Tolerant Ad Hoc Data Propagation with Error Quantification

Philipp Rösch

Database Technology Group
Technische Universität Dresden, Germany
philipp.roesch@tu-dresden.de

Abstract. Nowadays everybody uses a variety of different systems managing similar information, for example in the home entertainment sector. Unfortunately, these systems are largely heterogeneous, mostly with respect to the data model but at least with respect to the schema, making synchronization and propagation of data a daunting task. Our goal is to cope with this situation in a best-effort manner. To meet this claim, we introduce a symmetric instance-level matching approach that allows to establish mappings without any user interaction, schema information or dictionaries and ontologies. In awareness of dealing with inexact and incomplete mappings, the quality of the propagation has to be quantified. For this purpose, different quality dimensions like accuracy or completeness are introduced. Additionally, visualizing the quality allows users to evaluate the performance of the data propagation process.

1 Introduction

With the flood of gadgets managing personal information, data propagation becomes ever more important. Take the home entertainment sector, for example: a variety of products like MP3 players, hand-helds, mobile phones, car radios, hi-fi systems or PCs store your music files. Since users want to listen to their favorite songs wherever they are, there is a strong need for *ubiquitous access* to personal data, e.g. to favorite playlists. This process should be as simple as possible to allow for the smooth integration of new systems. However, aside from the propagation of data between users' gadgets, where once established mappings could be re-used for every data propagation process, there are also application scenarios where ad hoc data propagation is required. Imagine a rental car comes with a collection of music files. Here, users want to select songs according to their favorite playlists easily and quickly. Consequently, performing this task without the requirement of user interaction or expertise would be a great benefit. This scenario also applies to planes, trains or hotel rooms. Moreover, other types of personal data can be considered, like preferences and highscores of games or settings of programs.

Unfortunately, the huge variety of products and the many different manufacturers complicate this need. Different data models, like hierarchical, relational or flat models, as well as different schemas within the data models create a

large heterogeneity among the data. Together with the inconsistent availability of schema information, a perfect synchronization or propagation of the data will be impossible. Neither will the manufacturers provide mappings from their data models and schemas to all the other models nor does a standard format seem to evolve. Being in this unpromising situation, our goal has to be to propagate the data as efficiently as possible.

All in all, consider the application area: There are small datasets with no schema information available; the propagation of the data may not be perfect; and the user would not or even cannot manually establish these mappings. Therefore, we introduce an instance-level matching approach, where the key is to detect matchings between the current data in the documents. The pleasant consequence is that concentrating on the content allows mapping between documents with *arbitrary namings and structure*. The only prerequisites are 1) an intersection of the content, which is usually given in scenarios concerned with synchronization and data propagation, 2) the absence of complex data types, which is common if there is no schema, and 3) a hierarchical structure of the data.

Subsuming all demands above, we detect a strong need for

- a fully automated matching approach that does not rely on provided schema information, references like dictionaries or thesauri, and user interaction,
- a mapping and propagation technique which considers similarity matches and strongly varying sizes of the documents, and
- the quantification of the error which possibly occurred to give the user feedback on the quality of the results of the data propagation process.

The remainder of the paper is structured as follows: Section 2 gives a brief overview of related work. In Section 3, we introduce the symmetric instance-level matching, which avoids problems of schema matching approaches by efficiently comparing the content of the documents. In Section 4, we present a normalization approach of the matching results and show how to compute the mapping probabilities in order to establish meaningful mappings. The quantification of possible errors of the data propagation process is discussed in Section 5. Finally, a summary and open topics for future work are given in Section 6.

2 Related Work

In the area of data integration, the idea of best-effort data propagation is surprisingly disregarded. Here current solutions like [1–4] among others consider *schema matching*. These approaches require the existence of schema information as well as similar or semantically meaningful names of elements and attributes. Even if missing schema information may be created from the data, as shown in [5, 6], there often are generated schemas with cryptic namings. Additionally, even in the case of semantically meaningful names, application-specific dictionaries and synonym tables must be available. Furthermore, some approaches require user interaction [1, 7, 8] and can lead to incorrect interpretations of ambiguous names. Moreover, since these solutions claim to deliver perfect results and are

intended for large datasets containing hundreds of relations or documents with known schemas, they are oversized and inapplicable in this context.

Dealing with error quantification touches another research topic: *quality*. Unfortunately, there is no common definition in the literature for the ambiguous concept of quality. A variety of publications [9–14] show aspects of quality within different application scenarios and define more or less varying quality parameters. An overview can be found in [15, 16]. [9, 10, 17] care about the definition of an *overall measure* for the quality by combining different quality parameters. However, those approaches are mostly subjective and application-dependent. Quality-driven query optimization in integrated environments is given in [18]. Here, queries are optimized with respect to the quality of the result. Ideas from all of these different approaches can be adopted; however, they have to be adapted to create an objective quality measure of the data propagation process.

