IMPrECISE: Good-is-good-enough data integration

Ander de Keijzer•, Maurice van Keulen•

•University of Twente
Postbus 217, 7500AE Enschede
The Netherlands
{a.dekeijzer;m.vankeulen}@utwente.nl

***Abstract*— IMPrECISE is an XQuery module that adds probabilistic XML functionality to an existing XML DBMS, in our case MonetDB/XQuery. We demonstrate probabilistic XML and data integration functionality of IMPrECISE. The prototype is configurable with domain knowledge such that the amount of uncertainty arising during data integration is reduced to an acceptable level, thus obtaining a "good is good enough" data integration with minimal human effort.**

I. Introduction

Data integration is a challenging problem in many application areas as it usually requires manual resolution of semantic issues like schema heterogeneity, data overlap, and data inconsistency, before data sources can be meaningfully used in an integrated way [1]. We believe, however, that data integration can be made into less of an obstacle by striving for less perfect, but near-automatic integration, i.e., "good is good enough" data integration. Data integration problems are symptoms of semantic uncertainty. Therefore, being able to properly handle uncertainty in data can provide for near-automatic data integration. Parts of the data that require tighter integration can be improved incrementally while the integrated source is being used.

The basis of our approach is depicted in Figure 1 [2], [3]. We view a database as a representation of information about the real world based on observations. In this view, data integration is a means to combine observations stored in different data sources. Since observations may conflict, the DBMS may become uncertain about the state of the real world. In particular, the DBMS may be uncertain about data overlap, i.e., whether or not two data items refer to the same real-world object (rwo). We have chosen a representation of uncertain data that compactly represents in one XML tree all possible states the real world can be in, the *possible worlds*, for which an intuitive and consistent theory exists. In this way, it is not necessary that all semantic problems be solved before the integrated data can be used in a meaningful way. Posing queries to an uncertain database means that an application may receive several possible answers. In many application areas, this suffices if those answers can be properly ranked according to likelihood. Furthermore, a user interacting with an application can provide feedback on the correctness of these answers [4]. Feedback on query answers can be traced back to possible worlds and be used to remove data related to *impossible* worlds from the database, hence incrementally improving the integration result.

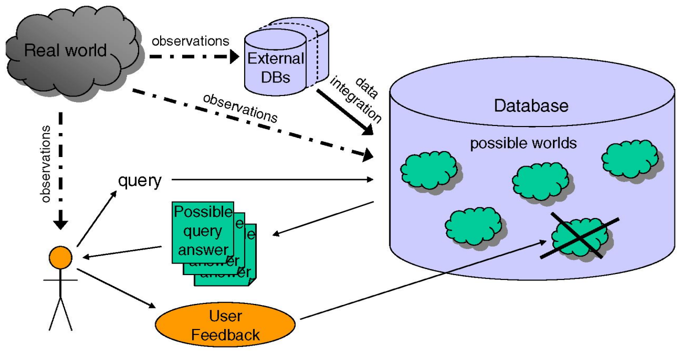

Fig. 1. Information cycle

Our ideas are consistent with those of the DSSP approach (DataSpace Support Platform) [5]. Several other groups are developing system support for managing uncertain data such as Trio[6], Orion [7] and MystiQ [8]. In contrast with these systems, IMPrECISE uses the XML data model instead of relational. The main reasons for this choice are that XML is the prominent data model for data exchange and integration, and its tree structure naturally resembles decision trees [2]. Other XML-based approaches are Fuzzy trees [9], PXML [10], and ProTDB [11].

II. Probabilistic XML

To capture uncertainty in the XML datamodel, we introduce two new node types: probability nodes ($\bigtriangledown$) and possibility nodes ($\circ$). The root node of the document is always a probability node. Child nodes of probability nodes are always possibility nodes. Each possibility node has an associated probability, which is the probability that the node and its subtree exists. Sibling possibility nodes are mutually exclusive, hence probability nodes indicate *choices*. Child nodes of possibility nodes are regular XML nodes ($\bullet$). Child nodes of regular XML nodes are probability nodes. This data model defines a layered XML document where all nodes on the same level have the same type. If all probability nodes have only one child node and these possibility nodes have an associated probability of 1, then the document is certain. A formal definition of the probabilistic XML data model is given in [2]. An example probabilistic XML tree is given in Figure 2 in which uncertainty about phone numbers of people named "John" is captured. The document could be the result of the

