A Probabilistic Relational Model and Algebra

DEBABRATA DEY and SUMIT SARKAR
Louisiana State University

Although the relational model for databases provides a great range of advantages over other data models, it lacks a comprehensive way to handle incomplete and uncertain data. Uncertainty in data values, however, is pervasive in all real-world environments and has received much attention in the literature. Several methods have been proposed for incorporating uncertain data into relational databases. However, the current approaches have many shortcomings and have not established an acceptable extension of the relational model. In this paper, we propose a consistent extension of the relational model. We present a revised relational structure and extend the relational algebra. The extended algebra is shown to be closed, a consistent extension of the conventional relational algebra, and reducible to the latter.

Categories and Subject Descriptors: F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages—*algebraic language theory*; G.3 [**Probability and Statistics**]: Statistical Computing; H.2.1 [**Database Management**]: Logical Design—*data models*; H.2.3 [**Database Management**]: Languages—*data manipulation languages (DML), query languages*; H.2.8 [**Database Management**]: Database Applications; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*uncertainty, "fuzzy," and probabilistic reasoning*

General Terms: Languages, Theory

Additional Key Words and Phrases: Data uncertainty, data incompleteness, probability calculus, probabilistic relation, relational model, relational algebra

1. INTRODUCTION

Over the last two decades, relational databases have gained widespread popularity and acceptance in business information systems. Early database systems, based on the hierarchical or the network models, are rapidly being replaced by newer products based on the relational model. The primary reason for this shift is the fact that relational databases provide a great

This research was partially supported by the College of Business Administration, Louisiana State University.

Authors' address: Department of Information Systems and Decision Sciences, College of Business Administration, Louisiana State University, Baton Rouge, LA 70803; Dey email: qmdey@unix1.sncc.lsu.edu; Sarkar email: qmsark@lsuvm.sncc.lsu.edu

range of advantages, such as access flexibility, logical and physical data independence, data integrity, reduced (and controlled) data redundancy, and enhanced programmer productivity. Unfortunately, the relational model does not have a comprehensive way to handle incomplete and uncertain data. Such data, however, exist everywhere in the real world. Having no means to model these data, the relational model ignores all uncertain data and focuses primarily on values that are known for sure; uncertain data items are represented using "null" values, which are special symbols often employed to represent the fact that the value is either unknown or undefined [Maier 1983; Date 1986]. Consequently, relational databases do not yield satisfactory results in many real-world situations. Consider, for example, the case where a database is used to monitor locations of battleships during a war [Wong 1982]. It is clear that, for every decision (regarding deployment, battle, or other operations), it is essential to consider the approximate locations of the ships. Since the relational model cannot represent the inherent uncertain nature of the data, it cannot be used directly.

Similarly, a financial institution may be interested in storing some non-deterministic attributes of different companies for the purpose of making the right investment decisions [Barbará et al. 1992]. Marketing and production decisions are mostly based on expected customer behavior patterns, which are seldom deterministic. Similar examples are plentiful in the literature [Wong 1982; Buckles and Petry 1983, 1984; Prade and Testemale 1984; Zemankova and Kandel 1985; Barbará et al. 1992]; they all point towards the need for an extension of the relational model so that uncertain data can be supported. To that end, we propose an extension of the relational model—the probabilistic relational model (PRM). The main contribution of this work is the following: (i) a probabilistic relational model with relations abiding by first normal form (1NF), and (ii) an associated algebra that is closed, a consistent extension of the traditional algebra, and reducible to the latter. We propose appropriate integrity constraints and discuss null values within the context of a probabilistic model.

The rest of this paper is organized as follows. Previous research dealing with data uncertainty is examined in Section 2. Section 3 discusses the proposed approach. The relational algebra for this approach is discussed in Section 4. Section 5 discusses some properties of the relational algebra. Incompleteness in the joint probability distribution of objects is treated in Section 6 as an extension of the proposed algebra. Section 7 concludes the paper and offers future research directions.

2. PREVIOUS RESEARCH

Previous research in the area of data incompleteness and uncertainty falls into four different categories. The first category deals with extending the relational algebra to handle "null" values and provides the semantics of "null" values [Codd 1979; Maier 1983; Date 1986]. These representations

using "null" values assume that data values are either known with certainty or they are unknown. This assumption, however, is too restrictive to model most real applications, and hence, this approach is not useful for our purpose. Lipski [1979] discusses the semantic issues of representing data incompleteness using "null" values. He introduces the concepts of *internal* and *external interpretations* of a query under the *closed world* and *open world assumptions*, respectively. Although his scheme is more expressive than the usual interpretation of "null" values, it still cannot express the stochastic nature of data.

The second category deals with the uncertainty involved in the retrieval of incomplete data and the costs of data incompleteness. Wong [1982], Mendelson and Saharia [1986], and Dey et al. [1995] assume that, while it is possible to do so, data items are not described completely because of the high associated cost. These investigations provide a framework for performing design time cost-benefit trade-off analyses to decide how much data to store. They all model the information retrieval and decision-making under uncertainty that results from data incompleteness, but ignore the inherent uncertainty associated with the data items themselves. However, many attributes of the real world—such as stock prices, weather reports, customer satisfaction, and future demand—are inherently stochastic. As a result, contrary to their assumption, the uncertainty associated with these data items cannot be completely avoided. In summary, these models are useful tools for deciding on desired levels of data storage, but are not adequate for the representation of data uncertainty.

The third category models data uncertainty using fuzzy set theory [Buckles and Petry 1983, 1984; Prade and Testemale 1984; Zemankova and Kandel 1985]. The typical assumption in these models is that some attributes (or data items) do not have precise values; rather, they take on "fuzzy" values. For example, the height of a person can have the values "tall" or "short" instead of values such as 5 feet 3 inches. This approach is an improvement over the conventional relational model, but is not well-suited to represent business data. In order to understand why this is so, we need to discuss the nature of uncertainty associated with data items. It is well-documented that there are two types of uncertainties in the real world: uncertainty due to *vagueness* and uncertainty due to *ambiguity* [Klir and Folger 1988]. Uncertainty due to vagueness is associated with the difficulty of making sharp or precise distinctions in the real world. For example, subjective terms such as tall, far, and heavy are vague. These cases can be modeled reasonably well with the help of tools such as fuzzy set theory. Uncertainty due to ambiguity, on the other hand, is associated with situations in which the choices among several precise alternatives are left unspecified. For example, we may know that a person's height is in the range of five to seven feet, but we may not know the exact height of that person. These situations need to be modeled using some kind of uncertainty measure.

In most business situations, the uncertainty about data arises from ambiguity and not from vagueness. If some data item is of interest in an

application, it is reasonable to assume that there is a way to measure and collect that data up to the level of precision that is necessary for that application. The uncertainty lies not with the reported value but with the correctness of the measurement or data collection techniques. For example, if a weighing scale reports a person's weight as 150 pounds, then there is no uncertainty with the reported value itself. It is, however, possible that the scale behaves correctly only 70% of the time, in which case the uncertainty arises from the erratic behavior of the scale. One would say that there is a probability of 0.7 that the weight of the above person is 150 pounds.[1] Such a case needs to be modeled using some uncertainty measure such as the probability measure [Klir and Folger 1988].

The fourth category of work uses that approach. For example, the work of Cavallo and Pittarelli [1987] extends the relational model to represent uncertainty due to ambiguity using the well-known probability calculus. They assign a probability measure with every tuple in a relation; it indicates the joint probability of all the attribute values in that tuple. They impose a restriction that the total probability assigned to all the tuples in a relation is exactly one. However, note that a relation contains information about the key, as well as the non-key, attributes, the key being the unique identifier of an object. If we know that the object exists in the real world, the probability of its existence (and hence the marginal probability of its key value) should be unity. An important limitation of their relational structure is that a separate relation would be required for every object that is known to exist with certainty. For example, the information about several hundred employees in an organization will have to be broken across several hundred relations. On the other hand, if all the information is to appear in a single relation, there is no way of asserting the certainty in the existence of objects and their key values in the above model (since all probabilities in a relation must add up to one). Moreover, the main focus of their work is on information content, and probabilistic functional and multi-valued dependency. As a result, they discuss only projection and join operations for their relational structure and do not define other useful operations and the relational algebra.

Raju and Majumdar [1988] generalize the basic relational concepts by using fuzzy relations. In their model, each tuple is assigned a possibility measure that represents the possibility of its membership in the relation. Moreover, their attributes may take fuzzy subsets as their values. Their model, however, suffers from the same problem with deterministic keys. If the existence of an object is known with certainty, and if its attribute values are uncertain, then their model cannot represent that fact. Second, since they use set-valued attributes, their model poses the usual implementation problem associated with all non-1NF relations. An associated concern is that of simplicity of user views. The non-1NF view of relations does

[1]To see the difference with vagueness, note that if the scale did not exist in the first place, we would have to resort to reporting the person's weight as "heavy" or "light" based on our subjective judgment of the person's appearance.

not have the simple flat file view of relations as in the original relational model. Finally, it should be mentioned that the use of possibility as a measure of uncertainty is often not appropriate. Instead, uncertainty in data can be modeled using probability theory which, due to its wide acceptance, is easier to interpret. Moreover, probability measures have a rich theoretical basis for representing uncertainty; they lend themselves to empirical testability and provide an easy-to-use semantics [Pearl 1986, 1989].

