13 Incorporating Uncertainty into Data Integration

Database systems typically model only certain data. That is, if a tuple is in the database, then it is true in the real world. There are many contexts in which our knowledge of the world is uncertain, and we would like to model that uncertainty in the database and be able to query it. For example, consider situations like the following:

- We are collecting information about theft sightings, and a witness does not remember whether the thief had blond or brown hair.
- We are recording temperature measurements from sensors in the field, and the accuracy of the sensors has a certain error bound or measurements may even be completely wrong if a sensor's battery is low.
- We are integrating data across multiple sites, and the answer to a query requires a join. This join includes an approximate string matching condition, which may yield a certain number of false positives.

The field of uncertain data management develops models for representing uncertainty in databases, semantics for query answering over such databases, and techniques for efficient query evaluation. Uncertainty can be introduced in multiple aspects of data integration:

DATA
To begin, some of the data may be extracted from unstructured data (as described in Chapter 9), and we may be uncertain about the accuracy of extraction. Furthermore, when we integrate data from many sources, they may be inconsistent with each other, because some of them are inaccurate or simply out of date. Even in enterprise settings, it is common for informational data about customers, such as gender and income level, to be dirty or missing, even when the data about the actual transactions concerning these customers are precise.

SCHEMA MAPPINGS
As described in Chapter 5, schema mappings may be generated using semiautomatic techniques. If we are dealing with a very large number of sources (e.g., on the World Wide Web) we may not have the resources to validate all these mappings. In other cases, we may not know the exact schema mapping because the meaning of the data may be ambiguous. Hence, uncertainty about schema mappings can be very common in data integration applications.

Principles of Data Integration, DOI: 10.1016/B978-0-12-416044-6.00013-2
Copyright © 2012 by Elsevier Inc. All rights reserved.

QUERIES
Some data integration contexts cannot assume that the users are familiar with the schema or that they have the skills to pose structured queries. Hence, the system may need to offer the user a keyword-query search interface, as we discuss in Chapter 16. The system needs to translate the keyword queries into some structured form so they can be reformulated onto the data sources. The translation step may generate multiple candidate structured queries, and therefore there will be uncertainty about which is the intended user query.

MEDIATED SCHEMA
When the domain of the integration system is very broad, there may even be some uncertainty about the mediated schema itself. As a simple example, if our mediated schema attempts to model all the disciplines of computer science, there will be uncertainty about the precise interpretation of each discipline and about the overlap between disciplines. For example, there may be overlap between the fields of databases and artificial intelligence, but quantifying the overlap may be difficult. In general, as we attempt to model broader domains, the terms in the domain may become more vague.

A data integration system that manages uncertainty would be able to model uncertain data, uncertain schema mappings, and uncertain queries. The system should attach probabilities to every object it manipulates and should incrementally produce *ranked* answers rather than finding all the certain answers.

Many of these challenges are the subject of ongoing research. This chapter introduces many of the basic concepts, especially as they pertain to data integration. Section 13.1 introduces the topic of modeling uncertainty associated with data. Section 13.2 then describes how uncertainty can be attached to schema mappings. We conclude with a brief discussion about the close connections between uncertainty and data provenance in Section 13.3. Further information on uncertainty as it pertains to queries in the context of keyword search, as well as general techniques for merging ranked results, can be found in Chapter 16. Finally, we refer the reader to the bibliographic references at the end of this chapter for more information about probabilistic modeling in databases and about how probabilities can be computed for the outputs of semiautomated schema matching techniques.

13.1 Representing Uncertainty

In general, uncertain data management is formulated in terms of *possible worlds*: instead of the database representing a single certain data instance, an uncertain database instead represents a set of possible instances or worlds. Each such instance may have some likelihood of being correct.

Uncertainty may be modeled to different degrees, depending on the circumstances.

- In perhaps the simplest form, we may know that there are several possible worlds, but we do not have any idea about the relative likelihood of each. Here we will often use *condition variables* to represent the set of possible values for an attribute (attribute A has values a_1, a_2, or a_3) or as Boolean conditions under which a tuple exists (tuple t exists if condition c_1 is true).

