

IMPRECISION AND UNCERTAINTY IN DATABASE SYSTEMS

Amihai Motro

Department of Information and Software Systems Engineering
George Mason University
Fairfax, VA 22030-4444
ami@gmu.edu

Abstract

Databases are models of the real world. Yet, our knowledge of the real world is often imperfect, thus challenging our ability to create databases of integrity. To uphold the integrity of a database in situations where knowledge of the real world is imperfect, one may either (1) restrict the model to that portion of the real world about which perfect information is available, or (2) develop formalisms that allow the representation of imperfect information. This paper surveys some of the better-known database formalisms for capturing imperfect information. Imperfections in the specification and processing of transactions also have important impact on the quality of the information delivered to users, and this survey discusses them as well.

Keywords: database, imperfect information, uncertain information, imprecise information, probabilistic database, fuzzy database, database soundness and completeness, vague query.

1 Introduction

A database is a computer model of the real world. Like any other model, it attempts to portray an abstracted version of reality, and the level of this abstraction is usually determined by the expected applications. As with any other model, the most important consideration is *integrity*: a

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

database should provide an accurate representation of the real world it presumes to capture. Yet, our knowledge of the real world is often imperfect, thus challenging our ability to create databases of integrity.

There are two solutions for upholding the integrity of a database in situations where knowledge of the real world is imperfect.

The first solution is to *restrict* the model to that portion of the real world about which perfect information is available. In the relational data model¹ this implies that a tuple that describes a particular employee will be excluded altogether from the database, if any of its attributes (e.g., age or salary) are missing or are suspect of imprecision. The resulting database will only model those employees for whom perfect information is available.

The second solution is to develop data models that allow the representation of imperfect information. Assume that the available information about the age of a particular employee is imperfect; for example, the age is only known to be within a specific range. If the data model had features for specifying and manipulating ranges, then this imperfect information could be captured in a database that maintains its integrity.

Because the second solution often permits additional applications, most database systems adhere to data models that include at least some features for capturing imperfect information; the ability to store “null values” being the most commonly available feature.

In this paper we survey some of the better known approaches for capturing imperfect information. Our emphasis is on relational databases (and to some degree also information retrieval systems and expert systems). The reader should be aware that the representation of imperfect knowledge and reasoning under uncertainty are active areas within artificial intelligence research [4, 31]. The management of imperfect information in both information systems and artificial intelligence is discussed in considerable detail in [29, 39].

There are different kinds of imperfections and Section 2 provides definitions for the basic kinds. Our introductory discussion focused on the *representation* of imperfect information, but other components of a databases system, e.g., query formulation or query processing, are susceptible to imperfections as well. Section 2 also uses a simple model of a

¹We use the term *model* for the actual *representation* of the real world, and the term *data model* for the *formalism* used in the representation.

database system to classify imperfections into three kinds: description, manipulation and processing. Our survey in Sections 3 and 4 follows this classification. Section 5 concludes this paper with a brief discussion.

2 Terminology

2.1 Three Basic Kinds of Imperfect Information

There have been many attempts to classify the various possible kinds of imperfect information. In this paper we concern ourselves with three basic kinds: error, imprecision, and uncertainty. We note that other kinds of imperfection have been observed, including vagueness and ambiguity [38], but they are not as important for database systems.

Error. Erroneous information is the simplest kind of imperfect information. Database information is erroneous when it is different from the true information. We shall take the approach that all errors, “large” or “small”, compromise the integrity of a database, and should not be tolerated. An important kind of erroneous information is *inconsistency*. Occasionally, the same aspect of the real world is represented more than once (this could be in the same database, or in different databases that are considered together). When the different representations are irreconcilable, the information is inconsistent. Issues of information inconsistency are of particular relevance, given the present interest in information integration [16, 27].

