# 22

# Management of Uncertainty in Database Systems

AMIHAI MOTRO

As models of the real world, databases are often permeated with forms of uncertainty, including imprecision, incompleteness, vagueness, inconsistency, and ambiguity. This chapter addresses issues of database uncertainty. It defines basic terminology, and it classifies the various kinds of uncertainty. It then surveys solutions that have been attempted, and it speculates on the reasons that have hindered the development of general-purpose database systems with powerful uncertainty capabilities. Finally, it describes challenging new applications that will require such capabilities, and it points to promising directions for research.

## 22.1 Introduction

Database systems model our knowledge of the real world.[1] This knowledge is often permeated with uncertainty. Hence, database systems must be able to deal with uncertainty. And, indeed, most database systems include capabilities for dealing with some kinds of uncertainty—for example, many systems are able to represent missing values with nulls, and most query languages admit some form of uncertain queries by allowing users to substitute patterns for constants.

Yet these capabilities are weak in comparison with the variety and degree of uncertainty that are encountered in practice. While the research community has shown persistent interest in this subject, most of these efforts have yet to transcend experimental prototypes. By and large, commercial database systems have been slow to incorporate capabilities for dealing with uncertainty.

Nevertheless, many new applications require uncertainty capabilities (see section 22.5), and without general systems that possess these capabilities, these applications are usually handled in an ad hoc manner. Undoubtedly, the development of a suitable database theory to deal with uncertain

1. We use the term *database system* in a broad sense, to include related types of information systems such as *information retrieval systems* and *expert systems*.

database information and uncertain database transactions, and the successful deployment of this theory in an actual database system, remain challenges that have yet to be met.

This chapter addresses issues of database uncertainty. It defines basic terminology and classifies the various kinds of uncertainty. It then surveys solutions that have been attempted, and it speculates on the reasons that have hindered the development of general-purpose database systems with powerful uncertainty capabilities. Finally, it describes new challenging applications that will require such capabilities, and it points to promising directions for research.

Undoubtedly, the best known issue of uncertainty is the representation of unknown values with nulls and the evaluation of queries in the presence of such nulls. Yet uncertainty permeates many additional aspects of databases, including the definitions of database transactions and even the processing of transactions.

We therefore devote a substantial part of this chapter (sections 22.2, 22.3, and 22.4) to study the different forms of uncertainty and to evaluate the work that has already been done in this area. Section 22.2 provides basic definitions, including a simple and general model of database systems. This model allows us to classify uncertainty into three main categories: *Description uncertainty* refers to uncertainty in the information stored in databases (e.g., data, constraints, or rules), *transaction uncertainty* refers to uncertainty in the operations that manipulate this information (e.g., queries, updates, and re-structuring operations), and *processing uncertainty* refers to uncertainty arising in the application of transactions to descriptions. The main approaches are then studied in sections 22.3 and 22.4. In sections 22.5 and 22.6, we speculate on the reasons commercial database systems have been slow to incorporate capabilities for dealing with uncertainty. We then describe several applications that challenge present database systems by requiring more powerful methods for dealing with uncertainty, and we point to some promising areas of research; in particular, we postulate that suitable solutions could come from fusing database technology with various theories of artificial intelligence.

## 22.2 Terminology

Our discussion of uncertainty in database systems begins with definitions of database systems and uncertainty.

### 22.2.1 Database System

A database system is a computer model of some portion of the real world. Like any other model, it abstracts reality to a level warranted by the

expected applications. Usually, such models include a declarative component for *describing* the real world and an operational component for *manipulating* this description. Typical manipulations are (1) *modifications* of the description, either to refine the model or to track any changes that may have occurred in the real world; and (2) *transformations* of the description, to derive implied descriptions.

Thus, a database system can be abstracted as a description $D$ of the real world, a stream of modifications, and a stream of transformations. Each modification $m$ replaces the present description with a new description; each transformation $t$ computes a new description from the present description (without changing the present description). A computer system $S$ is responsible for maintaining the description reliably and for processing the modifications and transformations correctly and efficiently.

As an example, in a relational database system a description is a set of tables (i.e., a database); a modification can affect either the definition or the contents of tables (i.e., restructuring or update); and a transformation reduces the set of tables into a single table (i.e., query evaluation).

### 22.2.2 Uncertainty

We assume that uncertainty permeates our *models* of the real world, not the real world itself. In other words, the real world is always certain; it is our knowledge of it that is sometimes uncertain. We use the term *uncertainty* to refer to any element of the model that cannot be asserted with complete confidence. In particular, uncertainty can be expected (1) in the descriptions of the real world, (2) in the modifications and transformations of these descriptions, and (3) in the execution of these operations. Referring again to the example of relational database systems, these correspond to the data, the definitions of restructuring, update and query transactions, and the processing of transactions.

We have offered the general term *uncertainty* to describe any element of the model that cannot be asserted with complete confidence. Within this general condition, we observe several distinct types of uncertainty:[2]

- *Uncertainty*—it is not possible to determine whether an assertion in the model is *true* or *false*. For example, there might be uncertainty about the database fact "the age of John is 38."
- *Imprecision*—the information available in the model is not as specific as it should be. For example, when a distinct value is required, the information available might be a *range* (e.g., "the age of John is between 37 and 43"), *disjunctive* (e.g., "the age of John is either 37 or 43"), *negative*

2. One of these specific types will also be termed *uncertainty*; thus, depending on the context, this term will be used both generically and specifically.

