P

Probabilistic Data Integration

Maurice Van Keulen
Faculty of EEMCS, University of Twente,
Enschede, The Netherlands

Synonyms

Uncertain data integration

Definitions

Probabilistic data integration (PDI) is a specific kind of data integration where integration problems such as inconsistency and uncertainty are handled by means of a probabilistic data representation. The approach is based on the view that data quality problems (as they occur in an integration process) can be modeled as uncertainty (van Keulen 2012), and this uncertainty is considered an important result of the integration process (Magnani and Montesi 2010).

The PDI process contains two phases (see Fig. 2): (i) a quick partial integration where certain data quality problems are not solved immediately, but explicitly represented as uncertainty in the resulting integrated data stored in a probabilistic database; (ii) continuous improvement by using the data – a probabilistic database can be queried directly resulting in possible or approximate answers (Dalvi et al. 2009) – and gathering evidence (e.g., user feedback) for improving the data quality.

A *probabilistic database* is a specific kind of DBMS that allows storage, querying, and manipulation of uncertain data. It keeps track of alternatives and the dependencies among them.

Overview

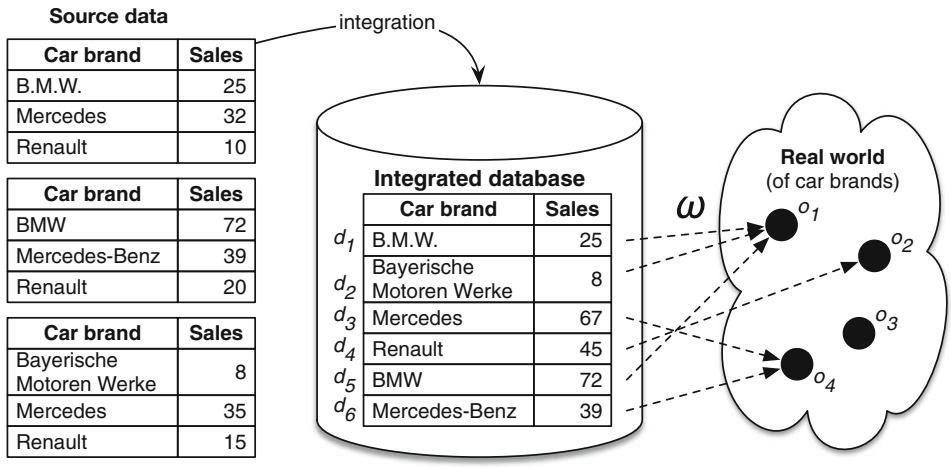
This chapter explains a special kind of data integration approach, called Probabilistic Data Integration. We first define the concept as well as the related notion of a Probabilistic Database. After presenting an illustrative example that is used throughout the chapter, a motivation for the PDI approach is given based on its strengths in dealing with the main problems in data integration.

The technical part of this chapter first addresses the notion of a probabilistic database capable of storing and querying probabilistic representations, and then explains which probabilistic representations can be used for various data integration problems. We conclude with example applications and future developments.

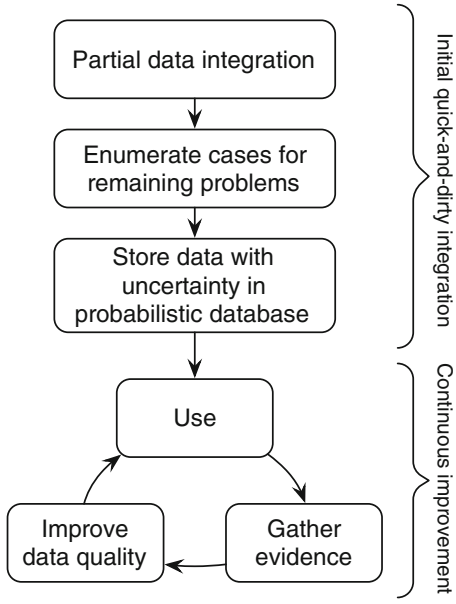
Example

As a running example taken from van Keulen (2012), imagine a manufacturer of car parts supplying major car brands. For a preferred customer program (preferred customer defined as one with sales over 100) meant to avoid loosing important customers to competitors, data of three production sites needs to be integrated.