Barbará et al. [1992] propose an extension of the relational model using probability theory. They also adopt a non-1NF view of probabilistic relations, and redefine the project, select, and join operations using semantics of probability theory. Although their approach overcomes some of the above shortcomings, it too has its own limitations. First, their model also has non-1NF relations and poses the usual implementation problem. Second, their definition of the join operation does not allow one to join on attributes that are parts of the key attributes of both the participating relations. This is a serious problem because sometimes navigational links cannot be established between two relations.[2] Third, their method of data organization presumes knowledge of probabilistic independence among attributes, which may not be known at design time. Consequently, the structure of a relation cannot be determined before the data is realized. Fourth, their assumption that key attributes are always deterministic is somewhat restrictive. For example, a relation may be used to represent a relationship that an employee works in a department. In that case, the key ({EMP#, DEP#}) itself may be uncertain. This cannot be represented if only deterministic keys are supported. Finally, they do not define their algebra in terms of valid objects and permissible operations on those objects.

Based on the model of Barbará et al. [1992], Tseng et al. [1993] develop a set of extended relational operations to represent uncertainty arising from data heterogeneity. They use probabilities at the tuple, as well as at the attribute, level. The regular relational operations are redefined, and a new operation called *integration* is introduced. In addition to the implementation problems of a non-1NF model, this proposal has several other shortcomings. First, the use of probability measures at two different levels introduces needless complexity in the representation of uncertainty; the semantics of their tuple probability is not discussed clearly. Second, their relational structure is not well-defined, and the algebra is not formally stated in terms of valid objects and operations. Third, their definitions of relational operations treat primary key attributes separately from the other attributes. Note that conventional definitions of relational operations treat all attributes uniformly; this feature is desirable for the ease of

[2]For example, consider the following three relations: an EMPLOYEE relation with EMP# as the primary key, a PROJECT relation with PROJ# as the primary key, and an ASSIGNMENT relation with {EMP#, PROJ#} as the primary key. The join operation that they define cannot be performed among these three relations. In other words, we will not be able to answer questions such as: "Which employees are assigned to work on the MICROCHIP project?"

Table I. EMPLOYEE: A Probabilistic Relation

EMP#	ssn	lName	fName	rank	salary	dept	pS
3025	086-63-0763	Lyons	James	clerk	15K	toy	0.2
3025	086-63-0763	Lyons	James	cashier	20K	shoe	0.6
3025	086-63-0763	Lyons	James	cashier	15K	auto	0.2
6723	089-83-0789	Kivari	Jack	clerk	18K	toy	0.4
6723	089-83-0789	Kivari	Jack	cashier	20K	auto	0.4
6879	098-84-1234	Peters	Julia	clerk	25K	toy	0.3
6879	098-84-1234	Peters	Julia	clerk	27K	toy	0.1
6879	098-84-1234	Peters	Julia	cashier	25K	shoe	0.6

implementation. Fourth, some of their extended operations—namely projection, intersection, difference, union, Cartesian product, and join—are defined only for relations with deterministic tuples (tuples with no associated probability). Consequently, relations with only deterministic key values can participate in these operations, although a relation with uncertain key values may result from such an operation. A related problem associated with this non-uniform treatment of participating and resulting relations is that an algebra based on their relational operations can never be closed. Finally, since they use the same definition as Barbará et al. [1992] for the join operation involving primary keys, the problems discussed earlier still persist.

It is evident from the above discussion that models dealing with data uncertainty suffer from several problems. In this research, we address these problems in a systematic manner, and provide a consistent extension of the relational model for representing uncertainty in business data.

3. PROBABILISTIC RELATIONS AND BASIC OPERATIONS

3.1 Structure and Meaning of Relations

We present a scheme to store probabilities associated with values of attributes of real-world objects in terms of a discrete probability distribution. Unlike Barbará et al. [1992], we do not make any restrictive assumption about the key values for a relation being deterministic. Deterministic keys can be modeled as a special case in our model. Informally speaking, we stamp every row (or tuple) of a relation with associated probability measure. For example, consider the probabilistic relation shown in Table I. In this table, the primary key is EMP#, and the last column pS denotes the probability associated with each row of the relation. The pS column for the first row has the value 0.2; it means that there is a probability of 0.2 that there exists an employee with the following associated values: 3025 for EMP#, 086-63-0763 for ssn, Lyons for lName, James for fName, clerk for rank, 15K for salary, and toy for dept. All other rows are interpreted in a similar fashion. The probability stamp of a tuple is, therefore, the joint probability of the given realizations of all the attributes (in that tuple) taken together. Probabilities of individual attributes can be derived by

appropriately marginalizing the distribution. For example, the first three rows indicate that it is known with certainty that (i) there exists an employee with EMP# 3025, and (ii) the ssn, lName and fName for this employee are 086-63-0763, Lyons and James, respectively. Similarly, the probability of an employee having EMP# = 3025 and rank = "cashier" is 0.8 (from the second and the third rows).

We do not allow two tuples with the same values for all non-probability attributes—defined as *value-equivalent* tuples in Section 4—to be present in a relation. Value-equivalent tuples are similar to duplicates in the conventional relational model. Analogous to elimination of duplicates, value-equivalent tuples must be *coalesced*.[3] The *coalescence* operations are formally defined in Section 4.

Ideally, probability stamps associated with a key value should add up to one. In that case, the existence of the object (identified by that key value) is certain and the joint probability distribution for all its attributes is completely specified. However, the complete distribution is not necessary in order to store probabilities about attributes. If the existence of the object itself is uncertain, then the probability stamps associated with the key value of that object should be less than one. The modified requirement is, then, that the probability stamps associated with any given key value must add up to no more than one. For instance, in the example shown in Table I, the probability masses associated with EMP# 6723 adds up to 0.8; this means that the existence of that employee is not certain and has a probability of 0.8.

There are several reasons why our representation is an improvement over the existing proposals in the literature:

—In existing proposals, the existence of an object is either certain [Barbará et al. 1992], or uncertain [Cavallo and Pittarelli 1987; Raju and Majumdar 1988]. It is not possible, using these models, to represent certainty of some objects and uncertainty about others. In our representation, certainty about an object is a special case where the probabilities associated with the key value for that object add up to exactly one.

—In our representation, all relations are in first normal form (1NF). Relations that are in 1NF do not pose the implementation problem associated with the non-1NF relations of Raju and Majumdar [1988], Barbará et al. [1992], and Tseng et al. [1993].

—Unlike Barbará et al. [1992] and Tseng et al. [1993], our definition of the *join* operation (to be discussed in Section 4) is not overly restrictive.

—Our algebra, like the traditional relational algebra, is unisorted because the only valid object in our algebra is a *relation*. We provide a very general definition of relations and relational operations.

[3]The notion of *value-equivalent* tuples and *coalescing* them was originally introduced by Snodgrass [1987] in his work related to temporal databases.

Table II. A Probabilistic Relation after Projection

EMP#	rank	pS
3025	clerk	0.2
3025	cashier	0.8
6723	clerk	0.4
6723	cashier	0.4
6879	clerk	0.4
6879	cashier	0.6

There are several other advantages with the representation that we have presented. First, this view is similar to the view of flat tables (1NF) of the original relational model. Second, using this approach, it is possible to specify the joint distribution among various data items, conditioned on the existence of the object (i.e., its primary key value).[4] Finally, in our model, prior knowledge of the dependence among attributes is not needed at design time. If such a dependence is discovered after realization of the data in the database, a relation can be decomposed based on that. As a related point, the apparent data redundancy in the relation shown in Table I can be eliminated easily by decomposing it into smaller relations. For example, if rank and salary are the only mutually dependent attributes, and if ssn, lName, and fName are deterministic, then it is possible to decompose it into three relations on the following schemes:

EMPLOYEE: [EMP#, ssn, lName, fName],
EMP_SAL: [EMP#, rank, salary, pS],
EMP_DEPT: [EMP#, dept, pS].

3.2 The Projection, Selection, and Join Operations

Here we briefly describe the three major relational operations, namely, projection, selection, and join. More formal definitions of each of these operations and several others are provided in Section 4.

Projection. The projection operation provides us with the marginal distribution of a subset of attributes. For example, in Table I, several attributes of employees are presented. The user, however, may want to view only the rank for all employees. This is accomplished by projecting this relation onto the attributes EMP#, rank and pS, as shown in Table II. The probability stamps for the resulting tuples are obtained by evaluating the appropriate marginal distribution from the joint distribution stored in the original relation. Thus, the three tuples associated with EMP# 3025 in Table I result in the two tuples (corresponding to the ranks "clerk" and "cashier") shown in Table II. The probability that an employee has EMP# = 3025 and rank = "cashier" is 0.8, which is consistent with the information

[4]This can be done by dividing the probability stamp of each tuple by the total probability mass associated with the key value of that tuple. This means that the conditional distribution of Barbará et al. [1992] is derivable from our representation as well.

Table III. A Probabilistic Relation after Selection

EMP#	ssn	lName	fName	rank	salary	dept	pS
3025	086-63-0763	Lyons	James	clerk	15K	toy	0.2
6723	089-83-0789	Kivari	Jack	clerk	18K	toy	0.4
6879	098-84-1234	Peters	Julia	clerk	25K	toy	0.3
6879	098-84-1234	Peters	Julia	clerk	27K	toy	0.1

stored in the original relation. Note that when using the projection operation, a candidate key should be included; otherwise, the pS-attribute may not provide reliable probability measures.

Selection. The selection operation is used to identify tuples that satisfy specified conditions on attributes and probability stamps. For instance, we may want to view all tuples with rank = "clerk." The result of this query on Table I is shown in Table III.

