Database Support for Probabilistic Attributes and Tuples

Sarvjeet Singh #1, Chris Mayfield #2, Rahul Shah *3, Sunil Prabhakar #4,
Susanne Hambrusch #5, Jennifer Neville #6, Reynold Cheng †7

#*Department of Computer Science, Purdue University*
West Lafayette, Indiana, USA
1sarvjeet@cs.purdue.edu
2cmayfiel@cs.purdue.edu
4sunil@cs.purdue.edu
5seh@cs.purdue.edu
6neville@cs.purdue.edu

**Department of Computer Science, Louisiana State University*
Baton Rouge, Louisiana, USA
3rahul@csc.lsu.edu

†*Department of Computing, Hong Kong Polytechnic University*
Kowloon, Hong Kong, China
7csckcheng@comp.polyu.edu.hk

***Abstract*— The inherent uncertainty of data present in numerous applications such as sensor databases, text annotations, and information retrieval motivate the need to handle imprecise data at the database level. Uncertainty can be at the attribute or tuple level and is present in both continuous and discrete data domains. This paper presents a model for handling arbitrary probabilistic uncertain data (both discrete and continuous) natively at the database level. Our approach leads to a natural and efficient representation for probabilistic data. We develop a model that is consistent with possible worlds semantics and closed under basic relational operators. This is the first model that accurately and efficiently handles both continuous and discrete uncertainty. The model is implemented in a real database system (PostgreSQL) and the effectiveness and efficiency of our approach is validated experimentally.**

I. Introduction

For many applications data is inherently uncertain. Examples include sensor databases (measured values have errors), text annotation (annotations are rarely perfect), information retrieval (the match between a document and a query is often a question of degree or confidence), scientific data (model outputs, estimates, experimental measurements, and hypothetical data), and data cleansing (multiple alternatives for an incorrect value). While existing databases offer great benefits for handling such data, they do not provide direct support for the uncertainty in the data. Consequently, these applications are either forced to manage the uncertainty outside the database, or coerce the data into a form that can be represented in the database model.

Due to the importance of the need for supporting uncertain data several researchers have addressed this problem. A wide body of work deals with fuzzy modeling of uncertain data [1]. In this paper we focus on probabilistic modeling. Recent work on the problem of handling uncertain data using probabilistic relational modeling can be divided into two main groups. One deals with modeling and the other with efficient execution of queries. Work on query processing over probabilistic data has assumed a simple model – a single (continuous or discrete) attribute that takes on probabilistic values [2], [3], [4], [5], [6], [7]. Most of this work is focussed on developing index structures for efficient query evaluation over probability distribution (or density) functions (pdf). While this work addresses specific queries (e.g. Range [8], nearest-neighbors [2]), it lacks a comprehensive model to handle complex database queries consisting of selects, projects and joins in a consistent manner. Most of the work is also focused on single table queries.

Recently proposed models for probabilistic relational data deal with the representation and management of *tuple uncertainty* (with the exception of [6]). These models are naturally well-suited for applications with categorical uncertainty. Under tuple uncertainty, the presence of a tuple in a relation is probabilistic, and multiple tuples can have constraints such as mutual exclusion among them. The recently proposed models [9], [10], [11] generalize most of the earlier models for probabilistic relational data. In contrast, *attribute uncertainty* models [6], [12] consider that a tuple is definitely part of the database, but one or more of its attributes is (are) not known with certainty. The model in [6] allows an uncertain value to take on a continuous ranges of values, but all other work has been focussed on the case of discrete uncertainty (i.e. an enumerated list of alternative values with associated probabilities). Continuous uncertainty models easily capture the case of discrete uncertainty. Discrete uncertainty models can handle continuous uncertainty by sampling the continuous pdf, but are forced to tradeoff accuracy (lots of samples) or efficiency (fewer samples).

This paper presents a new model for representing probabilistic data that handles both continuous and discrete domains and allows uncertainty at the attribute and tuple level. To the

978-1-4244-1837-4/08/$25.00 © 2008 IEEE

best of our knowledge, this is the first model that handles continuous pdfs and is closed under possible worlds semantics (Section I-A). The model can handle arbitrary correlations among attributes of a given tuple, and across tuples. Although this model is motivated by attribute uncertainty, it can directly handle tuple uncertainty, and thus is more general. The underlying representation for arbitrarily correlated uncertain data in our model is based upon multi-dimensional pdf attributes. Our approach results in a more natural representation for uncertain data primarily due to the fact that our chosen data representation better matches how uncertainty is modeled in applications. A second advantage of our model is its space efficient representation of uncertain data. This efficiency results in improved query result accuracy and lower processing time.

As an example, consider an application which uses sensors to measure locations of objects. For simplicity, assume that location is a 1-dimensional attribute. There is an uncertainty associated with readings of any sensor in the real world. We assume that the error for each reading is represented by a Gaussian distribution with a given variance around the observed sensor value (mean), in line with the well-known error for GPS devices. A large variance (i.e., large uncertainty in the reading) might be the result of poor quality of sensors or other environmental factors. Table I shows the values returned by the sensors. (*Gaus* represents a gaussian distribution followed by the parameters of the distribution – mean and variance).

TABLE I
EXAMPLE: SENSOR DATABASE

Sensor ID	Location
1	Gaus(20,5)
2	Gaus(25,4)
3	Gaus(13,1)

Now consider the case where we use tuple uncertainty (i.e., discrete uncertainty) to model the sensor database in Table I. Current tuple uncertainty models will be forced to make a discrete approximation of the pdf as they only support discrete uncertain data. This approach has a number of weaknesses. Firstly, such a representation is not efficient as we have to repeat certain attribute(s) (e.g., sensor id) along with each value instance of uncertain attribute(s). Secondly, either we have to sample many points (not practical) or sacrifice a great deal of accuracy (not desirable). On the other hand, if we use the symbolic form of a Gaussian distribution, obviously the answers will be more accurate as we are avoiding approximations. Furthermore, as we will see later, the usual database operations can be evaluated on symbolic pdfs in a more efficient manner. Note that this requires built-in support for symbolic pdfs (e.g., Gaussian) in the database. Our model provides this support, and for non-standard distributions, we support a *generic* pdf represented by histograms (*Hist*). Histograms give us an approximation for continuous pdfs, but this approximation is still more accurate than a discrete approximation. This issue is further explored in the experimental section.