S. Sakr, A. Zomaya (eds.), *Encyclopedia of Big Data Technologies*,
https://doi.org/10.1007/978-3-319-63962-8_18-1



Probabilistic Data Integration, Fig. 1 An uncareful data integration leading to a database with semantic duplicates



Probabilistic Data Integration, Fig. 2 Probabilistic data integration process (van Keulen and de Keijzer 2009)

Figure 1 shows an example data integration result and a part of the real world it is supposed to represent. Observe that at first glance there is no preferred customer due to *semantic duplicates*: The "same" car brand occurs more than once under different names because of different conventions. Importantly, data items d_3 and d_6 refer to the same car brand, and their combined sales is 106, so "Mercedes-Benz" should be a preferred customer.

Typical data-cleaning solutions support duplicate removal that merges data items when they likely refer to the same real-world object, such as d_1 and d_5 merged into a new data item d_{15}; $d_3, d_6 \mapsto d_{36}$ analogously. But, it is quite possible that an algorithm would not detect that also d_2 refers to "BMW." Note that this seemingly small technical glitch has a profound business consequence: it determines whether "BMW" is considered a preferred customer or not, risking loosing it to a competitor.

What do we as humans do if we suspect that "BMW" stands for "Bayerische Motoren Werke"? *We are in doubt*. Consequently, humans simply consider both cases, reason that "BMW" *might* be a preferred customer and act on it if we decide that it is important and likely enough. It is this behavior of "doubting" and "probability and risk assessment" that probabilistic data integration is attempting to mimic.

Motivation

"Data integration involves *combining* data residing in different *sources* and providing users with a *unified view* of them" (Lenzerini 2002).

Applications where uncertainty is unavoidable especially call for a probabilitic approach as the highlighted terms in the definition illustrate:

- It may be hard to extract information from certain kinds of sources (e.g., natural language, websites).
- Information in a source may be missing, of bad quality, or its meaning is unclear.
- It may be unclear which data items in the sources should be combined.
- Sources may be inconsistent complicating a unified view.

Typically, data integration is an iterative process where mistakes are discovered and repaired, analyses are repeated, and new mistakes are discovered . . . Still, we demand from data scientists that they act responsibly, i.e., they should know and tell us about the deficiencies in integrated data and analytical results.

Compared to traditional data integration, probabilistic data integration allows:

- postponement of solving data integration problems, hence, provides an initial integration result much earlier;
- better balancing of trade-off between development effort and resulting data quality;
- an iterative integration process with smaller steps (Wanders et al. 2015);
- leveraging human attention based on feedback; and
- more robustness being less sensitive to wrong settings of thresholds and wrong actions of rules (van Keulen and de Keijzer 2009).

Probabilistic Databases

Probabilistic data integration hinges on the capability to readily store and query a voluminous probabilistic integration result as provided by a probabilistic database. The two main challenges of a probabilistic database is that it needs both to *scale* to large data volumes and also to do *probabilistic inference* (Dalvi et al. 2009).

The formal semantics is based on *possible worlds*. In its most general form, a probabilistic database is a probability space over the possible contents of the database. Assuming a single table, let I be a set of tuples (records) representing that table. A probabilistic database is a discrete probability space $\text{PDB} = (W, \text{P})$, where $W = \{\text{I}_1, \text{I}_2, \ldots, \text{I}_n\}$ is a set of possible instances, called possible worlds, and $\text{P} : W \rightarrow [0, 1]$ is such that $\left(\sum_{j=1..n} \text{P}(\text{I}_j)\right) = 1$.

In practice, one can never enumerate all possible worlds; instead a more concise representation is needed. Many representation formalisms have been proposed differing a.o. in expressiveness (see Panse 2015, Chp.3 for a thorough overview).