3 Preprocessing: Symmetric Instance-Level Matching

Visualizing the problem, Fig. 1 shows fragments of a sample source and target document containing information about music albums. Despite being small, even this example already points out problems of schema matching techniques; e.g. consider the element <title> that represents the name of a music album in the first data source, whereas it contains the name of a track in the second data source. Similarly the element <track> would probably be erroneously identified as a match. Besides avoiding invalid mappings due to equal or similar element names, we also have to find valid mappings of elements with different names but equal content, such as information on the release date, the artist, and the position of the track in the current example.

```
<favorites>
  <disc>
    <published>2002</published>
    <title>Hits</title>
    <artist>The Singer</artist>
    <rating>2</rating>
    <song>
      <name>firstTrack</name>
      <track>1</track>
      <duration>2:35</duration>
    </song>
    <song>...</song>
  </disc>
  ...
</favorites>
```

```
<playlist>
  <entry>
    <album>Hits</album>
    <year>2002</year>
    <genre>Pop</genre>
    <tracks>
      <track>
        <singer>The Singer</singer>
        <trackno>1</trackno>
        <title>firstTrack</title>
      </track>
      <track>...</track>
    </tracks>
  </entry>
  ...
</playlist>
```

Fig. 1. XML-fragments of a source and a target document

While this small example identifies problems of finding matches solely with schema information, naive instance-level matching also comes with some challenges we have to face. Although the data sources in the application scenario of this project are moderate in size, it would be very expensive to compare each element of one data source with each element of the other data source. So, to avoid unnecessary comparisons between incompatible types and domains, a *fingerprint* is created on the fly for each element[1] by generating the most specific data type and the domain. Elements with 'compatible' fingerprints, i.e. with the same data types and overlapping domains, are put into a cluster. Figure 2 shows the clusters of the source document (top) and the target document (bottom). The first shaded box of the source document shows the cluster of integers ranging from 1980 to 2005 and the second cluster contains the elements of type string. In the third cluster, elements of type integer ranging from 1 to 20 are collected and the last cluster manages elements with time information.

Now, *cluster groups* can be established by combining each cluster of the source document with all clusters of the target document with compatible fingerprints. Clusters with no compatible target cluster remain unlinked. In Fig. 2, these cluster groups are denoted by arrows between the clusters. In the current simple example, each source cluster is only connected with at most one target cluster, but this is no limitation.

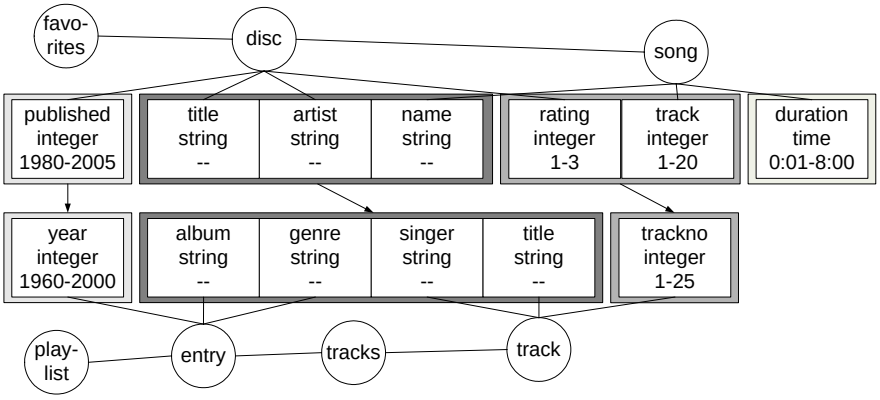

Fig. 2. Clusters (shaded) and cluster groups (linked by arrows)

For each cluster group a three-dimensional matrix can be constructed. The three dimensions are 1) the elements of the source document, 2) the elements of the target document, and 3) the direction of comparison. The reason for considering both directions of comparison is given in Sect. 4 while explaining the mapping. Figure 3 shows the matrix for elements of type integer ranging from 1 to 25.