Imprecision. Database information is imprecise when it denotes a *set* of possible values, and the real value is one of the elements of this set. Thus, imprecise information is not erroneous and does not compromise the integrity of a database. Specific kinds of imprecise information include *disjunctive* information (e.g., John’s age is either 31 or 32), *negative* information (e.g., John’s age is not 30), *range* information (e.g., John’s age is between 30 and 35, or John’s age is over 30), and information with *error margins* (e.g., John’s age is 34 ± 1 year). The two extreme kinds of imprecision are *precise* information (when the set of possibilities is a singleton), and *null values* (a null value usually denotes that no information is available, yet could be regarded as imprecise information where the set of possible values encompasses the entire domain of legal values).

Uncertainty. At times, our knowledge of the real world (precise or imprecise) cannot be stated with absolute confidence. This requires that we qualify our confidence in the information stated. Again, information with qualified certainty is not erroneous and does not compromise the integrity of a database. Whereas the statement “John’s age is either 31 or 32” denoted imprecision, the statement “John is probably 32” denotes uncertainty. At times, precision can be traded for certainty and vice-versa. A precise value may entail low certainty, but as this value is substituted by values that are progressively less precise, certainty increases gradually, until finally it is maximal for a value which is “minimally precise” (i.e., a null value).

2.2 A Model of Database Systems

The impact of imperfect information on a database system reaches beyond issues of the integrity of the representation. To explain this, we use a simple model of a database system.

A database system includes 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). Specifically, 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).

The objective of any 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 imperfect, either because D is imperfect, or because t is imperfect, or because the processing of t against D is imperfect. In

turn, the imperfection of D may owe to imperfections either in the initial description, or in some later modification m .

In the next two sections we discuss database imperfections in these three categories: descriptions, manipulations (transformations and modifications), and processing.

3 Imperfect Descriptions

Of the three categories mentioned above, the description of imperfect information has received the most attention, and in this section we briefly survey the major approaches to this issue. Of course, any approach for describing imperfect information must also address the manipulation of imperfect descriptions (e.g., querying and updating).²

3.1 Null and Disjunctive 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 precision and certainty. Occasionally, this problem may be evaded simply by not modeling any object whose description is imperfect. Often, however, the consequences of this approach are unacceptable, and imperfect descriptions must be admitted.

The least ambitious approach to admitting imperfect descriptions is to ignore all partial information about the imperfect parts of a description that may be available, and to model them with a pseudo-description, called *null*, that denotes unavailability [22, 9, 19, 45].³ The semantics of a null value is that *any* value from the corresponding domain of legal values is an equally probable candidate for the true value.

²Note the difference between manipulations of imperfect descriptions, which are discussed in this section, and imperfect manipulations, which are the subject of Section 4.

³Null values have also been used to denote *inapplicability*; i.e., that a specific attribute is inapplicable to a specific object. Such null values do not indicate imperfection of information. Hence, the term *null* will be used here to denote applicability but unavailability.

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 a simple task. For example, an extension to the relational calculus that is founded on a three-valued logic [8] has been the subject of criticism [10]. Inference in incomplete databases is discussed in [12]. Updates of incomplete databases are discussed in [1].

Null values model the ultimate lack of information about a value (except that it exists). Other kinds of null values have been suggested that express some additional information. For example, two values in a database may be unavailable, but are known to be identical. Such partial information may be modeled by using distinguishable instances of nulls (*marked nulls*) throughout the database in general, but the same null instance for these two values. Recording this partial information proves to be useful in the performance of joins.

At times, it is known that a missing value belongs to a more limited set of values (possibly, a range of values). This partial information has been modeled by *disjunctive values* (*Or-objects*). A disjunctive value is a set of values, that includes the true value. Hence, disjunctive values are more informative than nulls (a null value is a specific kind of disjunctive value, where the set of possibilities is the entire domain). Disjunctive databases are discussed in [20, 30].

Clearly, null and disjunctive values both express *imprecision*.

3.2 Confidence Factors

Confidence factors denote confidence in various elements of the description. Hence, they offer a simple tool for representing *uncertainty*. Confidence factors have been applied in both information retrieval systems and in expert systems, but not in database systems proper.