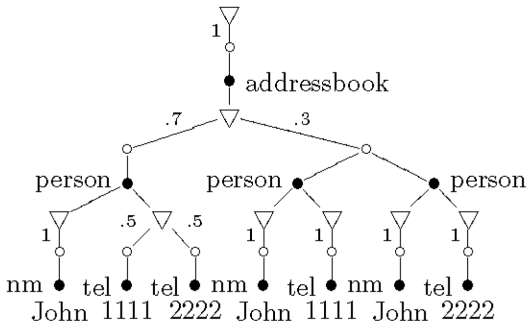

Fig. 2. Example probabilistic XML tree.

integration of two address books, both containing a person named "John", where the first address book lists "1111" as John's phone number, and the second "2222". The example tree represents three possible worlds:

- There is one person John with phone number 1111,
- There is one person John with phone number 2222,
- There are two persons named John, one with phone number 1111 and the other with phone number 2222.

III. Probabilistic Integration

Although a data integration system should definately support schema integration, we consider it to be a separate issue. We assume therefore that the schemas of data sources are already aligned.

The probabilistic integration process is executed in a recursive fashion starting from the roots of both source documents (see Figure 3). The integration function tries to match the child nodes of both sources. Two child nodes *match* if they refer to the same rwo. For example, two person elements match if they refer to the same persons in real life. In many cases, this can't be established with certainty, so the system needs to consider two cases: the two person elements refer to two different persons or they refer to the same person. For two sequences of persons, this may create many different combinations of possibilities limited by those possibilities the system can rule out based on a DTD or other semantical knowledge. In Figure 2, we depicted the final result where the DTD specified that persons also only have one phone number, hence the possibility of John having two phone numbers is rejected. A complete description of the integration process is given in [2]

IV. System Overview

The global architecture of the IMPrECISE system is given in Figure 4. The system is built as XQuery modules on top of the XML DBMS MonetDB/XQuery [12]. The bottom layer contains all functionality related to managing uncertainty in data based on the probabilistic XML approach. The middle layer contains the data integration functionality. A specific component, called "The Oracle", determines the probability that two XML elements refer to the same rwo based on knowledge rules (see Section V).

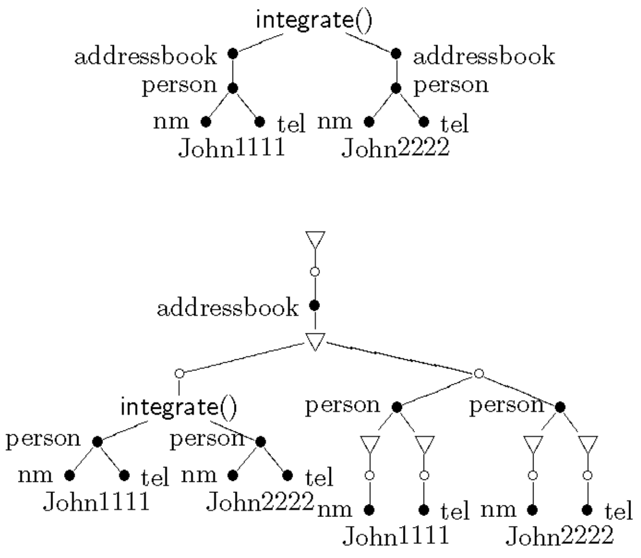

Fig. 3. Integration process

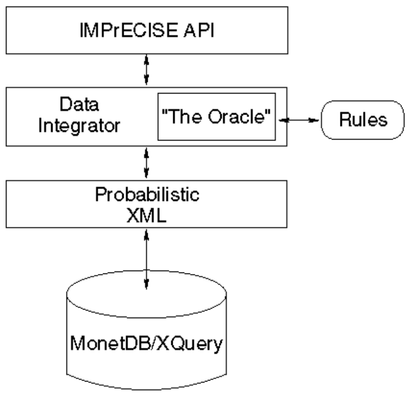

Fig. 4. Architecture of IMPrECISE

V. Possibility reduction

We experiment with integrating metadata of movies from two different data sources: IMDB and an MPEG-7 document. We aligned the schemas and can select some data about movies like title, year, genres and directors. The sources use different conventions for, e.g., naming directors, so these never match exactly.

In theory, data sources can be integrated fully automatically using our method. Data integration, however, quickly results in an exploding number of theoretical possibilities if the system contains too little semantical knowledge. Semantical knowledge is given to "The Oracle" in terms of rules, which make statements about when, with certainty, two elements match or not. The rules need to be as simple as possible, because the purpose of probabilistic integration is to significantly reduce manual effort, so rule specification overhead should be minimal. The number of possibilities the system needs to handle is related to the effectiveness of the rules to make absolute decisions.