(e.g., "the age of John is not 37"), or even *unknown* (often referred to as *incompleteness*).

- *Vagueness*—the model includes elements (e.g., predicates or quantifiers) that are inherently vague; for example, "John is in early middle age." A particular formalization of vagueness is based on the concept of *fuzziness*.
- *Inconsistency*—the model contains two or more assertions that cannot be true at the same time; for example, "the age of John is between 37 and 43" and "the age of John is 35."
- *Ambiguity*—some elements of the model lack complete semantics, leading to several possible interpretations. For example, it may not be clear whether stated salaries are per year or per month.

The objective of a database system is to provide its users with the information they need. In our terminology, such information is always the result $t(D)$ of transforming a description $D$ with a transformation $t$. Thus, it is the quality of $t(D)$, rather than the quality of $D$, that should concern the designers and users of database systems. A result $t(D)$ may be uncertain because $D$ is uncertain, or because $t$ is uncertain, or because the application of $t$ to $D$ involves uncertainty.[3]

We name these categories of uncertainty, respectively, *description uncertainty*, *transaction uncertainty*, and *processing uncertainty*. These categories serve as a framework for classifying the work that has been done in the area of uncertainty in database systems. This work is discussed in the next two sections.

## 22.3 Description Uncertainty

Undoubtedly, uncertainty that permeates the information stored in databases has received the most attention. In this section we consider some basic aspects of description uncertainty, and we review some of the more important solutions that have been suggested.

### 22.3.1 Introduction

Our discussion of description uncertainty begins by considering three issues: the different *elements* of descriptions that might be affected by uncertainty, the different *sources* for uncertainty, and the different *degrees* of uncertainty.

3. In turn, the uncertainty of $D$ may owe to uncertainty either in the initial description or in some later modification $m$.

Roughly, we ask *what* is uncertain, *why* is it uncertain, and *how* uncertain is it?

## Elements Affected by Uncertainty

Depending on the model used, descriptions may take different forms, and uncertainty could affect each of them.

Consider, for example, relational databases. The structures of the relational model admit different kinds of uncertainty. The first kind involves uncertainty at the level of data values; for example, some values of DEPARTURE_TIME in the relation FLIGHT (FLIGHT_NUMBER, DEPARTURE_TIME) might be uncertain. The second kind involves uncertainty at the level of the tuple; for example, there might be uncertainty regarding the membership of some tuples in the relation DIRECT_CONNECTION (ORIGIN, DESTINATION). A third kind involves uncertainty at the level of the relation (the structure); for example, there might be uncertainty whether a flight may have more than one destination, and hence what should be the proper description of this relationship [Borgida 1985].

As another example, consider an information retrieval system that models each document with an identifier and a vector of keywords. There might be uncertainty at the level of a keyword (i.e., the appropriateness of a specific keyword to a given document might be questionable). In addition, there might be uncertainty at the level of an entire document (i.e., the existence of a document might be in doubt).

As a third example, consider an expert system that models real-world knowledge with facts and rules expressed in logic. There might be uncertainty about specific facts (similar to the tuple uncertainty in relational databases) and about specific rules (i.e., a rule might be only an approximation of the behavior of the real world).

## Sources of Uncertainty

Having excluded the possibility that reality itself is subject to uncertainty, we can assume that a perfect description of any real-world object always exists. Thus, uncertain descriptions are solely due to the *unavailability* of these perfect descriptions. For example, the precise salary of Tom might be unknown, or the true relationship between Bordeaux wine and good health might be unclear. Yet within this generic unavailability we observe several specific sources of uncertainty [Kwan et al. 1992].

Uncertainty might result from using *unreliable information sources*, such as faulty reading instruments, or input forms that have been filled out incorrectly (intentionally or inadvertently). In other cases, uncertainty is a result of *system errors*, including transmission noise, delays in processing update transactions, imperfections of the system software, and corrupted data owing to failure or sabotage.

At times, uncertainty is the unavoidable result of information-gathering methods that require *estimation* or *judgment*. Examples include the determination of the subject of a document, the digital representation of a continuous phenomenon, and the representation of a phenomenon that is constantly varying.

In other cases, uncertainty is the result of restrictions imposed by the *model*. For example, if the database schema permits storing at most two occupations per employee, descriptions of occupation might exhibit uncertainty. Similarly, the sheer *volume* of the information that is necessary to describe a real-world object might force the modeler to turn to approximation and sampling techniques.

## Degrees of Uncertainty

The relevant information that is available in the absence of certain information may take different forms, each exhibiting a different level of uncertainty. The following discussion is independent of the particular structure affected by uncertainty. It assumes that an element *e* of the description models an object *o* of the real world. The element *e* might be a value, a fact, a tuple, a rule, and so on.

Uncertainty is highest when the mere *existence* of some real-world object is in doubt. The simplest solution is to ignore such objects altogether. This solution, however, is unacceptable if the model claims to be *closed world* (i.e., objects not modeled do not exist).

Uncertainty is reduced somewhat when elements of the model may be assigned values in a prescribed range, to indicate the certainty that the objects they model exist. When the element is a fact, this value can be interpreted as the *confidence* that the fact holds; when it is a rule, this value can be interpreted as the *strength* of the rule (percent of cases where the rule applies).