In information retrieval systems, confidence 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) [41, 37]. Methods have even been developed for computing confidence factors automatically, by scanning documents and applying keyword counting techniques. The manipulation of these confidence factors is relatively simple, as they are easily accommodated in the vector space models that are often used in information retrieval.

In expert systems, confidence factors have been used to denote confidence that stated facts and rules indeed describe real-world objects [15]. Such factors are usually declared by the knowledge engineers as part of the knowledge acquisition process, but can also be derived automatically as part of a knowledge discovery process [32]. The manipulation of confidence factors in expert systems is often straightforward; for example, assuming confidence factors in the range $[0,1]$, when a rule with confidence p is applied to a fact with confidence q , the generated fact is assigned a confidence 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 confidence factors, showing that the lack of firm semantics may lead to unintuitive results [31].

3.3 Probabilistic Databases

Probabilistic databases represent information with variables and their probability distributions. In a relational framework, the value of a particular attribute A for a specific tuple t is a variable $A(t)$, and this variable has an associated *probability distribution* $P_{A(t)}$. $P_{A(t)}$ assigns values in the range $[0,1]$ to the elements of the domain of attribute A , with the provision that the sum of all values assigned is 1. An example of a probabilistic value is the variable $Age(john)$ and this probability distribution:⁴

$$P_{Age(john)} = \begin{cases} 32 & 0.6 \\ 33 & 0.4 \end{cases}$$

The interpretation of this information is that with probability 0.6 the age of John is 32, with probability 0.4 it is 33, and the probability that John's age is some other value is 0.

A probabilistic relational model based on this approach and a suitable set of operators are defined in [3]. The model allows probability distributions that are incompletely specified: each such distribution is completed with a special null value which is assigned the balance of the probability. It is also possible to define probability distributions for combinations of interdependent attributes. A somewhat different (but equivalent) model is given in [7]. In this model a database is a collec-

⁴We use the key value *john* to identify a specific tuple.

tion of objects, where each object has an associated set of attributes (a scheme), a set of tuples over that scheme (a domain), and a probability distribution function that assigns each tuple of the domain a probability (the sum of the probabilities for the entire domain of an object is 1).

We have observed that disjunctive information; i.e., “32 or 33” is a form of imprecise information, whereas “32 with confidence 0.6” is a form of uncertain information. Consider now this statement “32 with probability 0.6 and 33 with probability 0.4”. In many ways, such probabilistic information is a combination of both imprecision and uncertainty. It is imprecise because it denotes several different alternatives, and it is uncertain because every alternative is associated with a likelihood.

3.4 Fuzzy and Possibilistic Databases

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. An example of a fuzzy set (using a common notation) is: $F = \{30/1.0, 31/1.0, 32/1.0, 33/0.7, 34/0.5, 35/0.2\}$. The elements 30, 31 and 32 are in this set with grade of membership 1.0, the elements 33, 34 and 35 have corresponding grades of membership 0.7, 0.5 and 0.2, and all elements not shown have grade of membership 0.

Several different models of databases have been based on fuzzy set theory. The simplest model extends relations, which are subsets of a Cartesian product of domains, to fuzzy relations; i.e., fuzzy subsets of the product of domains [6, 34]. Thus, each tuple in a relation is associated with a membership grade. For example, the tuple $(john, pascal)$ belongs to the relation $Proficiency(Programmer, Language)$ with membership grade 0.9. Associating a membership grade with each tuple may be regarded as a statement of *uncertainty*.

Alternatively, the same tuple may be interpreted as stating that John’s proficiency in Pascal is 0.9. In this interpretation, the membership grades indicate the “strength” of the association between the components of the tuple (in this case, a programmer and a programming language). These different interpretations should not be confused, and must be taken into account when defining operations for manipulating fuzzy relations.