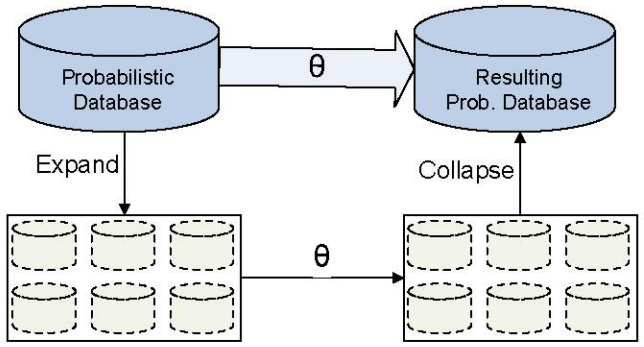

Fig. 1. Possible Worlds Semantics

In addition, even in situations where the base uncertain data is discrete, some queries (e.g. aggregates) can produce results that are very expensive to represent using discrete pdfs. The main reason is that the resulting uncertain attribute can have an exponential number of possible values. In such cases, one can save space as well as time by approximating with a continuous pdf. This is exactly what our model proposes.

While our model is tailored towards representing continuous distributions, it is general enough to be used for modeling discrete uncertainty as well.

In summary, the salient features of our model are:

1) It handle both continuous and discrete uncertainty (with arbitrary correlations) natively at the database level, and is consistent and closed under possible worlds semantics.
2) The first model for uncertain data that can accurately handle continuous pdfs.
3) The pdf approach leads to a more natural and efficient representation and implementation than a tuple uncertainty based approach.

A. Possible Worlds Semantics

The definition of relational operators for this model is based upon the Possible Worlds Semantics (PWS) [13] that has been commonly used for other work on uncertain databases. Under these semantics, a probabilistic relation is defined over a set of probabilistic events. Depending upon the outcome of each of these events, a possible world is defined. Thus given a probabilistic relation, we get a set of possible worlds corresponding to all possible combinations of the outcomes of the events in the relation. Figure 1 shows a graphical view of the possible worlds semantics. Given a probabilistic database and query θ to be evaluated over this database, conceptually we first *expand* the database to produce the set of all possible worlds. The query is then executed on each possible world. The resulting probabilistic database is defined as the database obtained by *collapsing* the possible worlds in which the query is satisfied.

Consider a database table with uncertain attributes a and b, as shown in Table II. It consists of two probabilistic tuples. The first tuple represents a total of 4 possibilities: (i.e. $\{0,1\},\{0,2\},\{1,1\},\{1,2\}$) and a single (certain) value for

TABLE II
EXAMPLE OF PROBABILISTIC TABLE

a	Pr(a)	b	Pr(b)
0	0.1	1	0.6
1	0.9	2	0.4
7	1.0	3	1.0

TABLE III
POSSIBLE WORLDS

Possible Worlds		Probability
0 7	1 3	0.06
0 7	2 3	0.04
1 7	1 3	0.54
1 7	2 3	0.36

the second tuple. The corresponding set of possible worlds are shown in Table III along with the associated probabilities for each world. The semantics of a query over this uncertain relation are defined as follows. The query is executed over each possible world (which has no uncertainty) to yield a set of possible results along with the probability of each result. The probability values of worlds that yield the same result are aggregated to yield the probability of that result for the overall query over the uncertain relation. Consider a selection query with predicate $a < b$, over the relation in Table II. Conceptually, this query is evaluated over each possible world. The probability that a tuple satisfies the query criterion is equal to the sum of the probabilities of the possible worlds in which the tuple satisfies the query. In practice, the number of possible worlds can be very large (even infinite for continuous uncertainty). The goal of a practical model is to avoid enumerating all possible worlds while ensuring that the results are consistent with PWS. Section III-C shows how our model handles this particular example.

II. MODEL

In this section, we formally define our model for representing and querying a database with probabilistic data. We allow two kinds of attributes – *uncertain* (or pdf attributes) and *certain* (or precise) attributes. The model represents a set of database tables **T**, with a set of probabilistic schemas $\{(\Sigma_T, \Delta_T) : \forall T \in \mathbf{T}\}$ and a history Λ for each dependent set of attributes in **T**. A database table T is defined by a *probabilistic schema* (Σ_T, Δ_T) consisting of a *schema* (Σ_T) and *dependency information* (Δ_T). The schema Σ_T is similar to the regular relational schema and specifies the names and data types of the table attributes (both certain and uncertain). The dependency information Δ_T identifies the attributes in T that are jointly distributed (i.e., correlated). The uncertain attributes are represented by pdfs (or joint pdfs) in the table. In addition to pdfs, for each dependent group of uncertain attributes we store its *history* Λ. We will now describe each of these concepts in detail.

A. Uncertain Data types and Correlations

There are two major kinds of uncertain data types that our model supports – discrete and continuous. These data types are represented using their pdfs. The uncertainty model in many real applications can be expressed using standard distributions. Our model has built in support for many commonly used continuous (e.g., Gaussian, Uniform, Poisson) and discrete (e.g., Binomial, Bernoulli) distributions. These distributions are stored symbolically in the database. The major advantage of using these standard distributions is efficient representation and processing. When the underlying data distribution cannot be represented using the standard distributions we revert to *generic* distributions – Histogram and Discrete sampling. The histogram distribution consists of buckets over the data domain, along with the probability density in each bucket. The discrete sampling simply consists of multiple value-probability pairs. The bin size (or number of sampling points) is an important parameter that decides the trade-off between accuracy and efficiency.

The simple pdf distributions discussed above can be used to represent 1-dimensional pdfs. But in many cases, there are *intra-tuple* correlations present within the attributes. For example, in a location tracking application, the uncertainty between the x- and y-coordinates of an object is correlated. These more complex distributions are supported in our model using joint probability distributions *across* attributes. For example, to represent the 2-D uncertainty in case of moving objects we represent the uncertainty by creating two uncertain attributes x and y which specify the x- and y-coordinates of the object, respectively. Instead of specifying two *independent pdfs* over x and y, we have a single *joint pdf* over these two attributes.