Figure 3 shows a probabilistic integration result of our running example of Fig. 1 where possible duplicates are probabilistically merged; see section "Record Level". The used representation formalism is based on U-relations (Antova et al. 2008), which allows for dependencies between tuples, for example, tuples d_3 and d_6 (top left in Fig. 3) both exist together or are both absent.

Relational probabilistic database systems that, to a certain degree, have outgrown the laboratory bench include MayBMS (Koch 2009; Antova et al. 2009), Trio (Widom 2004), and MCDB (Jampani et al. 2008). MayBMS and Trio focus on tuple-level uncertainty where probabilities are attached to tuples, while MCDB focuses on attribute-level uncertainty where a probabilistic value generator function captures the possible values for the attribute.

Besides probabilistic relational databases, probabilistic versions of other data models and associated query languages can be defined by attaching a probabilistic "sentence" to data items and incorporating probabilistic inference in the semantics of the query language that adheres to the possible worlds semantics (Wanders and van Keulen 2015). For example, several probabilistic XML (Abiteboul et al. 2009; van Keulen and de Keijzer 2009) and probabilistic logic formalisms have been defined (Fuhr 2000; Wanders et al. 2016; De Raedt and Kimmig 2015).

Probabilistic Data Integration, Fig. 3
Example of a probabilistic database (resulting from indeterministic deduplication of Fig. 1) with a typical query and its answer (Taken from van Keulen 2012)

PDB

	car	sales	
d_1	…	25	$(r_1 \mapsto 0)$
d_2	…	8	$(r_1 \mapsto 0)$
d_5	…	72	$(r_1 \mapsto 0)$
d_{15}	…	97	$(r_1 \mapsto 1)$
d_2	…	8	$(r_1 \mapsto 1)$
d_{125}	…	105	$(r_1 \mapsto 2)$
d_4	…	45	
d_3	…	67	$(r_2 \mapsto 0)$
d_6	…	39	$(r_2 \mapsto 0)$
d_{36}	…	106	$(r_2 \mapsto 1)$

Worlds

rva	P	
$(r_1 \mapsto 0)$	0.1	'd_1, d_2, d_5 different'
$(r_1 \mapsto 1)$	0.6	'd_1, d_5 same'
$(r_1 \mapsto 2)$	0.3	'd_1, d_2, d_5 same'
$(r_2 \mapsto 0)$	0.2	'd_3, d_6 different'
$(r_2 \mapsto 1)$	0.8	'd_3, d_6 same'

$Q =$
```
SELECT SUM(sales)
  FROM carsales
  WHERE sales≥ 100
```
'sales of preferred customers'

All possible worlds with their answer to Q

	World descr.	World	Probability	Q
I_1	$(r_1 \mapsto 0), (r_2 \mapsto 0)$	$\{d_1, d_2, d_3, d_4, d_5, d_6\}$	$0.1 \cdot 0.2 = 0.02$	0
I_2	$(r_1 \mapsto 1), (r_2 \mapsto 0)$	$\{d_{15}, d_2, d_3, d_4, d_6\}$	$0.6 \cdot 0.2 = 0.12$	0
I_3	$(r_1 \mapsto 2), (r_2 \mapsto 0)$	$\{d_{125}, d_3, d_4, d_6\}$	$0.3 \cdot 0.2 = 0.06$	105
I_4	$(r_1 \mapsto 0), (r_2 \mapsto 1)$	$\{d_1, d_2, d_{36}, d_4, d_5\}$	$0.1 \cdot 0.8 = 0.08$	106
I_5	$(r_1 \mapsto 1), (r_2 \mapsto 1)$	$\{d_{15}, d_2, d_{36}, d_4\}$	$0.6 \cdot 0.8 = 0.48$	106
I_6	$(r_1 \mapsto 2), (r_2 \mapsto 1)$	$\{d_{125}, d_{36}, d_4\}$	$0.3 \cdot 0.8 = 0.24$	211

Possible answers

sum(sales)	P
0	0.14
105	0.06
106	0.56
211	0.24

Other derivable figures

description	sum(sales)	P
Minimum	0	0.14
Maximum	211	0.24
Answer most likely world	106	0.48
Most likely answer	106	0.56
Sec. most likely answer	211	0.24
Expected value	116.3	*N.A.*

Probabilistic Data Integration

In essence probabilistic data integration is about finding probabilistic representations for data integration problems. These are discussed on three levels: attribute value, record, and schema level.

