Data Integration and Data Exchange: It's Really About Time

Mary Roth
IBM and UC Santa Cruz
torkroth@yahoo.com

Wang-Chiew Tan
UC Santa Cruz
tan@cs.ucsc.edu

ABSTRACT

With the deluge in the amount and variety of data in the world, it is rare for data that describes an entity to be completely contained and managed by a single data source. As a consequence, there is often great value in combining data about an entity from multiple sources, and also from versions of data reported by the same source over time. Data integration in which multiple dimensions of time may be expressed explicitly (e.g., as part of the data itself) or implicitly (e.g., the publication date of a data source), must be performed with great care. This is because each data source contains only partial (time-specific) knowledge about an entity, and thus their collective knowledge about the entity may contain conflicts that need to be resolved.

In this paper, we call for a formal framework for data integration and data exchange across time that would facilitate the creation of consistent and integrated longitudinal knowledge about entities. We call such longitudinal knowledge of an entity its *when-provenance*, which intuitively corresponds to when one knows what one knows about the entity. We believe that the vision and research directions described in this paper will serve to instigate the research and development of the next generation data integration and data exchange system, where both data and time can be reasoned on equal footing.

1. Introduction

With the deluge in the amount and variety of data in the world, it is rare for data that describes an entity to be derived from a single source. As a consequence, there is often great value in combining data about an entity from multiple sources, or from various versions of data reported by the same source over time. In fact, integrated data across time may unlock insights that may otherwise be hidden [60]. Rather surprisingly, even though there has been extensive prior research on data integration and data exchange, and in the related area of temporal databases, the problem of accurately and consistently combining heterogeneous data sources that contain time-specific information is still largely unexplored up to now.

Time-specific information can occur *implicitly* or *explicitly* in data. For example, scientific databases that publish data [51] on a regular basis have an implicit time component derived from the publication date. On the other hand, data records that contain birth and death dates, graduation dates, dates of purchase etc., are examples of data that contain explicit time-specific information. Other examples of data sources with interesting time-specific information include the Internet Archive [38], social media data (e.g., Twitter, Facebook, blogs etc.), and even public, curated, or enterprise data that are shared publicly or sold by data markets such as Amazon Web Services and Azure DataMarket.

In many cases, data contains both implicit and explicit types of time-specific information. An example of this type of data are the reports that are available via the Securities and Exchange Commission (SEC) EDGAR database [29]. The SEC requires that corporations regularly report information disclosing the financial health of the company and the stock transactions of its corporate officers and directors ("beneficial owners"). Each report includes the date the report was received and published by the SEC as well as explicit time-specific information, such as the date, number and type of shares involved in a stock transaction, and the title of the insider on the day of the transaction.

Even though every version of each data source contains valuable information, there is often even greater value for an aggregated view of the versions of data produced by the data sources. To exemplify, each SEC report contains only partial information; while a report may record that an executive held the title on certain date, it does not contain enough information to determine how long the title was held, or even if he is still employed at the company. However, such reports can be aggregated and coupled with information from other publicly available sources to build historical profiles of corporate executives, including employment history, stock holdings, and relationships to other executives over time. Such profiles are useful for a variety of scenarios including executive recruiting, regulatory compliance, and even fraud detection [19].

There is also significant value to produce aggregate electronic medical records. This is because a temporally consistent and integrated personal patient health care record provides complete knowledge of a patient's prescription history and medical procedures to ensure correct diagnosis and treatment. Patients typically visit multiple medical facilities and see multiple medical professionals, sometimes simultaneously, over the course of their lifetime, and each incidence may result in the diagnosis and treatment of medical conditions. Each of these interactions is associated with point-in-time information regarding when a clinic was visited, or when certain prescriptions and medical procedures were carried out. While any one medical facility may keep a complete record of a patient's medical history at that facility, it does not provide a complete view of the patient's health and medical care. Currently, medical profes-

6th Biennial Conference on Innovative Data Systems Research (CIDR '13) January 6-9, 2012, Asilomar, California, USA.

sionals rely on the patients to provide medical history information. However, this method is not always reliable, which can potentially lead to misdiagnoses and improper treatment. Indeed, adverse drug interaction is a serious cause of patient hospitalization and death [36, 41].

In this paper, we illustrate some of the challenges associated with data integration and data exchange across time by means of a running example derived from realistic data. We argue for a systemic approach to investigate and define a time-aware data model and schema mappings with the goal of developing a system that can efficiently support data integration and data exchange across time, longitudinal query answering, and evolution changes that may occur to the schema, data, and mappings.

1.1 Motivating Example

We now describe a simplified example in detail to exemplify the issues and challenges to create an integrated archive of profile data across time. As shown in Figure 1, the goal is to construct employment histories of individuals from various versions of SEC reports (sources S1 and S2) and additional data available via the Internet (source S3). Source instance S1 is derived from quarterly SEC Form 10Q reports and provides information that Ann Executive was Chief Financial Officer (CFO) at ABC corporation in Jan '05. Source instance S2 is derived from SEC Forms 3/4/5 that are periodically filed by corporations to report changes in the stock holdings of its officers and directors. The extract represented in the figure shows that Ann Executive was ABC's Chief Executive Officer (CEO) in Feb '10. Additional information from a Wikipedia entry published in Jan '09 (S3) also shows that Ann Executive was CFO at company XYZ in Jan '08. Each source instance tracks two dimensions of time. One time dimension states when the report to the SEC was made (or when the fact was reported on the Web) and the other time dimension states when the position held by Ann Executive was known to begin.

The integrated profile of Ann Executive is shown on the right of Figure 1, and we defer discussion of the middle of the figure to Section 2.2. Figure 2 presents a visualization of the changes in one's understanding of Ann Executive's employment history as facts are combined. The integrated profile shows a combined understanding of when Ann Executive worked for each company and also when Ann Executive held various positions in her lifetime.