The theory of possibility is based on fuzzy set theory. In a relational framework, the value of a particular attribute A for a specific tuple t is a variable t_A , and this variable has an associated *possibility distribution* $\pi_{A(t)}$. $\pi_{A(t)}$ assigns values in the range $[0, 1]$ to the elements of the domain of attribute A . Using the same variable $Age(john)$, an example of a possibilistic value is:

$$\pi_{Age(john)} = \begin{cases} 30 & 1.0 \\ 31 & 1.0 \\ 32 & 1.0 \\ 33 & 0.7 \\ 34 & 0.5 \\ 35 & 0.2 \end{cases}$$

The interpretation of this information is that it is completely possible that the age of John is 30, 31 or 32, it is “very” possible that it is 33, it is “somewhat” possible that it is 34, it is “remotely” possible that it is 35, and it is completely impossible that it is any other age. If this individual possibility distribution is assigned a name; e.g., *early-thirties*, then it is also possible to interpret it as a definition of the linguistic term “early thirties”: it is a term that refers to 30–32 year olds with possibility 1.0, to 33 year olds with possibility 0.7, etc. Thus, possibility distributions may be used to describe vague linguistic terms.

Consider now standard relations, but assume that the elements of the domains are not values, but possibility distributions [44, 33]. Having possibility distributions for values permits specific cases where a value is one of several kinds: (1) A vague term; for example, a value of *Age* can be *early-thirties*. (2) A disjunctive value; for example, a value of *Department* can be $\{shipping, receiving\}$, or a value of *Salary* can be $40,000-50,000$. (3) A null value. (4) A simple value. Our earlier observation, that probabilistic information expresses both uncertainty and imprecision, applies to possibilistic information as well.

To manipulate fuzzy databases, the standard relational algebra operators must be extended to fuzzy relations. The first approach, where relations are fuzzy sets but elements of domains are “crisp”, requires simple extensions to these operators. The second approach, where 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 mathe-

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 (vague) retrieval requests.

Although their basic axioms and even much of their semantics are very different, probability theory, possibility theory and confidence factors share a common intuitive concept, namely, they all *qualify* database values with numbers that attempt to describe the likelihood that these specific values are indeed the correct one. For a thorough survey of database models based on fuzzy set and possibility theory, see [5].

3.5 Data Distances

An approach to imprecision that has been applied successfully to both databases systems and information retrieval systems handles imprecise information with *distance*. The basic idea is to model the real world with apparently-precise descriptions, to define the notion of distance among two descriptions, and thus to create *neighborhoods* of descriptions.

Thus, any imprecision about a real-world object o is ignored, and an apparently-precise 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 [37, 41, 40]. 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 imprecisions 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 [24].

We assumed here that descriptions are subject to imprecision (which is ignored) and requests are precise. The same solution applies when descriptions are precise and requests are imprecise. Such requests would be specified with apparent-precision, and would be answered with the neighborhood of the request. Again, the negative effects of imprecision in the request would be moderated. Imprecise requests are discussed in more detail in Section 4.1.

A refinement of this general method is to admit confidence factors (Section 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 confidence that a specific keyword applies to the given document (or, alternatively, to denote the strength with which this keyword applies). Similarly, request vectors use such values to denote the relative weights of various keywords in the overall request.

3.6 Soundness and Completeness

The final approach that we discuss [25] does not attempt to model imperfect information; instead, declarations are made of the portions of the database (i.e., *views*) that are perfect models of the real world (and thereby the portions that are possibly imperfect).

Thus, like distances (Section 3.5) and unlike disjunctive values (Section 3.1), confidence factors (Section 3.2), probabilistic values (Section 3.3) or possibilistic values (Section 3.4), the descriptions themselves have no special features for representing imperfection (i.e., they appear perfect). However, meta-information provides the distinction between perfect and imperfect information.

This approach interprets perfectness, 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).

With this information included in the database, the database system

can *qualify* the perfectness of the answers it issues in response to queries: each answer is accompanied by statements that define the portions (i.e., views) that are guaranteed to be perfect. A technique is described in [25] 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 notion of view completeness is similar to an assumption that a certain view of the database is *closed world* [35]. Also, *open nulls* [14] are actually declarations of views that are non-complete. The notion of view soundness is shown to be a generalization of standard database integrity constraints.

