Uncertainty Handling in a Tabular Representation

Yiping Li, Jianwen Chen, and Ling Feng

***Abstract*—Nowadays a lot of uncertainty appears due to the burst of data on the internet. This challenges the traditional data processing methods because of its volume and variety. Such uncertainty is hard to be represented in simply continuous distribution functions when it is too hard or complicated to be obtained. Considering the realistic situations that people may be interested in queries falling into a scope rather than a specific value, we present a new data model, called N-DB model, where the attribute value is represented with a tabular form in an increasing order, denoted *N*-table (*i.e.*, a set of ordered pairs). This model can deal with queries in a scope efficiently, such as "movies with reputation level bigger than 5". We modify the relational algebra, including the aggregation query, and show the query processing in a running example. We also define an uncertain measurement (α-precision) to measure the precision and information of the *N*-table, which can be used in the computation of precision requirements. Through experimental results, we find queries can be evaluated efficiently by searching in the *N*-tables, and are returned with confidence intervals.**

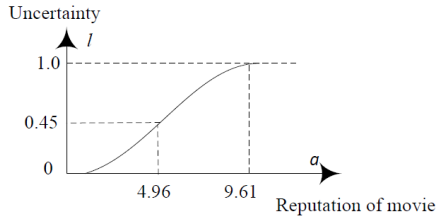
***Index Terms*—Uncertainty handling, uncertain databases, queries.**

I. Introduction

Large volumes of data spring up in recent years, such as microblog, mobile locations. Such data contains rich information and may be captured from different equipment, so the handling of this data may show the property of uncertainty. Querying over imprecise data seems to be urgent affairs. Among them, one kind of queries is found to appear quite frequently in searching for useful information, that is, queries falling into a scope. For example, movies may be scored by users in online video site. Users may be interested in the movie reputation before watching a movie. To obtain this information, they usually implement queries in the form of "movies with reputation level bigger than 5" rather than "movies with reputation level exactly 5". In the application of online video or shopping site, queries, in the form of "less than 5, between 3 and 5, bigger than 5", are more useful in helping people to find the appropriate information. Focused on this problem, we propose a new data model, called N-DB model, to handle uncertainty in imprecise databases and implement query processing.

In earlier work of queries over continuous attribute values, the distribution of an uncertain variable is presented with a probabilistic density function, which is usually uniform distribution or normal distribution. However, an uncertain variable in realistic situations is more complicated than the common distribution functions. The distribution can be quite irregular which is hard to be expressed by one single distribution function. For example, the reputations of movies can be quite different from each other, because the users who mark the reputations are different in the dimension of number and quality, and the reputation values can also change with the flow of users. Therefore, it results in an irregular distribution of reputation values. The presentation of *N*-table can help the research on this problem.

As shown in Fig. 1, the tabular representation of the uncertain attribute *Reputation* of a movie satisfies a partial property, corresponding to *N* points in the uncertain distribution curve. One pair (0.45 ≤ 4.96) means that the confidence of *Reputation* ≤4.96 is 0.45.



N-Table(N=50) depicting uncertain reputation of a movie (nt_1)

Uncertainty (l)	0.01	0.02	...	0.45	0.46	...	1.0
Reputation (a)	≤ 2.84	≤ 3.51	...	≤ 4.96	≤ 5.07	...	≤ 9.61

Fig. 1. An example of *N*-table.

The materialized storage of attribute values is convenient in the query process. Queries are evaluated over the *N*-tables by searching the nearest values instead of over probabilistic density functions by integration computation. Different from traditional probabilistic databases, each query result is returned along with a confidence interval rather than an exact probability value. *N*-tables are provided by users, corresponding to *N* points in the uncertainty distribution curve. Furthermore, we present a measure, called α-precision, to represent the precision and information of *N*-tables. Through the measurement, the precision of *N*-tables can be gained and adjusted according to requirements of queries.

The paper is organized in this way. In Section II, we introduce some earlier work related to uncertainty handling in databases. Then we detail our tabular representation model in Section III. Evaluation is implemented in Section IV, and we give concluding discussions in Section V.

II. Related Work

In dealing with uncertainty in the databases domain, a lot of previous work has been done [1]-[6]. In probabilistic databases, the attribute-wise uncertainty is represented with a set of discrete values with separate probabilities or a random variable with a probabilistic density function (PDF) [7]. Dalvi and Dan Suciu [8] study the query evaluations over probabilistic databases, and find some query complexity to be #P-complete. A lot of probabilistic database systems are

Manuscript received April 5, 2014; revised June 5, 2014.

The authors are with Tsinghua University, Beijing 100084, China (e-mail: liyp09@tsinghua.edu.cn).

presented, such as ORION [9], MystiQ [10], Trio [11], Urank [12], and MayBMS [13]. In [14], L. V. S. Lakshmanan, N. Lenoe, R. Ross, and V. S. Subrahmanian present the concept of probability range and corresponding possible worlds in the new ProbView database system, where attributes may take values from several alternatives with separate probability ranges, such as {(700, [0.4,0.7]), (800, [0.5, 0.9])}.