Value Level

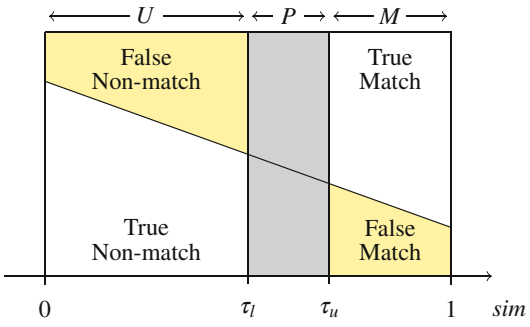
Inconsistency and ambiguity Integrated sources may not agree on the values of certain attributes, or it is otherwise unknown which values are correct. Some examples: text parsing may be ambiguous: in splitting my own full name "Maurice Van Keulen," is the "Van" part of my first name or my last name? Differences in conventions: one source may use first name-last name (as customary in the West) and another last name-first name (as customary in China). Information extraction: is a phrase a named entity of a certain type or not?

In the formalism of the previous section, this is represented as:

	Firstname	Lastname	
d_1^a	Maurice Van	Keulen	$(r_4 \mapsto 0)$
d_1^b	Maurice	Van Keulen	$(r_4 \mapsto 1)$
d_2^a	Zhang	Li	$(r_5 \mapsto 0)$
d_2^b	Li	Zhang	$(r_5 \mapsto 1)$
d_3	Paris	Hilton	$(r_6 \mapsto 0)$

where r_i $(i \in \{4, 5, 6\})$ govern the uncertainty which names are correct or preferred.

Data imputation A common approach to dealing with missing values is data imputation, i.e., using a most likely value and/or a value that retains certain statistical properties of the data



Probabilistic Data Integration, Fig. 4 Grey area in tuple matching (Taken from Panse et al. 2013)

set. Especially for categorical attributes, imputing with a wrong value can have grave consequences. In general, an imputation method is a classifier that predicts a most suitable value based on the other values in the record or data set. A classifier can easily not only predict one value but several possible ones each with an associated probability of suitability. By representing the uncertainty around the missing value probabilistically, the result is more informative and is more robust against imperfect imputations.

Record Level

Semantic duplicates, entity resolution A semantic duplicate is almost never detected with absolute certainty unless both records are identical. Therefore, there is a gray area of record pairs that may or may not be semantic duplicates. Even if an *identifier* is present, in practice it may not be perfectly reliable. For example, it has once been reported in the UK that there were 81 million National Insurance numbers but only 60 million eligible citizens.

Traditional approaches for deduplication are based on pairwise tuple comparisons. Pairs are classified into *matching* (M) and *unmatching* (U) based on similarity, then clustered by transitivity, and, finally, merged by cluster. The latter may require solving inconsistencies (Naumann and Herschel 2010).

In such approaches with an absolute decision for tuples being duplicates or not, many realistic possibilities may be ignored leading to errors in the data. Instead, a probabilistic database can directly store an indeterministic deduplication result (Panse et al. 2013). In this way, all significantly likely duplicate mergings find their way into the database, and any query answer or other derived data will reflect the inherent uncertainty.

Indeterministic deduplication deviates as follows (Panse et al. 2013). Instead of M and U, a portion of tuple pairs are now classified into a third set P of *possible matches* based on two thresholds (see Fig. 4). For pairs in this *gray area*, both cases are considered: a match or not. Duplicate clustering now forms clusters for $M \cup U$ (in Fig. 1, there are 3 clusters: $\{d_1, d_2, d_5\}$, $\{d_4\}$, $\{d_3, d_6\}$). For each cluster, the possible worlds are determined, e.g., d_1, d_2, d_5 all different, d_1, d_5 the same and d_2 different, or d_1, d_2, d_5 all the same. To represent the probabilistic end result, a random variable is introduced for each cluster with as many values as possible worlds for that cluster, and merged and unmerged versions of the tuples are added according to the situation in the world. Figure 3 shows the end result.