- In some cases, as in a Web search engine, we may annotate each data item with a score, whose value comes from a hand-tuned combination of factors. Here the rules for combining and composing scores may be fairly ad hoc.
- It is often desirable to have clean rules for *composing* scores as we apply query operations to tuples and tuple combinations. This leads to more formal models for computing and combining measures of uncertainty. A natural model is that of probabilities, which have a bounded range (the interval [0,1]) and can be cleanly composed even under negation (complement).

Our focus in this chapter will be on the last of these three cases, namely, using probabilistic models for uncertain data.

TYPES OF UNCERTAINTY

Uncertain data modeling generally considers two kinds of uncertainty. In *tuple-level uncertainty*, we do not know whether a data instance contains a tuple or not. We can consider the presence of the tuple to be a random variable that is true if the tuple is in the instance, or false if the tuple is not present in the instance. Alternatively, in *attribute-level uncertainty*, the value of an attribute is itself a random variable, whose domain is the set of values allowed for that tuple.

For regularity, we often convert attribute-level uncertainty into tuple-level uncertainty, as follows. For each possible value of attribute A, say a_1, a_2, a_3, in tuple t, we create a copy t_1, t_2, t_3 with attribute A set to a_1, a_2, a_3, respectively. Then we assign to each copied tuple t_i the probability of attribute A being a_i. Finally, we constrain t_1, t_2, t_3 to be mutually exclusive.

Example 13.1

Suppose for a theft sighting database, we are given that the witness is 60% confident he or she saw the thief, although the person who was seen might have been an innocent bystander. The person witnessed had blond hair. We can represent this using a tuple ($thief_1$, *Blondhair*) with tuple-level probability 0.6.

Alternatively, the witness may have directly seen the thief stealing an item, but due to the lighting, may not be certain about the thief's hair color (with 50% confidence it was brown, but with a 20% chance it was black and a 30% chance it was blond). We can represent this using attribute-level uncertainty given a tuple ($thief_1$, X). Random variable X is given the value *Brown* with probability 0.5, *Black* with probability 0.2, and *Blond* with probability 0.3.

Alternatively, we can convert the last case to a set of tuple-level probabilities given a data representation where we can mark tuples as mutually exclusive. We create the tuples

$t_1 = (thief_1, Brown)$ with probability 0.5, $t_2 = (thief_1, Black)$ with probability 0.2, and $t_3 = (thief_1, Blond)$ with probability 0.3. Then we make t_1, t_2, t_3 mutually exclusive.

This last example brings up a natural question, which is how to represent mutual exclusion among tuples and other kinds of constraints over how tuples may appear within possible worlds. From this point forward (given that we have the above conversion process), let us assume tuple-level uncertainty in our discussion.

13.1.1 Probabilistic Data Representations

In the database world, the starting point for uncertain data representations is that of the *conditional table* (c-table). In conditional tables every tuple is annotated with a Boolean condition over a set of variables. The tuple is present in the table if the condition evaluates to true, and absent if it evaluates to false. The c-table gives us a means of representing a set of possible worlds, which are the set of instances that arise due to different combinations of valuations to the variables. The various conditions restrict which combinations of tuples are allowed in the instances.

Example 13.2

Suppose Alice wishes to go on a vacation with Bob. They have two options as destinations, Tahiti and Ulaanbaatar. A third traveler, Candace, will be going to Ulaanbaatar. We can model this set of vacations as

Vacationer	Destination	Condition
Alice	Tahiti	x
Bob	Tahiti	x
Alice	Ulaanbaatar	$\neg x$
Bob	Ulaanbaatar	$\neg x$
Candace	Ulaanbaatar	true

where x represents the condition that they jointly choose Tahiti. Observe that there are only two possible worlds in the above table.

We can generalize the notion of the conditional table to the *probabilistic conditional table*, or pc-table, by letting the variables in the c-table be random variables and assigning a probability distribution to the random variables and their possible values.

Example 13.3

Given the above data instance, we might see Alice with a tan and know that it is winter. Hence odds are 80% that Alice and Bob went to Tahiti and not to Ulaanbaatar. We assign a probability of 0.8 to x.