Fuzzy theory [15]-[17] is another way to deal with uncertainty, especially in handing the vagueness of human language. It does well in the label classification problems. For example, the vague concept "about 7 discount" can be represented with a membership function, such as {0/3, 0.2/4, 0.8/6, 1.0/7, 0.6/8}. In this membership function, the 4 discount satisfies the label "about 7 discount" with a degree of 0.2.

There are several approaches to model data vagueness in fuzzy databases. One approach [18] considers a relation as a fuzzy set which includes each tuple as an element with a value representing its grade of membership. Another approach [19]-[21] considers the attribute level vagueness, where each attribute is represented as a possibility distribution. A third approach [18], [22] combines the above two approaches. Some survey works are done for uncertainty handling in fuzzy databases [23]-[27].

Besides, there are also some works using other uncertain handling theories. A Monte Carlo based uncertain database was presented in [28], and some characteristic values (e.g. mean and variance of a normal function) are stored associated with attributes instead of pre-stored probabilities. Evidence oriented databases were also presented in 1990s by Lee [29], where the appearance probability of a tuple is represented with a confidence value [belief, plausibility]. Based on fuzzy theory, fuzzy databases [18], [20] have been proposed to support data vagueness since 1980s in parallel to probabilistic databases.

III. Tabular Representation Model

Based on the tabular representation of uncertain information, we present the NDB model to deal with attribute-wise and tuple-wise uncertainty in a running example of movies' discount and reputation.

A. Definitions and Basic Notions

As shown in Table I, different from traditional databases, a relation in N-DB usually consists of three parts: attributes with certain values (*MID* and *Name*), attributes with uncertain values (*Discount* and *Reputation*), and Uncertainty Interval (U). *N*-table (e.g. nt_1, nt_2, nt_3, and nt_4) is used to represent an attribute's uncertain value, where a set of ordered pairs in an increasing order (Fig. 1) are used to describe the uncertain distribution, denoting the attribute-wise uncertainty. Besides, an extra column U (Uncertainty Interval) is used to represent the confidence interval of the corresponding tuple, which is tuple-wise uncertainty.

Definition 1. N-table is a set of ordered pairs $nt = \{(l_1, \le a_1), (l_2, \le a_2), \cdots, (l_n, \le a_n)\}$ corresponding to n points in the uncertain distribution curve as shown in Fig. 1, $0 \le l_1 \le l_2 \cdots \le l_n = 1.0$, $0 \le a_1 \le a_2 \cdots \le a_n$. Each pair $(l_i, \le a_i)$ represents the probability that the attribute value is less than a_i is l_i, i.e., prob. (attribute value $\le a_i$) = l_i. For simplicity, the symbol $\le$ before a_i in the *N*-table is usually omitted. $|nt| = n$ is called the size of the N-table. We can use $nt(a_i)$, $nt(l_i)$, and $nt(a_i, l_i)$ to represent the value of a_i, l_i, and a pair (a_i, l_i) in nt separately, and $\mathrm{nt}(A) = \{nt(a_i) | i = 1, \cdots, n\}$, $nt(L) = \{nt(l_i) | i = 1, \cdots, n\}$. Besides, the uncertainty increment $\Delta l_i = l_i - l_{i-1}$, and the attribute value increment $\Delta a_i = a_i - a_{i-1}$, $1 \le i \le n$, $l_0 = 0$, $a_0 = 0$.

Definition 2. Uncertainty interval (*U*) is represented in the form $[u_l, u_u]$ $(u_l \le u_u)$ along with initial and resulting tuples. $w = u_u - u_l$ is the uncertainty interval width. Users may require this value to be less than a precision threshold that they can accept.

TABLE I: A Running Example

(A) Uncertain Relation *MovieInfo*

MID	*Name*	*Discount*	*U*
*t*1	Iron Man	nt_1	[1,1]
*t*2	Thor	nt_2	[1,1]

(B) Uncertain Relation *MovieRepu*

MID	*Name*	*Reputation*	*U*
*r*1	Iron Man	nt_3	[1,1]
*r*2	Thor	nt_4	[1,1]

(C) *N*-table of Thor's Discount(nt_2)

Uncertainty(l)	0.01	⋯	0.65	0.69	⋯	0.86	0.88	⋯	1.0
Discount(a)	1.02	⋯	2.84	3.01	⋯	4.75	5.09	⋯	8.24

(D) *N*-table of Iron Man's Reputation(nt_3)

Uncertainty(l)	0.01	⋯	0.80	0.81	⋯	1.0
Reputation(a)	2.54	⋯	6.91	7.00	⋯	9.02

(E) *N*-table of Thor's Reputation(nt_4)

Uncertainty(l)	0.05	⋯	0.80	0.85	⋯	1.0
Reputation(a)	1.47	⋯	6.82	7.42	⋯	9.52

At the beginning, each initial tuple in databases is assigned a default uncertainty interval [1.0, 1.0], as shown in Table I. In the querying process, each resulting tuple is returned along with an uncertainty interval after searching and computing in the *N*-table.