A related problem is that of entity resolution (Naumann and Herschel 2010). The goal of data integration is often to bring together data on the same real-world entities from different sources. In the absence of a usable identifier, this matching and merging of records from different sources is a similar problem.

Repairs Another record-level integration problem is when a resulting database state does not satisfy some constraints. Here the notion of a *database repair* is useful. A repair of an inconsistent database I is a database J that is consistent and "as close as possible" to I (Wijsen 2005). Closeness is typically measured in terms

of the number of "insert," "delete," and "update" operations needed to change I into J. A repair, however, is in general not unique. Typically, one resorts to *consistent query answering*: the intersection of answers to a query posed on all possible repairs within a certain closeness bound. But, although there is no known work to refer to, it is perfectly conceivable that these possible repairs can be represented with a probabilistic database state.

Grouping data While integrating grouping data also inconsistencies may occur. A grouping can be defined as a membership of elements within groups. When different sources contain a grouping for the same set of elements, two elements may be in the same group in one source and in different groups in the other. Wanders et al. (2015) describe such a scenario with groups of orthologous proteins which are expected to have the same function(s). Biological databases like Homologene, PIRSF, and eggNOG store results of determining orthology by means of different methods. An automatic (probabilistic!) combination of these sources may provide a continuously evolving unified view of combined scientific insight of higher quality than any single method could provide.

Schema Level

Probabilistic data integration has been mostly applied to instance-level data, but it can also be applied on schema level. For example, if two sources hold data on entity types T and T', and these seem similar or related, then a number of hypotheses may be drawn up:

- T could have exactly the same meaning as T',
- T could be a subtype of T' or vice versa, or
- T and T' partially overlap and have a common supertype.

But it may be uncertain which one is true. It may even be the case that a hypothesis may only be partially true, for example, with source tables "student" and "PhD student." In most cases, a PhD student is a special kind of student, but in some countries such as the Netherlands, a PhD student is actually an employee of the university. Also employees from a company may pursue a PhD. In short, not all tuples of table "PhD student" should be integrated into "student." This also illustrates how this schema-level problem may be transformed into a record-level problem: a representation can be constructed where all tuples of a type probabilistically exist in a corresponding table. The uncertainty about two attributes being "the same" is an analogous problem.

Data Cleaning

Probabilistic data allows new kinds of cleaning approaches. High quality can be defined as a high probability for correct data and low probability for incorrect data. Therefore, cleaning approaches can be roughly categorized into uncertainty reducing and uncertainty increasing.

Uncertainty Reduction: Evidence If due to some evidence from analysis, reasoning, constraints, or feedback, it becomes apparent that some cases are definitely (not) true, then uncertainty may be removed from the database. For example, if in Fig. 3 feedback is given from which it can be derived that d_3 and d_6 are for certain the same car brand, then in essence $\mathrm{P}(r_2 \mapsto 0)$ becomes 0 and $\mathrm{P}(r_2 \mapsto 1)$ becomes 1. Consequently, all tuples that need $(r_2 \mapsto 0)$ to be true to exist can be deleted (d_3 and d_6). Furthermore, random variable r_2 can be abolished, and the term $(r_2 \mapsto 1)$ can be removed from all probabilistic sentences. It effectively removes all possible worlds that contradict with the evidence. van Keulen and de Keijzer (2009) have shown that this form of cleaning may quickly and steadily improve quality of a probabilistic integration result.

If such evidence cannot be taken as absolutely reliable, hence cannot justify the cleaning actions above, the actual cleaning becomes a matter of massaging of the probabilities. For example, $\mathrm{P}(r_2 \mapsto 1)$ may be increased only a little bit. In this approach, a probability threshold may be

introduced above which the abovedescribed random variable removal is executed. As evidence accumulates, this approach converges to a certain correct database state as well, so data quality improvement is only slowed down provided that the evidence is for a large part correct.