We can include explicit conditions on the probability stamp itself, and logical connectives may be used to combine multiple conditions. Such a query, for example, could select tuples with rank = "clerk" and $pS \geq 0.3$. The result of this query on Table I is shown in Table IV. Note that since the selection operation is defined at the tuple level, just the selection operation cannot provide a list of employees who are clerks with a probability of 0.3 or higher. In order to obtain such a list, it is necessary to combine the projection and selection operations. This will be illustrated with the help of some queries in Section 4.

Join. The join operation between two relations provides the joint distribution of all the attributes in the participating relations. In the join operation, every tuple in one relation is checked for a match (on attributes common to both the relations) with every tuple in the other relation; if a match is found, they are combined to form a new tuple in the resulting relation. The probability stamp of the new tuple is simply the product of the probability stamps of the participating tuples. An important implication of this is that the probability stamp for the resulting relation is a reliable probability measure only when the attributes in the two participating relations are independent.

The join operation is illustrated using the relations in Table V. The relations Employee and Department are joined to obtain the new relation Emp_Dept. Closer examination of the Department relation reveals that the total probability mass associated with each department is exactly one; this means that the existence of each department is certain. In other words, we can also interpret this relation as the conditional distribution of the attribute "mgr" given the attribute "dept." This is the reason why the join operation is meaningful in this case. Symbolically,

Table IV. A Probabilistic Relation after Selection with Composite Selection Condition

EMP#	ssn	lName	fName	rank	salary	dept	pS
6723	089-83-0789	Kivari	Jack	clerk	18K	toy	0.4
6879	098-84-1234	Peters	Julia	clerk	25K	toy	0.3

Table V. Illustration of Join Operation on Probabilistic Relations

Relation: Employee

EMP#	dept	*pS*
3025	shoe	0.6
3025	toy	0.4
6637	toy	0.3
6637	auto	0.5

Relation: Department

DEPT	mgr	*pS*
shoe	Joe	0.8
shoe	Bill	0.2
toy	Bob	0.5
toy	Bill	0.5

Relation: Emp_Dept
Employee ⋈ Department

EMP#	dept	mgr	*pS*
3025	shoe	Joe	0.48
3025	shoe	Bill	0.12
3025	toy	Bob	0.20
3025	toy	Bill	0.20
6637	toy	Bob	0.15
6637	toy	Bill	0.15

$$\Pr[\text{emp\#} = e, \text{dept} = d, \text{mgr} = m]$$

$$= \Pr[\text{emp\#} = e, \text{dept} = d] \times \Pr[\text{mgr} = m | \text{dept} = d]$$

$$= \frac{\Pr[\text{emp\#} = e, \text{dept} = d] \times \Pr[\text{dept} = d, \text{mgr} = m]}{\Pr[\text{dept} = d]},$$

assuming "mgr" depends only on "dept." However, since $\Pr[\text{dept} = d] = 1$ for all d, the join would yield meaningful probability measures. If the existence of a department were not certain, we could not use the join operation directly to obtain the joint distribution of the combined attributes. In that case, first we would have to find the conditional probability of manager given the department name. Let us consider the relation Department′ shown in Table VI. In this case, since there is uncertainty about the existence of the departments themselves, we should not directly

Table VI. Relation: Department'

DEPT	mgr	pS
shoe	Joe	0.4
shoe	Bill	0.1
toy	Bob	0.4
toy	Bill	0.4

join this with the Employee relation. In order to make the join yield a meaningful joint distribution, we need to first transform Department' (to relation Department, as in Table V) such that the distribution is conditionalized on "dept"; it can then be joined with the Employee relation. In Section 4, we introduce a new operation called *conditionalization* that allows us to make such transformations.

3.3 Modifications of Existing Data

In a probabilistic database, arrival of new information should lead to revision of beliefs stored about the objects in the database. In the most general case, this means that the database management system should, based on the new information, recalculate the new probability distribution of the relevant objects. However, that is a difficult task to accomplish.

In order to illustrate this, consider the relation "EMPLOYEE" shown in Table I. Suppose that we are informed, at this point, that there exists an employee with EMP# = 6723 and rank = "clerk." Clearly, the last tuple for EMP# = 6723 should disappear. We could use Jeffrey's probability kinematics [Jeffrey 1983] for revising our belief about the tuple ⟨6723, clerk, 18K⟩. Assuming the conditional distribution of all other attributes (i.e., ssn, lName, fName, salary, and dept), given employee number and rank, has not changed, we know the probability associated with that tuple is 1. In a similar fashion, it is also possible to use Jeffrey's kinematics to handle cases where the incoming information itself is probabilistic. In either case, the belief revision involves combining the incoming information with the stored data in a manner consistent with the axioms of probability theory. However, since the incoming information may be specified in many different ways, it is not possible to establish a single operation that can accomplish the task of belief revision. For this reason, we feel it is not appropriate to make belief revision a part of the algebra. The task of belief revision may be relegated to an auxiliary support system that interfaces with the database management system and helps the user specify new distributions about objects in an interactive fashion. We are currently investigating the issues associated with building such a system.

For the purpose of this paper, we assume that the user can, based on the stored data and the incoming information, specify the new distribution of an object. Once the new distribution is known, the updating is a simple task of deleting the old tuples and inserting the new ones. Below we redefine the union and the difference operations that can be used for this purpose.

Table VII. Two Probabilistic Relations on the Same Scheme

Relation: EMP		
EMP#	dept	pS
3025	shoe	0.6
3025	toy	0.3
6637	toy	0.8
6637	auto	0.1
Relation: EMP'		
EMP#	dept	pS
3025	shoe	0.7
3025	toy	0.1
6637	toy	0.3
6637	shoe	0.5

Union. The union operation is useful in inserting new data into a relation. When applied to two relations with the same set of attributes, it builds a new relation consisting of all tuples appearing in either or both relations. The only restriction is that if there are two value-equivalent tuples, the one with the higher probability stamp is included in the new relation; the other one is discarded. Consider the relations EMP and EMP' shown in Table VII. The union operation between them will generate the relation shown in Table VIII(a).

The union operation may lead to a semantically inconsistent probability distribution for an object. In Table VIII(a), the total probability associated with EMP# 6637 is 1.4. This is a data integrity issue that is subsequently discussed in Section 4.2. Here we note that similar inconsistencies occur in the conventional relational model as well. Consider, for example, two deterministic relations with the same structure and assume that there are two tuples—one in each relation—with the same primary key value. If the other attributes in these tuples have different values, the conventional union operation on these two relations would yield a relation with tuples where the primary key value is not unique. Thus, the union operation should not be considered as a way to reconcile conflicts between the data contained in two relations. When there are conflicts, we assume that the user can—possibly with the help of an auxiliary support system—resolve them. Once that is done, the user can use the union operation to insert the revised distribution into the relation.

Difference. The difference operation is useful in deleting old data from a relation. For example, the user can replace the old distribution by a revised one (based on some new information) with the help of the difference and union operations. The difference operation between two relations with the same set of attributes builds a new relation consisting of all tuples appearing in the first relation that are not value equivalent to any tuple in the second relation. However, if a tuple in the first relation is value-equivalent to a tuple in the second relation and has a higher probability

Table VIII. Illustration of Union and Difference Operations

(a) EMP ∪ EMP′

EMP#	dept	pS
3025	shoe	0.7
3025	toy	0.3
6637	toy	0.8
6637	auto	0.1
6637	shoe	0.5

(b) EMP − EMP′

EMP#	dept	pS
3025	toy	0.2
6637	toy	0.5
6637	auto	0.1

stamp, then a new tuple is included in the resulting relation; this new tuple is also value-equivalent to the other two and has a probability stamp equal to the difference in the probability stamps of the participating tuples. The difference operation between the relations EMP and EMP′ (shown in Table VII) will generate the relation shown in Table VIII(b).

4. RELATIONAL ALGEBRA

4.1 Basic Definitions

Let $\mathcal{N} = \{1, 2, \ldots, n\}$ be an arbitrary set of integers. A *relation scheme* R is a set of *attribute names* $\{A_1, A_2, \ldots, A_n\}$, one of which may be a probability stamp pS. Corresponding to each attribute name A_i, $i \in \mathcal{N}$, is a set D_i called the *domain* of A_i. If $A_i = pS$, then $D_i = (0, 1]$. The *multiset* $\boldsymbol{D} = \{D_1, D_2, \ldots, D_n\}$ is called the domain of R. A *tuple* x over R is a function from R to $\boldsymbol{D}$ ($x : R \rightarrow \boldsymbol{D}$), such that $x(A_i) \in D_i$, $i \in \mathcal{N}$. In other words, a tuple x over R can be viewed as a set of attribute name-value pairs: $x = \{\langle A_i, v_i \rangle | \forall i \in \mathcal{N} (A_i \in R \wedge v_i \in D_i)\}$. We write $x(R)$ to denote that x is a tuple on scheme R. *Restriction of a tuple* x over S, $S \subset R$, written $x(S)$, is the sub-tuple containing values for attribute names in S only, i.e., $x(S) = \{\langle A, v \rangle \in x | A \in S\}$.