In this example, we apply a "one company, one job-title" constraint (i.e., every individual can only be employed by at most one company and hold one job position at any point in time) to obtain the integrated profile. It should be noted that a given application may require many constraints, and different applications may require different sets of constraints depending on the desired structure of the data. For example, instead of "one company, one job-title", a more realistic application may allow executives to hold multiple titles at the same time, and allow an executive to simultaneously hold multiple stocks in his portfolio, but hold only one quantity of shares of a given stock at a time.

Observe that the process of combining data from different sources is not a simple matter of taking a "union" of facts that are migrated to the target. Based on the "one company, one job-title" constraint, the time periods associated with a company (resp. positions) of a person are dependent on the time periods associated with other companies (resp. positions) of the same person over time. For this example, source instance S1 provides the information that Ann Executive was the CFO of organization ABC since at least Jan '05 (i.e., active-time) and this fact was reported in Mar '05 (i.e., reported-time). Pictorially, the current state of knowledge can be visualized as the largest rectangle (P1,P2,P3,P4) in the graph in Figure 2(a). In other words, in the absence of other information, one could assume that this fact continues to remain true up to now. However, with the new knowledge that Ann Executive was the CEO of ABC in Feb '10 (reported in a Form 3/4/5 filed in Feb '10) from S2, and because of the "one-company, one job-title" constraint, one can deduce that the prior fact that Ann Executive was the CFO of ABC Company can only be true from Jan '05 to *at most* Feb '10. Hence, the new retroactive knowledge about when Ann Executive was the CFO of ABC is now given by the rectangle (P5,P2,P3,P6) shown in Figure 2(a). The graph essentially reads as follows: *Since Mar '05, Ann Executive was known to be the CFO of ABC in the time period (Jan '05-Feb '10).* In addition, the rectangle given by "CEO of ABC" states that *since Feb '10, Ann Executive was known to be the CEO of ABC in the time period (Feb '10-*now*)*. With the knowledge brought about by S2, a choice that Ann Executive was the CEO of ABC (as opposed to the CFO of ABC) is made in the active time period (Feb '10-now) because the knowledge from S2 is based on a later reported-time, and therefore assumed to be more accurate. Similarly, when the third piece of information arrives from S3, one can deduce the information that is given by the three rectangles labeled, respectively, "CFO of ABC", "CFO of XYZ", and "CEO of ABC" in Figure 2(b). Essentially, the new evidence from S3 requires one to adjust the knowledge of when Ann Executive was the CFO of ABC again, which is now given by the rectangle (P7,P2,P3,P8).

This example illustrates how the understanding of Ann Executive's employment history evolves as new knowledge arrives. Figure 2(b) depicts the most accurate and consistent understanding of the data provided by all data sources S1, S2, and S3 under the given policies and semantics for the time dimensions. In general, the integration process must include a mechanism by which one can specify different application-specific policies to govern the merge process. Such policies can be invoked to resolve conflicts that arise as new data is integrated that may introduce constraint violations when combined with existing information in the integrated result. For example, it may be that information reported at a later time should be favored as new evidence, and the time periods associated with any older evidence with which the new evidence has time-conflicts should be adjusted to remove the time-conflict. Alternatively, it may be that sources have different levels of authoritativeness, with one reporting definitive information, and another reporting information that *might* be true only in the absence of other conflicting information. When resolving time-based constraint violations in this scenario, the policy may be to favor the second source unless or until conflicting information arrives from the first.

It is worth noting that in this example, it is important to track both dimensions of time in the integrated result in order to fully understand the *when-provenance*, or *when one knows what one knows* about the employment history of an individual. For example, a complete understanding is especially important for audit purposes, where it is crucial to not only understand when a person held a particular position but also when that fact was *reported*. To exemplify, suppose a question arises as to the earliest possible time that the SEC could have had knowledge that Ann Executive participated in fraudulent activities in Jan '05 as the CFO of company ABC. The answer is Mar '05, because that was the earliest time it was reported to the SEC that Ann Executive was employed as the CFO of ABC in Jan '05.

Just as constraints are application-specific, the use of time can vary in semantics across different applications. For example, temporal databases [56] couple valid-time, the time a record is valid with respect to the application, with transaction-time, a system-

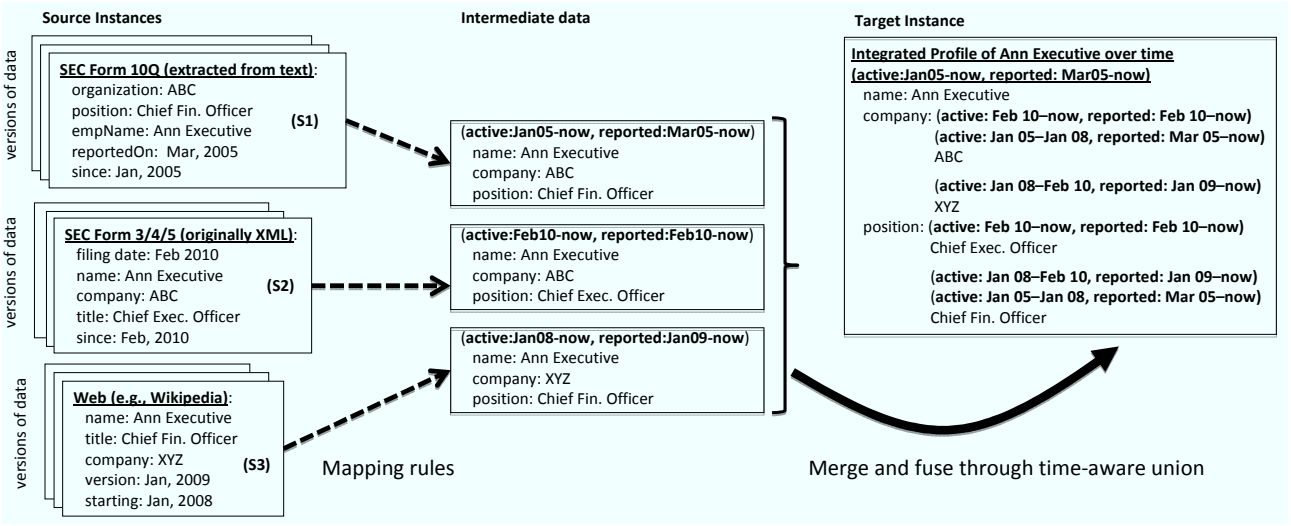

Figure 1: Combining knowledge across different times from different SEC filings and the Web.