Assume now that existence is assured, but some or all of the information with which the model describes an object is *unknown*. Such information has also been referred to as *incomplete, missing,* or *unavailable*.

Uncertainty is reduced when the information that describes an object is known to come from a limited set of alternatives (possibly a range of values). This uncertainty is referred to as *disjunctive* information. Note that when the set of alternatives is simply the entire domain of admissible values, disjunctive information reverts to unknown information.

Uncertainty is reduced even further when each alternative is accompanied by a number describing the probability that it is indeed the true description (and the sum of these numbers for the entire set is 1). In this case, the uncertain information is *probabilistic*. Again, when the probabilities are unavailable, probabilistic information reverts to disjunctive information.

## 22.3.2 Solutions

Space considerations forbid discussion of all the different solutions that have been attempted for accommodating uncertainty in descriptions. We sketch here five approaches that are different from each other; they also exhibit sufficient generality to be applicable in different kinds of information systems.

### Null Values

Most data models insist that similar real-world objects be modeled with similar descriptions. The simplest example of this approach are models that use tabular descriptions. Each such table models a set of similar real-world objects: Each row describes a different object, and the columns provide the different components of the description. Often, some elements of a particular description cannot be stated with certainty. Occasionally, this problem may be evaded simply by not modeling any object whose description is incomplete. Often, however, the consequences of this approach are unacceptable, and incomplete descriptions must be admitted. Not surprisingly, incompleteness is a lesser issue in models that do not rely on mandatory information.

The least ambitious approach to admitting incomplete descriptions is to ignore all partial information about the uncertain parts of a description that may be available and to model them with a pseudo-description, called *null*, that denotes uncertainty [Date 1986; Imielinski 1989; Maier 1983; Zicari 1992].

Once null values are admitted into descriptions, the model must define the behavior of transformations and modifications in the presence of nulls. This is not always a simple task. For example, an extension to the relational calculus that is founded on a three-valued logic [Codd1979] has been the subject of criticism [Date 1990]. Inference in incomplete databases is discussed in Demolombe and Farinas del Cerro 1988. Updates of incomplete databases are discussed in Abiteboul and Grahne 1985.

Various refinements to this approach have been suggested. For example, a distinction has been made between incompleteness due to *unavailability* and incompleteness due to *inapplicability*.[4] Similarly, it has been suggested to use *distinct* nulls for unavailable descriptions that are known to be identical. These and other refinements constitute attempts to apply whatever partial information is available.

4. Strictly speaking, however, inapplicability is not a case of uncertainty. Inapplicability indicates that specific objects cannot be accommodated in the general description schemes used in the model.

## Certainty Factors

Certainty factors denote confidence in various elements of the description. They offer a simple tool for representing uncertainty and have thus been applied in both information retrieval systems and expert systems.

In information retrieval systems, certainty factors (often called *weights*) have been used to denote confidence that a specific keyword describes a given document (or alternatively, to denote the strength with which this keyword applies to the document) [van Rijsbergen 1979; Salton and McGill 1983; Turtle and Croft 1992]. Methods have even been developed for computing certainty factors automatically, by scanning documents and applying keyword counting techniques. The manipulation of these certainty factors is relatively simple, as they are easily accommodated in the vector space models that are often used in information retrieval.

In expert systems, certainty factors have been used to denote confidence that stated facts and rules indeed describe real-world objects [Harmon and King 1985]. Such factors are usually declared by the knowledge engineers as part of the knowledge acquisition process, but they can also be derived automatically as part of a knowledge discovery process [Piatetsky-Shapiro 1991]. The manipulation of certainty factors in expert systems is often straightforward; for example, assuming certainty factors in the range [0,1], when a rule with certainty $p$ is applied to a fact with certainty $q$, the generated fact is assigned a certainty factor $p \cdot q$. Pragmatic considerations may have been the reason that commercial expert systems often prefer this mostly informal representation of uncertainty over more formal approaches that are based on probability theory. However, many objections have been raised against certainty factors, showing that the lack of firm semantics may lead to unintuitive results [Pearl 1988].

## Possibility and Probability

Fuzzy set theory offers a comprehensive approach to modeling uncertainty in information systems. The approach described here is derived from Prade and Testemale 1984; Raju and Majumdar 1988; and Zemankova and Kandel 1985.

The basic concept of fuzzy set theory is the *fuzzy set*. A fuzzy set $F$ is a set of elements where each element has an associated value in the interval [0,1] that denotes the *grade* of its membership in the set. For example, a fuzzy set may include the elements 20, 30, 40, and 50, with grades of membership 1.0, 0.7, 0.5, and 0.2, respectively.

As a relation is a subset of the product of several domains, one approach is to define relations that are fuzzy subsets of the product of several domains. Since each such relation is a fuzzy set, each of its tuples is associated with a membership grade. This definition admits uncertainty at the tuple level. For example, the tuple **(Dick, Pascal)** could belong to the relation PROFICIENCY(PROGRAMMER, LANGUAGE) with membership grade

0.9 (alternatively, this tuple may be interpreted as stating that Dick's proficiency in Pascal is 0.9).

