella and W. M. Caminhas. Review: A review of machine learning approaches to spam filtering. *Expert Syst. Appl.*, 36(7):10206–10222, 2009.
[43] A. Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.
[44] K. W. Hart, D. B. Searls, and G. C. Overton. Sortez: A relational translator for ncbi's asn. 1 database. *Bioinformatics*, 10(4):369–378, 1994.
[45] M. Herschel, R. Diestelkämper, and H. Ben Lahmar. A survey on provenance: What for? what

form? what from? *VLDB J.*, 26(6):881–906, 2017.
[46] S. Huang, L. Xu, J. Liu, A. J. Elmore, and A. G. Parameswaran. Orpheusdb: Bolt-on versioning for relational databases. *PVLDB*, 10(10):1130–1141, 2017.
[47] D. Ignatius. How to protect against fake facts. The Washington Post, November 2017.
[48] D. V. Kalashnikov, L. V. S. Lakshmanan, and D. Srivastava. Fastqre: Fast query reverse engineering. In *SIGMOD*, pages 337–350, 2018.
[49] G. Karvounarakis and T. J. Green. Semiring-annotated data: queries and provenance? *SIGMOD Record*, 41(3):5–14, 2012.
[50] M. Kaufmann, P. M. Fischer, N. May, and D. Kossmann. Benchmarking bitemporal database systems: Ready for the future or stuck in the past? In *EDBT*, pages 738–749, 2014.
[51] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, J. Dong, M. Y. T. Made K. Prasadha and, Jacqueline Pei and, J. Zhu, C. Li, S. Hewett, J. Dong, I. Ziyar, R. Z. Alexander Shi and, L. Zheng, R. Hou, W. Shi, X. Fu, Y. Duan, V. A. Huu, C. Wen, E. D. Zhang, C. L. Zhang, O. Li, M. A. S. Xiaobo Wang and, X. Sun, J. Xu, A. Tafreshi, M. A. Lewis, H. Xia, and K. Zhang. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172:1122–1131.e9, 2018.
[52] K. Kulkarni and J.-E. Michels. Temporal features in sql:2011. *SIGMOD Rec.*, 41(3):34–43, 2012.
[53] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila. Finding effectors in social networks. In *SIGKDD*, pages 1059–1068, 2010.
[54] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.
[55] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 468–477. IEEE Press, 2017.
[56] P. P. Ltd. Retrieved September 12 2018.
[57] M. Maddox, D. Goehring, A. J. Elmore, S. Madden, A. G. Parameswaran, and A. Deshpande. Decibel: The relational dataset branching system. *PVLDB*, 9(9):624–635, 2016.
[58] P. Missier and L. Moreau. PROV-dm: The PROV data model. W3C recommendation, W3C, Apr. 2013. http://www.w3.org/TR/2013/REC-prov-dm-20130430/.
[59] H. J. Moon, C. Curino, M. Ham, and C. Zaniolo. PRIMA: archiving and querying historical data with evolving schemas. In *SIGMOD*, pages 1019–1022, 2009.
[60] L. Moreau and P. Groth. "provenance: an introduction to prov". *Synthesis Lectures on the Semantic Web: Theory and Technology*, 3(4):1–129, 2013.
[61] L. Moreau, B. Ludäscher, I. Altintas, R. S. Barga, S. Bowers, S. Callahan, G. Chin Jr, B. Clifford, S. Cohen, S. Cohen-Boulakia, et al. Special issue: The first provenance challenge. *Concurrency and computation: practice and experience*, 20(5):409–418, 2008.
[62] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer. Provenance-aware storage systems. In *USENIX Annual Technical Conference, General Track*, pages 43–56, 2006.
[63] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008.
[64] F. Poursabzi-Sangdeh, D. G. Goldstein, J. M. Hofman, J. W. Vaughan, and H. M. Wallach. Manipulating and measuring model interpretability. *CoRR*, 2018.
[65] B. A. Prakash, J. Vreeken, and C. Faloutsos. Spotting culprits in epidemics: How many and which ones? In *IEEE ICDM*, pages 11–20, 2012.
[66] S. Pröll and A. Rauber. Scalable data citation in dynamic, large databases: Model and reference implementation. In *IEEE Intl. Conf. on Big Data*, pages 307–312, 2013.
[67] Research Data Alliance (RDA), Data Citation WG.
[68] J. F. Roddick. A survey of schema versioning issues for database systems. *Information & Software Technology*, 37(7):383–393, 1995.
[69] N. Ruchansky, S. Seo, and Y. Liu. CSI: A hybrid deep model for fake news detection. In *CIKM*, pages 797–806, 2017.
[70] M. Schuetz. Startup codex brings blockchain to art with backing from pantera, February 2018. Retrieved September 2018.
[71] P. Senellart. Provenance and probabilities in relational databases. *SIGMOD Record*, 46(4):5–15, 2017.
[72] D. Shah and T. Zaman. Rumors in a network: Who's the culprit? *IEEE Trans. Inf. Theor.*, 57(8):5163–5181, 2011.
[73] E. Stahl. "blockchain provenance: Wheres my new sweater really from?", November 2017. Retrieved September 2018.
[74] B. ten Cate, C. Civili, E. Sherkhonov, and W.-C. Tan. High-level why-not explanations using ontologies. In *PODS*, pages 31–43, 2015.
[75] B. ten Cate, P. G. Kolaitis, K. Qian, and W.-C. Tan. Active learning of gav schema mappings. In

PODS, pages 355–368, 2018.

[76] S. Veenadhari, B. L. Misra, and C. Singh. Machine learning approach for forecasting crop yield based on climatic parameters. *2014 International Conference on Computer Communication and Informatics*, pages 1–5, 2014.

[77] X. Wang, C. Yu, S. Baumgartner, and F. Korn. Relevant document discovery for fact-checking articles. In *WWW*, pages 525–533, 2018.

[78] Y. R. Wang and S. E. Madnick. A polygen model for heterogeneous database systems: The source tagging perspective. In *VLDB*, pages 519–538, 1990.

[79] A. Woodruff and M. Stonebraker. Supporting fine-grained data lineage in a database visualization environment. In *ICDE*, pages 91–102, 1997.

[80] Y. Wu, P. K. Agarwal, C. Li, J. Yang, and C. Yu. Computational fact checking through query perturbations. *TODS*, 42(1):4:1–4:41, 2017.

[81] W. Zhou, Q. Fei, A. Narayan, A. Haeberlen, B. T. Loo, and M. Sherr. Secure network provenance. In *SOSP*, pages 295–310, 2011.

[82] W. Zhou, M. Sherr, T. Tao, X. Li, B. T. Loo, and Y. Mao. Efficient querying and maintenance of network provenance at internet-scale. In *SIGMOD*, pages 615–626, 2010.