The pc-table is an elegant abstraction for capturing probabilistic information, although it leaves open precisely how we capture the probability distributions of the variables. One possibility is to use concepts from machine learning such as graphical models (Markov networks, Bayesian networks, and their relatives), which capture the correlation structure among random variables. Significant work has been done on building database query models and systems that return probabilistic results under such models.

Sometimes we use much simpler models, which greatly limit the set of possible worlds that can be captured — but which make the query answering problem much more tractable. We briefly describe two such models that are often used.

THE TUPLE-INDEPENDENT MODEL

The simplest probabilistic representation of a set of possible worlds uses a separate Boolean variable to annotate each tuple in a pc-table. We then assign a probability to each variable separately: this probability is with respect to the *event* that the tuple is a member of the instance. Observe that since each tuple has its own variable, there are no correlations among the tuples. Hence we can find the set of possible models by enumerating all possible subsets of tuples from the relation and computing the probability of each instance as the product of all of the probabilities of the member tuples. The resulting model is, rather naturally, called the *tuple-independent* model.

The tuple-independent model is, unfortunately, too simple to capture correlations or mutual exclusions. In our vacationers example, there is no way of indicating that Alice can only go to one of Tahiti or Ulaanbaatar, as opposed to having some independent probability of going to each. There is also no way of capturing the fact that Alice and Bob will go together. The next model captures mutual exclusions, although it still cannot express correlations.

THE BLOCK-INDEPENDENT-DISJOINT (BID) MODEL

Here we divide the table into a set of *blocks*, each of which is independent of the others. Any instance of the table must have a tuple from each block, and each tuple in the block is a *disjoint* (i.e., mutually exclusive) probabilistic event. We can think of this as assigning a random variable x_b to each block b, and annotate each tuple in the block $t_1, t_2, t_3, \ldots$ with the condition $x_b = 1$, $x_b = 2$, $x_b = 3, \ldots$, respectively. Since each random variable only appears once per block, in writing the BID table, we will typically replace the condition $x_b = y$ with its associated event probability.

Example 13.4

We can capture the various choices with the following BID table, where the long horizontal lines separate the different blocks.

Vacationer	Destination	Probability
Alice	Tahiti	0.8
Alice	Ulaanbaatar	0.2
Bob	Tahiti	0.8
Bob	Ulaanbaatar	0.2
Candace	Ulaanbaatar	1.0

However, we still cannot constrain Alice and Bob to go to the same destination. Graphical models offer a more powerful formalism for capturing correlations between tuples. We discuss them briefly in the bibliographic notes.

13.1.2 From Uncertainty to Probabilities

A probabilistic model for representing uncertainty has many positives. However, a natural question is how one goes from confidence levels in data, mappings, queries, or schemas to actual probability values. After all, for example, converting a string edit distance score to a probability requires a model of how typographical errors or string modifications are introduced. Such a model is likely highly dependent on the particular data and application — and thus unavailable to us.

The answer to the question of where probabilities come from is typically application specific, and often not formally justified. In the best cases, we do have probabilistic information about distributions, error rates, etc. to build from. In a few of these cases, we may even have models of how data values correlate.

However, in many other cases, we only have a subjective confidence level that gets converted to a [0,1] interval and gets interpreted as a probability. Much as in Web search, the ultimate question is whether the system assigns a higher score to answers composed from good (high-confidence) values than to poor ones — not whether we have a mathematically solid foundation to the generation of the underlying scores.

Within this section, our focus has been on representing uncertainty associated with data. We next describe how we can ascribe uncertainty to another key ingredient in data integration, namely, schema mappings.

13.2 Modeling Uncertain Schema Mappings

Schema mappings describe the relationship between the terms used in the mediated schema and the terms used in the sources. In this section we describe probabilistic schema mappings (p-mappings) that are intended to capture uncertainty on mappings.

We describe the possible semantics of probabilistic mappings and their effect on the complexity of query answering.