Consider the fuzzy set defined earlier, and assume it is named YOUNG. It is also possible to interpret this set as the definition of the term *young*: It is a term that refers to 20-year-olds with possibility 1.0, to 30-year-olds with possibility 0.7, and so on. Thus, fuzzy sets may be applied to describe vague terms.

Consider now standard (nonfuzzy) relations, but assume that the elements of the domains are not values but fuzzy sets of values. This definition admits uncertainty at the data-value level. Having fuzzy sets for values permits specific cases where a value is one of four kinds:

1. A set or a range—for example, the value of DEPARTMENT can be **{shipping, receiving}**, or the value of SALARY can be **40,000-50,000**. Note that the interpretation of such sets and ranges is purely *disjunctive*: Exactly one element of the set or range is the correct value.
2. A fuzzy value—for example, the value of AGE can be **young**.
3. A null value.
4. A simple value (no uncertainty).

Finally, by defining a fuzzy relation as a fuzzy subset of the product of domains of fuzzy sets, both kinds of uncertainty can be accommodated.

To manipulate fuzzy databases, the standard relational algebra operators must be extended to fuzzy relations. The first approach, in which relations are fuzzy sets but elements of domains are crisp, requires simple extensions to these operators. The second approach, in which relations are crisp but elements of domains are fuzzy, introduces more complexity because the softness of the values in the tuples creates problems of value identification (e.g., in the join or in the removal of replications after projections). Also, in analogy with standard mathematical comparators such as = or <, which are defined via sets of pairs, the second approach introduces fuzzy comparators such as *similar-to* or *much-greater-than*, which are defined via fuzzy sets of pairs. These fuzzy operators offer the capability of expressing fuzzy (uncertain) retrieval requests.

In Barbara et al. 1992, a model is described that handles uncertainty with traditional probability distributions (rather than the possibility distributions of the fuzzy models). A *probability distribution function* of a variable $X$ over a domain $D$ assigns each value $d \in D$ a value between 0 and 1, as the probability that $X = d$. One important difference between probability and possibility distribution functions is that the sum of the probabilities assigned to the elements of $X$ must be exactly 1. The definition of a probabilistic database is similar to the second definition of fuzzy databases: standard relations but with domain values that are, in general, probability distribution functions. A feature of this model is that it allows probability

distributions that are incompletely specified: Each such distribution is complemented with a *missing value* that is assigned the balance of the probability.

## Distances

An approach to uncertainty that has been applied successfully to both databases systems and information retrieval systems handles uncertainty with *distance*. The basic idea is to model the real world with apparently certain descriptions and to rely on definitions of distances among descriptions to create *neighborhoods* of descriptions.

Thus, any uncertainty about a real-world object $o$ is ignored, and an apparently certain description of it $e$ is stored. It is then hoped that this negligence would be compensated by having $e$ somewhere in the neighborhood of the true description. When a request for information specifies this true description, $e$ would be retrieved along with the other neighbors of the true description.

As an example, consider an information retrieval system that describes documents with sets of keywords [Salton and McGill 1983; Turtle and Croft 1992; van Rijsbergen 1979]. Such systems often represent keyword sets with vectors: The dimension of each vector is the number of possible keywords, and a specific vector position is 1 if a particular keyword is in the set and 0 otherwise. Often, there is uncertainty whether a specific vector is the true description of a given document. By establishing a distance among document descriptions, usually with some vector metric, and retrieving all the information in the neighborhood of a request-vector, the negative effects of inaccuracies in the description are diminished.

As another example, consider relational database systems. Such systems describe objects with tuples, and often there is uncertainty regarding the value of some attribute in a given tuple. It is possible to establish a distance among the elements of the domain of this attribute. Then, when a query specifies a value of this attribute, all the tuples would be retrieved whose value for that attribute is in the neighborhood of the specified value [Motro 1988].

We assumed here that descriptions are subject to uncertainty (which is ignored) and requests are certain. The same solution applies when descriptions are certain and requests are subject to uncertainty. Such requests would be specified with apparent certainty and would be answered with the neighborhood of the request. Again, the negative effects of inaccuracies in the request would be moderated. Uncertainty in requests is discussed in more detail in section 22.4.1.

A refinement of this general method is to admit certainty factors (section 22.3.2) in the descriptions and in the requests. For example, vectors describing keyword sets use values in the range [0,1] to denote the certainty that a specific keyword applies to the given document (or, alterna-

tively, to denote the strength with which this keyword applies). Similarly, request-vectors use such values to denote the weights of various keywords in the overall request.

### Soundness and Completeness

The final approach we discuss has been developed in the context of relational databases [Motro 1989]. Instead of modeling imperfections in the information (i.e., uncertainty), it suggests declaring the portions of the database that are prefect models of the real world (and thereby the portions that are possibly imperfect).

Thus, like distances and unlike incompleteness, certainty factors, or fuzziness, the descriptions themselves have no special features for uncertainty (i.e., they appear certain). However, meta-information provides the distinction between certain and uncertain information. (Recall that fuzziness and distances also require meta-information: the definitions of fuzzy terms or the measures of proximity.)

With this information included in the database, the database system can *qualify* the accuracy of the answers it issues in response to queries: Each answer is accompanied by statements that define the portions that are guaranteed to be perfect.