[1] attributes are treated like elements

	$d_s \rightarrow d_t / d_t \rightarrow d_s$
d_s.rating (7)	7/15
d_s.track (84)	84/60
	d_t.trackno (60)

Fig. 3. Matrix for elements of type integer and with domain from 1 to 25

Within the cluster groups, each element of the source document is compared with each element of the target document and vice versa. While integer and floating-point numbers are compared for equality, strings additionally allow similarity matching, like edit distance or soundex. In general, specific comparison operations can be chosen for each data type. The number of matches for each element is annotated in the corresponding matrix. In Fig. 3, for example, the given numbers can be explained with the following sample: The source document d_s contains information about seven albums with a total of 84 tracks, while the target document d_t manages five albums with a total of 60 tracks. The number of occurrences of each element is given in brackets. The rating ranges from 1 to 3, so each of the seven rating entries matches with some track numbers. Moreover, consider that each possible rating occurs. Consequently, each of the first three <trackno>-elements matches with a <rating>-element resulting in a count of 15; that is, 7 of the 7 <rating>-elements of d_s match with some <trackno>-elements of d_t while only 15 of the 60 <trackno>-elements match with some <rating>-elements. For the track numbers, consider a simplified scenario where each album has 12 tracks resulting in 84 and 60 matches respectively.

4 Element Mapping

The generation of a mapping raises another challenge: How can documents of strongly varying sizes or with small intersections be compared? For example, consider the case that both the source and the target document manage 100 albums each and that the intersection consists of just one album. Most of the matches would provide values close to one, making conclusions about the mapping quality difficult. To overcome this problem, we propose a normalization of the matching results. This normalization requires the number of common objects managed in both documents, that is, the intersection size $s_\cap$. Unfortunately, $s_\cap$ is not known in advance; however it can be *estimated*. If there is a pair of elements with unique values[2] having the same number of matches in both directions, this number can be regarded as intersection size. Together with the number of occurrences of these elements, the scaling factors can be calculated. For the source document the scaling factor is given by

$$SF_{d_s} = \frac{occ_x}{s_\cap} \tag{1}$$

[2] The uniqueness of an element can be determined from its fingerprint.

with occ_x as the number of occurrences of the source element. The scaling factor for the target document SF_{d_t} can be calculated in the same way.

For the string-valued elements of the current example we have the unnormalized matching values given in Fig. 4.

	$d_s \rightarrow d_t / d_t \rightarrow d_s$	$d_s \rightarrow d_t / d_t \rightarrow d_s$	$d_s \rightarrow d_t / d_t \rightarrow d_s$	$d_s \rightarrow d_t / d_t \rightarrow d_s$
d_s.title (7)	4/4	0/0	0/0	2/2
d_s.artist (7)	0/0	0/0	6/60	0/0
d_s.name (84)	2/2	0/0	0/0	48/48
	d_t.album(5)	d_t.genre(5)	d_t.singer(60)	d_t.title(60)

Fig. 4. Matrix for elements of type string

Assuming that album names are unique, the pair (d_s.title, d_t.album) indicates an intersection size of $s_\cap = 4$. Using this intersection size and the number of occurrences of these two elements together with (1), we get the scaling factors of $SF_{d_s} = \frac{7}{4}$ for the source document and $SF_{d_t} = \frac{5}{4}$ for the target document. Once the scaling factors have been determined, they are applied to *all* values in *all* clusters where the normalized number of matches are calculated by

$$\bar{m}_{x,y} = SF_{d_p} * m_{x,y} \tag{2}$$

with $m_{x,y}$ as the number of matches of element x with respect to element y and $p \in s, t$. Obviously, an element cannot have more matches than occurrences, thus (2) has to be extended to

$$\bar{m}_{x,y} = max(SF_{d_p} * m_{x,y}, occ_x) \ . \tag{3}$$

Figure 5 shows the normalized values of the string-typed elements.

	$d_s \rightarrow d_t / d_t \rightarrow d_s$	$d_s \rightarrow d_t / d_t \rightarrow d_s$	$d_s \rightarrow d_t / d_t \rightarrow d_s$	$d_s \rightarrow d_t / d_t \rightarrow d_s$
d_s.title (7)	7/5	0/0	0/0	3.5/2.5
d_s.artist (7)	0/0	0/0	7/60	0/0
d_s.name (84)	3.5/2.5	0/0	0/0	84/60
	d_t.album(5)	d_t.genre(5)	d_t.singer(60)	d_t.title(60)