The information about intra-tuple dependencies is captured by the schema dependency information Δ_T. Δ_T is a partition of all the uncertain attributes present in the table T. It consists of multiple sets of attributes that are correlated within a tuple. These sets are called *dependency sets*. It also contains *singleton sets* containing attributes that are uncertain but are not dependent on any other attributes. The attributes not listed in Δ_T are assumed to be certain.

To illustrate, let us consider a table T with schema $\Sigma_T = (a_1{:}d_1, a_2{:}d_2, a_3{:}d_3, a_4{:}d_4)$, where d_i represents the data type of attribute a_i. If all the attributes in the table are certain, $\Delta_T = \phi$. On the other hand, if a_1, a_2 and a_3 are uncertain and a_1, a_2 are correlated, this information is represented by defining the dependency information as $\Delta_T = \{a_1, a_2\}, \{a_3\}$. For the example presented in Table I, $\Sigma_T = \{id : int, x : real\}$ and $\Delta_T = \{x\}$ (x represents the 1-D location). To model the location as a jointly distributed 2-D attribute, $\Sigma_T = \{id : int, x : real, y : real\}$ and $\Delta_T = \{x, y\}$.

Consider the special case when all the attributes in a table T are jointly distributed (i.e. $\Delta_T = \{\Sigma_T\}$). This extreme case captures tuple uncertainty as the complete value of the tuple is uncertain. The joint pdf over the attributes implicitly represents a group of dependent tuples. In addition, we can define tuples which are continuous and thus an infinite number

of alternatives are possible for each tuple. This representation is more powerful that the tuple uncertainty models in which each tuple can only have a finite number of alternatives.

We allow the dependency information Δ_T to contain *phantom attributes* which are not present in Σ_T. These extra attributes and their corresponding joint distribution are needed for ensuring that the correlation information of the attributes that are projected out is not lost during projections (See Section III-B for more information). However, only the attributes in Σ_T are visible to the user.

Definition 1: A probabilistic tuple t of table $T(\Sigma_T, \Delta_T)$ is represented by values $t.a_j$ for all certain attributes a_j and pdf $f_t(S_i)$ for all sets of uncertain attributes $t.S_i \in \Delta_T$.

To be precise, let us define $X^t_{S_i}$ to be the random variable for an attribute set $t.S_i$. Thus, $f_t(S_i)$ returns a pdf function that is defined over $X^t_{S_i}$. That is, $f_t : S_i \rightarrow f(X^t_{S_i})$. In the rest of this paper, whenever we refer to $f_t(S_i)$, it is understood that we are referring to the underlying distribution $f(X^t_{S_i})$.

B. Partial pdfs

In traditional databases, NULL is used to represent unknown or missing data. We also use NULL values in our model to signify *missing attribute values*. However, there is another way of representing missing data. The semantics of these two approaches differ from each other. To illustrate this point, let us consider the example presented in Table IV. The first tuple has missing (unknown) values for attribute b and c. However, the presence of the tuple itself is certain as the probability $Pr(b, c)$ adds up to 1. The other approach for representing missing data uses a closed world assumption to represent unknown information with *partial pdfs*. The probability that the second tuple exists in the table is 0.8 ($=\sum Pr(b, c)$) and thus with 0.2 probability the tuple does not exist in the table. Although both these approaches signify missing data their probabilistic interpretations are quite different.

The usual definition of a pdf requires that it sums up (or integrates) to 1. We remove this restriction in our model in order to represent *missing tuples* with partial pdfs. The support for partial pdfs is crucial in our model to ensure that database operations such as selection are consistent with PWS. A partial pdf is a pdf where only the events associated with the existence of the tuple are explicitly represented. If the joint pdf of a tuple sums to x, then $1 - x$ is the probability that the tuple does not exist, under a closed world assumption. In this paper, we use the terms pdf and partial pdf interchangeably.

TABLE IV
EXAMPLE: MISSING ATTRIBUTES VALUES VS MISSING TUPLES

a	b	c	Pr(b, c)
1	2	3	0.8
	NULL	*NULL*	0.2
2	4	7	0.2
	4.1	3.7	0.6

C. History

As discussed in the previous section, we allow multiple attributes to be jointly distributed in our model. This flexibility makes the model very powerful in terms of data representation, by allowing *intra-tuple dependencies* (i.e. correlation between attributes). But for the model to be closed and correct under the usual database operations, we need to handle *inter-tuple dependencies* as well. History captures dependencies among attribute sets as a result of prior database operations. It is used to ensure that the results of subsequent database operations are consistent with PWS. This is described in more detail in Section III. A similar concept is used in many tuple uncertainty models to track correlations between tuples. [9] uses lineage and [14] uses factor tables to capture such dependencies. As we are interested in capturing historical dependencies between attributes of tuples, our concept of dependencies is different from this related work, which capture these dependencies on a per tuple basis.

We maintain the history of uncertain attributes by storing the top-level *ancestors* of each dependency set in a tuple. The function Λ maps each pdf $t.S$ of a tuple t, to a set of pdfs that are its ancestors.

Definition 2: For a newly inserted tuple t in table T, $\Lambda(t.S) = t.S$, $\forall S \in \Delta_T$. If a new pdf $t'.S'$ is *derived* from pdfs $t.S_i$ via a database operation, then $\Lambda(t'.S') = \bigcup_i \Lambda(t.S_i)$.

In other words, the ancestors are the base pdfs which are inserted in the database by the user. We assume that the base tuples are independent. All the *derived* attributes point back to the base pdfs from which they are derived.

Definition 3: If $\Lambda(t.S_1) \cap \Lambda(t.S_2) \neq \phi$, then the nodes $t.S_1$ and $t.S_2$ are said to be *historically dependent.*

Note that the deletion of a base tuple will cause dependency sets of its derived tuples to lose their ancestor information. Thus, while deleting a tuple from the base table, we first check if any other tuple in the database is referencing any dependency set within the tuple. If there is a reference, we delete the tuple but keep the dependency set and its pdf as a *phantom* node until its reference count falls to zero. Definition 2 assumes that the base tuples are historically independent. This is not limiting since a historical dependency between attribute sets of a base table, can be captured by creating a *phantom* ancestor and pointing the dependent attribute sets to this common phantom ancestor.