This approach interprets certainty, which it terms *integrity*, as a combination of *soundness* and *completeness*. A description is sound if it includes *only* information that occurs in the real world; a description is complete if it includes *all* the information that occurs in the real world. Hence, a description has integrity if it includes the whole truth (completeness) and nothing but the truth (soundness).

The approach uses the mechanism of *views* to declare the portions of the database or the portions of an answer that have integrity. It describes a technique for inferring the views of individual answers that are guaranteed to have integrity from the views of the entire database that are known to have integrity.

The concept of view completeness is similar to an assumption that a certain view of the database is *closed world* [Reiter 1978]; the notion of view soundness is shown to be a generalization of standard database integrity constraints.

## 22.4 Transaction Uncertainty and Processing Uncertainty

While most of the work in uncertainty has focused on description uncertainty, transaction and processing uncertainty have just as important impact on the quality of the information delivered to users. In this section we discuss briefly issues and solutions that concern uncertainty in the

definition of transformations (e.g., queries), in the definition of modifications (e.g., updates or restructuring operations), and in the processing of such transactions.

### 22.4.1 Transformations

Transformations are operations that derive new descriptions from stored descriptions. The most frequent type of transformation is requests from users for information (queries). Uncertain requests may occur for different reasons.

At times, users of database systems have insufficient knowledge of the database and database system they are using: They might not have a clear idea of the information available in the database (or how it is organized), or they might not know how to formulate their requests with the tools provided by the system.

Requests for information formulated by such *naïve* users exhibit a high level of uncertainty. They range from requests that cannot be interpreted by the system (for reasons that are either syntactic or semantic) to requests that do not achieve correctly the intentions of the users (or achieve them only in part).

Regardless of their level of expertise, occasionally users may try to access a database system with only a *vague* idea of the information they seek. For example, a user may be accessing an electronic catalogue for a product that would be interesting or exceptional. Alternatively, users could have a clear idea of the information they want but might lack the information necessary to specify it to the system. An example is a user who wishes to look up the meaning of a word in a dictionary but cannot provide its correct spelling.

To summarize, we distinguish among (1) uncertainty about the *information available* (or how it is organized), (2) uncertainty about the *information needed* (or how to denote it in terms acceptable to the system), and (3) uncertainty about the *system languages and tools* that are used to formulate requests.

To address all these, the approach has been to develop alternative access tools. Browsers allow users to access information in either situation discussed above [D'Atri et al. 1992; Motro 1986; Rogers and Cattell 1987]. Interactive query constructors conduct user-system dialogues to arrive at satisfactory formulations of user requests [Williams 1984]. Vague query processors allow users to embed uncertain specifications in their requests (e.g., neighborhood queries) [Ichikawa and Hirakawa 1986; Motro 1988]. Error-tolerant interfaces use relaxed formalisms in their interpretation of requests [Motro 1990].

As mentioned above, even after a request for information has been accepted by the system and its answer delivered to the user, uncertainty might still persist because it is not always possible to verify that the request

has indeed achieved correctly the intention of the user. Often, the only assurance that the information delivered is the information needed is that the user is somewhat familiar with it. In the absence of such familiarity, uncertainty will persist.

Hence, one must accept that there would be frequent instances where the answer that is delivered is inaccurate, yet both the system and the user are unaware of this inaccuracy. Often, the uncertainty of apparently certain answers is revealed only when a conflicting answer to the same request is received from another information system.

### 22.4.2 Modifications

Modifications (update and restructuring) are operations that affect the descriptions stored in information systems. Like transformations (queries), modifications are defined by users, and here also we distinguish among three main sources of uncertainty: (1) insufficient knowledge of the system, (2) insufficient knowledge of the specific database that is being modified, and (3) uncertainty about the information embedded in a modification.

Many of the approaches aimed at alleviating the problems of transformation uncertainties (section 22.4.1 above) are also applicable to modification uncertainties. However, fewer tools have been developed to address modification uncertainties. A possible explanation is that, while modifications may be attempted by users at all levels of expertise, it is often assumed that users who modify databases should have good familiarity with the database and database system.

The third source of uncertainty, vague or imprecise modifications, is unrelated to expertise. One example is a request to add information that is uncertain—for example, "the new manager is either Paul or John." Another example is a request to delete information, with uncertainty about what exactly should be deleted—for example, "some of the telephone numbers are no longer valid."

However, this kind of uncertainty is not any different from the description uncertainty that was the subject of section 22.3. Thus, the first request would be accommodated as any information of the kind "exactly one of the following values holds," and the second request would be accommodated as any information of the kind "some of the following values hold." (Of course, if the system cannot model these kinds of uncertainty, then it would not be able to handle these modifications.)

### 22.4.3 Processing

Even when a description $D$ and a transformation $t$ are free of uncertainty, the result $t(D)$ may be uncertain because of the methods used by the system to process requests. In certain applications, an information system might

allocate only limited computational resources to process a request [Imielinski 1987]. For example, a recursive query to a genealogical database to list all the ancestors of a specific individual might be terminated after a predetermined period of time (presumably the number of ancestors retrieved by then would be sufficient). In other applications, query processing might involve randomizations, sampling, or other estimation techniques [Kwan et al. 1992]. For example, a statistical database system might introduce perturbations into its answers deliberately, for reasons of security. In each case, the answers would exhibit uncertainty.