For brevity, we consider very simple schema mappings here, specified as attribute correspondences. Specifically, a mapping consists of a set of *attribute correspondences*. An attribute correspondence is of the form $c_{ij} = (s_i, t_j)$, where s_i is a *source attribute* in the schema S and t_j is a *target attribute* in the schema T. Intuitively, c_{ij} specifies that there is a relationship between s_i and t_j. The correspondence may specify that the two attributes are equal to each other or that there may be a transformation function involved (e.g., from Celsius to Fahrenheit). We consider simple schema mappings defined as follows.

Definition 13.1 (Schema Mapping). *Let $\bar{S}$ and $\bar{T}$ be relational schemas. A relation mapping M is a triple (S, T, m), where S is a relation in $\bar{S}$, T is a relation in $\bar{T}$, m is a set of attribute correspondences between S and T, and each source and target attribute occurs in at most one correspondence in m.*

A schema mapping $\overline{M}$ is a set of relation mappings between relations in $\bar{S}$ and in $\bar{T}$, where every relation in either $\bar{S}$ or $\bar{T}$ appears at most once. □

Example 13.5

Consider a data source S that describes a person by her email address, current address, and permanent address, and the mediated schema T that describes a person by her name, email, mailing address, home address, and office address (both of these schemas include a single relation):

```
S=(pname, email-addr, current-addr, permanent-addr)
T=(name, email, mailing-addr, home-addr, office-addr)
```

The following is a possible relation mapping between S and T.

```
{
  (pname, name),
  (email-addr, email),
  (current-addr, mailing-addr),
  (permanent-addr, home-addr)
}
```

13.2.1 Probabilistic Mappings

As described earlier, we may not be sure about the mapping between two schemas. Continuing with Example 13.5, a semiautomatic schema mapping tool may generate three possible mappings between S and T, assigning each a probability, as shown in Figure 13.2. Whereas the three mappings all map pname to name, they map other attributes in the source and the target differently. For example, mapping m_1 maps current-addr to mailing-addr, but mapping m_2 maps permanent-addr to mailing-addr. Because of the uncertainty about which mapping is correct, we would like to consider all of these mappings in query answering.

We can now formalize the notion of p-mappings. Intuitively, a p-mapping describes a probability distribution over a set of *possible* schema mappings between a source schema and a target schema.

Definition 13.2 (Probabilistic Mapping). *Let $\bar{S}$ and $\bar{T}$ be relational schemas. A probabilistic mapping (p-mapping), pM, is a triple $(S, T, \mathbf{m})$, where $S \in \bar{S}$, $T \in \bar{T}$, and $\mathbf{m}$ is a set $\{(m_1, Pr(m_1)), \ldots, (m_l, Pr(m_l))\}$, such that*

- *for $i \in [1,l]$, m_i is a mapping between S and T, and for every $i,j \in [1,l]$, $i \neq j \Rightarrow m_i \neq m_j$.*
- *$Pr(m_i) \in [0,1]$ and $\sum_{i=1}^{l} Pr(m_i) = 1$.*

A schema p-mapping, $\overline{pM}$, is a set of p-mappings between relations in $\bar{S}$ and in $\bar{T}$, where every relation in either $\bar{S}$ or $\bar{T}$ appears in at most one p-mapping. □

13.2.2 Semantics of Probabilistic Mappings

Given a p-mapping, *pM*, there are (at least) two ways to interpret uncertainty about schema mappings:

1. a single mapping in *pM* is the correct one and it applies to all the data in the source *S*, or
2. several mappings in *pM* are *partially* correct and each is suitable for a different subset of tuples in *S*. Furthermore, we do not know which mapping is the right one for a specific tuple.

In our running example, both interpretations are equally valid. While one of the mappings may be correct for all the data, it may also be true that some people may choose to use their current address as their mailing address while others use their permanent address as their mailing address. In the latter case, the correct mapping depends on the particular tuple.

We define query answering semantics with respect to two interpretations, *by-table* semantics and *by-tuple* semantics. The two semantics turn out to have different computational properties. Note that the needs of the application dictate which is the appropriate semantics.