III. PROBABILISTIC OPERATIONS

We begin by defining some basic operations on pdfs that underly the implementation of the usual database operations for our model. These operators are not directly accessible by users. One of the strengths of our model is that correctness with respect to PWS is achieved by manipulating the pdfs. Next, we present the usual relational operations under our model. The section concludes with a discussion of new operators that directly operate on the pdfs and are available to users as extensions to SQL.

A. Preliminaries

Here we describe some basic operations that are needed to define the usual relational database operations.

`marginalize`(f, A): Given a pdf f over attributes A_f, and a subset of attributes $A \subseteq A_f$: the operation produces the pdf function f' over attributes A. This is done by marginalizing the distribution f, i.e. $f' = \int_{A_f - A} f$. For discrete distributions, the integral is replaced by sum. It is easy to show the consistency wrt PWS because the probability of an event is the sum of probabilities of all the possible worlds in which the event occurs.

`floor`(f, F): Given a pdf f, on a domain D and given a subset $F' \subseteq D$, operation `floor`(f, F) produces a new pdf f' such that values of $f'(x) = 0$ whenever $x \in F$ and $f'(x) = f(x)$ otherwise. This `floor` operation corresponds to a selection predicate. The values in F are those which do not pass the selection criteria and hence do not exist in the resulting pdf. Going by the PWS, this means that in the possible world where x takes the value in F, this tuple does not meet the selection criteria and hence it does not exist. Multiple `floor` operations can be successively applied over a pdf in any order and the result would be `floor`$(f, F_1 \cup ... F_k)$ regardless of the order in which they are applied.

The application of `floor` on a symbolic distribution (e.g. Gaus) will, in general, result in a non-standard partial pdf. This partial pdf could be potentially captured by a histogram representation. But, we can optimize the floor operation (and subsequent operations) significantly, if we store *symbolic floors* to represent the flooring operation along with the original (symbolic) distribution. Our model has built-in support for simple symbolic floors which result from some common selection predicates. To illustrate, if the distribution of an attribute x is given by Gaus(5,1) and we apply the selection predicate $x < 5$, the resulting pdf will be floored when $x \geq 5$ (and its value is given by Gaus(5,1) when $x < 5$). This resulting distribution is represented as [Gaus(5,1), Floor$\{[5, \infty]\}$] in our implementation.[1]

`product`(f_1, f_2): Given two pdfs f_1 and f_2 over attribute value sets S_1 and S_2 (in a given tuple t) respectively, the operation `product` gives their joint pdf f (over $S' = S_1 \cup S_2$). We have to consider the following two cases:

f_1 and f_2 are historically independent: In this case, $f(x) = f_1(x_1) f_2(x_2)$ where $x \in S_1 \times S_2$ and $x = (x_1, x_2)$. To illustrate, assuming the pdfs shown in Figure 2(a), (b) are historically independent, the result of performing the product operation is shown in Figure 2(c).

f_1 and f_2 are historically dependent: Let $t_j.N_j, 1 \leq j \leq m$ be the common ancestors of $t.S_1$ and $t.S_2$ (i.e. $t_j.N_j \in \Lambda(t.S_1) \cap \Lambda(t.S_2)$). Each $t_j.N_j$ represents the distribution of an attribute set (N_j) of a given tuple (t_j). Thus N_j denotes the set of attributes in $t_j.N_j$. We define $C_j = N_j \cap S'$ and $D_i = S_i - \bigcup C_j$, $i = 1$ or 2. Thus C_j is the set of attributes

[1]Similar implementation optimizations are possible for other operations presented in this paper. We skip their discussion in this paper due to space limitation.

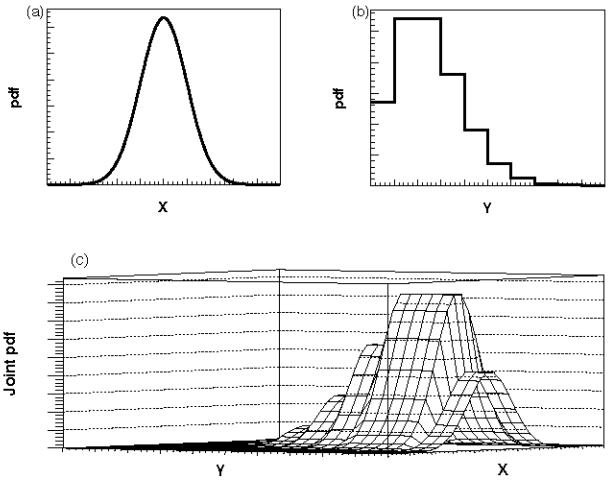

Fig. 2. Example of product operation

that the ancestor $t_j.N_j$ shares with either S_1 or S_2 . D_1 (D_2) is the set of attributes in S_1 (S_2) that are not shared with any common ancestor. Let X_S^t be the random variable for an attribute set $t.S$. Let x_S^t be an instance of X_S^t. With these notations, the joint pdf of resulting set $t.S'$ is:

$$f(x_{S'}^t) = \begin{cases} 0, & \text{if } f(x_{S_1}^t) \text{ or } f(x_{S_2}^t) = 0 \\ f(x_{D_1}^t) f(x_{D_2}^t) \prod_{j=1}^{m} f(x_{C_j}^{t_j}), & \text{otherwise} \end{cases}$$

where, $x_{S'}^t \in X_{D_1}^t \times X_{D_2}^t \times X_{C_1}^{t_1} \times X_{C_1}^{t_1} \ldots \times X_{C_m}^{t_m} \times X_{C_m}^{t_m}$

In other words, we first find the group of attribute sets (D_1, D_2 and $C_j, \forall j$) that are independent of each other. We can multiply the distributions of these nodes as they are independent. But, that would ignore any `floors` that were applied during database operations from ancestor nodes $t_j.N_j$ to $t.S_1$ or $t.S_2$. One potential solution is to keep track of all the operations and re-apply them[2] but we observe that we can infer the final floors from the distributions of $t.S_1$ and $t.S_2$. The regions where they were floored are the regions whose corresponding possible worlds did not "survive" the selection conditions. Thus, we propagate the floors of $t.S_1$ and $t.S_2$ to the joint distribution. This operator is used for defining selection and is further discussed in Section III-C. Note that this operator is associative and hence can be used over more than two pdfs as well.