Effective rules	**#nodes** (×1000)
none	13958
Genre rule	6015
Movie title rule	243
Genre and movie title rule	154
Genre, movie title and year rule	29

TABLE I
EFFECT OF RULES ON UNCERTAINTY

We claim that in typical situations only simple rules suffice for reduction to an acceptable level [3]. For example, integrating 6 movies produced in 1995 from the MPEG-7 source with 60 movies from the IMDB-source (of which two refer to the same rwo), only on two occasions "The Oracle" could not make an absolute decision. The integrated document of about 3500 nodes compactly stores the resulting 4 possible worlds.

The abovementioned rules can be divided into generic and domain-specific rules. Examples of generic rules:

- Two `deep-equal` elements refer to the same rwo.
- No two siblings in one source refer to the same rwo.

Example of domain-specific rules:

- *Genre rule*: no typos occur in genres
- *Title rule*: two movies cannot match if their titles are not sufficiently similar.
- *Year rule*: movies of different years cannot match.

To put the integration system to the test, we also experimented with confusing conditions such as integrating sources that contain sequels. For example, taking 2 'Mission Impossible' sequels, 2 'Die Hard' sequels, and 2 'Jaws' sequels for which only 1 each refers to the same rwo as in the other source, results in an integrated document of 14 million nodes with only the generic rules. By adding the simple domain-specific rules below, the amount of uncertainty, hence also the number of nodes can be brought down to 29 thousand (see Table I), which is *good enough for querying*.

The amount of uncertainty is often measured in terms of the number of possible worlds. We find this a rather deceiving measure, because in the presence of many *independent* possibilities, the measure grows exponentially. For obtaining a good view on scalability, we prefer to look at the number of nodes used to represent these possible worlds in the database. In Figure 5, we show the results of integrating 6 movies of our MPEG-7 source with a growing number of movies from the IMDB-source. Again to put the integration method to the test, we selected only sequels, TV-shows, etc. with 'Impossible Mission', 'Jaws', and 'Die Hard' in the title. In such a confusing setting, the amount of uncertainty grows quickly.

Note that the latter experiments are executed under very confusing conditions, so confusing that even humans cannot make absolute decisions. When comparing the integration of 6 with 60 movies under confusing and typical conditions, we see that the size of the integration result jumps from 3500 nodes to 1,5 million, a significant increase of course, but still manageable by our system. Note also that reduction should not be pushed too far, because eliminating valid possibilities

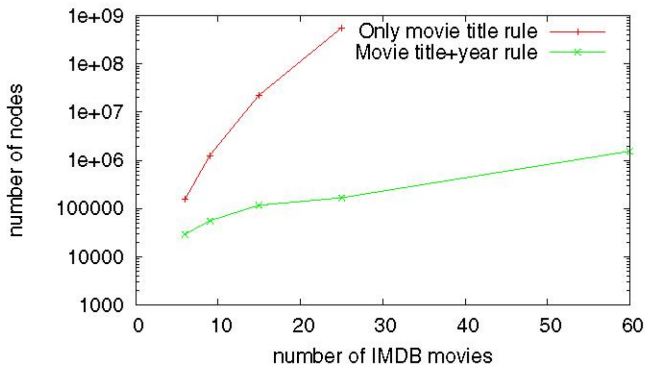

Fig. 5. Influence of rules on scalability

reduces the quality of query answers. We are currently setting up answer quality experiments.

VI. PROBABILISTIC QUERYING

Even in the presence of much uncertainty, a probabilistic database can still be queried effectively. In theory, the semantics of a query is the set of possible answers obtained by evaluating the query in each of the possible worlds separately. Although this ordinarily creates many possible answers, query answers from different possible worlds are often the same. Because XQuery answers are always sequences, we can construct an amalgamated answer by merging and ranking the elements of all possible answers.

The effectiveness of our approach to querying a probabilistic database can be shown with a few examples posed to an integration result under confusing conditions, more specifically a probabilistic database of 33856 possible worlds. Our first example is a query asking for horror movies:

```
//movie[.//genre="Horror"]/title
```

Even though the integrated document contains thousands of possible worlds, the ranked answer contains only two movies: 'Jaws' and 'Jaws 2' with an equal rank of 97%. These were indeed the only two movies classified as 'Horror' in the data sources. The 'missing' 3% are due to some worlds that are, though very unlikely, still possible under the given set of rules. Note that although there is much confusion between the two movies, the query has a perfectly usable answer.