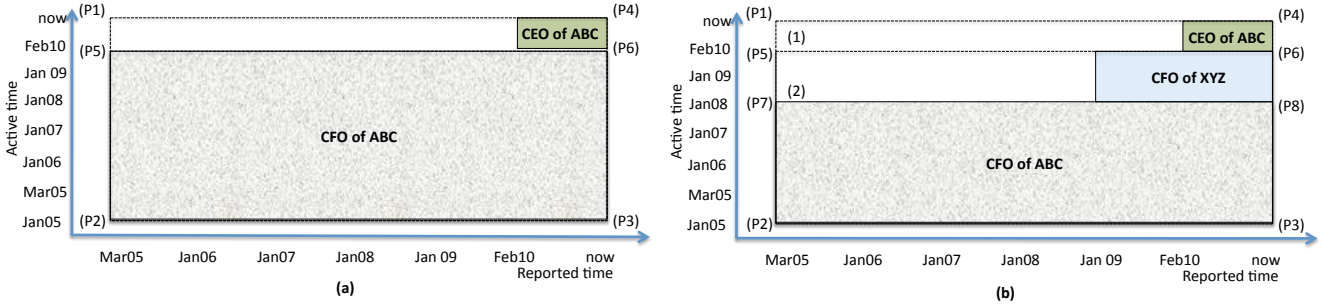

Figure 2: An illustration of the process of integrating facts from the middle of Figure 1.

generated clock time for when a record enters a database. Valid-time can be used to record when something was, is, or will be true (e.g., past, present, or future). In our example, active-time is similar to valid-time, though the update semantics of a standard relational bi-temporal database would replace the active time currently associated with the record, rather than merge and fuse the active-time from various sources as desired in our example and as shown in Figure 1. Reported-time, however, is not like transaction-time, because it is not a system-generated value; instead, it represents the time at which a report was filed with the SEC, not the time the record derived from the report was inserted into the database. In other data integration scenarios, it may also be important to track the departure time from the original source and/or its arrival time at the target integrated database.

While single and two dimensional (bi-temporal) time is well studied in the literature, use cases also exist for higher dimensions. For example, data in SEC reports filed by companies might record a time for when the data was valid, a time to capture when the report was filed with the SEC, and a time to record when the report was published by the SEC, each of which might be crucial for compliance verification and fraud detection. Additionally, streaming and complex event processing systems [5] that model sliding windows of valid time rely on three dimensions of time: valid-time, an *occurrence-time* to record the window in which the valid time occurred, and transaction-time.

Even though we use a fictitious corporate executive to illustrate our ideas, the running example is realistic in that one can obtain the various types of information that we depict from the SEC EDGAR database and the Web. In this discussion, we assume that the source instances contain structured information that may be the result of an extraction process that is applied on "raw" data sources (e.g., text files, Web pages, blogs, etc.) which may be unstructured. Furthermore, we assume that these source instances are already mapped into a form that is ready for integration. The example is intentionally kept simple to focus our discussion on the key issues of this proposed framework. However, in practice, added complexities can arise in several different dimensions. For example, the structure of the integrated profile may not resemble the structure of the source data as closely as it does in this example. In addition, data can be much richer in both structure and content; data can be *hierarchical* (whether in the source or target), it may be qualified with multiple *dimensions of time*, and may contain more attributes, records, and types than what is shown in Figure 1.

1.2 Related work

Related work on data integration and data exchange Data integration and data exchange are long standing problems that are known to be difficult and time-consuming tasks [9, 27].

A primary task to combine data from multiple sources, whether through data integration or data exchange, is the precise specification of *schema mappings* between the source schemas and the

global or target schema. These schema mappings are typically specified through the non-trivial efforts of data architects[1]. Several commercially available tools and research prototypes help alleviate the task of integrating and exchanging data. They range from graphical user interfaces (e.g., Altova Mapforce, Stylus Studio, [35, 57]), and high-level declarative languages (e.g., [14, 52]) to specify schema mappings, to the study of operators for manipulating schema mappings (e.g., [7]) and more recently, the use of data examples (e.g., [1, 2, 3, 53]) to design and understand schema mappings. However, prior techniques and systems are largely agnostic to time, and hence, they cannot be immediately applied to build an integrated dataset over time without at least manually introducing ad-hoc time-manipulation functions into the mapping process. Except for [62], which discusses three types of temporal heterogeneity that need to be resolved when integrating data over time, the issue of time, to the best of our understanding, has not been systematically and thoroughly addressed in prior work in this area.

Our call to investigate techniques for resolving conflicts across time can be seen as a generalization of the study of data conflict resolution for integration [12, 13, 28, 59]. The difference is, again, that existing techniques for data conflict resolution are agnostic to time. Our proposed time-aware union (described in Section 2.2) resolves data conflicts at any time point through a broader strategy that takes into account constraints and policies that may be based on time.

Related work on bi-temporal databases There is a large body of work on bi-temporal databases and [22, 40] provides a comprehensive overview of related work and concepts in this area.