The by-table and by-tuple semantics are natural extensions of certain answers (see Section 3.2.1). We first briefly review certain answers in our simplified context. Recall that a mapping defines a relationship between instances of *S* and instances of *T* that are *consistent* with the mapping.

Definition 13.3 (Consistent Target Instance). *Let $M = (S, T, m)$ be a relation mapping and D_S be an instance of S.*

An instance D_T of T is said to be consistent with D_S and M if for each tuple $t_s \in D_S$, there exists a tuple $t_t \in D_T$, such that for every attribute correspondence $(a_s, a_t) \in m$, the value of a_s in t_s is the same as the value of a_t in t_t. □

Example 13.6

In Figure 13.1, (a) is an instance of the source that is consistent with (b) under the mapping m_1 and is consistent with (c) under the mapping m_2.

*p*name	*e*mail-addr	*c*urrent-addr	*p*ermanent-addr
Alice	alice@	Mountain View	Sunnyvale
Bob	bob@	Sunnyvale	Sunnyvale

(a)

*n*ame	*e*mail	*m*ailing-addr	*h*ome-addr	*o*ffice-addr
Alice	alice@	Mountain View	Sunnyvale	office
Bob	bob@	Sunnyvale	Sunnyvale	office

(b)

*n*ame	*e*mail	*m*ailing-addr	*h*ome-addr	*o*ffice-addr
Alice	alice@	Sunnyvale	Mountain View	office
Bob	bob@	Sunnyvale	Sunnyvale	office

(c)

FIGURE 13.1 (a) is an instance of the source schema, and (b) and (c) are instances of the target schema. (a) is consistent with (b) and (c) under different schema mappings.

For a relation mapping M and a source instance D_S, there can be an infinite number of target instances that are consistent with D_S and M. We denote by $Tar_M(D_S)$ the set of all such target instances. The set of answers to a query Q is the intersection of the answers on all instances in $Tar_M(D_S)$:

Definition 13.4 (Certain Answer). *Let $M = (S, T, m)$ be a relation mapping. Let Q be a query over T and let D_S be an instance of S.*

A tuple t is said to be a certain answer of Q with respect to D_S and M if for every instance $D_T \in Tar_M(D_S)$, $t \in Q(D_T)$. □

13.2.3 By-Table Semantics

We now generalize certain answers to the probabilistic setting, beginning with the by-table semantics. Intuitively, a p-mapping pM describes a set of possible worlds. In each of these worlds, a different mapping in pM applies to the source. The worlds are weighted by the probability of each mapping in pM. Following this intuition, we define target instances that are *consistent with* the source instance.

Definition 13.5 (By-Table Consistent Instance). *Let $pM = (S, T, \mathbf{m})$ be a p-mapping and D_S be an instance of S.*

An instance D_T of T is said to be by-table consistent with D_S and pM if there exists a mapping $m \in \mathbf{m}$ such that D_S and D_T satisfy m. □

Example 13.7

In Figure 13.1, both (b) and (c) are by-table consistent with (a) with respect to the probabilistic mappings in Figure 13.2.

Possible Mapping		Prob
$m_1 =$	{(pname, name), (email-addr, email), (current-addr, mailing-addr), (permanent-addr, home-addr)}	0.5
$m_2 =$	{(pname, name), (email-addr, email), (permanent-addr, mailing-addr), (current-addr, home-addr)}	0.4
$m_3 =$	{(pname, name), (email-addr, mailing-addr), (current-addr, home-addr)}	0.1

(a)

FIGURE 13.2 A probabilistic mapping, consisting of three alternative relation mappings.

Given a source instance D_S and a possible mapping $m \in \mathbf{m}$, there can be an infinite number of target instances that are consistent with D_S and m. We denote by $Tar_m(D_S)$ the set of all such instances.

In the probabilistic context, we assign a probability to every answer. Intuitively, we consider the certain answers with respect to each possible mapping in *isolation*. The probability of an answer t is the sum of the probabilities of the mappings for which t is deemed to be a certain answer. We define by-table answers as follows:

Definition 13.6 (By-Table Answer). *Let $pM = (S, T, \mathbf{m})$ be a p-mapping. Let Q be a query over T and let D_S be an instance of S.*

Let t be a tuple. Let $\bar{m}(t)$ be the subset of $\mathbf{m}$, such that for each $m \in \bar{m}(t)$ and for each $D_T \in Tar_m(D_S)$, $t \in Q(D_T)$.