View soundness and view completeness have been extended in [27] to measure *relative* soundness and completeness of views (with respect to the real world). Let v denote the extension of some view in the database and let v_0 denote the real-world extension of the same view. Let $|v|$ denote the number of tuples in v . Then

$$\frac{|v \cap v_0|}{|v|}$$

expresses the proportion of the database answer that appears in the true answer. Hence, it is a measure of the soundness of v . Similarly,

$$\frac{|v \cap v_0|}{|v_0|}$$

expresses the proportion of the true answer that appears in the database answer. Hence, it is a measure of the completeness of v . Relative soundness and completeness are similar to the *precision* and *recall* measures used in information retrieval [37, 41], and are used to guide the *harmonization* of inconsistent answers in a multi-database environment.

4 Imperfect Manipulation and Processing

While most of the work on imperfect databases has focused on description imperfection, transaction and processing imperfection also have important impact on the quality of the information delivered to users. In this section we discuss briefly issues and solutions that concern imperfections 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.

4.1 Transformations

Transformations are operations that derive new descriptions from stored descriptions. The most frequent type of transformation are queries. Imperfect queries 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 naive users exhibit a high level of imperfection. They range from requests that cannot be interpreted by the system (for reasons that are either syntactical or semantical) 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) insufficient knowledge of the *information available* (or how it is organized), (2) vagueness with respect to the *information needed* (or how to denote it in terms acceptable to the system), and (3) insufficient knowledge of 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 [23, 36, 2, 11]. Interactive query constructors conduct user-system dialogues to arrive at satisfactory formulations of user requests [43]. Vague query processors allow users to embed imprecise specifications in their requests (e.g., neighborhood queries, as in Section 3.5 above) [17, 24]. Error-tolerant interfaces use relaxed formalisms in their interpretation of requests [26].

As mentioned above, even after a request for information had 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 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 imperfect, yet both the system and the user are unaware of this imperfection. Often, the imperfection of apparently-perfect answers is revealed only when a conflicting answer to the same request is received from another information system.

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 we identify three similar main sources of imperfection: (1) insufficient knowledge of the system, (2) insufficient knowledge of the specific database that is being modified, and (3) uncertainty or imprecision with respect to the information embedded in a modification.

Many of the approaches aimed at alleviating the problems of transformation imperfections (Section 4.1 above) are also applicable to modification imperfections. However, fewer tools have been developed to address modification imperfections. 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 imperfection, uncertain or imprecise information, is unrelated to expertise. One example is a request to add imprecise information; for example, “the new manager is either Paul or John”. Another example is an imprecisely-specified request to delete information; for example, “some of the telephone numbers are no longer valid”.

However, this kind of imperfection is not any different from the description imperfections, that were the subject of Section 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 imperfection, then it would not be able to handle these modifications.)

4.3 Processing

Even when a description D and a transformation t are free of imperfections, the result $t(D)$ may be imperfect 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 [18]. 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). A query processor that provides monotonically improving partial answers under constraints of time or unavailability is described in [42]. In other applications, query processing might involve randomizations, sampling or other estimation techniques [21]. For example, a statistical database system might introduce perturbations into its answers deliberately, for reasons of security. In each case, the answers would exhibit imperfections.

Finally, it is sometime considered advantageous to sacrifice accuracy for the sake of *simplicity*. Recent research on *intensional answers* [28] 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).

5 Conclusion

Much of the work described in this paper is either theoretical or its implementation has been limited to research prototypes. Commercial database systems have been slow to incorporate capabilities for representing imperfect data or for specifying imperfect transactions. Imperfect data must be stored as null values or it must be excluded from the database, and users with vague requests can use queries with simple patterns or they must browse the database to find their answers. To accommodate new applications that are now emerging, future database systems would have to include stronger capabilities for dealing with imperfections. We describe here briefly three such applications.