The problem of integrating data with time-specific information cannot be immediately resolved with bi-temporal databases. As mentioned earlier, bi-temporal databases have only two specific notions of time, namely that of *valid-time* and *transaction-time* (which are also known as *application-time* and, respectively, *system-time* in the recent standard [39]). Transaction time denotes the time at which updates are applied to the database and hence, they can only "increase", while valid time denotes the time at which a tuple is valid in the real world. However, in data integration and data exchange settings over heterogeneous distributed data sources, there is little control over the order in which facts arrive to be merged into the target. As can be seen in our running example, facts may arrive in the target "out-of-order", whether we consider reported-time or active-time. Hence, it is important to provide idempotent, commutative, and associative properties as facts are integrated to ensure a time-consistent result, regardless of the order in which the facts are learned. While a standard bi-temporal database, for example, could be used to track when facts are learned, it does not guarantee that the most current understanding of the facts will be the same, regardless of the order in which updates occur. Suppose, for example, we did use a bi-temporal relational database to store records about corporate executives like Ann Executive in Figure 1, and that valid-time was used to capture the time at which a particular job description was valid in the executive's career. Consider the following three updates that correspond to the arrival of the three facts shown in the middle of Figure 1:

```
UPDATE PROFILE FOR PORTION OF BUSINESS_TIME
FROM '01/01/2005' to CURRENT DATE
SET POSITION = 'CFO', SET COMPANY = 'ABC'
WHERE NAME = 'Ann Executive'

UPDATE PROFILE FOR PORTION OF BUSINESS_TIME
FROM '02/01/2010' to CURRENT DATE
SET POSITION = 'CEO', SET COMPANY = 'ABC'
WHERE NAME = 'Ann Executive'

UPDATE PROFILE FOR PORTION OF BUSINESS_TIME
FROM '01/01/2008' to CURRENT DATE
SET POSITION = 'CFO', SET COMPANY = 'XYZ'
WHERE NAME = 'Ann Executive'
```

The update statements above are written with the DB2 syntax. The database will add a system generated transaction-time with an open-ended time interval for each update that reflects the time at which the update was made, and it will automatically 'close-out' the transaction-time of the existing version of the same record with an open-ended time interval (if one exists). Thus, the version of the record with the open-ended transaction-time reflects the current understanding of the record. If the updates are executed in this order, then the current understanding is that Ann Executive has been CFO of XYZ since January 2008. However, if the order of the statements is reversed, the database will record the current understanding to be that Ann Executive has been CFO of ABC since January 2005. While all three facts illustrated in the middle of Figure 1 were true at different points in time, it is not possible to rely on a system-generated transaction-time to accurately portray the time at which the facts were reported to be true according to the application. Is the last update a correction to the first and second, or just a fact that arrived out of order?

Even if facts from different data sources are migrated in an order that can be treated as transaction-time order, the combined knowledge of when Ann Executive is the CFO or CEO of a company under the valid-transaction-time semantics is different from the semantics that are depicted in Figure 1. As mentioned earlier, each update to the valid-time columns of a record in a bi-temporal database effectively replace the previous values of valid-time columns for that record, rather than merging the new and existing values as shown in our example.

In principle, one could re-engineer bi-temporal databases to adopt our desired semantics and to handle out-of-order updates. We can add additional columns to a bi-temporal database to capture time-specific information that may exist in the data (e.g., active and reported time). However, this will require ad-hoc functions to be introduced, perhaps through the use of triggers or user-defined functions, to manipulate the time in a way that fits the application. As we described in Section 1.1, applications may require different semantics to integrate data with time-specific information in general. Since the "correct" semantics may depend only on the application at hand, our running example points out the need to provide an extensible framework that goes beyond the valid-transaction-time semantics of bi-temporal databases so that alternative semantics can be adopted as needed.

With the exception of [24] and [49], most implementations of bi-temporal databases have been relational, which do not naturally capture the use cases and the time-aware nested relational structure that we have described in our running example. Furthermore, regardless of existing implementations of bi-temporal databases, to the best of our knowledge, there currently does not exist a framework where one can declaratively specify schema mapping rules for data integration and data exchange across time.

Related work on complex event processing, streams, and uncertain data Complex event processing and data streams that continually react and respond to many inputs from multiple sources is another area of related research [5, 15, 58]. The goal of such systems is to make decisions based on continuously streaming data that may arrive in order or out-of-order [46], and for which the time

[1]The term "data architect" is used by [10] to refer to the role of the person who designs and develops schema mappings.

element associated with data values may be known with certainty or may be imprecise [54, 61]. For example, [5] introduces an occurrence time set by an application to track changes in valid time for a tuple that represents an event. However, it does not address how conflicts should be resolved if the valid times for a set of tuples (with different values) that refer to the same event overlap. In fact, the time-aware union operator could conceivably be used to resolve such conflicts between multiple sources contributing to the stream, and adjust the occurrence time appropriately before tuples enter the stream. In data integration and data exchange, conflicts need to be resolved when multiple sources supply multiple distinct values of time-aware data that introduce constraint violations. In our running example, the time-aware union operator enforces the "one-company, one job-title" constraint for an executive. In general, the system should enable support for other reasonable constraints, such as to allow an executive to hold multiple different stocks in his portfolio simultaneously, but only one quantity of a given stock at any time instant, or that the time periods associated with salary information reported for an executive at a particular company be contained within the time periods of his known tenure at that company. What is needed is a formal foundation by which to model such constraints, and enable specification of application-specific policies to resolve violations as part of the integration process to produce a consistent integrated result.

Related work on versioning, archiving, and annotation systems Different techniques for archiving data exists, going back to multi-version control systems [8] with certain ACID guarantees, diff-based version management systems (e.g., [48]), reference-based approaches (e.g., [20]) for hierarchical data, to techniques that compact versions based on key constraints [16, 18, 43].