B. Projections

Given a table T, we define $R = \Pi_A(T)$ as the table which contains a tuple t' corresponding to *each* tuple $t \in R$ ($t \rightarrow t'$), such that the resulting schema $\Sigma_R = A$. The new dependency information Δ_R can contain some of the attributes that are projected away. These attributes and their corresponding distributes are kept to ensure that we do not loose any floors associated with the projected out attributes.

[2]This method, though correct, is very inefficient and will not scale with database size and number of operations.

$\forall S_i \in \Delta_T$, where $S_i \cap A \neq \phi$ or $\int f_t(S_i) \neq 1$, we keep $S_i \in \Delta_R$. A number of optimizations are possible to reduce the number of extra attributes that are kept in Δ_R. For example, instead of the complete set S_i, we can keep a subset S_i' such that for each tuple, S_i' functionally determine S_i.

The history of the new sets is updated to history of sets from which they are derived i.e. $\forall t' \in R$ and $\forall S_k \in \Delta_R$ where $t \rightarrow t'$ and $S_k \subseteq S_i$ ($S_i \in \Delta_T$), we have $\Lambda(t'.S_k) = \Lambda(t.S_i)$.

Similar to other models for uncertain data, we do not address the issue of duplicate elimination in projections in this paper. This is because the concept of duplicate elimination for probabilistic data in general leads to complex historical dependencies. As part of our ongoing work, we are extending our model to address duplicate elimination.

C. Selections

Given a table T with attributes Σ_T and a boolean predicate $\Theta(A)$ defined over a subset of attributes A of table T, the result of the selection operator is $R = \sigma_{\Theta(A)}(T)$. If all the attributes in A are certain then we can simply use the "usual" definition of select operator to get the result. If not, selection will introduce new dependencies in the resulting set R, as explained below.

Case 1 All the attributes $a_i \in A$ are certain: The schema $\Sigma_R = \Sigma_T$ and the dependency information $\Delta_R = \Delta_T$. A tuple $t \in T$ maps to a tuple $t' \in R$ (i.e. $t \rightarrow t'$), if $\Theta(t.A)$ is true. That is, $t'.a_i = t.a_i$, $\forall$ certain a_i and, $f_{t'}(S_i) = f_t(S_i), \forall S_i \in \Delta_R$. The history is simply "copied over" for all the dependency sets i.e. $\forall S_i, \Lambda(t'.S_i) = \Lambda(t.S_i)$. As an example, the result of performing a selection $\sigma_{id=1}(T)$ on the relation T presented in Table I would give us a single tuple $t = [1, Gaus(20, 5)]$.

Case 2 At least one of the attributes $a_i \in A$ is uncertain: The schema $\Sigma_R = \Sigma_T$ and dependency information $\Delta_R = \Omega(\Delta_T \cup \{A\})$. The closure Ω is defined as follows:

Definition 4: Given a set system $\{S_1, S_2, \ldots, S_m\}$ representing a hyper-graph, the closure $\Omega(\{S_1, S_2, \ldots, S_m\})$ produces a set system $\{S_1', S_2', \ldots, S_{m'}'\}$ such that $S_1', S_2', \ldots, S_{m'}'$ represent the hyper-graph produced by merging all the connected components of $\{S_1, S_2, \ldots, S_m\}$.

To illustrate, if $\Delta_T = \{\{a, b\}, \{c, d\}, \{e, f\}\}$ and $A = \{b, c, g\}$ (g is certain), then $\Omega(\Delta_T \cup \{A\}) = \{\{a, b, c, d, g\}, \{e, f\}\}$. Note that the sets $\{a, b\}$ and $\{c, d\}$ were *merged* due to the condition on A. The dependency set $\{e, f\}$ was not affected as it is disjoint from A. Note that some of the certain attributes in T may become uncertain in R.

Let us assume that a tuple $t \in T$ maps to a tuple $t' \in R$ (i.e. $t \rightarrow t'$). For all the certain attributes a_j in R, we have $t'.a_j = t.a_j$ (i.e., they are copied over). For the dependency sets that were disjoint from A, we do not need to do anything special. For the merged sets, we need to evaluate the resulting pdf. Thus, for $\forall S_k \in \Delta_R$, we have the following cases:

Case 2(a) ($A \cap S_k = \phi$): This is the case when S_k does not share any attributes with the selection set A, and thus using Definition 4 and the fact that all $S_i \in \Delta_T$ are disjoint, we can see that S_k is derived from exactly one attribute set $S_i \in \Delta_T$, i.e. $f_{t'}(S_k) = f_t(S_i)$.

Case 2(b) ($A \cap S_k \neq \phi$): Using Definition 4 it is easy to see that ($A \subseteq S_k$). In this case, S_k can be potentially derived from multiple attribute sets $S_i \in \Delta_T$. These attribute sets S_i are the sets for which ($A \cap S_i \neq \phi$). Let us assume $f_i, 1 \leq i \leq n$ are their respective pdfs. S_k consists of all the attributes in such sets S_i and A. Let us assume that C is set of all certain attributes ($C \subset A$) and c is the value of C in t. We define the identify pdf f_0 over C as $f_0(c) = 1$ and 0 otherwise. Now, we can derive the resulting pdf of S_k by performing a `product` operation over $f_0, f_1, \ldots, f_m$ and `floor`ing the resulting pdf in the region where $\Theta(A)$ is false. If the pdf of S_k is completely floored (i.e. the resulting probability of the tuple becomes 0), we remove that tuple from the result.

Similar to the previous case, the histories of the new dependency sets are updated to the combined histories of sets from which they are derived i.e. $\forall t' \in R$ and $\forall S_k \in \Delta_R$ where $t \rightarrow t'$, we have:

$$\Lambda(t'.S_k) = \bigcup_{\forall S_i \subseteq S_k, S_i \in \Delta_T} \Lambda(t.S_i)$$

Consider the example shown in Table II. The probabilistic schema of that relation in our model would be represented as $\Sigma = (a : int, b : int)$ and $\Delta = \{\{a\}, \{b\}\}$. There are two tuples t_1 and t_2 in that relation with pdfs $f_{t_1}(\{a\}) = Discrete(0 : 0.1, 1 : 0.9)$ and $f_{t_1}(\{b\}) = Discrete(1 : 0.6, 2 : 0.4)$ (this notation represents a discrete pdf, whose parameters $x_i : y_i$ denote the probability y_i for value x_i). Similarly, we can write the pdfs of t_2 as $f_{t_2}(\{a\}) = Discrete(7 : 1.0)$ and $f_{t_2}(\{b\}) = Discrete(3 : 1.0)$. Applying a selection predicate $\sigma_{a<b}$ results in a table with schema $\Sigma = (a : int, b : int)$ and $\Delta = \{\{a, b\}\}$. This table consists of a single tuple t' with the joint distribution $f_{t'}(\{a, b\}) = Discrete(\{0, 1\} : 0.06, \{0, 2\} : 0.04, \{1, 2\} : 0.36)$. The history $\Lambda(t'.\{a, b\}) = \{t_1.\{a\}, t_1.\{b\}\}$.