Finally, it is sometime considered advantageous to sacrifice accuracy for the sake of *simplicity*. Recent research on *intensional answers* [Cholvy and Demolombe 1986; Shum and Muntz 1988; Pirotte et al. 1991; Motro 1994] has focused on the generation of abstract answers that describe the exhaustive answers compactly albeit imperfectly. For example, a query to list the employees who earn over $50,000 might be answered simply and compactly "engineers," even when the set of engineers and the set of employees who earn over $50,000 are not exactly the same (e.g., when the two sets overlap substantially or when one set contains the other).

## 22.5 Hindrances and Challenges

Commercial database systems have been slow to incorporate uncertainty capabilities. While solutions based on fuzzy set theory are gaining acceptance in several technologies, commercial database systems have not embraced this solution yet. Often, the only capabilities widely available are for handling null values and for specifying specifying uncertain queries by allowing the substitution of patterns for constants.

Examining the possible reasons for this slow acceptance may suggest directions for further research. First, database systems practitioners are concerned primarily with the *performance* of their systems. Here, many of the algorithms for matching uncertain data or for processing uncertain requests are highly complex and inefficient.

Practitioners are also concerned with *compatibility*. This dictates that capabilities for accommodating uncertainty should be offered as strict extensions of existing standards. Additionally, database practitioners have often been dissatisfied with various *idiosyncratic implementations* of uncertainty capabilities. This may have had a chilling effect on further implementations.

Another hindrance for database systems with uncertainty capabilities may lie in the *expectations of users*. A fundamental principle of database systems has been that queries and answers are never open to subjective interpretations, and users of database systems have come to expect their queries to be interpreted unambiguously and answered with complete accuracy. In contrast, users of information retrieval systems would be

pleased with a system that delivers a high rate of recall (proportion of relevant material retrieved) and precision (proportion of retrieved material that is relevant). Similarly, users of expert systems are aware that conclusions offered by automated experts are often questionable, and they would be satisfied with systems that have shown to have a high rate of success. A database system that must adhere to the principles of unambiguity and complete accuracy cannot accommodate the full range of uncertain information; for example, it can accommodate null values or disjunctive data but not (because of its more subjective nature) probabilistic or possibilistic data.

Recently, new applications have emerged that require database systems with uncertainty capabilities. Several of these challenging applications are described below. Their description is followed by a discussion of a possible source of uncertainty technology and the challenges involved in adapting it for database systems.

### 22.5.1 Heterogeneous Database Environments

In recent years, the integration and interchange of information among heterogeneous database systems has been recognized as an important area of database system research and development. Multidatabase environments present a strong case for having uncertainty management capabilities. While individual database systems are usually careful to avoid internal inconsistencies (mostly with methodical design that avoids repeating the same information at multiple locations in the system), when information from independent systems with overlapping information is integrated, inconsistencies will often surface. Thus, even when individual answers are free from any uncertainty, their integration in a global answer would introduce uncertainty. Therefore, database systems in a multidatabase environment should be capable of combining conflicting answers into a single (uncertain) answer and then storing and manipulating such information [Motro 1993].

In a heterogeneous multidatabase environment, the integration and interchange of information requires protocols for translating information among the different models. The possibility of having database systems that are based on different uncertainty formalisms presents an additional challenge: to develop protocols for finding *common* uncertainty formalisms so that uncertain information can be integrated with minimal loss of information.

### 22.5.2 Multimedia Databases

As discussed in section 22.3, retrieval from traditional databases is usually based on *exact matching*, where a request for data establishes a specific

retrieval goal, and the database system retrieves the data that match it exactly. Presently, the management of image databases largely follows the same paradigm: While images are stored (in digitized form) in large databases, retrieval is performed on *textual descriptions* of these images that are stored along with the images themselves.

A more ambitious approach, such as IBM's Query By Image Content (QBIC) [Faloutsos et al. 1994], is to retrieve images that match a given *image*. Image-matching techniques are usually based on algorithms of *best matching*. As with the information retrieval systems discussed earlier, the use of uncertainty formalisms is essential. A similar problem is to match handwritten addresses against a database of addresses.

### 22.5.3 Imputation and Knowledge Discovery

In various applications, notably in scientific and statistical projects, it is necessary to estimate missing data (nulls) from other data that are available. For example, a missing measurement is estimated by other measurements made by the same instrument at other times, as well as by measurements made by other instruments at the same time. This process, usually referred to as *imputation*, yields information of varying degrees of uncertainty. The management of this information cannot be done by traditional database techniques and requires the use of uncertainty techniques.

The inference of missing values from their contexts is related to the more general issue of discovering new knowledge in large databases. Database knowledge is usually *declared*: Rules and constraints are assumed to be definite and accurate and to hold in any database instance (exceptions, if any, must be stated). In contrast, *discovered* knowledge is always subject to uncertainty: The patterns and rules are often tenuous and could be refuted by future data. Again, the management of such information requires uncertainty techniques.

### 22.5.4 Adapting Artificial Intelligence for Databases

Modeling uncertainty in database systems involves a classical dilemma. On one hand, we want a model as rich and powerful as possible—for example, a single concept of information that is capable of representing elegantly all known kinds of uncertain information, as well as certain information. On the other hand, database systems must abide by crucial constraints of simplicity and efficiency, of both their descriptions and the manipulations of their descriptions.