Archiving can be construed as a form of data integration across versions of data. All the systems above, however, apply only to a single dimension of time (i.e., versions of data) and cannot be immediately generalized to manage multiple dimensions of time.

Time-specific information can be seen as a special type of annotation and in this context, our work is a first step towards a general and extensible framework for understanding how to integrate data with different types of annotations in general. There has been significant prior work on manipulating annotations of data [11, 33, 44] and the study of provenance semirings [34], which allows different types of annotations on data. It is conceivable that the "additive" commutative monoid of a provenance semiring can be applied to obtain a union of such annotated data sources. However, a mechanism for understanding how conflicts are to be resolved in the process of combining annotations will be required to ensure that constraints in the target schema and associated policies are satisfied.

2. Challenges

The final integrated profile of our running example consolidates the individual data sources into one unified view and thus facilitates longitudinal querying. For example, it is easier to pose and answer queries against the integrated profile, such as "Who were the other executives of XYZ during the time when Ann Executive was CFO of XYZ? Are they also now executives at ABC? Answering such queries directly over the individual source instances is likely to involve complex logic that manipulates data from many reports from the SEC and the Web.

This paper argues for the need to investigate a solid foundation and develop a system for integrating and exchanging data across time, as well as answering queries over such settings. Towards this goal, we will investigate and address questions including: What are an appropriate data model, constraints, and schema mapping language that are "time-aware"? What is the precise semantics for such schema mappings? How does one efficiently exchange data according to such schema mappings? How does one efficiently answer queries over a data integration or data exchange specification under this framework? How can a data integration system efficiently perform incremental view maintenance and view adaptation, and handle schema evolution?

2.1 Appropriate data model and schema mapping language

Time-aware data model For the purpose of describing the research challenges, we describe a basic data model that captures both time and data as first-class citizens:

$$\tau ::= \mathsf{now} \mid \text{Str} \mid \text{Int} \mid (\tau, \tau) \mid \text{SetOf } \tau \mid \text{Rcd}[l_1 : \tau_1, \ldots, l_n : \tau_n] \mid \text{Pair}[l_1 : \tau_1, l_2 : \tau_2]$$

The symbol now is a special keyword that refers to the current time, and Str and Int are the only atomic types for the time being. A pair (τ, τ) is used to specify a time interval, Rcd$[l_1 : \tau_1, \ldots, l_n : \tau_n]$ is a record type with fields (or attributes) $l_1, \ldots, l_n$ and corresponding types $\tau_1, \ldots, \tau_n$, and a Pair$[l_1 : \tau_1, l_2 : \tau_2]$ is a special record type with two fields. Data is modeled in a tree-like structure using set types, records, and pairs similar to the nested relational model used in data integration and data exchange [52].

Here, a *temporal context* can be associated with every element in the tree in the data since every element can, in principle, vary over time. Intuitively, the *temporal context of an element* is a set of n-dimensional time intervals that is used to capture the time periods, under the different dimensions, at which this element exists. In the case where only two time dimensions are involved, a temporal context is also known as a *bi-temporal element* in the literature of bi-temporal databases [40]. However, in order to avoid confusion with the term "elements" (i.e., nodes of a tree) in our setting, we will use the term temporal context instead.

A data model for our running example is defined in Figure 3. ActRep, shown in the first three lines, is the type of a temporal context, which is defined to be a set of records with two fields, "active" and "reported", denoting the two dimensions of time in our running example. The target schema (defined as Data) is shown afterwards in the same figure.

```
starttime ::= Int
endtime ::= Int | now
ActRep ::= SetOf Rcd[active:(starttime,endtime),
                     reported:(starttime,endtime)]

Data ::=
   Pair[C: ActRep,
        DB: SetOf Pair[C: ActRep, Person:
             Rcd[name*: Str,
                 company:
                     SetOf Pair[C: ActRep, value*: Str]
                 position:
                     SetOf Pair[C: ActRep, value*: Str]
               ] ] ]
```

Figure 3: A data model for our running example.

Intuitively, the label DB is the label for the set of tuples in our target database. The labels DB, Person, and the values of company and position are each associated with a field C that is of ActRep type. In other words, one can capture the temporal context of these elements across time. Observe that DB is written as a field of a Pair type instead of SetOf type because there is only one DB element. In contrast, there can be multiple Person elements under the DB. Next, observe that name is defined to be of type Str, while company

and positions are defined with SetOf Pair type. This is because we assume that name is the *key* of the relation (denoted with a "*"), which means that every person element can be uniquely identified by name. In contrast, there can be multiple values for company and position, since these values can vary across time. However, these values are unique within the set denoted by company and respectively, position. Hence, the values of company and position are also keys. It should be noted that even though there is a temporal context associated with many elements of Data, an implementation of this model can achieve substantial space savings by storing the temporal context at an element only if it is different from its parent element [18]. In fact, this data model is arguably a generalization of the data model that is implicit in [8, 17, 18, 50], where the focus was on only a single semantics and single dimension of time (i.e., version number).

Time-aware Schema Mappings There should be a high-level language that would allow one to declaratively specify the desired integration and exchange semantics across time, and at the same time, hide the procedural and physical plans. Such a language should give rise to an accompanying graphical user interface that allows one to visually specify such mappings.

For example, to declaratively specify the evolution of facts in our running example from Form 10Q to the target, we may state the following:

$$\begin{aligned}\forall o,p,e,r,s.(&\text{Form10Q}(o,p,e,r,s) \rightarrow \exists C_1,C_2,C_3,O,P.\\ &\text{Data.C}((s,\mathsf{now}),(r,\mathsf{now}))\wedge\\ &\text{Data.DB}(C_1,(e,O,P))\wedge\\ &O(C_2,o)\wedge P(C_3,p)\wedge C_1((s,\mathsf{now}),(r,\mathsf{now}))\wedge\\ &C_2((s,\mathsf{now}),(r,\mathsf{now}))\wedge C_3((s,\mathsf{now}),(r,\mathsf{now})))\end{aligned} \quad (1)$$

For compactness, we have presented the above time-aware schema mappings in a notation that is similar to source-to-target tuple generating dependencies (s-t tgds) [31] over an alternative query-like notation [52] that is more verbose. These dependencies, also known as Global-Local-As-View dependencies, have been studied extensively in data integration and data exchange [42, 45].