Theorem 1: The new pdf generated by selection operation is consistent with PWS.

Proof: This follows from PWS consistency for the operators `product` and `floor`. The `product` operation on contributing pdfs results in a joint pdf which is consistent with the PWS semantics for all the non-zero values of the new pdf. Now, the various selection criteria can be considered as multiple applications of the `floor` operation which set the pdf to zero for all possible worlds where the corresponding attribute values do not pass the selection criteria. In these possible worlds, the tuple containing this pdf will not exist. Since operation `floor` can be applied in any order, one does not need to re-apply selection criteria which were already captured by some dependency set S_i. ■

D. Joins

The join of two tables $T_1 \bowtie_{\Theta(A)} T_2$ can be written as $\sigma_{\Theta(A)}(T_1 \times T_2)$. Thus, to define the semantics of joins, we can use the semantics of selection and cross-product. We have already seen selection, the cross-product $R = T_1 \times T_2$ is

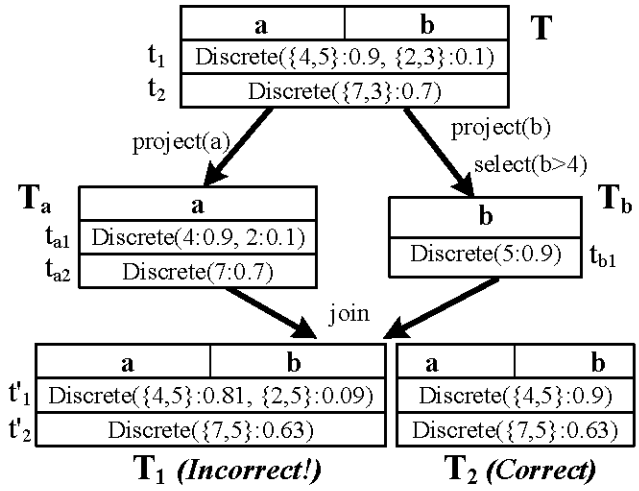

Fig. 3. Example illustrating histories

defined as follows. $\Sigma_R = \Sigma_{T_1} \cup \Sigma_{T_2}$ and $\Delta_R = \Delta_{T_1} \cup \Delta_{T_2}$. Let us assume a tuple $t \in R$ is derived from tuples $t_1 \in T_1$ and $t_2 \in T_2$ (i.e. $(t_1, t_2) \rightarrow t$). $\forall S_k \in \Delta_R$ and the corresponding $S_i \in \Delta_{T_c}, c = 1$ or 2 we have, $f_t(S_k) = f_{t_c}(S_i)$. Similarly, the history is also copied over for the new sets, $\Lambda(t'.S_k) = \Lambda(t_c.S_i)$.

Thus, conceptually joins are an application of cross-product followed by selection (as defined in Section III-C). The tuples that are produced as a result of join may contain some dependencies (implied by history Λ) which are not captured by the attribute dependencies (implied by Δ_T). We can, in principle, apply the algorithm explained in Section III-C to collapse the intra-tuple dependencies implied by Λ into Δ_T. This decision will not affect the correctness or the semantics of the operations defined in this section but will have a significant effect on performance. The definition of the operations in this section assumes a lazy merging of dependencies and evaluation of joint pdfs. In practice, a combination of these techniques can be used to improve performance. Thus, the decision of whether to merge the intra-tuple dependencies eagerly or lazily is left to the implementation.

Consider as an example, a table T with $\Sigma_T = (a : int, b : int)$ and $\Delta_T = \{\{a, b\}\}$ as shown in Figure 3. We perform operations $\Pi_a(T)$ and $\Pi_b(\sigma_{b>4}(T))$ to obtain the tables T_a and T_b (In this example, we do not need to keep the projected out attributes, as both the attributes a and b functionally determine each other in both the tuples). Clearly, $\Sigma_{T_a} = (a : int)$ and $\Delta_{T_a} = \{\{a\}\}$ for T_a; and $\Sigma_{T_b} = (b : int)$ and $\Delta_{T_b} = \{\{b\}\}$ for T_b. Now, if we join T_a and T_b *without* considering historical dependencies we would get an incorrect result T_1. The tuple $(2, 5)$ in t'_1 can never exist because it do not exist in any possible world corresponding to table T. Similarly, the probability of tuple $(4, 5)$ in T_1 is incorrect as the pdfs of t_{a1} and t_{b1} share common ancestor $t_1.\{a, b\}$ and thus the two events cannot be considered independent. Our model detects the historical dependency between tuples t_{a1} and t_{b1} and uses that information to correctly calculate the distribution of tuple t'_1 in the final table T_2 by considering the joint distribution of attributes a and b in T. In addition, as part of the tuple value $(2, 3)$ $(\in T)$ was floored in table T_b, we correctly floored that value in the distribution of $t'_1.\{a, b\}$.

The correctness of the project and join operations with respect to the possible world semantics follows from the correctness of the selection operation and are thus omitted. Given the definition and the correctness of the selection, project, and join operations, we obtain the following theorem.

Theorem 2: Our model is closed under selection, projection, and join operations.

E. Operations on Probability Values

We also allow queries based on the probability values of the tuples in our model. One example of such queries are threshold queries. Given a table T with probabilistic schema (Σ_T, Δ_T), a threshold query $R = \sigma_{Pr(A)>p}(T)$, where $A \subseteq \Sigma_T$ and p is the probability threshold, returns all tuples whose probability over the attribute set A is greater than p. As the operations on probability values act on the probabilistic model instead of a *possible world*, the possible worlds semantics described in Section I is not be used to define the semantics of these operations.