Like database systems, the field of artificial intelligence has been concerned with modeling accurately our knowledge of the real world. Numerous uncertainty theories have been developed; mostly they are founded on

nonclassical logics, on probability theory, on belief functions, or on possibility theory [Lea Sombe 1991; Motro and Smets 1992; Smets and Clarke 1991; Smets and Motro 1993].

It might be said that, traditionally, research in artificial intelligence has been emphasizing rich and powerful models, striving to model accurately all the minute nuances of reality. On the other hand, research in database systems has been emphasizing economical representations with lower expressivity but with small representational overhead and high processing efficiency.

A notable case in point is the so-called *semantic data models* that were defined in the late 1970s to enhance the modeling capabilities of early database systems [Hammer and McLeod 1981; Hull and King 1987]. These models incorporated modeling features, such as generalization and aggregation hierarchies, that had been adapted from research in artificial intelligence. An even earlier example is the adaptation of *semantic networks* for databases [Roussopoulos and Mylopoulos 1975]. A third and more recent example is *knowledge-rich database systems* that incorporate inference rules expressed in mathematical logic [Ullman 1988; Ullman 1989; Ceri et al. 1990]. Not surprisingly, a major research thrust among database researchers who have been working in this area has been the development of *highly efficient* inference techniques for a *limited* class of rules.

Thus, database systems may be said to have embraced poor-person's AI or, more graciously, to have adapted AI concepts to work under the strict constraints of database systems. The adaptation of uncertainty theories that have been developed for artificial intelligence to working database techniques is an important challenge for database researchers. The recent success of several information-extensive diagnostic and decision-support systems, such as QMR-BN [Henrion and Suermondt 1993], hints at the possible rewards that research in this direction may provide.

## 22.6 Conclusion

Database systems of the current generation are not designed to manage information that is permeated with uncertainty. Uncertain data is stored as null values or it must be excluded from the database, and users with uncertain requests can use queries with simple patterns or they must browse the database to find their answers.

To provide satisfactory solutions to new applications, such as information integration in multidatabase environments, retrieval of images by content, and inference of missing values or new knowledge, future database systems would have to include stronger capabilities for dealing with uncertainty.

To assure the success of any future endeavors in this area, various hindrances must be overcome; in particular, issues of performance and compatibility must be addressed. Research into uncertainty in the field of artificial intelligence may provide a suitable source of technology for database systems.

## ACKNOWLEDGMENTS

This work was supported in part by NSF Grant No. IRI-9007106 and by a ARPA grant, administered by the Office of Naval Research under Grant No. N0014-92-J-4038.

## REFERENCES

Abiteboul, S., and Grahne, G. 1985. Update Semantics for Incomplete Databases. *Proceedings of the Eleventh International Conference on Very Large Data Bases*, Stockholm, 1–12.

Barbara, D., Garcia-Molina, H., and Porter, D. 1992. The Management of Probabilistic Data. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4, No. 5, 487–502.

Borgida, A. 1985. Language Features for Flexible Handling of Exceptions in Information Systems. *ACM Transactions on Database Systems*, Vol. 10, No. 4, 565–603.

Ceri, S., Gottlob, G., and Tanka, L. 1990. *Logic Programming and Databases*. Springer-Verlag, Berlin.

Cholvy, L., and Demolombe, R. 1986. Querying a Rule Base. *Proceedings of the First International Conference on Expert Database Systems*, Charleston, South Carolina, 365–371.

Codd, E. F. 1979. Extending the Database Relational Model to Capture More Meaning. *ACM Transactions on Database Systems*, Vol. 4, No. 4, 397–434.

Date, C. J. 1986. *Relational Database: Selected Writings*. Addison-Wesley, Reading, Mass.

Date, C. J. 1990. NOT is Not "Not"! In: *Relational Database Writings 1985-1989*. Addison-Wesley, Reading, Mass.

D'Atri, A., Motro, A., and Tarantino, L. 1992. ViewFinder: An Object Browser. Technical report, Department of Information and Software Systems Engineering, George Mason University.

Demolombe, R., and Farinas del Cerro, L. 1988. An Algebraic Evaluation Method for Deduction in Incomplete Data Bases. *Journal of Logic Programming*, Vol. 5, 183–205.

Faloutsos, C., Equitz, W., Flickner, M., Niblack, W., Petkovic, D., and Barber, R. 1994. Efficient and Effective Querying by Image Content. *Journal of Intelligent Information Systems* (in press).

Hammer, M., and McLeod, D. 1981. Database Description with SDM: A Semantic Database Model. *ACM Transactions on Database Systems*, Vol. 6, No. 3, 351–386.

Harmon, P., and King, D. 1985. *Expert Systems—Artificial Intelligence in Business.* John Wiley & Sons, New York.

Henrion, M., and Suermondt, J. 1993 Probabilistic and Bayesian Representations of Uncertainty in Information Systems: A Pragmatic Introduction. *Proceedings of the Workshop on Uncertainty Management in Information Systems: From Needs to Solutions,* Avalon, Cal., 71–90.

Hull, R., and King, R. 1987. Semantic Database Modeling: Survey, Applications and Research Issues. *Computing Surveys,* Vol. 19, No. 3, 201–260.