Definition 3. α-precision measures the precision of the *N*-table, that is, the error size. It is gained by taking the maximum uncertainty increment between two adjacent points in a *N*-table, and is helpful in computing the precision requirements.

$$\alpha(nt) = \max_{i=1}^{|nt|} \Delta l_i, 0 \le \alpha \le 1.0 . \quad (1)$$

The smaller the α-precision value is, the smaller the uncertainty interval width may be. It means that N-table that takes small α-precision value can result in answers with high precision requirements.

B. Relational Algebra

The relational algebra in N-DB model is quite like the traditional databases, except the computation involving in the *N*-tables and uncertainty intervals. Among them, two

N-tables nt_1 and nt_2 are considered to be equal if and only if $|nt_1| = |nt_2|$, $nt_1(a_i) = nt_2(a_i)$, $nt_1(l_i) = nt_2(l_i)$, $i = 1, \cdots, |nt_1|$. Besides, the confidence intervals of resulting tuples are also updated.

1) Union

In the union operation, the uncertainty interval corresponding to each resulting tuple should also be computed. If two tuples t_1 and t_2 are identical with separate *U* values $[u_{l_1}, u_{u_1}]$ and $[u_{l_2}, u_{u_2}]$, then they are merged to one result tuple with uncertainty interval

$$\begin{aligned} U(t_1 \vee t_2) &= [u_{l_1}, u_{u_1}] \vee [u_{l_2}, u_{u_2}] \\ &= [u_{l_1} + u_{l_2} - u_{l_1} \times u_{l_2}, u_{u_1} + u_{u_2} - u_{u_1} \times u_{u_2}]. \end{aligned} \tag{2}$$

2) Intersection

Similarly, if two tuples t_1 and t_2 with separate *U* values $[u_{l_1}, u_{u_1}]$ and $[u_{l_2}, u_{u_2}]$ result in one answer, then the corresponding uncertainty interval is

$$U(t_1 \wedge t_2) = [u_{l_1}, u_{u_1}] \wedge [u_{l_2}, u_{u_2}] = [u_{l_1} \times u_{l_2}, u_{u_1} \times u_{u_2}]. \tag{3}$$

3) Selection

Selection operation on uncertain attributes includes two types: $A \le c$ and $A \ge c$ (*A* is the attribute name, and *c* is a constant). For each tuple, we search for *c* in the N-table of uncertain attribute until we find two adjacent points (a_k, l_k) and (a_{k+1}, l_{k+1}), satisfying $a_k \le c \le a_{k+1}$, $1 \le k \le n-1$. Then the resulting tuple of $A \le c$ selection is returned along with the uncertainty interval $[l_k, l_{k+1}]$, and the resulting tuple of $A \ge c$ selection is returned with the uncertainty interval $[1-l_{k+1}, 1-l_k]$.

4) Projection

If two tuples t_1 and t_2 are projected to the same resulting tuple on the projected attributes with separate U values $[u_{l_1}, u_{u_1}]$ and $[u_{l_2}, u_{u_2}]$, then

$$\begin{aligned} U(t_1 \vee t_2) &= [u_{l_1}, u_{u_1}] \vee [u_{l_2}, u_{u_2}] \\ &= [u_{l_1} + u_{l_2} - u_{l_1} \times u_{l_2}, u_{u_1} + u_{u_2} - u_{u_1} \times u_{u_2}]. \end{aligned} \tag{4}$$

5) Join

If two tuples t_1, t_2 with separate U values $[p_{l_1}, p_{u_1}]$, $[p_{l_2}, p_{u_2}]$ can result in one resulting tuple through the join operation, then the resulting uncertainty interval is

$$\begin{aligned} U(t_1 \wedge t_2) &= [u_{l_1}, u_{u_1}] \wedge [u_{l_2}, u_{u_2}] \\ &= [u_{l_1} \times u_{l_2}, u_{u_1} \times u_{u_2}]. \end{aligned} \tag{5}$$

C. Aggregate Query

The aggregation query over uncertain attributes in N-DB needs computation among N-tables.