Uncertainty Increase: Casting Doubt Perhaps counterintuitive, but increasing uncertainty may improve data quality, hence could be an approach for cleaning. For example, if due to some evidence it becomes unlikely that a certain tuple is correct, a random variable may be introduced, and possible repairs for tuples may be inserted. In effect, we are casting doubt on the data and insert what seems more likely. Consequently the uncertainty increases, but the overall quality may increase, because the probability mass associated with incorrect data decreases and the probability mass for correct data increases (assuming the evidence is largely correct).

Measuring uncertainty and quality The above illustrates that uncertainty and quality are orthogonal notions. Uncertainty is usually measured by means of entropy. Quality measures for probabilistic data are introduced by (van Keulen and de Keijzer 2009): expected precision and expected recall. These notions are based on the intuition that the quality of a correct query answer is better if the system dares to claim that it is correct with a higher probability.

Example Applications

A notable application of probabilistic data integration is the *METIS* system, "an industrial prototype system for supporting real-time, actionable maritime situational awareness" (Huijbrechts et al. 2015). It aims to support operational work in domains characterized by constantly evolving situations with a diversity of entities, complex interactions, and uncertainty in the information gathered. It includes natural language processing of heterogeneous (un)structured data and probabilistic reasoning of uncertain information. METIS can be seen as an open-source intelligence (OSINT) application.

Another notable and concrete example of an existing system is *MCDB-R* (Arumugam et al. 2010). It allows risk assessment queries directly on the database. Risk assessment typically corresponds to computing interesting properties of the upper or lower tails of a query result distribution, for example, computing the probability of a large investment loss.

Probabilistic data integration is in particular suited for applications where much imperfection can be expected but where a quick-and-dirty integration and cleaning approach is likely to be sufficient. It has the potential of drastically lowering the time and effort needed for integration and cleaning, which can be considerable since "analysts report spending upwards of 80% of their time on problems in data cleaning" (Haas et al. 2015).

Other application areas include:

- Machine learning and data mining: since probabilistically integrated data has a higher information content than "data with errors," it is expected that models of higher quality will be produced if probabilistic data is used as training data.
- Information extraction from natural language: since natural language is inherently ambiguous, it seems quite natural to represent the result of information extraction as probabilistic data.
- Web harvesting: websites are designed for use by humans. A probabilistic approach may lead to more robust navigation. Subtasks like finding search results (Trieschnigg et al. 2012) or finding target fields (Jundt and van Keulen 2013) are typically based on ranking "possible actions." By executing not only one but a top-k of possible actions and representing resulting data probabilistically, consequences of imperfect ranking are reduced.

Future Developments

Probabilistic data integration depends on scalable probabilistic database technology. An important direction of future research is the development of probabilistic database systems and improving their scalability and functionality. Furthermore, future research is needed that compare the effectiveness of probabilistic data integration vs. non-probabilistic data integration approaches for real-world use cases.

Cross-References

- Data Cleaning
- Data Deduplication
- Data Integration
- Graph Data Integration and Exchange
- Holistic Schema Matching
- Record Linkage
- Schema Mapping
- Semantic Interlinking
- Truth Discovery
- Uncertain Schema Matching

References

Abiteboul S, Kimelfeld B, Sagiv Y, Senellart P (2009) On the expressiveness of probabilistic xml models. VLDB J 18(5):1041–1064. https://doi.org/10.1007/s00778-009-0146-1

Antova L, Jansen T, Koch C, Olteanu D (2008) Fast and simple relational processing of uncertain data. In: Proceedings of ICDE, pp 983–992

Antova L, Koch C, Olteanu D (2009) $10^{(10^6)}$ worlds and beyond: efficient representation and processing of incomplete information. VLDB J 18(5):1021–1040. https://doi.org/10.1007/s00778-009-0149-y

Arumugam S, Xu F, Jampani R, Jermaine C, Perez LL, Haas PJ (2010) MCDB-R: risk analysis in the database. Proc VLDB Endow 3(1–2):782–793. https://doi.org/10.14778/1920841.1920941