In general, consider the boolean predicate given by $\Theta(S)$, where $S = \{Pr(s_1), Pr(s_1), \ldots, Pr(s_m)\}$ and $s_i \subseteq \Sigma_T$. The result R of applying this selection on T consists of all tuples $t \in T$ such that t satisfies $\Theta(S)$. The semantics of this operation and effect on histories is similar to Case 1 defined in Section III-C.

IV. Experimental Evaluation

We have implemented our model in *Orion*, a publicly available extension to PostgreSQL that provides native support for uncertain data [8]. This system not only allows us to validate the accuracy of our methods in a realistic runtime environment, it also gives additional insight into the overall effect our techniques have on probabilistic query processing in an industrial-strength DBMS. The following experiments were conducted on a Sun-Blade-1000 workstation with 2 GB RAM, running SunOS 5.8, PostgreSQL 8.2.4, and Orion 0.2.

Using a series of synthetically generated datasets, we explore the performance and accuracy of our model's operations over pdfs. Each dataset consists of random "sensor readings," using the schema `Readings(rid, value)`. The uncertain pdfs (e.g. reported from the sensors) are Gaussians, with their means distributed uniformly from 0 to 100, and their standard deviations distributed normally using $\mu = 2$ and $\sigma = 0.5$. We also generate numerous range queries, with midpoints distributed uniformly between 0 and 100, but with interval lengths distributed normally using $\mu = 10$ and $\sigma = 3$.

For simplicity, we omit the initial results of evaluating pdfs symbolically because they produce no approximation error and incur negligible overhead. Instead, our results focus on the relative performance of approximating symbolic pdfs with histograms as opposed to discrete sampling. Although it's obvious

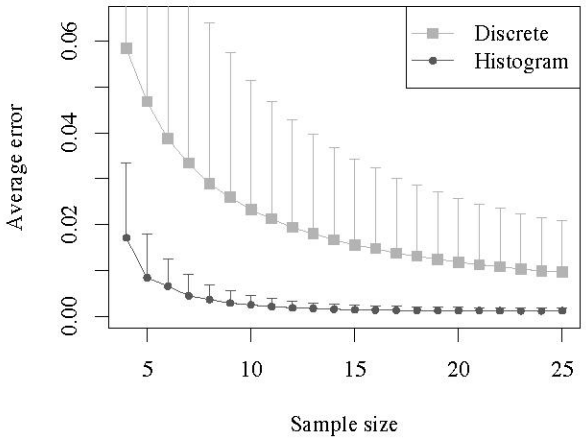

Fig. 4. Accuracy vs Sample Size

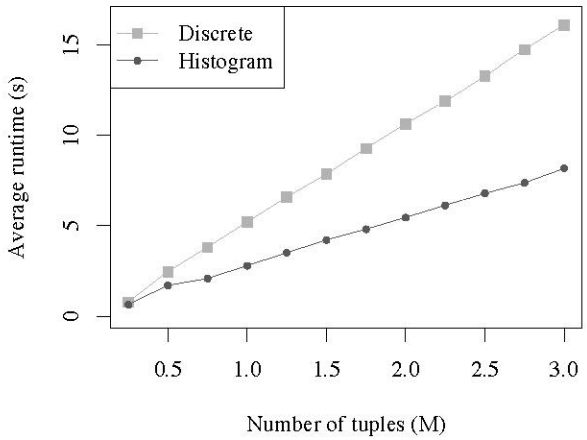

Fig. 5. Performance of Discretized PDFs

theoretically that histograms will generally outperform discrete representations, we wish to quantify the observed difference of these two approximations in our actual implementation.

A. Accuracy vs Sample Size

The first experiment shows the average error when answering range queries over histogram and discrete approximations of symbolic pdfs. We first discretize our dataset of random Gaussian pdfs, varying the number of sample points. Figure 4 shows the average approximation error of the cdf values returned at each sample size. The standard error over these averages is negligible. As expected, the histogram representation outperforms the discrete, even in the worst case (not shown). With only five sampling points, the accuracy is around ± 0.01 probability mass. A discrete approximation requires over twenty-five sampling points, which greatly increases the size of each tuple and thus the overall I/O cost. Of course, a symbolic representation is both ideal in storage size and accuracy.

We also show the standard deviation of the error values themselves, at each sample size, plotted only in the positive direction for clarity. As expected, a discrete representation has a considerably higher variance in approximation error than a histogram. Sometimes the error is quite large, for example in boundary cases when the query barely misses a discrete point. Continuous representations (including histograms) avoid this issue altogether because they can accurately estimate probability mass at arbitrary points. The difference in error is likely to be even greater in more complex pdfs.

B. Performance of Discretized PDFs

For this experiment, we compare the performance of the aforementioned approximate representations. We fix the number of histogram bins at five and the number of discrete sample points at twenty-five, in order to compare runtimes at an equivalent level of accuracy. As shown in Figure 5, discretizing the data not only takes additional processing time, but also incurs more disk reads, yielding a steeper rise in cost. Runtimes for the symbolic representation are just under the five-bin histogram times, but we do not show these here since they give an even higher level of accuracy.

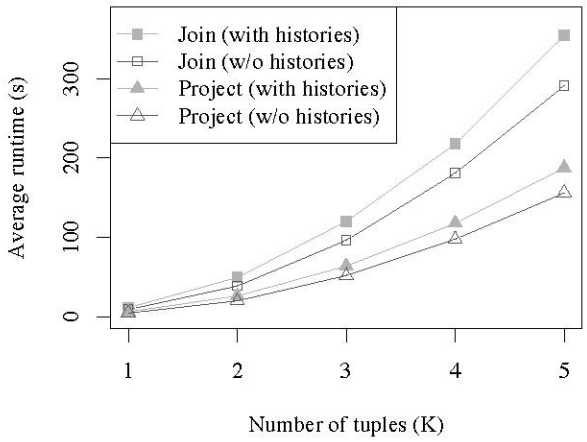

Fig. 6. Overhead of Histories

C. Overhead of Histories

The final experiment shows the overall performance of the implementation of our proposed model inside PostgreSQL. We run two types of queries: joins over range queries (which involve floors and products), and projections of the resulting correlated data (triggering a collapse of the 2D pdfs). Figure 6 compares the average runtime of these queries with and without the overhead of maintaining histories for correctness. Note that ignoring histories will result in incorrect answers. The overhead shown in this figure ranges between 5-20%. Thus, although the proposed model is complex, it is efficient to implement and we pay a small overhead for correctness.