We now give a formal interpretation of a tuple. A tuple x over R represents our belief about attributes (in R) of a real world object. If $pS \in R$, then we assign a probability of $x(pS) > 0$ to the fact that an object has the values $x(R - \{pS\})$ for the corresponding attributes. In other words, the attribute pS represents the joint distribution of all the attributes taken together. Symbolically,

$$x(pS) = \Pr[R - \{pS\} = x(R - \{pS\})].$$

If $pS \notin R$; i.e., if the relation scheme R is deterministic, then every tuple on R is assigned a probability of one, and is not explicitly written. However,

if x is a tuple on the scheme R, and $pS \notin R$, it will be implicitly assumed that $x(pS) = 1$. This assumption—called the *deterministic assumption* hereafter—allows us to provide generalized definitions of different components of the model, for probabilistic as well as deterministic relation schemes. When present in a scheme, renaming pS (using the *rename* operation to be defined later in this section) would make it lose its special meaning; renaming pS is allowed in this algebra, but not recommended.

Two tuples x and y on relation scheme R are *value-equivalent* (written $x \simeq y$) if and only if, for all $A \in R$, $(A \neq pS) \Rightarrow (y(A) = x(A))$. Value-equivalent tuples are not allowed in a relation; they must be *coalesced*. We define two types of *coalescence* operations on value-equivalent tuples:

(1) The *coalescence-PLUS* operation is used in the definition of the projection operation. Coalescence-PLUS (denoted by $\oplus$) on two value-equivalent tuples x and y is defined as:

$$z = x \oplus y \Leftrightarrow (x \simeq y) \wedge (z \simeq x) \wedge (z(pS) = \min\{1, x(pS) + y(pS)\}).$$

(2) The *coalescence-MAX* operation is used in the definition of the union operation. Coalescence-MAX (denoted by $\odot$) on two value-equivalent tuples x and y is defined as:

$$z = x \odot y \Leftrightarrow (x \simeq y) \wedge (z \simeq x) \wedge (z(pS) = \max\{x(pS), y(pS)\}).$$

The idea of value-equivalent tuples and coalescence operations need not be confined to just two tuples. Given m tuples $x_1, x_2, \ldots, x_m$, all of which are on the same relation scheme, they are said to be value-equivalent if $x_i \simeq x_j$ for all i, j; $1 \leq i, j \leq m$. Coalescence-PLUS, for example, on all these value-equivalent tuples will recursively coalesce all the tuples pair-wise, i.e.,

$$\bigoplus_{i=1}^{m} x_i = (\ldots((x_i \oplus x_2) \oplus x_3) \oplus \ldots \oplus x_{m-1}) \oplus x_m.$$

We are now ready to define a *relation*. Let R be a relation scheme. A *relation* r on the scheme R is a finite collection of tuples x on R such that no two tuples in r are value-equivalent. We provide a few example relations in Table IX. Note from this table that a relation can be either *deterministic* (i.e., a relation on a scheme without probability stamps) or *probabilistic* (i.e., a relation on a scheme with a probability stamp). The above definition of a relation is general enough to include both possibilities, and, in what follows, the relational operations are defined in such a manner that both types of relations can be supported.

Table IX. Examples of Relations in an Employee Database

Relation: Employee

EMP#	ssn	lName	fName
3025	086-63-0763	Lyons	James
6723	089-83-0789	Kivari	Jack

Relation: Emp_Sal

EMP#	rank	salary	*pS*
3025	clerk	15K	0.2
3025	cashier	20K	0.8
6723	clerk	18K	0.4
6723	cashier	20K	0.4
6723	cashier	21K	0.1

Relation: Emp_Dept

EMP#	dept	*pS*
3025	toy	0.2
3025	shoe	0.6
3025	auto	0.2
6723	toy	0.4
6723	auto	0.4

4.2 Primary and Foreign Keys

In the relational model, every tuple in a relation represents a unique object (i.e., an entity or a relationship) from the real world; a *superkey* is a set of attributes that uniquely identifies a tuple, and hence an object. A superkey, in that sense, is an *object surrogate*, one that uniquely identifies every object. A *candidate key* is a minimal superkey, minimal in the sense that no attribute can be dropped without sacrificing the property of uniqueness. For each relation, only one candidate key is chosen as the *primary key* of that relation.

In the probabilistic extension, where every tuple has a probability stamp that represents the joint probability of occurrence of the attribute values in that tuple, each tuple cannot stand for a unique object. Associated with every object there may be several tuples representing the complete joint distribution of its attributes. This suggests that we must retain the *object surrogate* interpretation of the primary key (i.e., unique identifier of real world objects) and discard the notion of the primary key as a unique identifier of tuples.

The term *foreign key* retains the usual meaning in this model. In other words, a foreign key of a relation scheme R is a set of attributes $F \subset R$ that *refers* to a primary key K of some relation scheme S. Attributes in F and K may have different names, but they relate to the same real-world property of an object and come from the same domain. If r and s are relations on schemes R and S respectively, we call r the *referring* (or *referencing*) relation and s the *referred* (or *referenced*) relation. This is written symbol-

ically as: $r.F \rightarrow s.K$. The possibility that r and s are the same relation is not excluded. Primary and foreign keys are useful in enforcing important integrity constraints on probabilistic relations.

Intra-Relational Integrity Constraints: Let r be any relation on scheme R with primary key K. The following intra-relational constraints are imposed on r:

(1) The total probability associated with a primary key value must be no more than one. In other words, for all $x \in r$,

$$\sum_{\substack{y \in r \\ y(K) = x(K)}} y(pS) \leq 1.$$

Since it has been implicitly assumed that the probability value for a tuple on a deterministic scheme is always unity (the *deterministic assumption*), the above constraint reduces to key uniqueness for each tuple when deterministic relations are considered.

(2) For all $x \in r$, no part of $x(K)$ can be null. (Null values in probabilistic relations are discussed in Section 6 of this paper.)

(3) For all $x \in r$, if $pS \in R$, then $x(pS) \in (0, 1]$ and $x(pS)$ is not null.

Referential Integrity Constraints: Let r and s be two relations on schemes R and S respectively. Let K_R and K_S be the primary keys of R and S, and let $r.F \rightarrow s.K_S$ for some $F \subset R$. The following referential constraints are imposed on r and s:

(1) For all $x \in r$, if there exists an attribute $A \in F$ such that $x(A)$ is null, then for all other attributes $B \in F$, $x(B)$ is also null. This ensures that the foreign key value of a tuple is not partially null.

(2) For all $x \in r$, either $x(F)$ is null (fully), or there exists $y \in s$ such that

$$\sum_{\substack{z \in r \\ z(K_R F) = x(K_R F)}} z(pS) \leq \sum_{\substack{z \in s \\ z(K_S) = y(K_S)}} z(pS),$$

where $K_R F$ is a shorthand for $K_R \cup F$. This ensures that the probability assigned for a set of attributes must be consistent with the probability of existence of the object that these attributes refer to. For example, if 0.6 is the probability that an employee named James Lyons works in the shoe department, then the probability of existence of the shoe department must not be less than 0.6.

Consider the relations Employee and Department in Table V; here Employee is the referring relation and Department is the referred relation. It can be easily verified that these relations follow the referential integrity constraints. Consider, on the other hand, the relation Department′ in Table VI. This violates the above referential constraint; the probability associated with dept = "shoe" is only 0.5 in Department′, whereas the probability that EMP# = 3025 and dept = "shoe" is given in the relation Employee as 0.6 (greater than 0.5).

Because of the deterministic assumption, this constraint conveniently reduces to the conventional referential integrity rule in the deterministic case.

4.3 Relational Operations

In this section, we redefine the basic relational operations and introduce a new operation called *conditionalization*. The conditionalization operation is useful in deriving conditional joint distribution of different attributes given other attributes. Note that the deterministic assumption—stated as $x(pS) = 1$ for all tuples x on scheme R where $pS \notin R$—allows us to provide generalized definitions for all the relational operations; these definitions work well for both probabilistic and deterministic relations.

(1) *Union.* Let r and s be relations on the same scheme R. Then the union of these two relations is defined as:

$$r \cup s = \{x(R) | ((x \in r) \wedge (\forall y \in s(y \neq x))) \vee ((x \in s) \wedge (\forall y \in r(y \neq x))) \vee (\exists y \in r \exists z \in s(x = y \odot z))\}.$$

It can be easily verified that union is *commutative*, *associative*, and *idempotent*.[5]

(2) *Difference.* Let r and s be as above. Then the difference of these two relations is given by:

$$r - s = \{x(R) | ((x \in r) \wedge (\forall y \in s(y \neq x))) \vee (\exists y \in r \exists z \in s((x \simeq y \simeq z) \wedge (y(pS) > z(pS)) \wedge (x(pS) = y(pS) - z(pS))))\}.$$

(3) *Projection.* Let r be a relation on scheme R, and let $S \subset R$. The projection of r onto S is defined as:

$$\prod_S(r) = \left\{ x(S) | x = \bigoplus_{\substack{y \in r \\ y(S) = x}} y(S). \right.$$

Note that if $Q \subset S$, then $\Pi_Q(\Pi_S(r)) = \Pi_Q(r)$.

(4) *Selection.* Let r be a relation on scheme R. Let Θ be a set of comparators over domains of attribute names in R. Let P be a predicate (called

[5]An alternative definition of the union operation may be obtained by replacing $\odot$ with $\oplus$ in the above definition. In such a case, however, the union operation would not be idempotent.

the *selection predicate*) formed by attributes in R, comparators in Θ, constants in the domain of A for all $A \in R$, and logical connectives. The selection on r for P, written $\sigma_P(r)$, is the set $\{x \in r | P(x)\}$.