1) Max/Min

The *Max* operation over a set of *N*-tables $(nt_1, nt_2, \cdots, nt_m)$ also returns a *N*-table as the result. Firstly, to make the attribute values in different *N*-tables to be identical and comparable, we populate data in nt_j $(j = 1, \cdots, m)$ to an expanded form nt_j'. For each $a_i \in \bigcup_{j=1}^{m} nt_j(A)$,

$$nt_j'(a_i, l_i) = \begin{cases} nt_j(a_i, l_i) \ if \ a_i \in nt_j(A) \\ (a_i, nt_j(l_k)) \\ where \ nt_j(a_k) \le a_i \le nt_j(a_{k+1}) \end{cases} if \ a_i \notin nt_j(A) \tag{6}$$

From this equation, we know the size of nt_j' is $|\bigcup_{j=1}^{m} nt_j(A)|$. It populates the missing attribute values in nt_j by taking the uncertainty level of the adjacent points. Besides, the uncertainty level takes 0 if a_i is less than $nt_j(a_1)$, and takes 1 if a_i is bigger than $nt_j(a_{|nt_j|})$.

Then the computation of max aggregation over the expanded *N*-tables is shown in Fig. 2. The max attribute value is a_i if and only if nt_1 takes values less than $a_i, \cdots$, and nt_m takes values less than a_i, *i.e.*, $nt_1'(l_i) nt_2'(l_i) \ldots nt_m'(l_i)$. The computation of min aggregation is similar.

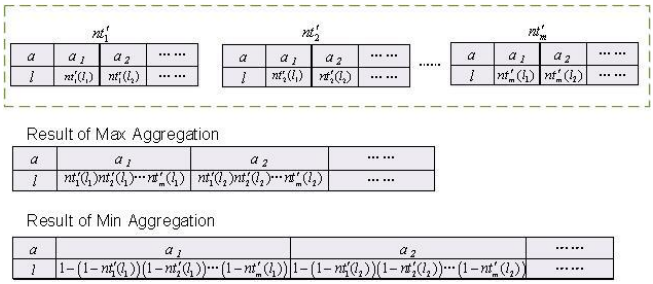


Fig. 2. Max and min aggregation.

2) SUM

For *SUM* operation, we use a recursion method to generate results which is also a N-table. We sum the *N*-tables one by one, and use nt_{sum_j} to represent the results of summing the first *j* *N*-tables. Let $nt_{sum_1} = nt_1 \bullet L_{sum_j(a)}$ denote the uncertainty that the attribute value in nt_{sum_j} is less than *a*. Then

$$\begin{aligned} & L_{sum_j}(x) \\ &= \sum_{i=1}^{|nt_j|} P(nt_j(a_{i-1}) \le A \le nt_j(a_i)) L_{sum_{j-1}}(x - nt_j(a_i)) \\ &= \sum_{i=1}^{|nt_j|} nt_j(\Delta l_i) L_{sum_{j-1}}(x - nt_j(a_i)) \end{aligned} \tag{7}$$

3) Average

The computation process of the *Average* operation is

similar to the *SUM* operation. $L_{avg_j(a)}$ denotes the uncertainty that the attribute value in nt_{avg_j} is less than *a*, then

$$L_{avg_j}(x) = \sum_{i=1}^{|nt_j|} P(nt_j(a_{i-1}) \le A \le nt_j(a_i)) L_{avg_{j-1}}(\frac{x - nt_j(a_i)}{j-1}) \quad (8)$$

$$= \sum_{i=1}^{|nt_j|} nt_j(\Delta l_i) L_{avg_{j-1}}(\frac{x - nt_j(a_i)}{j-1})$$

D. Query Processing

In the following, we show the query processing of the model.

Query 1: "*Find the movies whose discount is not greater than 5 discount*."

SELECT *Name* FROM *MovieInfo*

WHERE *Discount* ≤ 5

For Iron Man in Table I, we need to search the *N*-table of nt_1 (*i.e.*, Fig. 1) to find the probability that its discount isn't greater than 5 discount. We find 5 discount is between the point (0.45, 4.96) and (0.46, 5.07), therefore Iron Man's discount isn't greater than 5 discount within the uncertainty interval [0.45, 0.46]. Similarly, the uncertainty interval of Thor can also be found from Table I. The query results are shown in Table II.

Query 2: "F*ind the restaurants whose discount is between 3 and 5*."

SELECT *Name* FROM *Restaurant*

WHERE *Discount* $\ge$ 3 AND *Discount* ≤ 5

For Iron Man, according to Fig. 1, we find the uncertainty interval of *Discount* ≤ 3 and *Discount* ≤ 5 to be [0.01, 0.02] and [0.45, 0.46] separately. Then the uncertainty interval for the query result can be gained by the latter interval minus the former, i.e. [0.45-0.02, 0.46-0.01]=[0.43, 0.45]. The query results are shown in Table II.

TABLE II: RESULT OF QUERY 1 AND QUERY 2

(A) RESULT OF QUERY 1

Name	*U*
Iron Man	[0.45,0.46]
Thor	[0.86,0.88]

(B) RESULT OF QUERY 2

Name	*U*
Iron Man	[0.43,0.45]
Thor	[0.17,0.23]

Query 3: "*Find the movies whose discount isn't greater than 5 and reputation isn't less than 7*."