Dalvi N, Ré C, Suciu D (2009) Probabilistic databases: diamonds in the dirt. Commun ACM 52(7):86–94. https://doi.org/10.1145/1538788.1538810

De Raedt L, Kimmig A (2015) Probabilistic (logic) programming concepts. Mach Learn 100(1):5–47. https://doi.org/10.1007/s10994-015-5494-z

Fuhr N (2000) Probabilistic datalog: implementing logical information retrieval for advanced applications. J Am Soc Inf Sci 51(2):95–110

Haas D, Krishnan S, Wang J, Franklin M, Wu E (2015) Wisteria: nurturing scalable data cleaning infrastructure. Proc VLDB Endow 8(12):2004–2007. https://doi.org/10.14778/2824032.2824122

Huijbrechts B, Velikova M, Michels S, Scheepens R (2015) Metis1: an integrated reference architecture for addressing uncertainty in decision-support systems. Proc Comput Sci 44(Supplement C):476–485. https://doi.org/10.1016/j.procs.2015.03.007

Jampani R, Xu F, Wu M, Perez LL, Jermaine C, Haas PJ (2008) MCDB: a monte carlo approach to managing uncertain data. In: Proceeding of SIGMOD. ACM, pp 687–700

Jundt O, van Keulen M (2013) Sample-based XPath ranking for web information extraction. In: Proceeding of EUSFLAT. Advances in intelligent systems research. Atlantis Press. https://doi.org/10.2991/eusflat.2013.27

Koch C (2009) MayBMS: a system for managing large probabilistic databases. In: Aggarwal CC (ed) Managing and mining uncertain data. Advances in database systems, vol 35. Springer. https://doi.org/10.1007/978-0-387-09690-2_6

Lenzerini M (2002) Data integration: a theoretical perspective. In: Proceeding of PODS. ACM, pp 233–246. https://doi.org/10.1145/543613.543644

Magnani M, Montesi D (2010) A survey on uncertainty management in data integration. JDIQ 2(1):5:1–5:33. https://doi.org/10.1145/1805286.1805291

Naumann F, Herschel M (2010) An introduction to duplicate detection. Synthesis lectures on data management. Morgan & Claypool. https://doi.org/10.2200/S00262ED1V01Y201003DTM003

Panse F (2015) Duplicate detection in probabilistic relational databases. PhD thesis, University of Hamburg

Panse F, van Keulen M, Ritter N (2013) Indeterministic handling of uncertain decisions in deduplication. JDIQ 4(2):9:1–9:25. https://doi.org/10.1145/2435221.2435225

Trieschnigg R, Tjin-Kam-Jet K, Hiemstra D (2012) Ranking xpaths for extracting search result records. Technical report TR-CTIT-12-08, Centre for telematics and information technology (CTIT)

van Keulen M (2012) Managing uncertainty: the road towards better data interoperability. IT – Inf Technol 54(3):138–146. https://doi.org/10.1524/itit.2012.0674

van Keulen M, de Keijzer A (2009) Qualitative effects of knowledge rules and user feedback in probabilistic data integration. VLDB J 18(5):1191–1217

Wanders B, van Keulen M (2015) Revisiting the formal foundation of probabilistic databases. In: Proceeding of IFSA-EUSFLAT. Atlantis Press, p 47. https://doi.org/10.2991/ifsa-eusflat-15.2015.43

Wanders B, van Keulen M, van der Vet P (2015) Uncertain groupings: probabilistic combination of group-

ing data. In: Proceeding of DEXA. LNCS, vol 9261. Springer, pp 236–250. https://doi.org/10.1007/978-3-319-22849-5_17

Wanders B, van Keulen M, Flokstra J (2016) Judged: a probabilistic datalog with dependencies. In: Proceeding of DeLBP. AAAI Press

Widom J (2004) Trio: a system for integrated management of data, accuracy, and lineage. Technical report 2004-40, Stanford InfoLab. http://ilpubs.stanford.edu:8090/658/

Wijsen J (2005) Database repairing using updates. ACM TODS 30(3):722–768. https://doi.org/10.1145/1093382.1093385