Fig. 5. Normalized elements of type string

Now, the mappings from elements of the source document to elements of the target document can be established. Therefore, the ratio of the (normalized) matches and the occurrences of an element are regarded, resulting in a mapping probability of

$$P_{x,y} = \frac{\bar{m}_{x,y}}{occ_x} \ . \tag{4}$$

However, this may lead to bad results. Consider the matching result of d_s.rating and d_t.trackno, which is $\bar{m}_{\text{rating,trackno}} = 7$. This would erroneously generate a perfect mapping $P_{\text{rating,trackno}} = 7/7 = 1$. Otherwise, if the mapping would be the other way around, the considered matching result would be $\bar{m}_{\text{trackno,rating}} = 18.75$, leading to a reasonable result of $P_{\text{trackno,rating}} = 18.75/60 \approx 0.3$. Actually, this is the reason for the *symmetric* instance-level matching. Now, the computation of the mapping probability is extended to

$$P_{x,y} = P_{y,x} = min\Big(\frac{\bar{m}_{x,y}}{occ_x}, \frac{\bar{m}_{y,x}}{occ_y}\Big) \tag{5}$$

taking both directions of comparison into account.

Figure 6 shows the mapping probabilities of the integer-valued elements with domain 1 to 25 (left) and the string-valued elements (right) of the current example. The possible mappings between elements which are not used for the final mapping generation are illustrated with dashed lines.

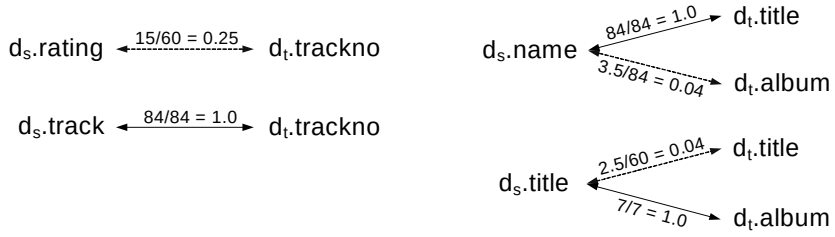

Fig. 6. Mapping probabilities

Now, we have mappings based on the content of the leaves of the hierarchical structure of the documents. However, this hierarchical structure gives further information about semantically meaningful mappings. The relationship between the elements represents *context information*. Consequently, the structure of the content has to be taken into account to further improve the results.

To utilize this additional information, we infer *higher-level mappings*. Beginning at the bottom of the hierarchy, for each node we check if all the mapping partners of its child nodes themselves are children of one common parent node. In this case, the corresponding parents can be seen as mapping candidates. Child nodes without mapping partners are dismissed and considered as not transferable. Now, this information can be used to adapt the mapping probabilities in a way that mappings between child nodes of higher-level mapping candidates are regarded as more likely, since they appear in a similar context, while possible mappings between elements within different contexts are accounted as rather unlikely.

Take the <song>-node of the current example. Without a matching partner, <duration> is dismissed. The other child nodes (<name> and <track>)

both have matching partners (<title> and <trackno>), which are child nodes of <track>, thus leading to a higher-level mapping between d_s.song and d_t.track (cf. Fig. 7).

```
<song>                                  <track>
  <name>firstTrack</name>                 <singer>who knows</singer>
  <track>1</track>                        <trackno>1</trackno>
  <duration>2:35</duration>               <title>firstTrack</title>
</song>                                 </track>
```

Fig. 7. Section of the XML-fragments illustrating higher-level mappings

Obviously, a somewhat weaker proceeding is reasonable since the structure of the given documents would rather be equal. Consider the direction from the target to the source document. Here, the element <singer> would prevent the meaningful higher-level mapping between d_s.song and d_t.track since the mapping partner <artist> sits on an upper level. Consequently, the results of both directions have to be regarded in this stage as well in order to identify some higher-level mappings. Finally, the partner with the highest mapping probability is chosen for each element to establish the mapping.

5 Error Quantification

Since the data propagation process is fully automated and thus not supervised, some errors may occur. In order to give the user feedback on the quality of the data propagation process, possible errors have to be quantified. One aspect of the result's quality is its *completeness*. This aspect is generally independent of the matching and mapping procedure and is caused by the differences within the schemas of the source and the target document. There are two facets of completeness. On the one hand, we have elements of the source document having no counterpart in the target document and thus cannot be transferred. Those information is *lost*. On the other hand, we have elements of the target document without any counterpart in the source document, which can thus not be filled. The information is *unknown*.

Another aspect of the result's quality depends on the matching procedure. Here, the algorithms for defining similarity of values affect the overall result. If they are too 'tight' some meaningful mappings may be missed; otherwise, if they are too 'lax' some senseless mappings may occur. This aspect is referenced to as *matching accuracy*.

Closely related to the matching accuracy is the *mapping accuracy*. Again, two facets can be considered. The two factors which contribute to the mapping probability and thus to the mapping accuracy are the number of similar or equal values as well as the respective similarity value itself.