SELECT *MovieInfo.Name* FROM *MovieInfo, MovieRepu*

WHERE *MovieInfo.Name = MovieRepu.Name*

AND *Discount* $\le$ 5 AND *Reputation* ≥ 7

Take Iron Man for example. From Table I, we need to search the nt_1 (Fig. 1) to find the probability that Iron Man's discount isn't greater than 5. In nt_1, 5 discount is between the point (0.45, 4.96) and (0.46, 5.07), then the query *Discount* ≤ 5 is returned with the uncertainty interval [0.45, 0.46] for Pizza Hut. Similarly, from nt_3, we find the probability that its reputation isn't less than 7 turns out to be 1-0.81=0.19.We consider restaurants' discount and reputation to be independent. Then the query result is returned with the uncertainty interval [0.45 $\times$ 0.19, 0.46 $\times$ 0.19] = [0.0855, 0.0874], as shown in Table III.

Query 4: "*Find the movies whose discount isn't greater than 5 or reputation isn't less than 7*."

SELECT *Name* FROM *MovieInfo, MovieRepu*

WHERE *MovieInfo.Name = MovieRepu.Name*

AND *Discount* $\le$ 5 OR *Reputation* ≥ 7

The query result can be returned with the uncertainty interval as shown in Table III. Take Iron Man for example, [0.45, 0.46] $\vee$ 0.19 = [0.45 + 0.19 – 0.45 $\times$ 0.19, 0.46 + 0.19 -0.46 $\times$ 0.19] = [0.5545,0.5626].

TABLE III: RESULT OF QUERY 3 AND QUERY 4

(A) RESULT OF QUERY 3

Name	*U*
Iron Man	[0.0855,0.0874]
Thor	[0.129,0.176]

(B) RESULT OF QUERY 4

Name	*U*
Iron Man	[0.5545,0.5626]
Thor	[0.881,0.904]

Query 5: "*Find the max discount of movies*."

SELECT *Max(Discount)* FROM *MovieInfo*

For example, we can find the two pairs (0.01, 2.84) and (0.65, 2.84) from nt_1 and nt_2, thus there exists one pair (0.0065, 2.84) in the resulting *N*-table.

E. Precision Requirements

As shown in the above query examples, the results are usually returned with an uncertainty interval which represents the truth degree. If this interval is too wide, it provides little information for users. Therefore, the uncertainty interval width (*w*) should be constrained to satisfy users' precision requirement threshold (α_h), $w \le \alpha_h$. In the following, we show the relationships of α-precision and the result width w in the $\wedge$ and $\vee$ operations.

$$t_1 \wedge t_2$$

We assume the uncertainty interval of tuples t_1 and t_2 to be $U(t_1) = [u_{l_1}, u_{u_1}]$ and $U(t_2) = [u_{l_2}, u_{u_2}]$ separately. Let *A* denote an uncertain attribute, then t_1 and t_2 may be 1) initial tuples, $t_1.A = nt_1$, $t_2.A = nt_2$, or 2) gained through the selection operation ($A \le c$ or $A \ge c$) over initial tuples t_1' and t_2' separately, and $t_1'.A = nt_1$, $t_2'.A = nt_2$. For the first case 1), t_1 and t_2 are initial tuples, then $w_1 = u_{u_1} - u_{l_1} = 0 \le \alpha(nt_1)$, $w_2 = u_{u_2} - u_{l_2} = 0 \le \alpha(nt_2)$. For the second case 2), as we know, α-precision is the maximum uncertainty increment between two adjacent points in a *N*-table, while the uncertainty interval of resulting tuples in the selection operation is usually gained by taking the uncertainty values

of two adjacent points. Therefore, the uncertainty interval width of resulting tuple in the selection operation must be less than the α -precision value of the initial tuples. That is, $w_1 = u_{u_1} - u_{l_1} \leq \alpha(nt_1)$, and $w_2 = u_{u_2} - u_{l_2} \leq \alpha(nt_2)$. Then we can conclude $w_1 = u_{u_1} - u_{l_1} \leq \alpha(nt_1)$ and $w_2 = u_{u_2} - u_{l_2} \leq \alpha(nt_2)$ for both cases. We know $U(t_1 \wedge t_2) = [u_{l_1} \times u_{l_2}, u_{u_1} \times u_{u_2}]$, so its width is

$$
\begin{aligned}
w &= u_{u_1} \times u_{u_2} - u_{l_1} \times u_{l_2} \\
&= (u_{l_1} + w_1) \times (u_{l_2} + w_2) - u_{l_1} \times u_{l_2} \\
&= u_{l_2} \times w_1 + u_{l_1} \times w_2 + w_1 \times w_2 \\
&= u_{l_2} \times w_1 + (u_{l_1} + w_1) \times w_2
\end{aligned}
$$