For two successive selection operations, the order is unimportant, i.e.,

$$\sigma_{P_1}(\sigma_{P_2}(r)) = \sigma_{P_2}(\sigma_{P_1}(r)) = \sigma_{P_1 \wedge P_2}(r).$$

(5) *Natural Join.* Let r and s be any two relations on schemes R and S respectively, and let $R' = R - \{pS\}$ and $S' = S - \{pS\}$. The natural join of r and s is defined as:

$$r \bowtie s = \{x(R \cup S) | \exists y \in r \, \exists z \in s((x(R') = y(R')) \wedge (x(S') = z(S')) \wedge (x(pS) = y(pS)z(pS)))\}.$$

Note that the attributes in R and S should be independent for the natural join operation to yield meaningful results. It can be easily verified that the natural join is *commutative* and *associative*, but it is not *idempotent*.

(6) *Rename.* The rename operation (ρ) is used to change the names of some attributes of a relation. Let r be a relation on scheme R, and let A and B be attributes satisfying $A \in R$, $B \notin R$. Let A and B have the same domain, and let $S = (R - \{A\}) \cup \{B\}$. Then r with A renamed to B is given by:

$$\rho_{A \leftarrow B}(r) = \{y(S) | \exists x \in r((y(S - B) = x(R - A)) \wedge (y(B) = x(A)))\}.$$

Thus, the rename operation remains the same in this algebra. If pS is renamed, it loses its special meaning and behaves like just another user-defined attribute.

(7) *Conditionalization.* Let r be a relation on scheme R, and $S \subset R - \{pS\}$. The conditionalization of r on S is given by:

$$\Upsilon_S(r) = \left\{ x(R) \,\middle|\, \exists y \in r\left((x \simeq y) \wedge \left(x(pS) = \frac{y(pS)}{\eta_{S,r}(y)} \right) \right) \right\},$$

where $\eta_{S,r}(x)$ is a function defined on a tuple $x \in r$ if $pS \in R$, and is given by:

$$\eta_{S,r}(x) = \max\left\{ 1, \sum_{\substack{y \in r \\ y(S) = x(S)}} y(pS) \right\}.$$

The conditionalization operation on S revises the probability stamp associated with each tuple by changing the marginal probability of the values for attributes in S to unity. In other words, after conditionalization, the relation can be interpreted as the joint conditional distribution of all attributes in $(R - S - \{pS\})$, given the values of attributes in S.

As a result, this operation is useful, for example, in answering queries about non-key attributes of a relation for a given key value, or before performing the join operation to obtain meaningful results. Note that for the conditional probabilities to be meaningful, it may be necessary to include a candidate key as part of S.

Other relational operations such as intersection and Cartesian product can be expressed in terms of the above basic operations:

—*Intersection.* Let r and s be relations on the same scheme R. Then the intersection of these two relations is given by:

$$r \cap s = \{x(R) \mid \exists y \in r\ \exists z \in s((x = y = z) \wedge (x(pS) = \min\{y(pS), z(pS)\}))\}.$$

It can be easily verified that $y \cap s = y - (y - s)$.

—*Cartesian Product.* The Cartesian product of two relations is a special case of a natural join [Codd 1990], where the relations do not have any common attribute name (with the possible exception of pS). Let R and S be two relation schemes satisfying $(R \cap S) - \{pS\} = \emptyset$. Let r and s be relations on the schemes R and S, respectively. The Cartesian product of r and s is a relation on scheme $(R \cup S)$ given by: $r \times s = r \bowtie s$.

—*Theta-join.* Let R, S, r and s be as above. Let Θ be a set of comparators over domains of attributes in $(R \cup S)$. Let P be any predicate formed by attributes in $(R \cup S)$, comparators in Θ, constants in the domain of A for all $A \in (R \cup S)$, and logical connectives. The theta-join between r and s is given by: $r \bowtie_P s = \sigma_P(r \bowtie s)$.

—*Alpha-cut.* The alpha-cut operation selects only those tuples from a relation that have a probability of α or more. Let r be a relation on scheme R. Let $R' = R - \{pS\}$. Then alpha-cut of r, denoted $\Phi_\alpha(r)$, is $\{x(R') \mid (x \in r) \wedge (x(pS) \geq \alpha)\}$. It is easy to verify that $\Phi_\alpha(r) = \Pi_{R'}(\sigma_{pS \geq \alpha}(r))$.

The relational algebra can now be defined formally in a fashion similar to [Maier 1983]:

Relational Algebra. Assume that $\boldsymbol{U}$ is a set of attribute names, called the *universe*. $\boldsymbol{U}$ may have the probability stamp pS as only one of its elements. Let $\mathcal{S}$ be a set of domains, and let **dom** be a total function from $\boldsymbol{U}$ to $\mathcal{S}$. Let $\boldsymbol{R} = \{R_1, R_2, \ldots, R_p\}$ denote a set of distinct relation schemes, where $R_i \subset \boldsymbol{U}$, for $1 \leq i \leq p$. Let $d = \{r_1, r_2, \ldots, r_p\}$ be a set of relations, such that r_i is a relation on R_i, $1 \leq i \leq p$. Θ denotes a set of comparators over domains in $\mathcal{S}$. The *relational algebra* over $\boldsymbol{U}$, $\mathcal{S}$, **dom**, $\boldsymbol{R}$, d, and Θ is the 7-tuple $\mathcal{R} = (\boldsymbol{U}, \mathcal{S}, \textbf{dom}, \boldsymbol{R}, d, \Theta, O)$, where O is the set of operators union, difference, natural join, projection, selection, rename, and consolidation, using attributes in $\boldsymbol{U}$ and comparators in Θ, and logical connectives. An *algebraic expression* over $\mathcal{R}$ is any expression formed legally (according

to the restrictions on the operators) from the relations in d and constant relations over schemes in $\boldsymbol{U}$, using the operators in O.

The *relational algebraic expressions* and their *schemes* over $\mathcal{R}$ are defined recursively (according to the restrictions on the operators) as follows:

(1) Let $Q = \{C_1, C_2, \ldots, C_k\} \subset \boldsymbol{U}$ be any relational scheme, and let $c_i \in \text{dom}(C_i)$, $1 \leq i \leq k$. Then $\{\langle c_1 : C_1, c_2 : C_2, \ldots, c_k : C_k\rangle\}$ is a relational expression over scheme Q called a constant.

(2) Each $r_i \in d$ is a relational expression over the scheme R_i, $1 \leq i \leq p$.

(3) If E_1 and E_2 are relational expressions over the same scheme Q, then so are the following: (i) $E_1 \cup E_2$, (ii) $E_1 - E_2$, and (iii) $\sigma_P(E_1)$, where P is a selection predicate.

(4) If E is a relational expression over the scheme Q, and $S \subset Q$, then $\Pi_S(E)$ is a relational expression over the scheme S.

(5) If E_1 and E_2 are relational expressions over schemes Q_1 and Q_2, then so is $E_1 \bowtie E_2$ over the scheme $Q_1 \cup Q_2$.

(6) If E is a relational expression over Q, and A and B are attributes with the same domain, then $\rho_{A \leftarrow B}(E)$ is a relational expression over $(Q - \{A\}) \cup \{B\}$.

(7) If E is a relational expression over Q, then so is $\Upsilon_S(E)$, for all $S \subset (Q - \{pS\})$.

4.4 Relational Algebra as a Query Language

The relational operations described above could be used in formulating different queries about the probabilistic data stored in the form of relations. In order to illustrate how this can be done, we adopt the relations shown in Table IX. For convenience, we will use the following abbreviations in the queries:

E = Employee	ES = Emp_Sal	ED = EMP_Dept
E# = EMP#	l = lName	f = fName
r = rank	s = ssn	d = dept

Let us now consider the following queries and their formulation using the relational algebra:

Query 1. Find rank and salary information about employees with rank = "clerk."

$$\sigma_{\text{r="clerk"}}(\text{ES}).$$

Query 2. What is the joint distribution of EMP#, rank, salary, and department for employee number 3025?

$$\sigma_{\text{E\#=3025}}(\text{ES}) \bowtie \sigma_{\text{E\#=3025}}(\Upsilon_{\{E\#\}}(\text{ED})), \underline{or}, \sigma_{\text{E\#=3025}}(\Upsilon_{\{\text{E\#}\}}(\text{ES})) \bowtie \sigma_{\text{E\#=3025}}(\text{ED}).$$

Note that for the join to be meaningful, one must use conditionalization before performing the join.

Query 3. List employee numbers of all employees who have rank = "clerk" with probability greater than or equal to 0.8.

$$\prod_{\{E\#\}}(\sigma_{(pS \geq 0.8)\ (r = \text{"clerk"})}\left(\prod_{\{E\#, r, pS\}}(ES)\right).$$

Note that the inner projection operation is necessary to calculate the marginal distribution before the selection operation is performed on the marginal probabilities.

Query 4. What is James Lyons' rank?

$$\prod_{\{E\#, r, pS\}}(\Upsilon_{\{E\#\}}(ES) \bowtie \sigma_{(f = \text{"James"}) \wedge (l = \text{"Lyons"})}(E)).$$

Query 5. Find the conditional distribution of salary given EMP# = 6723.

$$\Upsilon_{\{E\#\}}\left(\prod_{\{E\#, s, pS\}}(\sigma_{E\# = 6723}(ES))\right).$$

5. PROPERTIES OF RELATIONAL ALGEBRA

In this section, several properties of the proposed probabilistic relational algebra are described. This algebra is *closed*, meaning that all of the algebraic operators produce valid objects, in this case, relations. It is also a *consistent extension* of the conventional relational algebra and *reduces* to the latter when the probability stamp is not part of the relation schemes.

We have stated some basic equivalencies of algebraic expressions in Section 4. In the next theorem we state a few others.