The second example exhibits stronger effects of uncertainty during data integration. We query for movies directed by somebody named 'John':

```
//movie[some $d in .//director
        satisfies contains($d,"John")]/title
```

'Mission: Impossible II' is directed by 'John Woo' and 'Die Hard: With a Vengeance' by 'John McTiernan'. Due to the possibility that that the 'II' may be a typing mistake, the query produces the answer below. The incorrect third answer has a low probability though.

100%	Die Hard: With a Vengeance
96%	Mission: Impossible II
21%	Mission: Impossible

VII. The demonstration

The IMPrECISE system is a probabilistic XML database system which supports near-automatic integration of XML documents. What is required of the user is to configure the system with a few simple knowledge rules allowing the system to sufficiently eliminate nonsense possibilities. We demonstrate the integration process using varying degrees of confusion and different sets of rules.

Even when an integrated document still contains much uncertainty, it can be queried effectively. The system produces a sequence of possible result elements ranked by likelihood. User feedback on query results further reduces uncertainty which in a sense continues the semantic integration process incrementally. We demonstrate querying on integrated documents and measure answer quality with adapted precision and recall measures [13]. The user feedback mechanism has not been implemented, hence cannot be demonstrated yet.

IMPrECISE has been implemented as an XQuery module for the XML DBMS MonetDB/XQuery. Therefore, the demo also illustrates the power of this XML DBMS and of XQuery as both a query and programming language.

References

[1] A. Doan and A. Halevy, "Semantic integration research in the database community: A brief survey," *AI Magazine*, 2005.

[2] M. v. Keulen, A. d. Keijzer, and W. Alink, "A probabilistic XML approach to data integration," in *Proceedings of ICDE, Tokyo, Japan*, 2005, pp. 459–470. [Online]. Available: http://db.cs.utwente.nl/Publications/PaperStore/db-utwente-41064AD3.pdf

[3] A. de Keijzer, M. van Keulen, and Y. Li, "Taming data explosion in probabilistic information integration," in *On-line Pre-Proceedings of IIDB, Munich, Germany*, 2006, pp. 82–86, position paper. http://ssi.umh.ac.be/iidb.

[4] A. de Keijzer and M. van Keulen, "User feedback in probabilistic xml," Centre for Telematics and Information Technology, Univ. of Twente, Enschede, The Netherlands, Tech. Rep. TR-CTIT-07-25, March 2007, iSSN 1381-3625.

[5] A. Halevy, M. Franklin, and D. Maier, "Principles of dataspace systems," in *Proceedings of PODS, Chicago, IL, USA*, 2006, pp. 1–9.

[6] M. Mutsuzaki, M. Theobald, A. de Keijzer, J. Widom, P. Agrawal, O. Benjelloun, A. D. Sarma, R. Murthy, and T. Sugihara, "Trio-One: Layering uncertainty and lineage on a conventional DBMS," in *Proceedings of CIDR, Monterey, USA.* Online publication: www.crdrdb.org, 2007, pp. 269–274.

[7] R. Cheng, S. Singh, and S. Prabhakar, "U-DBMS: A database system for managing constantly-evolving data," in *Proceedings of VLDB, Trondheim, Norway*, 2005, pp. 1271–1274.

[8] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu, "MYSTIQ: a system for finding more answers by using probabilities," in *Proceedings of SIGMOD, Baltimore, Maryland, USA*, 2005, pp. 891–893.

[9] S. Abiteboul and P. Senellart, "Querying and updating probabilistic information in XML," in *Proceedings of EDBT, Munich, Germany*, 2006, pp. 1059–1068, INCS 3896.

[10] E. Hung, L. Getoor, and V. Subrahmanian, "PXML: A probabilistic semistructured data model and algebra," in *Proceedings of ICDE*, 2003.

[11] A. Nierman and H. Jagadish, "ProTDB: Probabilistic data in XML," in *Proceedings of VLDB*, 2002. [Online]. Available: citeseer.nj.nec.com/nierman02protdb.html

[12] P. Boncz, T. Grust, M. van Keulen, S. Manegold, J. Rittinger, and J. Teubner, "MonetDB/XQuery: a fast XQuery processor powered by a relational engine," in *Proceedings of SIGMOD, Chicago, IL, USA*, 2006, pp. 479–490.

[13] A. de Keijzer and M. van Keulen, "Quality measures in uncertain data management," in *Proceedings of SUM, Washington, DC, USA*, ser. LNCS, vol. 4772, 2007, pp. 104–115.