In recent years, the integration of information from multiple and heterogeneous database systems has been recognized as an important area of database system research and development. Multi-database environments present a strong case for having imprecision and uncertainty management capabilities. While individual database systems are usually careful to avoid internal inconsistencies, when information from independent systems with overlapping information is integrated, inconsistencies will often surface. Thus, even when individual answers are free from any imperfections, they might be introduced by their integration in a global answer. Therefore, database systems in a multi-database environment should be capable of combining conflicting answers into a single (imperfect) answer, and then store and manipulate such information [27].

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) [13], 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.

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 perfectness. The management of this information cannot be done by traditional database techniques and requires the use of uncertainty techniques.

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

References

- [1] S. Abiteboul and G. Grahne. Update semantics for incomplete databases. In *Proceedings of the Eleventh International Conference on Very Large Data Bases*, pages 1–12, 1985.
- [2] R. Agrawal, N. H. Gehani, and J. Srinivasan. OdeView: The graphical interface to Ode. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 34–43, 1990.
- [3] D. Barbara, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, 4(5):487–502, October 1992.
- [4] Lea Sombe (P. Besnard, M.-O. Cordier, D. Dubois, L. Farinas del Cerro, C. Froidevaux, Y. Moinard, H. Prade, C. Schwind, and P. Siegel). Special issue on reasoning under incomplete information in artificial intelligence. *International Journal of Intelligent Systems*, 5(4), September 1991.
- [5] P. Bosc and H. Prade. An introduction to fuzzy set and possibility theory-based approaches to the treatment of uncertainty and imprecision in database management systems. In A. Motro and P. Smets, editors, *Proceedings of the Workshop on Uncertainty Management in Information Systems: From Needs to Solutions*, pages 44–70, 1993.
- [6] B. P. Buckles and F. E. Petry. A fuzzy representation of data for relational databases. *Fuzzy Sets and Systems*, 7(3):213–226, May 1982.
- [7] R. Cavallo and M. Pittrelli. The theory of probabilistic databases. In *Proceedings of the Thirteenth International Conference on Very Large Data Bases*, pages 71–81, 1987.
- [8] E. F. Codd. Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems*, 4(4):397–434, December 1979.
- [9] C. J. Date. *Relational Database: Selected Writings*. Addison Wesley, Reading, Massachusetts, 1986.
- [10] C. J. Date. NOT is not “not”! In *Relational Database Writings 1985–1989*. Addison Wesley, Reading, Massachusetts, 1990.

- [11] A. D’Atri, A. Motro, and L. Tarantino. ViewFinder: an object browser. Technical report, Department of Information and Software Systems Engineering, George Mason University, June 1992.
- [12] R. Demolombe and L. Farinas del Cerro. An algebraic evaluation method for deduction in incomplete data bases. *Journal of Logic Programming*, 5:183–205, 1988.
- [13] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, forthcoming.
- [14] G. Gottlob and R. Zicari. Closed world assumption opened through null values. In *Proceedings of the Fourteenth International Conference on Very Large Data Bases*, pages 50–61, 1988.
- [15] P. Harmon and D. King. *Expert Systems — Artificial Intelligence in Business*. John Wiley & Sons, New York, New York, 1985.
- [16] A. R. Hurson, M. W. Bright, and S. H. Pakzad, editors. *Multi-database Systems: An Advanced Solution for Global Information Sharing*. IEEE Computer Society Press, Los Alamitos, California, 1994.
- [17] T. Ichikawa and M. Hirakawa. ARES: A relational database with the capability of performing flexible interpretation of queries. *IEEE Transactions on Software Engineering*, SE-12(5):624–634, May 1986.
- [18] T. Imielinski. Intelligent query answering in rule based systems. *Journal of Logic Programming*, 4(3):229–257, September 1987.
- [19] T. Imielinski. Incomplete information in logical databases. *Data Engineering*, 12(2):29–40, June 1989.
- [20] T. Imielinski and K. Vadaparty. Complexity of querying databases with or-objects. In *Proceedings of PODS-89, the 8th Symposium on Principles of Database Systems*, 1989.
- [21] S. Kwan, F. Olken, and D. Rotem. Uncertain, incomplete, and inconsistent data in scientific and statistical databases. In A. Motro and P. Smets, editors, *Proceedings of the Workshop on Uncertainty Management in Information Systems: From Needs to Solutions*, pages 64–91, 1992.
- [22] D. Maier. *The Theory of Relational Databases*. Computer Science Press, Rockville, Maryland, 1983.