THEOREM 5.1. *Let Q and R be two relation schemes. Let q be a relation on scheme Q, and let r, r_1, r_2 be relations on scheme R. Let P be any selection predicate involving attributes of R. Then, the following are identities:*

(a) $q \bowtie (r_1 \cup r_2) = (q \bowtie r_1) \cup (q \bowtie r_2)$,

(b) $q \bowtie (r_1 - r_2) = (q \bowtie r_1) - (q \bowtie r_2)$,

(c) $\sigma_P(r_1 \cup r_2) = \sigma_P(r_1) \cup \sigma_P(r_2)$,

(d) $\sigma_P(r_1 - r_2) = \sigma_P(r_1) - \sigma_P(r_2)$,

(e) $\sigma_P(q \bowtie r) = q \bowtie \sigma_P(r)$, if P does not involve attributes in Q.

PROOF. Straightforward from the definitions of the operators. □

THEOREM 5.2. *The proposed relational algebra is closed.*

PROOF. We must show that all the basic operations in this algebra result in a relation as defined in the algebra. A relation must satisfy three

criteria: (i) the values must come from an appropriate domain, (ii) no two tuples in a relation are value-equivalent, and (iii) it must be a finite collection of tuples.

For all attributes other than pS, it is easy to see that the first criterion is satisfied. For pS, the domain is (0,1]. We will show that if the pS values come from (0,1] before a relational operation is applied, then they would also be in (0,1] after the operation. If pS-values are all greater than zero in the participating relations, then they would clearly be so when the union, projection, selection, natural join, rename, or consolidation operation is applied. The difference operation is defined in such a fashion that a tuple may be generated from two value-equivalent tuples where the pS-value of the new tuple is the difference between the pS-values of the participating tuples. The definition, however, explicitly verifies that the resulting pS-value is strictly greater than zero. For example, $\{\langle 3025, 0.4\rangle\} - \{\langle 3025, 0.5\rangle\}$ would evaluate to an empty relation ("null set" of tuples) in our algebra, and not to $\{\langle 3025, -0.1\rangle\}$. Similarly, if pS-values are all less than or equal to one in the participating relations, then they would clearly be so when the difference, selection, natural join, rename, or consolidation operation is applied. That the same applies for the union and projection operations as well is ensured in the definition of coalescence, which does not allow the pS-value of a tuple to become greater than one.

The fact that no two value-equivalent tuples are produced when the basic operators are used is explicitly ensured in the definitions of the operators. If there is a possibility of generation of value-equivalent tuples, then those tuples would be automatically coalesced in this algebra.

One can prove that the third criterion is satisfied by constructing the resulting relation from the operands. Let us consider the union operation. Assume that r_1 and r_2 are relations on the same scheme. For every tuple $x \in r_1$, there are two possibilities. Either there is only one value-equivalent tuple $y \in r_2$, in which case $(x \odot y) \in (r_1 \cup r_2)$; or there is no such tuple in r_2, in which case $x \in (r_1 \cup r_2)$. A similar observation is also true about every tuple in r_2. It is then clear that the total number of tuples in $(r_1 \cup r_2)$ is no more than $(|r_1| + |r_2|)$, where $|r|$ denotes the number of tuples in a relation r. It can be shown in an analogous manner that (a) the number of tuples in $(r_1 - r_2)$ cannot exceed $|r_1|$, (b) the number of tuples in $(r_1 \ltimes r_2)$ cannot exceed $|r_1| \times |r_2|$, (c) the number of tuples in $\Pi_S(r)$ cannot exceed $|r|$, (d) the number of tuples in $\sigma_P(r)$ cannot exceed $|r|$, and (e) the number of tuples in $\rho_{A \leftarrow B}(r)$ or in $\Upsilon_S(r)$ is exactly equal to $|r|$. Therefore, if the participating relations were finite, so must be the resulting relations. □

This algebra is a consistent extension of the conventional relational algebra, and reduces to the latter when there is no uncertainty associated with attribute values. Before these properties can be proven, we need to define two operators. These operators are not part of the algebra, but allow us to transform deterministic relations to probabilistic relations and vice versa. For these definitions and the related discussion, we use "prime" (′)

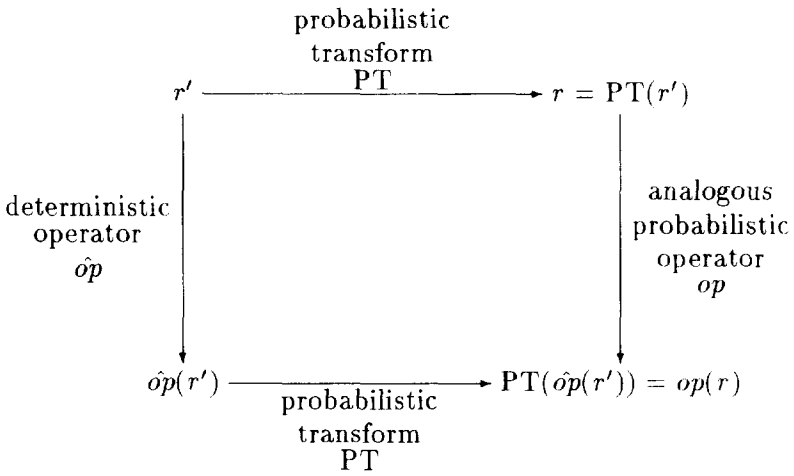

Fig. 1. Outline of an equivalence proof.

for deterministic relations and relation schemes, and "hat" (ˆ) over conventional deterministic operations exclusively.

Definition 5.1. (*Probabilistic transform*) Let R and R' be relation schemes satisfying $pS \notin R'$ and $R = R' \cup \{pS\}$. Let r' be any relation on R'. The *probabilistic transform* of r', written $PT(r')$, is a relation r on R given by:

$$r = \mathrm{PT}(r') = \{x(R) | (x(R') \in r') \wedge (x(pS) = 1)\}.$$

Definition 5.2. (*Deterministic transform*) Let R and R' be as above. Let r be any relation on R. The *deterministic transform* of r, written $\mathrm{DT}(r)$, is a relation r' on R' given by:

$$r' = \mathrm{DT}(r) = \{x(R') | (x \in r) \wedge (x(pS) = 1)\}.$$

THEOREM 5.3. *The probabilistic relational algebra is a consistent extension of the conventional algebra.*

PROOF. A probabilistic algebra is a consistent extension of the conventional algebra if any relation or algebraic expression that can be represented in the conventional algebra has a counterpart in the probabilistic algebra. In other words, the algebra should be at least as powerful as the conventional algebra. Figure 1 gives an outline of the equivalence proof for a unary operator. We show the proof for the binary operation natural join. The others follow in an analogous manner.

Let R' and S' be two relation schemes such that $pS \notin R'$ and $pS \notin S'$, and let r' and s' be relations on R' and S', respectively. Also let $R = R' \cup \{pS\}$ and $S = S' \cup \{pS\}$. We need to show that: $\mathrm{PT}(r') \bowtie \mathrm{PT}(s') = \mathrm{PT}(r' \bowtie s')$.

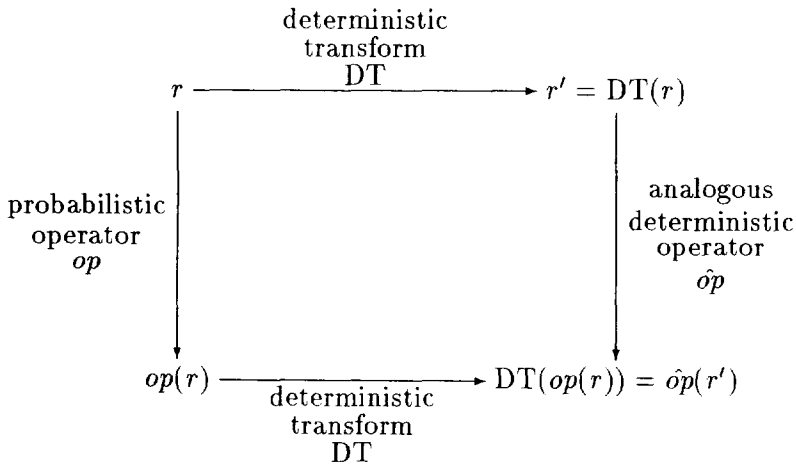

Fig. 2. Outline of a reduction proof.

If a tuple $x \in \mathrm{PT}(r') \ltimes \mathrm{PT}(s')$, then there exist $y \in \mathrm{PT}(r')$ and $z \in \mathrm{PT}(s')$ such that $y(R') = x(R')$ and $z(S') = x(S')$. Clearly, $y(R') \in r'$ and $z(S') \in s'$. As a result, $x(R' \cup S') \in (r' \ltimes s')$. Also note that $x(pS) = y(pS) \times z(pS) = 1$. This implies that $x \in \mathrm{PT}(r' \ltimes s')$.

Alternatively, if $x \in \mathrm{PT}(r' \ltimes s')$, then $x(R' \cup S') \in (r' \ltimes s')$. There must exist $y' \in r'$ and $z' \in s'$ such that $y' = x(R')$ and $z' = x(S')$. This, of course, implies that $\langle y', 1\rangle \in \mathrm{PT}(r')$ and $\langle z', 1\rangle \in \mathrm{PT}(s')$. Since $x(pS) = 1$, we conclude that $x \in \mathrm{PT}(r') \ltimes \mathrm{PT}(s')$. □

THEOREM 5.4. *The probabilistic relational algebra reduces to the conventional algebra.*

PROOF. A probabilistic relational algebra reduces to the conventional algebra if the semantics of the algebra is consistent with that of the conventional algebra. The reduction proof for any unary operator is outlined in Figure 2. Again we prove the case of the binary operation natural join, and leave the other cases for the reader to verify.