Let $p = \sum_{m \in \bar{m}(t)} Pr(m)$. If $p > 0$, then we say (t, p) is a by-table answer of Q with respect to D_S and pM. □

Example 13.8

Consider the query that retrieves all the mailing addresses in the target relation. Figure 13.3(a) shows the answers to the query under by-table semantics. As an example, for tuple t =('Sunnyvale'), we have $\bar{m}(t) = \{m_1, m_2\}$, so the possible tuple ('Sunnyvale', 0.9) is an answer.

13.2.4 By-Tuple Semantics

Whereas by-table semantics modeled a possible world for each choice of mapping in $\mathbf{m}$, by-tuple semantics need to consider that each tuple may choose a different mapping. Hence, a possible world is an assignment of a possible mapping in $\mathbf{m}$ to each tuple in D_S.

Tuple (mailing-addr)	Prob
('Sunnyvale')	0.9
('Mountain View')	0.5
('alice@')	0.1
('bob@')	0.1

(a)

Tuple (mailing-addr)	Prob
('Sunnyvale')	0.94
('Mountain View')	0.5
('alice@')	0.1
('bob@')	0.1

(b)

FIGURE 13.3 Finding all the mailing addresses according to by-table semantics (a) and according to by-tuple semantics (b).

Formally, the key difference in the definition of by-tuple semantics from that of by-table semantics is that a consistent target instance is defined by a mapping *sequence* that assigns a (possibly different) mapping in $\mathbf{m}$ to each source tuple in D_S. We assume, without loss of generality, that we have an ordering on the tuples in D_S.

Definition 13.7 (By-Tuple Consistent Instance). *Let d denote the number of tuples in D_S. Let $pM = (S,T,\mathbf{m})$ be a p-mapping and let D_S be an instance of S with d tuples.*

An instance D_T of T is said to be by-tuple consistent with D_S and pM if there is a sequence $\langle m^1,\ldots,m^d\rangle$ such that for every $1 \le i \le d$,

- *$m^i \in \mathbf{m}$, and*
- *for the ith tuple of D_S, t_i, there exists a target tuple $t'_i \in D_T$ such that for each attribute correspondence $(a_s,a_t) \in m^i$, the value of a_s in t_i is the same as the value of a_t in t'_i.* □

Given a mapping sequence $seq = \langle m^1,\ldots,m^d\rangle$, we denote by $Tar_{seq}(D_S)$ the set of all target instances that are consistent with D_S and *seq*. Note that if D_T is by-table consistent with D_S and m, then D_T is also by-tuple consistent with D_S and a mapping sequence in which each mapping is m.

We can think of every sequence of mappings $seq = \langle m^1,\ldots,m^d\rangle$ as a separate event whose probability is $Pr(seq) = \Pi_{i=1}^{d} Pr(m^i)$. If there are l mappings in pM, then there are l^d sequences of length d, and their probabilities add up to 1. We denote by $\mathbf{seq}_d(pM)$ the set of mapping sequences of length d generated from pM.

Definition 13.8 (By-Tuple Answer). *Let $pM = (S,T,\mathbf{m})$ be a p-mapping. Let Q be a query over T and D_S be an instance of S with d tuples.*

Let t be a tuple. Let $\overline{seq}(t)$ be the subset of $\mathbf{seq}_d(pM)$, such that for each $seq \in \overline{seq}(t)$ and for each $D_T \in Tar_{seq}(D_S)$, $t \in Q(D_T)$.

Let $p = \sum_{seq \in \overline{seq}(t)} Pr(seq)$. If $p > 0$, we call (t,p) a by-tuple answer of Q with respect to D_S and pM. □

Example 13.9

Continuing with Example 13.8, Figure 13.3(b) shows the answers to the query under by-tuple semantics. Note that the probability of tuple t=('Sunnyvale') in the by-table answers is different from that in the by-tuple answers.