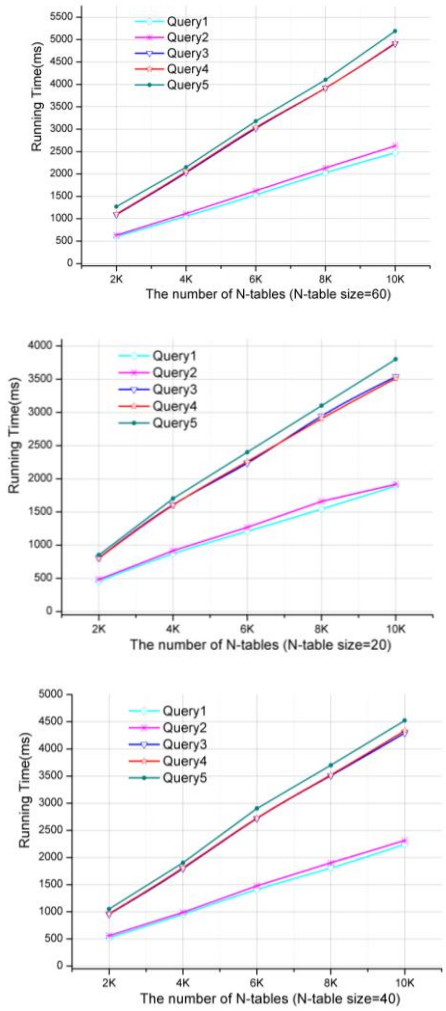
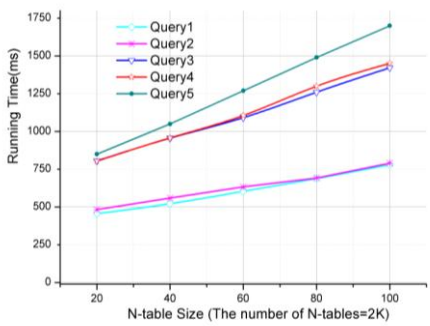


Fig. 3. The running time of different numbers of *N*-table.



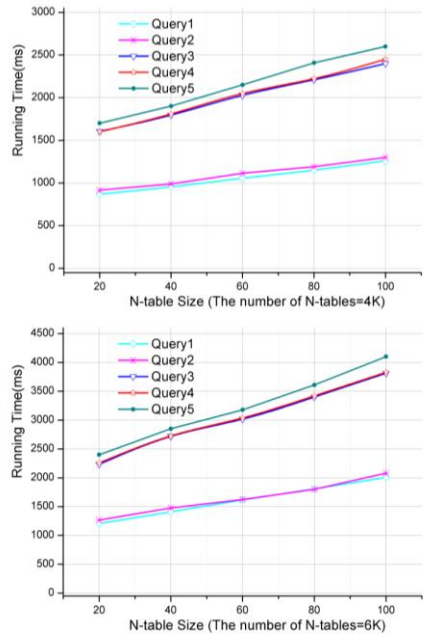


Fig. 4. The running time of different *N*-table size.

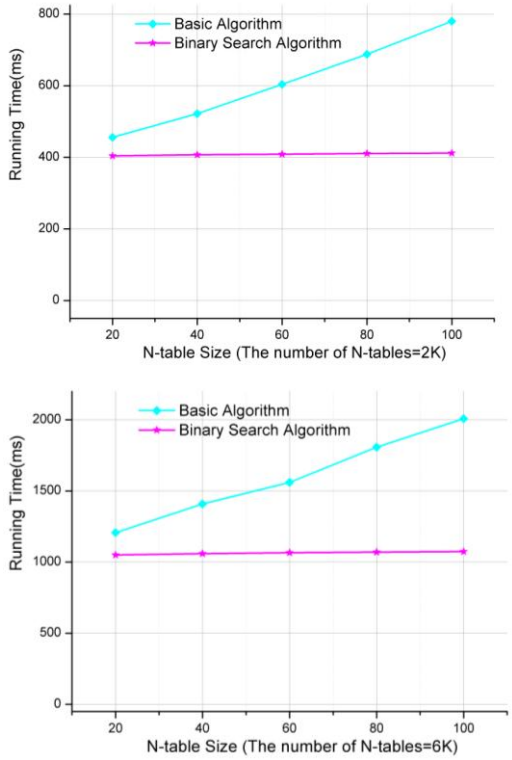


Fig. 5. The running time of Query 1 in basic algorithm and binary search algorithm in different *N*-table size.

$$
\begin{aligned}
&= u_{l_2} \times w_1 + u_{u_1} \times w_2 \\
&\leq w_1 + w_2 \leq \alpha(nt_1) + \alpha(nt_2)
\end{aligned}
\tag{9}
$$

If $\alpha(nt_1) \leq \alpha_h / 2$ and $\alpha(nt_2) \leq \alpha_h / 2$, then we can get $w \leq \alpha_h / 2 + \alpha_h / 2 = \alpha_h$ from the above equation.