Let R and S be two relation schemes such that $pS \in R$ and $pS \in S$, and let r and s be relations on R and S, respectively. Also let $R' = R - \{pS\}$ and $S' = S - \{pS\}$. We must prove that: $\mathrm{DT}(r) \bowtie \mathrm{DT}(s) = \mathrm{DT}(r \bowtie s)$.

If a tuple $x' \in \mathrm{DT}(r) \ltimes \mathrm{DT}(s)$, then there must exist tuples $y' \in \mathrm{DT}(r)$ and $z' \in \mathrm{DT}(s)$ such that $x'(R') = y'$ and $x'(S') = z'$. This implies that $\langle y', 1\rangle \in r$ and $\langle z', 1\rangle \in s$. In other words, $\langle x', 1\rangle \in (r \bowtie s)$. Clearly then, $x' \in \mathrm{DT}(r \bowtie s)$.

On the other hand, let us assume that $x' \in \mathrm{DT}(r \bowtie s)$. So, $\langle x', 1\rangle \in (r \bowtie s)$. This implies the existence of $y \in r$ and $z \in s$ satisfying $y(R') = x'(R')$, $z(S') = x'(S')$, and $y(pS) = z(pS) = 1$. Clearly, then $y(R') \in \mathrm{DT}(r)$ and $z(S') \in \mathrm{DT}(s)$. This means $x' \in \mathrm{DT}(r) \ltimes \mathrm{DT}(s)$. □

Table X. EMPLOYEE: A Probabilistic Relation with Null Values

EMP#	rank	salary	dept	pS
3025	clerk	15K	toy	0.2
3025	cashier	20K	shoe	0.6
3025	cashier	15K	auto	0.2
6723	clerk	18K	toy	0.4
6723	cashier	20K	auto	0.4
6723	*	*	*	0.1
6879	clerk	25K	toy	0.3
6879	clerk	*	toy	0.1
6879	cashier	*	*	0.6

6. INCOMPLETE DISTRIBUTION AND NULL VALUES

We now turn our attention to the case where the joint probability distribution of the attributes of an object is partially specified. For example, it is possible that the existence of an employee is certain (i.e., the marginal probability of the key EMP# is one), but the marginal distribution of the salary of that employee is not completely specified. This scenario is illustrated in the relation shown in Table X, where the existence of an employee with EMP# = 6879 is known with certainty; the marginal distribution of rank is completely specified for this employee, but the marginal distribution for salary and department information is not completely available. Similarly, the probability of existence of an employee with EMP# = 6723 is 0.9, but only 0.8 out of this total probability mass of 0.9 is specified for all the other attributes.

The relation in Table X models this type of incompleteness with the help of a *null value* "$*$." This null value is similar to the null value in the traditional relational model; it means that a portion of the probability mass is associated with a value that is unknown. For example, out of a total of 1.0, only 0.3 is associated with a known value of salary for EMP# = 6879; the remaining 0.7 is given to the null value.

It should be noted that the admission of null values into a relation increases the complexity of the model significantly. This is because if a is a constant, neither $(a = *)$ or $(a \neq *)$ can be guaranteed to hold. As a result, operations on a relation with null values may yield inaccurate results or may lose some semantic information available in the original relation. For this reason, we do not consider null values as a part of our basic algebra.

Of course, the occurrence of null values could be attributed to the fact that several independent attributes are represented together. For example, if we assume that the attributes "rank," "salary," and "dept" are all independent, we may decompose the EMPLOYEE relation in Table X into smaller relations on four different schemes {EMP#, pS}, {EMP#, rank, pS}, {EMP#, salary, pS}, and {EMP#, dept, pS}. With such a decomposition, it is possible to remove the null values completely, while retaining the actual meaning of the original relation. Although that may be an acceptable solution in some cases, one must note that, in general, the original relation

(with nulls) cannot be recreated from the smaller relations. If the overall view of the relation is important, such a decomposition may not be without loss.

Date [1986] discusses the problems associated with representing null values in the traditional relational model. These problems persist when we try to represent nulls in the probabilistic relational model. Date also describes a practical approach of extending the relational operations. We extend Date's approach in this section as a practical method of handling nulls, and note that semantic loss of information may result in some cases.

6.1 Interpretation of Partial Distribution

An important question is the interpretation of the probability stamp when the joint probability distribution is not fully specified. How we interpret the probability stamp has to do with the interpretation given to the portion of the total probability mass (associated with a key value) that is not specified, called the *missing probability* in Barbará et al. [1992]. There are two possible interpretations that may be given to the missing probability. The first is that the missing probability is associated with realizations of those values of attributes that are not already included in the relation. Thus, in Table X, the missing probability of 0.1 for EMP# 6723 could be distributed in any manner over those joint realizations for rank, salary, and department that are not already included in the table. With this interpretation, the probability stamps for tuples that do appear in the relation are construed as point estimates of the conditional probabilities for given values of the attributes. Therefore, the probability that EMP# = 6723, rank = "clerk," salary = 18K and dept = "toy" is interpreted to be 0.4. Similarly, the probability that EMP# = 6879 and dept = "toy" is 0.4.

The second interpretation for the missing probabilities is that they could be distributed over the entire set of realizations of the attributes, including the ones that already appear in the relation. In that case, the uncertainty associated with the attribute values for tuples that appear in the relation are represented by probability intervals, and not point estimates. The probability stamp associated with a tuple is then the lower bound of the probability interval for that tuple (as in Barbará et al. [1992]). Consider the previous example of EMP# 6723; this key value has a missing probability of 0.1. Since this probability mass could be assigned to any value, including those that have already appeared, the probability that EMP# = 6723, rank = "clerk," salary = 18K, and dept = "toy" lies in the interval [0.4, 0.5]. Similarly, the probability that EMP# = 6879 and dept = "toy" lies in the interval [0.4, 1.0]. When the distribution is completely specified, the interval clearly reduces to a point.

6.2 Extended Relational Operations

In this section, the basic algebraic operations are extended to incorporate the null values as possible attribute values. An important feature of this extension is that the semantics associated with each of the above two

interpretations is preserved as a result of all the basic relational operations; i.e., the extended operations can handle both interpretations of missing probabilities. Consequently, depending on their preference, users can represent uncertainties regarding attribute values either as point estimates or as intervals. The result of the relational operations will be consistent with the user's interpretation of the original tables. First, a few definitions similar to Maier's [1983] are necessary.

Let x be any tuple on scheme R. If $A \in R$ and $x(A)$ is not null, x is called *definite* on A, written $x(A) \downarrow$. For $S \subset R$, $x(S) \downarrow$ if $x(A) \downarrow$ for all $A \in S$. A tuple x is said to *subsume* a tuple y, both on scheme R, written $x \geq y$, if for all $A \in R$, $y(A) \downarrow$ implies $x(A) = y(A)$.

We now redefine the concept of value-equivalent tuples for those that might have null values. Let R be a relation scheme and let $R' = R - \{pS\}$. For any two tuples x and y on R,

$$(x = y) \Leftrightarrow (x(R') \geq y(R')) \wedge (y(R') \geq x(R')).$$

Again, value-equivalent tuples are not allowed to co-exist in a relation; they must be coalesced. The coalescence-PLUS and the coalescence-MAX operations as defined in Section 4 can also be used here.

We can now redefine the basic relational operations for relations containing null values. The definitions of the union, difference, and projection operations from Section 4 can be used with the extended definition of value-equivalent tuples.[6] The rename operation also remains the same for relations with nulls. Thus, only the selection, natural join, and conditionalization operations have to be redefined.

Selection. Let R, r, Θ, P be as in the definition of the *selection* operation in Section 4. Let $S \subset R$ be the set of attributes involved in P. Then, $\sigma_P(r) = \{x \in r | x(S) \downarrow \wedge P(x)\}$. In other words, tuples with null values for attributes involved in the selection predicate are not considered.

Natural Join. Let r and s be any two relations on schemes R and S respectively. Let $Q = R \cap S$, $R' = R - \{pS\}$ and $S' = S - \{pS\}$. Then,

$$r \bowtie s = \{x(R \cup S) | \exists y \in r\ \exists z \in s (y(Q) \downarrow \wedge z(Q) \downarrow \wedge (x(R') = y(R')) \wedge (x(S') = z(S')) \wedge (x(pS) = y(pS) z(pS)))\}.$$

In other words, the join operation matches tuples on non-null attribute values only.

[6]These definitions of the relational operations, despite their similarity to the previous definitions in Section 4, are essentially different from the latter because the definition of value-equivalent tuples has been extended. Our approach is similar to the one taken by Date [1986] for the conventional relational operations; his definition of algebraic operations relies on redefining the concept of duplicate tuples to incorporate null values. Our algebra is based on generalizing the concept of duplicate tuples to value-equivalent tuples, and hence, value-equivalent tuples are redefined for the case where null values may exist.

Conditionalization. Let r be a relation on scheme R, and $S \subset R - \{pS\}$. The conditionalization of r on S is given by:

$$\Upsilon_S(r) = \left\{ x(R) \middle| \exists y \in r\left(y(S)\downarrow \wedge (x \simeq y) \wedge \left(x(pS) = \frac{y(pS)}{\eta_{S,r}(y)} \right) \right) \right\},$$

where $\eta_{S,r}(y)$ is as defined in Section 4. Again, tuples with null values for attributes in S are excluded in performing the conditionalization operation.