Intuitively, the schema mapping above asserts the following: for every record (o, p, e, r, s) in Form 10Q (i.e., the source), where the variables o, p, e, r, and s bind to "company", "position", "name", "reportedOn", and "since" values respectively in Form 10Q, there must exist variables C_1, C_2, C_3, O, P such that the following is true in the target: The pair $((s, \mathsf{now}), (r, \mathsf{now}))$ are time intervals that can be found in the temporal context given by Data.C. Furthermore, the record $(C_1, (e, O, P))$ occurs in Data.DB, where C_1 denotes the temporal context associated with the person record (e, O, P). The variables O and P denote the set of companies, and the set of positions respectively. The record (C_2, o) exists in the set O and the record (C_3, p) exists in the set P. Finally, the record $((s, \mathsf{now}), (r, \mathsf{now}))$ occurs in the sets C_1, C_2, and C_3.

Similar time-aware schema mappings can be written to migrate data from S2 and S3 to the target. In general, the language should allow more complex time-aware schema mappings which may specify selections, projections, or joins on source instances and temporal contexts to obtain the desired result. The proposed schema mapping language should be sufficiently expressive to capture most, if not all, practical use-cases. The "right" schema mapping language is likely to borrow experience from temporal query languages, (e.g., [21, 23]), operators proposed for reasoning about time intervals (e.g., [4]), and/or logical and modal operators (e.g., from Linear Temporal Logic (e.g., [32])).

2.2 Data Integration and Data Exchange across Time

Time-aware union Traditionally, a *data exchange specification* is a triple $(\mathbf{S}, \mathbf{T}, \Sigma)$, where $\mathbf{S}$ is a source schema, $\mathbf{T}$ is the target schema and Σ is a set of schema mappings given by s-t tgds and target dependencies [30]. Given a source instance I of $\mathbf{S}$, the goal of data exchange is to materialize a target instance J of $\mathbf{T}$ so that I and J together satisfy Σ. A fundamental assumption that is often implicit in the traditional data exchange framework is that the target instance is created as a *union* of facts that are obtained from the result of the data exchange. Under the set union semantics, the target instance can be obtained by applying the chase procedure [6, 47] or by executing scripts that implement the chase [52] to obtain facts in the target. Subsequently, all target facts are union-ed, and under set union semantics, the set of all identical facts are fused into one.

To meaningfully integrate and exchange data across time, we advocate the use of time-aware union instead of set union. Since "name" is the key of the Persons relation, we can deduce that at any point in time, there can only be at most one person with the name Ann Executive. This type of constraint is called a *sequenced key constraint* in relational temporal databases [40]. Based on the semantics of sequenced key constraints and structural constraints that are imposed by the data model of our target schema defined in Figure 3, we propose to perform a *time-aware union* of facts that are migrated to the target. The time-aware union operator will adjust the temporal contexts accordingly and then *fuse* the three records, shown in the middle of Figure 1, to obtain the final result shown on the right of the same figure.

A key challenge is to understand and define the desirable properties that this operator should possess. For example, it will be highly desirable that the proposed semantics for time-aware union satisfy properties such as commutativity and associativity. In other words, if T_1, T_2, and T_3 denote three source instances under the same data model, then $T_1 \uplus T_2 = T_2 \uplus T_1$ (commutativity), where $\uplus$ denotes the time-aware union operator. Furthermore, we must have $T_1 \uplus (T_2 \uplus T_3) = (T_1 \uplus T_2) \uplus T_3$ (associativity). These properties are crucial in the context of data integration and data exchange because facts from different source instances may arrive out-of-order in any time dimension. The commutative and associative properties will ensure that regardless of the order in which the facts are combined in the target, the same equivalent target instance, modulo the representation of time, will be obtained.

The time-aware union operator is similar to the deep union operator [17] and the merge operator [18], where in deep union, every node in the deterministic tree can be uniquely identified by a path from the root to that node. Nodes of different trees are merged if they have identical paths. The merge operator of [18] extends deep union where *timestamps* can be associated with nodes and are manipulated as appropriate when elements are merged.

Variations of time-aware union, policies, and constraints Sequenced key constraints will help determine which facts that are migrated to the target are to be combined together. However, they do not specify *how* such facts should be combined. As described in Section 1.1, there are multiple ways to combine identical entities together. Examples of alternatives include choices that can be made between preferring information that has a later reported time or a later active time, and whether or not one should retroactively adjust understanding about the entity as new knowledge is received.

It will be desirable to investigate natural semantic variations of the time-aware union operator that make sense in practice, and determine which of the alternatives possess properties such as commutativity and associativity. Finally, it will be highly attractive to provide an extensible framework which allows policies and constraints to be specified so that different time-aware union semantics

can be applied to different types of elements in the same global or target schema.

2.3 Query Answering

Along with a data integration and data exchange framework that puts both time and data on equal footing, the classical query answering problem in data integration and data exchange will need to be re-examined in the new setting.

The investigation in this direction will first require an adequate understanding as to what constitutes an appropriate query language for the proposed data model, especially in the context of work that has already been done on the language, semantics, and expressiveness of query languages in temporal databases (e.g., see [21, 23, 55]). In addition, when a query q is posed against the target schema, how do we efficiently process q? There are two general approaches towards a solution for this problem. The first approach aims to answer q directly over the target instance, assuming that the target instance is materialized. This is especially the case for data exchange since the target instance is materialized. However, it has been shown that a direct evaluation of q over the target instance produces the correct answer only in certain settings [30]. It will be interesting to see whether similar results will hold in the new data exchange framework.