Now, to quantify these quality parameters, information of the matching and mapping procedure have to be evaluated. The number of lost and unknown elements can be determined from the clustering process, where unlinked clusters lead to incompleteness. Also the elements which are part of cluster groups but are not part of the final mapping cause incomplete results. Accuracy information can be computed with the number and the similarity of the matchings managed in the matrices of the cluster groups.

6 Conclusion and Future Work

In this paper, we presented an approach for propagating data, like personal information, in a fast and easy way. By concentrating on the content, we avoid—contrary to existing techniques—the requirement of schema information, user interaction and references like dictionaries or thesauri. Furthermore, we presented a mapping technique that copes with strongly varying sizes of documents and defined some quality parameters to give the user feedback on the quality of the results of the data propagation process.

Our prototype TRANSMITTER (ToleRANt ScheMa-Independent daTa propagaTion with ERror quantification), a JAVA implementation of the functionality described above, shows already promising results. Nevertheless, some tasks remain open. To reduce the complexity of the matching, the fingerprints have to be refined. Here, domain information for strings, like patterns of regular expressions, have to be regarded. Also, it has to be examined if histograms can be used in a meaningful way. The resolution of conflicts within the establishment of mappings has to be considered. Moreover, the adaptation of the mapping probabilities based on the higher-level mappings has to be advanced.

In addition, the quality feedback can be extended by taking further quality parameters into account, and even more important, a convenient representation of the current quality for an intuitive interpretation has to be developed.

References

1. Do, H.H., Rahm, E.: COMA - A System for Flexible Combination of Schema Matching Approaches. In: Proceedings of the 28th VLDB Conference, Hong Kong, China. (2002) 610–621
2. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic Schema Matching with Cupid. In: Proceedings of the 27th VLDB Conference, Rome, Italy. (2001) 49–58
3. Milo, T., Zohar, S.: Using Schema Matching to Simplify Heterogeneous Data Translation. In: Proceedings of the 24th VLDB Conference, New York City, USA. (1998) 122–133
4. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In: Proceedings of the 18th International Conference on Data Engineering (ICDE'02), San Jose, USA. (2002) 117–128
5. Wang, Q.Y., Yu, J.X., Wong, K.F.: Approximate Graph Schema Extraction for Semi-Structured Data. In: Proceedings of the 7th EDBT Conference, Konstanz, Germany. (2000) 302–316

6. Goldman, R., Widom, J.: DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In: Proceedings of the 2rd VLDB Conference, Athens, Greece. (1997) 436–445
7. Mandreoli, F., Martoglia, R., Tiberio, P.: Approximate Query Answering for a Heterogeneous XML Document Base. In: Proceedings of WISE 2004, Brisbane, Australia. (2004) 337–351
8. Rahm, E., Bernstein, P.A.: On Matching Schemas Automatically. Technical Report MSR-TR-2001-17, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399 (2001)
9. Bovee, M., Srivastava, R.P., Mak, B.: A Conceptual Framework and Belief-Function Approach to Assessing Overall Information Quality. International Journal of Intelligent Systems **18**(1) (2003) 51–74
10. Lee, Y.W., Strong, D.M., Kahn, B.K., Wang, R.Y.: AIMQ: A Methodology for Information Quality Assessment. Information & Management **40** (2002) 133–146
11. Martinez, A., Hammer, J.: Making Quality Count in Biological Data Sources. In: Proceedings of the IQIS Workshop, Baltimore, USA. (2005) 16–27
12. Motro, A., Rakov, I.: Estimating the Quality of Databases. In: Proceedings of the 3rd FQAS Conference, Roskilde, Denmark. (1998) 298–307
13. Naumann, F., Rolker, C.: Do Metadata Models meet IQ Requirements? In: Proceedings of the 4th IQ Conference, Cambridge, USA. (1999) 99–114
14. Naumann, F., Rolker, C.: Assessment Methods for Information Quality Criteria. In: Proceedings of the 5th IQ Conference, Cambridge, USA. (2000) 148–162
15. Scannapieco, M., Missier, P., Batini, C.: Data Quality at a Glance. Datenbank-Spektrum **14** (2005) 6–14
16. Tayi, G.K., Ballou, D.P.: Examining Data Quality - Introduction. Communications of the ACM **41**(2) (1998) 54–57
17. Pipino, L., Lee, Y.W., Wang, R.Y.: Data Quality Assessment. Communications of the ACM **45**(4) (2002) 211–218
18. Naumann, F.: From Databases to Information Systems - Information Quality Makes the Difference. In: Proceedings of the 6th IQ Conference, Cambridge, USA. (2001) 244–260