- [23] A. Motro. BAROQUE: A browser for relational databases. *ACM Transactions on Office Information Systems*, 4(2):164–181, April 1986.
- [24] A. Motro. VAGUE: A user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems*, 6(3):187–214, July 1988.
- [25] A. Motro. Integrity = validity + completeness. *ACM Transactions on Database Systems*, 14(4):480–502, December 1989.
- [26] A. Motro. FLEX: A tolerant and cooperative user interface to databases. *IEEE Transactions on Knowledge and Data Engineering*, 2(2):231–246, June 1990.
- [27] A. Motro. 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, October 1993.
- [28] A. Motro. Intensional answers to database queries. *IEEE Transactions on Knowledge and Data Engineering*, 6(3):444–454, June 1994.
- [29] A. Motro and P. Smets, editors. *Proceedings of Workshop on Uncertainty Management in Information Systems: From Needs to Solutions*, 1992.
- [30] A. Ola and G. Ozsoyoglu. Incomplete relational database models based on intervals. *IEEE Transactions on Knowledge and Data Engineering*, 5(2):293–308, April 1993.
- [31] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.
- [32] G. Piatetsky-Shapiro. Discovery, analysis and presentation of strong rules. In G. Piatetsky-Shapiro and W. Frawley, editors, *Knowledge Discovery in Databases*, pages 227–248. AAAI Press/MIT Press, Menlo Park, California, 1991.
- [33] H. Prade and C. Testemale. Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries. *Information Sciences*, 34(2):115–143, November 1984.
- [34] K. V. S. V. N. Raju and A. Majumdar. Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database

- systems. *ACM Transactions on Database Systems*, 13(2):129–166, June 1988.
- [35] R. Reiter. On closed world data bases. In *Logic and Databases*, pages 55–76. Plenum Press, New York, New York, 1978.
- [36] T. R. Rogers and R. G. G. Cattell. Object-oriented database user interfaces. Technical report, Information Management Group, Sun Microsystems, 1987.
- [37] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, New York, 1983.
- [38] P. Smets. Imperfect information: Imprecision – uncertainty. In A. Motro and P. Smets, editors, *Proceedings of the Workshop on Uncertainty Management in Information Systems: From Needs to Solutions*, pages 1–28, 1993.
- [39] P. Smets and A. Motro, editors. *Proceedings of Workshop on Uncertainty Management in Information Systems: From Needs to Solutions*, 1993.
- [40] H. R. Turtle and W. B. Croft. Uncertainty in information retrieval systems. In A. Motro and P. Smets, editors, *Proceedings of the Workshop on Uncertainty Management in Information Systems: From Needs to Solutions*, pages 111–137, 1992.
- [41] C. J. van Rijsbergen. *Information Retrieval (Second Edition)*. Butterworths, London, 1979.
- [42] S. V. Vrbsky and J. W. S. Liu. APPROXIMATE: A query processor that produces monotonically improving approximate answers. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):1056–1068, December 1993.
- [43] M. D. Williams. What makes RABBIT run? *International Journal of Man-Machine Studies*, 21(4):333–352, October 1984.
- [44] M. Zemankova and A. Kandel. Implementing imprecision in information systems. *Information Sciences*, 37(1,2,3):107–141, December 1985.
- [45] R. Zicari. Databases and incomplete information. In A. Motro and P. Smets, editors, *Proceedings of the Workshop on Uncertainty Management in Information Systems: From Needs to Solutions*, pages 52–63, 1992.