V. Related Work

Barbará *et al.* [12] and Dey *et al.* [15] proposed the first of the probabilistic models. Building on their work, many robust models for managing tuple uncertainty have been proposed recently. A significant challenge when modeling uncertain data is tracking arbitrary correlations both within and between tuples. These dependencies are not only present in real-world data, they are more commonly introduced by applying operations to independent base data. Benjelloun *et al.* have proposed a novel technique that combines uncertainty with data lineage to solve this problem [9]. The ProbView system [16] took a similar approach by propagating the formulas necessary to evaluating the resulting probabilities. Sen *et al.* have more recently proposed an alternative approach to represent tuple correlations using probabilistic graphical models [14]. They use factored representations of the relations to represent their dependencies. Antova *et al.* developed a compact representation called world-set decompositions which captures the correlations in the database by representing the finite sets of worlds [17]. Dalvi *et al.* introduced safe plans [18], [10] in an attempt to avoid probabilistic dependencies in queries.

An important area of uncertain reasoning and modeling deals with fuzzy sets [1]. The work on fuzzy models is not immediately related to our work as we assume a probabilistic model.

None of the aforementioned tuple uncertainty models can fully support continuous probability distributions. They suffer from loss of accuracy and efficiency. Parallel to this modeling effort, there has also been a lot of recent work on querying and indexing pdf attributes in databases [2], [3], [4], [5], [6], [7].

In previous work, we have proposed preliminary models for attribute uncertainty that overcome these limitations [19], [20]. We have also studied indexing methods for attribute uncertainty, both for continuous [6] and categorical [7] distributions. Apart from our work, there has been other work by [2], [3], [5] on indexing pdfs. However, none of this work considers PWS and hence its appeal is limited to solving specific problems. In this paper we have shown the first model for handling pdfs which can pave the way for more complex and useful operations involving pdfs.

VI. Conclusion

We have presented a new model for handling arbitrary pdf (both discrete and continuous) attributes natively at the database level. Our approach allows a more natural and efficient representation and implementation for continuous domains. The model can handle arbitrary intra- and inter-tuple correlations. We show that our model is complete and closed under the fundamental relational operations of selection, projection, and join. In our previous work we have developed Orion – an extension of PostgreSQL that provides native support for attribute uncertainty with procedural semantics. We have extended Orion to support our new model. The experiments performed in Orion show the effectiveness and efficiency of our approach.

Acknowledgments

This work was supported by NSF grants IIS 0534702, IIS 0415097, CCF 0621457, AFOSR award FA9550–06–1–0099, ARO grant DAAD19–03–1–0321 and by the Research Grants Council of Hong Kong CERG PolyU 5138/06E. We would also like to thank the Trio group at Stanford University for alerting us to an inconsistency in an earlier version of this model.

References

[1] J. Galindo, A. Urrutia, and M. Piattini, *Fuzzy Databases: Modeling, Design, and Implementation*. Idea Group Publishing, 2006.

[2] V. Ljosa and A. Singh, "APLA: Indexing arbitrary probability distributions," in *Proceedings of 23rd International Conference on Data Engineering (ICDE)*, 2007.

[3] C. Böhm, A. Pryakhin, and M. Schubert, "The gauss-tree: Efficient object identification in databases of probabilistic feature vectors," in *Proceedings of International Conference on Data Engineering (ICDE)*, 2006.

[4] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Querying imprecise data in moving object databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, 2004.

[5] A. Faradjian, J. Gehrke, and P. Bonnet, "GADT: A probability space ADT for representing and querying physical world," in *Proceedings of International Conference on Data Engineering (ICDE)*, 2002.

[6] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. Vitter, "Efficient indexing methods for probabilistic threshold queries over uncertain data," in *Proceedings of International Conference on Very Large Data Bases (VLDB)*, 2004.

[7] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, and S. Hambrusch, "Indexing uncertain categorical data," in *Proceedings of 23rd International Conference on Data Engineering (ICDE)*, 2007.

[8] "http://orion.cs.purdue.edu/," 2006.

[9] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom, "ULDBs: Databases with Uncertainty and Lineage," in *Proceedings of the 32nd International Conference on Very Large Data Bases*, 2006, pp. 953–964.

[10] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu, "MYSTIQ: a system for finding more answers by using probabilities," in *Proceedings of ACM Special Interest Group on Management Of Data*, 2005.

[11] A. Deshpande and S. Madden, "MauveDB: supporting model-based user views in database systems," in *Proceedings of ACM Special Interest Group on Management Of Data*, 2006, pp. 73–84.

[12] D. Barbará, H. Garcia-Molina, and D. Porter, "The management of probabilistic data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 4, no. 5, pp. 487–502, 1992.

[13] J. Y. Halpern, *Reasoning about Uncertainty*. The MIT Press, 2003.

[14] P. Sen and A. Deshpande, "Representing and querying correlated tuples in probabilistic databases," in *Proceedings of 23rd International Conference on Data Engineering (ICDE)*, 2007.

[15] D. Dey and S. Sarkar, "A probabilistic relational model and algebra," *ACM Transactions of Database Systems*, vol. 21, no. 3, pp. 339–369, 1996.

[16] L. Lakshmanan, N. Leone, R. Ross, and V. Subrahmanina, "Probview: A flexible probabilistic database system," *ACM Transactions on Database Systems*, vol. 22, no. 3, pp. 419–469, 1997.

[17] L. Antova, C. Koch, and D. Olteanu, "10^10^6 worlds and beyond: Efficient representation and processing of incomplete information," in *Proceedings of 23rd International Conference on Data Engineering (ICDE)*, 2007.

[18] N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," in *Proceedings of International Conference on Very Large Data Bases (VLDB)*, 2004.

[19] R. Cheng, S. Singh, and Prabhakar, "U-DBMS: A database system for managing constantly-evolving data," in *Proceedings of Very Large Databases Conference (VLDB)*, 2005.

[20] R. Cheng, S. Singh, S. Prabhakar, R. Shah, J. Vitter, and Y. Xia, "Efficient join processing over uncertain data," in *Proceedings of ACM 15th Conference on Information and Knowledge Management (CIKM)*, 2006.