Finally, we would like to introduce a new operation called the *N-th moment*. This operation allows us to obtain interesting aggregate properties of different attribute names based on the original distribution of those attribute names represented in the form of a relation. Before we define this operation, let us first define the N-th moment of a distribution from a statistical point of view and discuss its usefulness. Let ψ be a random variable with domain Ψ and probability density function $f_\psi(x)$, $x \in \Psi$. Then, its N-th moment, $\mu_N(\psi)$, is defined as:

$$\mu_N(\psi) = \mathrm{E}[\psi^N] = \int_{x \in \Psi} x^N f_\psi(x)\mathrm{d}x.$$

Moments of a distribution are useful in obtaining aggregate properties of a distribution such as mean, standard deviation, skewness, and kurtosis. For example, the standard deviation of the random variable ψ can be easily obtained from its first and second moments:

$$\mathrm{stdv}(\psi) = \sqrt{\mu_2(\psi) - (\mu_1(\psi))^2}.$$

These aggregate properties are useful not only in understanding the overall nature of a distribution, but also in comparing two different distributions. This is why moments are a very important tool in statistical analysis. The N-th moment operation helps to form an overall opinion about the nature of real-world objects, as well as allowing various statistical analysis to be performed on the stored data.

N-th Moment. Let r be a relation on scheme R. Let $R' = R - \{pS\}$ and $S \subset R'$. The N-th moment of r given S, written $\mu_{S,N}(r)$, is defined as:

$$\mu_{S,N}(r) = \{x(R') | \exists y \in r(y(S)\downarrow \wedge (x(S) = y(S))$$

$$\wedge (\forall A \in (R' - S)(x(A) = m_{S,r,N}(y, A)))\},$$

Table XI. EMPLOYEE Relation after First Moment Operation

EMP#	rank	salary	dept
3025	Ω	18K	Ω
6723	Ω	19K	Ω
6879	Ω	25K	Ω

where,

$$m_{S,r,N}(x, A) = \begin{cases} \sum_{\substack{y \in r \\ y(A) \cdot \\ y(S) = x(S)}} (y(A))^N y(pS) \\ \sum_{\substack{y \in r \\ y(A) \\ y(S) = x(S)}} y(pS) \\ \Omega, \end{cases} \quad \begin{array}{l} \text{if } pS \in R \text{ and } A \in R' \text{ is numeric,} \\ \\ \text{otherwise} \end{array}$$

A few comments are in order about the *N-th moment* operation. First, note that it is really a family of operations; this is because we get a different operation for each positive integer N. For example, to obtain the expected value of different attributes, we can use the *first moment*, i.e., $N = 1$. If we apply the first moment operation on the EMPLOYEE relation shown in Table X with S = {EMP#}, we would get the expected value of all other attributes given the attribute EMP#; this is illustrated in Table XI. Second, it is possible to define other operations—such as *standard deviation*, *skewness* and *kurtosis*—based on the above class of operations. Third, as can be seen from the definition, null values (*) are not considered in calculating moments. In other words, only the explicitly specified part of the distribution is considered in the calculation of moments. Finally, we do not consider the *N-th moment* as a part of our basic algebra due to the possibility of generation of a special kind of null value (Ω) for non-numeric attributes.

7. CONCLUSIONS

Although relational databases enjoy a very widespread popularity in modern business information systems, they lack the power to model uncertainty in data items. In this paper, we present an extension of the relational model and an algebra that uses classical probability theory to express uncertainty about object properties. This model overcomes some of the major problems associated with existing proposals in the literature.

We discuss the basic structure of probabilistic relations, and formally define the necessary operations. Our representation of relations abides by first normal form (1NF), and hence is easier to implement. The associated relational algebra is shown to be closed. It is also a consistent extension of the conventional relational algebra and reduces to the latter. We feel that the conventional relational operations (having been redefined for the

extended structure) and the two new operations, namely, conditionalization and N-th moment, are enough to express the majority of queries that may be asked about the probabilistic data. The N-th moment operation can be used to compare two different distributions; as a result, we do not need operations such as ϵ-join or ϵ-select [Barbará et al. 1992]. Another strength of this model is that it can support more than one interpretation of the probability stamp in the case of partially specified distributions. If the relational operations are applied to relations with a particular interpretation of the probability stamp, that meaning is preserved after the operations as well.

An issue of practical significance is how to obtain the probabilistic data. Several methods have been presented in the literature for this purpose. Pearl [1986] discusses how the probability values could be assigned based on the user's confidence. It is also possible to assign probability values based on sampling, where a portion of the population is sampled to estimate the distribution [Barbará et al. 1992]. Yet a third method, based on maximizing the entropy, subject to a set of known constraints, is described in the classical work by Jaynes [1968].

Our representation considers only discrete joint probability distributions; this may be restrictive in some situations. However, business data are often collected and represented in the form of discrete distributions. Moreover, when the underlying distribution is continuous, usually it can be represented as a discrete distribution at a desired level of granularity. The issues of storage and query evaluation for data with continuous distribution are significantly more complex and need to be addressed in future research.

There are several other directions for future research. Issues such as storage structure, access paths, and query optimization need to be addressed for successful implementation of the model. It is also necessary to develop a non-procedural query language (like SQL or Quel) for this model. Belief revision of probabilistic data is another important area of research. We are currently examining these issues so that a prototype probabilistic database management system can be developed.

ACKNOWLEDGMENTS

The authors wish to thank Terence M. Barron, Amit Basu, Debashis Ghosh, Aditya N. Saharia, and Veda C. Storey for their helpful comments. Thanks are also in order for the associate editor, Hector Garcia-Molina, and the four anonymous reviewers of *ACM Transactions on Database Systems*; this paper has greatly benefited from their careful reading and constructive criticisms.

REFERENCES

BARBARA, D., GARCIA-MOLINA, H., AND PORTER, D. 1992. The Management of Probabilistic Data. *IEEE Trans. Knowl. Data Eng. 4*, 5 (Oct.), 487–502.

BUCKLES, B. P. AND PETRY, F. E. 1983. Information-Theoretical Characterization of Fuzzy Relational Databases. *IEEE Trans. Syst. Man Cybern. 13,* 1 (Jan./Feb.), 74–77.

BUCKLES, B. P. AND PETRY, F. E. 1984. Extending the Fuzzy Database with Fuzzy Numbers. *Inf. Sci.* (New York) 34, 145–155.

CAVALLO, R. AND PITTARELLI, M. 1987. The Theory of Probabilistic Databases. In *Proceedings of the 13th VLDB Conference.* Brighton, Eng., 71–81.

CODD, E. F. 1979. Extending the Database Relational Model to Capture More Meaning. *ACM Trans. Database Syst. 4,* 4 (Dec.), 397–434.

CODD, E. F. 1990. *The Relational Model for Database Management: Version 2.* Addison-Wesley, Reading, Mass.

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

DEY, D., BARRON, T. M. AND SAHARIA, A. N. 1995. Logical Design of Temporal Databases: A Decision Theoretic Approach. Working paper. Louisiana State University.

JAYNES, E. T. 1968. Prior Probabilities. *IEEE Trans. Syst. Sci. Cybernetics. 4,* 3 (Sept.), 227–241.

JEFFREY, R. 1983. *The Logic of Decision.* University of Chicago Press, Chicago, Ill.

KLIR, G. J. AND FOLGER, T. A. 1988. *Fuzzy Sets, Uncertainty, and Information.* Prentice Hall, Englewood Cliffs, N.J.

LIPSKI, W., JR. 1979. On Semantic Issues Connected with Incomplete Information Databases. *ACM Trans. Database Syst. 4,* 3 (Sept.), 262–296.

MAIER, D. 1983. *The Theory of Relational Databases.* Computer Science Press, Rockville, MD.

MENDELSON, H. AND SAHARIA, A. N. 1986. Incomplete Information Costs and Database Design. *ACM Trans. Database Syst. 11,* 2 (June), 159–185.

PEARL, J. 1986. Fusion, Propagation, and Structuring in Belief Networks. *Artif. Intell. 29,* 241–288.

PEARL, J. 1989. Probabilistic Semantics for Nonmonotonic Reasoning: A Survey. In *Proceedings of the First Conference on Principles of Knowledge Representation and Reasoning.* Morgan Kaufmann, 505–516.

PRADE, H. AND TESTEMALE, C. 1984. Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries. *Inf. Sci.* (New York) 34, 115–143.

RAJU, K. V. S. V. N. AND MAJUMDAR, A. K. 1988. Fuzzy Functional Dependencies and Lossless Join Decomposition of Fuzzy Relational Database Systems. *ACM Trans. Database Syst. 13,* 2 (June), 129–166.

SNODGRASS, R. T. 1987. The Temporal Query Language TQuel. *ACM Trans. Database Syst. 12,* 2 (June), 247–298.

TSENG, F. S. C, CHEN, A. L. P. AND YANG, W.-P. 1993. Answering Heterogeneous Database Queries with Degrees of Uncertainty. *Distributed and Parallel Databases: An International Journal. 1,* 3 (July), 281–302.

WONG, E. 1982. A Statistical Approach to Incomplete Information in Database Systems. *ACM Trans. Database Syst. 7,* 3 (Sept.), 470–488.

ZEMANKOVA, M. AND KANDEL, A. 1985. Implementing Imprecision in Information Systems. *Inf. Sci.* (New York) 37, 107–141.

Received March 1994; revised October 1995; accepted November 1995