The second approach aims to rewrite q into a query (or queries) against the source instances so that q can be evaluated against the source. This approach is usually done in data integration since the target instance is unavailable. As far as we know, while there is extensive literature on query rewriting [37], there has been little or no prior work on query rewriting in the context of time-aware schema mappings and a data model where both data and time are first-class citizens.

2.4 Managing changes

Another challenging research topic is to understand how one can gracefully manage changes that may occur to source instances (aka incremental view maintenance), schema mappings (aka view adaptation), and schemas (aka schema evolution). When such changes occur, it will be highly desirable to simply compute the updates that are required without recomputing the entire target instance. There is a large body of work on temporal view maintenance that has focused primarily on valid-transaction-time semantics. There is also a large body of work on view adaptation and schema evolution. However, to the best of our knowledge, there has been little or no prior work on view adaptation or schema evolution in the context of data integration and data exchange across time.

2.5 Efficiency, efficiency, efficiency

In order to support efficient data integration and data exchange, efficient query processing over large databases, and efficient management of changes, it will be imperative to investigate physical design decisions, such as the appropriate physical storage, indexes, access methods, and even compression techniques that support such efficient operations. As we borrow experience from prior work in temporal databases and in parallel database systems (e.g., see [26]), new investigations will need to be conducted to leverage parallel database systems or the Hadoop MapReduce [25] framework to develop an efficient system.

3. Conclusion

Building a consistent profile of an entity from multiple data sources over time requires time-specific knowledge to be evolved as new facts are integrated. Such facts may have multiple dimensions of time associated with them, such as when they were true, and when the fact was made known to be true. Those dimensions may be imprecise, for example, indicating that the fact is true from a certain point in time forward unless or until conflicting information is made available. As updates may be integrated in any order in the context of data integration and data exchange, it is necessary to re-adjust the current or historical understanding of entities as new data is integrated. The problem of how one should re-adjust knowledge as a new data source is integrated is often application-specific. In this paper, we have illustrated some of the challenges associated with data integration and data exchange across time, and we laid out a vision and some research directions for the next generation data integration and data exchange system, where we believe that both time and data should be placed on equal footing.

Acknowledgements This work is partially supported by NSF grant IIS-0905276. Part of this work was done while Tan was at IBM Research - Almaden.

4. References

[1] B. Alexe, B. ten Cate, P. G. Kolaitis, and W. C. Tan. Characterizing schema mappings via data examples. *ACM TODS*, 36(4):23, 2011.

[2] B. Alexe, B. ten Cate, P. G. Kolaitis, and W. C. Tan. Designing and refining schema mappings via data examples. In *ACM SIGMOD*, pages 133–144, 2011.

[3] B. Alexe, B. ten Cate, P. G. Kolaitis, and W. C. Tan. Eirene: Interactive design and refinement of schema mappings via data examples. *PVLDB (Demonstration)*, 4(12):1414–1417, 2011.

[4] J. F. Allen. Maintaining knowledge about temporal intervals. In *CACM*, pages 832–843, 1983.

[5] R. S. Barga, J. Goldstein, M. Ali, and M. Hong. Consistent streaming through time: A vision for event stream processing. In *In CIDR*, pages 363–374, 2007.

[6] C. Beeri and M. Y. Vardi. A Proof Procedure for Data Dependencies. *JACM*, 31(4):718–741, 1984.

[7] P. A. Bernstein. Applying Model Management to Classical Meta-Data Problems. In *CIDR*, pages 209–220, 2003.

[8] P. A. Bernstein and N. Goodman. Concurrency control in distributed database systems. *ACM Comput. Surv.*, 13(2), 1981.

[9] P. A. Bernstein and L. M. Haas. Information integration in the enterprise. *CACM*, 51(9):72–79, 2008.

[10] P. A. Bernstein and S. Melnik. Model management 2.0: manipulating richer mappings. In *ACM SIGMOD*, pages 1–12, 2007.

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

[12] J. Bleiholder and F. Naumann. Declarative data fusion: syntax, semantics, and implementation. In *ADBIS*, pages 58–73, 2005.

[13] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1–41, 2009.

[14] A. Bonifati, E. Q. Chang, T. Ho, and L. V. S. Lakshmanan. HepToX: Heterogeneous Peer to Peer XML Databases, 2005.

[15] L. Brenna, A. Demers, J. Gehrke, M. Hong, J. Ossher, B. Panda, M. Riedewald, M. Thatte, and W. White. Cayuga: a high-performance event processing engine. In *ACM SIGMOD*, pages 1100–1102, 2007.

[16] P. Buneman, J. Cheney, S. Lindley, and H. Müller. The database wiki project: A general purpose platform for data curation and collaboration. *Sigmod Record*, 40(3):15–20, 2011.

[17] P. Buneman, A. Deutsch, and W.-C. Tan. A deterministic model for semistructured data. In *Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats*, 1998.

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

[19] D. Burdick, M. A. Hernández, H. Ho, G. Koutrika, R. Krishnamurthy, L. Popa, I. Stanoi, S. Vaithyanathan, and S. R. Das. Extracting, linking and integrating data from public sources: A financial case study. *IEEE Data Eng. Bulletin.*, 34(3):60–67, 2011.

[20] S.-Y. Chien, V. J. Tsotras, and C. Zaniolo. Efficient management of multiversion documents by object referencing. In *VLDB*, pages 291–300, 2001.

[21] J. Chomicki. Temporal query languages: A survey. In *ICTL*, pages 506–534, 1994.

[22] J. Chomicki and D. Toman. Temporal databases. In *Foundations of Artificial Intelligence*, pages 429–467. Elsevier, 2005.