Remark 13.1 (Computational Complexity). By-table semantics have an attractive property: it can be shown that we can find all the certain answers to a select-project-join query in time that is polynomial in the size of the data and the mappings. By-tuple semantics do not enjoy the same computational properties as by-table semantics. It can be shown that in general, finding all the certain answers under by-tuple semantics is #P-complete with respect to the size of the data. □

13.3 Uncertainty and Data Provenance

Ultimately, particularly in a data integration scenario, the score (whether a probability or some other measure) associated with a tuple should depend on a wide variety of factors: the probability of correctness of the query, the probability of correctness of individual schema mappings, the probability of correctness of the source data, and in fact the overall sample space (the set of possible worlds).

As has been observed by many researchers in the probabilistic and data integration realms, there is a very close tie between the provenance of query answers, i.e., the explanations for how the answers were produced and from where, and the scores assigned to the answers. A variety of techniques have been proposed for separately computing query results with provenance, and then computing probabilities over these results. Other techniques have been proposed for learning the scores for individual data sources, schema mappings, or queries, given a correct ranking of the query answers, as well as provenance information that explains the relationships among sources and results. Work continues in this very rich area. We discuss data provenance in Chapter 14, and we detail how it is useful in computing and learning scores in Chapter 16.

Bibliographic Notes

An excellent overview text about probabilistic databases appears in [535]. Our discussion of probabilistic data modeling is heavily inspired by that text. Some of the foundational work includes representations for incomplete information, including c-tables, in [313], as well as the early probabilistic database work in [236, 364] and the model of pc-tables from [271]. More recently, a wide variety of work on probabilistic data management has arisen in the database, rather than data integration, context. Several well-known recent systems include Trio [67], MystiQ [95, 160, 496], MaybMS [32], PrDB [518], and BayesStore [561]. A popular scheme for computing approximate results, first used in MystiQ and studied in greater detail in MCDB [330] and [569], is to use Monte Carlo simulation. One motivating application for such systems is to support the execution of probabilistic computations such as information extraction from text or Web pages [560].

Managing and mining uncertain data have been a subject of research for quite some time. Some recent progress is described in a collection of articles edited by Aggrawal [17].

The by-table and by-tuple semantics for probabilistic schema mappings were introduced in [189]. That paper also establishes the complexity results on answering queries in the presence of p-mappings. There have been various models proposed to capture uncertainty on mappings between attributes. Gal et al. [239] propose keeping the top-K mappings between two schemas, each with a probability (between 0 and 1) of being true. In [242] they propose assigning a probability for matching of every pair of source and target attributes. A more comprehensive treatment of the topic can be found in the book [241].

Magnani and Montesi [411] have empirically shown that top-k schema mappings can be used to increase the recall of a data integration process, and Gal [238] described how to generate top-k schema matchings by combining the matching results generated by various matchers. Nottelmann and Straccia [462] proposed generating probabilistic schema matchings that capture the uncertainty on each matching step.

He and Chang [293] considered the problem of generating a mediated schema for a set of Web sources. Their approach was to create a mediated schema that is statistically maximally *consistent* with the source schemas. To do so, they assume that the source schemas are created by a *generative model* applied to some mediated schema, which can be thought of as a probabilistic mediated schema. Magnani et al. [412] proposed generating a set of alternative mediated schemas based on probabilistic relationships between *relations* (such as an Instructor relation intersects with a Teacher relation but is disjoint with a Student relation) obtained by sampling the overlapping of data instances.

Chiticariu et al. [135] studied the generation of multiple mediated schemas for an existing set of data sources. They consider multitable data sources and explore interactive techniques that aid humans in arriving at the mediated schemas.

Lu et al. [399] describe a system in which multiple data owners can share data, each using their own terminology. The data are inserted into a *wide table* that has a column for each attribute of each data provider. The system automatically determines the similarity between pairs of tags used by multiple data providers by inspecting the data itself and represents the similarity with a probability. Users can ask queries using any schema they want, and the system uses probabilistic query answering to retrieve data that do not conform to the schema used in the query.