Ichikawa, T., and Hirakawa, M. 1986. ARES: A Relational Database with the Capability of Performing Flexible Interpretation of Queries. *IEEE Transactions on Software Engineering,* SE-12(5), 624–634.

Imielinski, T. 1987. Intelligent Query Answering in Rule Based Systems. *Journal of Logic Programming,* Vol. 4, No. 3, 229–257.

Imielinski, T. 1989. Incomplete Information in Logical Databases. *Data Engineering,* Vol. 12, No. 2, 29–40.

Kwan, S., Olken, F., and Rotem, D. 1992. Uncertain, Incomplete, and Inconsistent Data in Scientific and Statistical Databases. *Proceedings of the Workshop on Uncertainty Management in Information Systems: From Needs to Solutions,* Mallorca, Spain, 64–91.

Lea Sombe, et al. 1991. Special issue on reasoning under incomplete information in artificial intelligence. *International Journal of Intelligent Systems,* Vol. 5, No. 4.

Maier, D. 1983. *The Theory of Relational Databases.* Computer Science Press, Rockville, Md.

Motro, A. 1986. BAROQUE: A Browser for Relational Databases. *ACM Transactions on Office Information Systems,* Vol. 4, No. 2, 164–181.

Motro, A. 1988. VAGUE: A User Interface to Relational Databases That Permits Vague Queries. *ACM Transactions on Office Information Systems,* Vol. 6, No. 3, 187–214.

Motro, A. 1989. Integrity = Validity + Completeness. *ACM Transactions on Database Systems,* Vol. 14, No. 4, 480–502.

Motro, A. 1990. FLEX: A Tolerant and Cooperative User Interface to Databases. *IEEE Transactions on Knowledge and Data Engineering,* Vol. 2, No. 2, 231–246.

Motro, A. 1993. A Formal Framework for Integrating Inconsistent Answers from Multiple Information Sources. Technical Report ISSE-TR-93-106, Department of Information and Software Systems Engineering, George Mason University.

Motro, A. 1994. Intensional answers to database queries. *IEEE Transactions on Knowledge and Data Engineering* (in press).

Motro, A., and Smets, P., eds. 1992. *Proceedings of the Workshop on Uncertainty Management in Information Systems: From Needs to Solutions,* Mallorca, Spain.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, Cal.

Piatetsky-Shapiro, G. 1991. Discovery, Analysis and Presentation of Strong Rules. In: *Knowledge Discovery in Databases,* 227–248. G. Piatetsky-Shapiro and W. Frawley, eds. AAAI Press/MIT Press, Menlo Park, Cal.

Pirotte, A., Roelants, D., and Zimanyi, E. 1991. Controlled Generation of Intensional Answers. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 3, No. 2, 221–236.

Prade, H., and Testemale, C. 1984. Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries. *Information Sciences*, Vol. 34, No. 2, 115–143.

Raju, K. V. S. V. N., and Majumdar, A. 1988. Fuzzy Functional Dependencies and Lossless Join Decomposition of Fuzzy Relational Database Systems. *ACM Transactions on Database Systems*, Vol. 13, No. 2, 129–166.

Reiter, R. 1978. On Closed World Data Bases. In: *Logic and Databases*, 55–76. Plenum Press, New York.

Rogers, T. R., and Cattell, R. G. G. 1987. Object-Oriented Database User Interfaces. Technical report, Information Management Group, Sun Microsystems.

Roussopoulos, N., and Mylopoulos, J. 1975. Using Semantic Networks for Data Base Management. *Proceedings of the First International Conference on Very Large Data Bases*, 144–172.

Salton, G., and McGill, M. J. 1983. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York.

Shum, C. D., and Muntz, R. 1988. An Information-Theoretic Study on Aggregate Responses. *Proceedings of the Fourteenth International Conference on Very Large Data Bases*, Los Angeles, 479–490.

Smets, P., and Clarke, M. R. B. 1991. Special issue on uncertainty, conditionals, and non-monotonicity. *Journal of Applied Non-Classical Logics*, Vol. 1, No. 2.

Smets, P., and Motro, A., eds. 1993. *Proceedings of the Workshop on Uncertainty Management in Information Systems: From Needs to Solutions*, Avalon, Cal.

Turtle, H. R., and Croft, W. B. 1992. Uncertainty in Information Retrieval Systems. *Proceedings of the Workshop on Uncertainty Management in Information Systems: From Needs to Solutions*, Mallorca, Spain, 111–137.

Ullman, J. D. 1988. *Database and Knowledge-Base Systems, Volume I.* Computer Science Press, Rockville, Md.

Ullman, J. D. 1989. *Database and Knowledge-Base Systems, Volume II.* Computer Science Press, Rockville, Md.

van Rijsbergen, C. J. 1979. *Information Retrieval*, 2nd Edition. Butterworths, London.

Williams, M. D. 1984. What Makes RABBIT Run? *International Journal of Man-Machine Studies*, Vol. 21, No. 4, 333–352.

Zemankova, M., and Kandel, A. 1985. Implementing Imprecision in Information Systems. *Information Sciences*, Vol. 37, Nos. 1–3, 107–141.

Zicari, R. 1992. Databases and Incomplete Information. *Proceedings of the Workshop on Uncertainty Management in Information Systems: From Needs to Solutions*, Mallorca, Spain, 52–63.