[23] J. Chomicki, D. Toman, and M. H. Böhlen. Querying ATSQL databases with temporal logic. *ACM TODS*, 26(2):145–178, 2001.

[24] F. Currim, S. Currim, C. E. Dyreson, S. Joshi, R. T. Snodgrass, S. W. Thomas, and E. Roeder. τxschema: Support for data- and schema-versioned xml documents. Technical Report TR-91, TimeCenter at Aalborg University, Sep. 2009.

[25] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *CACM*, 51(1):107–113, Jan. 2008.

[26] D. DeWitt and J. Gray. Parallel database systems: the future of high performance database systems. *CACM*, 35(6):85–98, 1992.

[27] A. Doan, A. Halevy, and Z. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[28] X. L. Dong and F. Naumann. Data fusion - resolving data conflicts for integration. *PVLDB*, 2(2):1654–1655, 2009.

[29] The EDGAR Public Dissemination Service. http://www.sec.gov/edgar.shtml.

[30] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *TCS*, 336(1):89–124, 2005.

[31] A. Fuxman, P. G. Kolaitis, R. J. Miller, and W.-C. Tan. Peer data exchange. In *ACM PODS*, pages 160–171, 2005.

[32] D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-dimensional modal logics: theory and applications*. Elsevier, 2003.

[33] F. Geerts, A. Kementsietsidis, and D. Milano. Mondrian: Annotating and querying databases through colors and blocks. In *ICDE*, page 82, 2006.

[34] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.

[35] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio Grows Up: From Research Prototype to Industrial Tool. In *ACM SIGMOD*, pages 805–810, 2005.

[36] K. M. Hakkarainen, K. Hedna, M. Petzold, and S. Hägg. Percentage of patients with preventable adverse drug reactions and preventability of adverse drug reactions – a meta-analysis. *PLoS ONE*, 7:e33236, 03 2012.

[37] A. Halevy. Answering Queries Using Views: A Survey. *VLDB Journal*, pages 270–294, 2001.

[38] Internet Archive Wayback Machine. http://waybackmachine.org, Aug. 26, 2011.

[39] Information technology – Database languages – SQL – Part 2: Foundation (SQL/Foundation), 2011. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=53682,Dec.15,2011.

[40] C. S. Jensen and R. T. Snodgrass, editors. *Temporal Database Entries for the Springer Encyclopedia of Database Systems*. Springer, 2009. http://www.cs.arizona.edu/~rts/pubs/TRmerged.pdf.

[41] A. Jha. Meaningful use of electronic health records: The road ahead. *JAMA*, 304(15):1709–1710, 2010.

[42] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *ACM PODS*, pages 61–75, 2005.

[43] I. Koltsidas, H. Müller, and S. D. Viglas. Sorting hierarchical data in external memory for archving. *PVLDB*, 1(1):1205–1216, 2008.

[44] E. V. Kostylev and P. Buneman. Combining dependent annotations for relational algebra. In *ICDT*, pages 196–207, 2012.

[45] M. Lenzerini. Data Integration: A Theoretical Perspective. In *ACM PODS*, pages 233–246, 2002.

[46] M. Liu, M. Li, D. Golovnya, E. A. Rundensteiner, and K. Claypool. Sequence pattern query processing over out-of-order event streams. In *ICDE*, pages 784–795, 2009.

[47] D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing Implications of Data Dependencies. *ACM TODS*, 4(4):455–469, 1979.

[48] A. Marian, S. Abiteboul, G. Cobena, and L. Mignet. Change-centric management of versions in an xml warehouse. In *VLDB*, pages 581–590, 2001.

[49] H. J. Moon, C. Curino, A. Deutsch, C.-Y. Hou, and C. Zaniolo. Managing and querying transaction-time databases under schema evolution. *PVLDB*, 1(1):882–895, 2008.

[50] H. Müller, P. Buneman, and I. Koltsidas. Xarch: archiving scientific and reference data. In *ACM SIGMOD*, pages 1295–1298, 2008.

[51] NCBI. National Center for Biotechnology Information (NCBI) Ftp site. http://www.ncbi.nlm.nih.gov/Ftp/, Jun 10,2012.

[52] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating Web Data. In *VLDB*, pages 598–609, 2002.

[53] L. Qian, M. J. Cafarella, and H. V. Jagadish. Sample-driven schema mapping. In *ACM SIGMOD*, pages 73–84, 2012.

[54] C. Ré, J. Letchner, M. Balazinksa, and D. Suciu. Event queries on correlated probabilistic streams. In *ACM SIGMOD*, pages 715–728, 2008.

[55] R. T. Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer, 1995.

[56] R. T. Snodgrass and I. Ahn. Temporal databases. *IEEE Computer*, 19(9):35–42, 1986.

[57] The ++Spicy Schema Mapping System. http://www.db.unibas.it/projects/spicy.

[58] U. Srivastava and J. Widom. Flexible time management in data stream systems. In *ACM PODS*, pages 263–274, 2004.

[59] N. E. Taylor and Z. G. Ives. Reconciling while tolerating disagreement in collaborative data sharing. In *SIGMOD Conference*, pages 13–24, 2006.

[60] G. Weikum, N. Ntarmos, M. Spaniol, P. Triantafillou, A. A. Benczúr, S. Kirkpatrick, P. Rigaux, and M. Williamson. Longitudinal analytics on web archive data: It's about time! In *CIDR*, pages 199–202, 2011.

[61] H. Zhang, Y. Diao, and N. Immerman. Recognizing patterns in streams with imprecise timestamps. *PVLDB*, 3(1-2):244–255, 2010.

[62] H. Zhu, S. E. Madnick, and M. D. Siegel. Effective data integration in the presence of temporal semantic conflicts. In *Intl. Symp. on Temporal Representation and Reasoning*, TIME, pages 109–114, 2004.