$t_1 \vee t_2$.

Similarly, for $t_1 \vee t_2$, $U(t_1 \vee t_2) = [u_{l_1} + u_{l_2} - u_{l_1} \times u_{l_2}, u_{u_1} + u_{u_2} - u_{u_1} \times u_{u_2}]$, and its width is

$$
\begin{aligned}
w &= (u_{u_1} + u_{u_2} - u_{u_1} \times u_{u_2}) - (u_{l_1} + u_{l_2} - u_{l_1} \times u_{l_2}) \\
&= w_1 + w_2 - u_{u_1} \times u_{u_2} + u_{l_1} \times u_{l_2} \\
&= w_1 + w_2 - (u_{l_1} + w_1)(u_{l_2} + w_2) + u_{l_1} \times u_{l_2} \qquad (10) \\
&= w_1 + w_2 - (u_{l_1} \times w_2 + u_{l_2} \times w_1 + w_1 \times w_2) \\
&\le w_1 + w_2 \le \alpha(nt_1) + \alpha(nt_2).
\end{aligned}
$$

We can also get $w \le \alpha_h$ if $\alpha(nt_1) \le \alpha_h / 2$ and $\alpha(nt_2) \le \alpha_h / 2$. For the above two situations, we can conclude that the query precision requirements α_h can be satisfied if the α-precision of involved *N*-tables is less than $\alpha_h / 2$.

For a compound event $ee = ((t_1 \wedge t_2) \vee (t_3)) \wedge t_4$ where t_1, t_2, t_3, t_4 are initial tuples or initial tuples over selection operation, we define the level according to the times of $\wedge$, $\vee$ operations for the tuples. For t_4, one $\wedge$ operation is to be computed in the computation process, then its level $level(t_4) = 1$. For t_1, two $\wedge$ operations and one $\vee$ operation are to be computed, then its level $level(t_1) = 3$. Similarly, $level(t_2) = 3$, $level(t_3) = 2$. Therefore, as we have concluded above, the precision requirement α_h for the compound event *ee* can be traced back to all initial tuples according to the levels, that is, $\alpha_h / level(t_i)$ for t_i.

IV. Evaluation

We evaluate the N-DB model by implementing queries on a Windows PC with 4 core CPU and 2GB RAM. Queries are evaluated on Mysql database of movie data in a size of 10K *N*-tables. Based on the model of N-DB, we evaluate the five queries in Section III We change the number of N-tables and the size of a *N*-table to find the scalability of the model.

Different numbers of N-tables. Fig. 3 shows the running time with the change of the numbers of *N*-tables. In *N*-table size of 20, 40, and 60, the running time all increases almost linearly with the increase of *N*-table numbers. That is because the increase of *N*-table numbers results in the increase of database size. Therefore, we need search for information in more *N*-tables. Query 3 and Query 4 are compound queries which are composed of simple queries like Query 1 and Query 2, thus it costs more running time. In the aggregation process, we need to go through the whole *N*-table, and there is extra expanding time. Thus the running time of Query 5 is even longer.

Different size of a N-table. The change of *N*-table size can also influence the efficiency, as shown in Fig. 4. In traditional databases, the value of an attribute is a single value which can be obtained quickly. However, the uncertain attribute value in N-DB is stored in the tabular form, *i.e.*, *N*-table. The time cost of searching in a *N*-table is sensitive to its size, which is usually in $O(n)$ time complexity. Fig. 4 shows the linear increment relationship between the running time of queries and the *N*-table size. As we know, a *N*-table is a set of ordered pairs. Thus, the time cost can be improved by taking better searching algorithms applicable to ordered lists, such as $O(\log_2 n)$ in binary search algorithm.

Basic algorithm vs. Binary search algorithm. Then we compare the basic algorithm ($O(n)$ search algorithm) and the binary search algorithm in Fig. 5. We find the growth trend of the binary search algorithm is much slower than the basic algorithm. When the number of *N*-tables gets larger (from 2K to 6K), the optimization effects are more obvious.

V. Concluding Discussions

We introduce a new uncertain handling data model which can deal with uncertainty in visual media metadata. By storing some materialized data in N-tables in an increasing order, it can work efficiently and implement queries with sensible answers. We modify the relational algebra, show query examples, and prove the model efficiency through experiments.

References

[1] T. Imielinski and W. Lipski, "Incomplete information in relational databases," *J. of ACM*, vol. 31, no. 4, pp. 761-791, 1984.

[2] A. Abiteboul, P. Kanellakis, and G. Grahne, "On the representation and querying of sets of possible world," *SIGMOD Record*, vol. 16, no. 3, pp. 34-48, 1987.

[3] N. Fuhr and T. Rolleke, "A probabilistic relational algebra for the integration of information retrieval and database systems," *ACM Trans. Information Systems*, 1997.

[4] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, 1970.

[5] J. Pei, M. Hua, Y. Tao, and X. Lin, "Query answering techniques on uncertain and probabilistic data: Tutorial summary," *SIGMOD*, 2008.

[6] A. D. Sarma, O. Benjelloun, A. Halevy, S. Nabar, and J. Widom, "Representing uncertain data: models, properties, and algorithms," *Journal of VLDB*, vol. 18, no. 5, 2009.

[7] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," *SIGMOD*, 2003.

[8] N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," *VLDB*, 2004.

[9] C. Mayfield, S. Singh, R. Cheng, and S. Prabhakar. Orion: A database system for managing uncertain data. [Online]. Available: http://orion.cs.purdue.edu.

[10] J. Boulos, N. Dalv *et al*., "Mystiq: A system for finding more answers by using probabilities," *SIGMOD*, 2005.

[11] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom, "Trio: A system for data, uncertainty, and lineage," *VLDB*, 2006.

[12] M. A. Soliman, I. F. Ilyas, and K. Chang, "Urank: Formulation and efficient evaluation of top-k queries in uncertain databases," *SIGMOD*, 2007.

[13] J. Huang, L. Antova, C. Koch, and D. Olteanu, "Maybms: A pobabilistic database management system," *SIGMOD*, 2009.

[14] L. V. S. Lakshmanan, N. Lenoe, R. Ross, and V. S. Subrahmanian, "ProbView: A flexible probabilistic database system," *ACM Transaction on Database System*, vol. 22, no. 3, 1997, pp. 419-469.

[15] L. A. Zadeh, "Fuzzy sets," *J. of Information and Control*, vol. 8, 1965, pp. 338-353.

[16] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *J. of Fuzzy Sets and Systems*, vol. 1, 1978, pp. 3-28.

[17] L. A. Zadeh, "Fuzzy logic," *IEEE Comput. Mag.*, vol. 21, no. 4, April 1988, pp. 83-93.

[18] J. Baldwin, "A fuzzy relational inference language for expert systems," in *Proc. 13th IEEE Int'l Symp. Multiple-Valued Logic*, pp. 416-423, 1983.

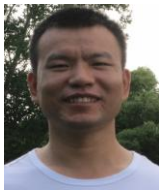
[19] H. Prade and C. Testemale, "Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries," *Information Sciences*, vol. 34, 1984, pp. 115-143.

[20] H. Prade and C. Testemale, "Fuzzy relational databases: Representational issues and reduction using similarity measures," *J. Am. Soc. Information Science*, vol. 38, no. 20, 1988, pp. 118-126.

[21] D. Dubois and H. Prade, *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, New York: Plenum Press, 1988.

[22] P. Boscand and O. Pivert, "Imprecise data management and flexible querying in databases," *Fuzzy Sets, Neural Networks and Soft Computing, Van Nostrand Reinhold*, 1994, pp. 368-395.
[23] M. Kamel, B. Hadfield, and M. Ismail, "Fuzzy query processing using clustering techniques," *Information Processing and Management*, vol. 26, 1990, pp. 279 – 293.
[24] A. Yazici, B. P. Buckles, and F. E. Petry, "A survey of conceptual and logical data models for uncertainty management," *Fuzzy Logic for Management of Uncertainty*, John Wiley and Sons Inc., 1992, pp. 607-644.
[25] E. E. Kerre and G. Q. Chen, "An overview of fuzzy data modeling," *Fuzziness in Database Management Systems*, Physica-Verlag, 1995, pp. 23-41.
[26] Z. M. Ma and L. Yan, "A literature overview of fuzzy database models," *J. of Information Science and Engineering*, vol. 24, pp. 189-202, 2008.
[27] Z. M. Ma and L. Yan, "A literature overview of fuzzy conceptual data modeling," *J. of Information Science and Engineering*, vol. 26, pp. 427-441, 2010.
[28] R. Jampani, F. Xu, and M. Wu, "Mcdb: A monte carlo approach to managing uncertain data," *SIGMOD*, 2008.
[29] S. K. Lee, "Imprecise and uncertain information in databases: an evidential approach," *ICDE*, 1992.

Yiping Li received her bachelor degree from Beijing University of Technology in China in 2008. She is now the PhD candidate in computer science and technology at Tsinghua University in China. Her research area is uncertain data managements.



Jianwen Chen received his bachelor degree from Nanjing University in China in 2000. He is now the PhD candidate in computer science and technology at Tsinghua University in China. His research area is uncertain data management.

Ling Feng is now working in Tsinghua University in China. Her research interests include context aware data management towards ambient intelligence, knowledge-based information systems, data mining and warehousing, and distributed object-oriented database management systems, etc. She has published over 150 scientific articles in high-quality international conferences or journals, and received the 2004 Innovational VIDI Award by the Netherlands Organization for Scientific Research, 2006 Chinese ChangJiang professorship Award by the Ministry of Education, and 2006 Tsinghua Hundred-Talents Award.