Neighborhood/Conceptual Query Answering with Imprecise/Incomplete Data*

Show-Jane Yen and Arbee L.P. Chen

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
Email: alpchen@cs.nthu.edu.tw

Abstract

The evolution of database systems tends to the development of higher degree of user-friendliness such that the system can be directly handled by nonprofessionals. In order to reach this goal, the database system needs to provide a query language by which queries can be specified conceptually. Also, the query condition may be relaxed such that information within a certain semantic distance to the exact answer can be obtained. Moreover, the real-world information is usually imprecise and incomplete. It is therefore important to store imprecise and incomplete information in a database, and to manipulate this information accordingly.

In this paper, we provide a conceptual query language by which fuzzy query conditions and neighborhood query conditions can be specified. Query processing strategies for these two query conditions are proposed considering imprecise and incomplete information. A domain concept hierarchy is constructed on top of a numerical domain to handle imprecise data, while dependencies between database attributes are derived for incomplete information. An application of the techniques for processing queries under network partitioning is also discussed.

1 Introduction

When a user queries a database, if an exact answer is unavailable, the system returns a null answer. A *neighborhood answer* is one which is close to an exact answer within certain semantic distance. Wesley and Qiming [4][5][6][7] presented cooperative query answering which provided neighborhood query answering. The information for providing neighborhood query answering comes from a *type abstraction hierarchy* which is an information of is-a and part-of views of type hierarchy. However, for many applications, such a type abstraction hierarchy cannot be created. Motro [19][20] presented a way to handle *goal queries* which establish target qualifications and are concerned with data which are close to these targets. Processing a goal query can yield neighborhood answers. The language for specifying a goal query is based on tuple calculus which is not user-friendly. Moreover, the evaluation of goal queries is rather complicate. Ichikawa and Hirakawa [17] presented a relational database system, ARES, which has the capability of performing flexible query interpretation. However, only the "similar to" or "approximately equal to" operations are considered.

In many cases, users do not have a precise view about information stored in the database. Traditional query processing requires that queries be specified precisely, and thus requires users to fully understand the database structure and content. In recent years, database researchers have focused on how to provide high-level, user-friendly environment for end-users. By using the same type abstraction hierarchy, Wesley and Qiming [4][5] provided an approach to answer *conceptual queries*. Takahashi [24] and Zemankava and Kandel [25] proposed fuzzy query languages for querying relational databases with the same purpose.

The information contained in a real-world database is usually incomplete and imprecise. Prad [21][22] and Buckles and Petry [1][2] described an extended relational algebra which deals with fuzzy information, and used possibility distributions and related concepts to model imprecise information. For incomplete information, several approaches

*This work was partially supported by the Republic of China National Science Council under Contract No. NSC 82-0408-E-007-098

have been proposed for null values [13][16][18]. They generally attempt to find appropriate responses to queries from such incomplete data.

In this paper, we provide a conceptual query language which contains *fuzzy query conditions* and *neighborhood query conditions* such that users can pose queries accordingly. We make use of a *domain concept hierarchy* to handle imprecise information. The database relations can store null values and various fuzzy terms specified on numerical attribute domains. Furthermore, a new method is proposed to estimate the value for a null value, using the dependencies between attributes.

When network partitioning occurs in a distributed database system, some queries may not be able to be processed because some needed data are inaccessible. These inaccessible data can be regarded as null values. Using the same way which handle null values, our approach can also estimate these inaccessible data to answer the queries.

The rest of this paper is organized as follows. Section 2 describes the basic concepts for the domain concept hierarchy and the dependencies between attributes. Section 3 presents the approach to estimate the value for a fuzzy term or null value. Section 4 presents the query processing for neighborhood query conditions and fuzzy query conditions. Fault tolerant distributed query processing as an application of our estimation techniques is discussed in Section 5. Finally, we conclude this paper and present some directions for future research in Section 6.

2 Semantic Information

In order to handle null values and fuzzy terms, we propose a method to estimate their values from other attribute values. For example, if there exists a statistic "if degree is master and years-of-experience is five, then salary is about $52000," we can estimate a null value or fuzzy term for attribute "salary" in a tuple based on this statistic and the values of the associated attributes "degree" and "years-of-experience." In this section, we shall discuss mechanisms for estimating attribute values. The procedure for the estimation will be given in Section 3.

2.1 Domain concept hierarchy

According to the meanings of a numerical domain, we can divide its elements into groups, and represent each group by a concept word. For example, suppose "age" is a numerical domain ranging from 0 to 120. The ages from 13 to 19 can be represented by a concept word "teenager." The hierarchy constructed from these concept words on a domain is called a *domain concept hierarchy*. If the concept words are applicable to all related relations, we call the associated domain concept hierarchy a *global domain concept hierarchy*. On the other hand, if they are applicable only to a specific relation, it is called a *local domain concept hierarchy*.

Consider a COMPANY database consisting of the following relations:

MANAGER(id, name, age, degree, years-of-experience)
ASSISTANT(id, name, age, degree, years-of-experience)

$40000 for an assistant is considered a "high" salary, but for a manager, it may just be a "median" salary. In this case, two local domain concept hierarchies are constructed for "salary" in the relations of ASSISTANT and MANAGER, respectively. There may also exist a global domain concept hierarchy for "salary" if we do not care about a person's job position. Tables 1 and 2 give two relations, and Figures 1 - 5 show the associated domain concept hierarchies for later reference.

id	name	age	degree	years-of-experience	salary
11	Alex	36	PHD	4	null
12	Anndy	adult	master	median	high
13	Frank	youth	bachelor	high	52000
14	Jane	adult	master	7	high
15	Tony	youth	senior	8	low
16	Keith	teenager	senior	6	25000

Table1. relation MANAGER

id	name	age	degree	years-of-experience	salary
101	John	youth	master	median	45000
102	Mary	30	bachelor	very low	high
103	Ammi	35	senior	6	24000
104	Jack	prime	senior	8	median
105	Nancy	teenager	junior	low	18000

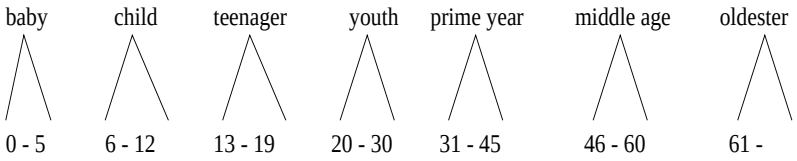

Figure 1: Global domain concept hierarchy for "age."

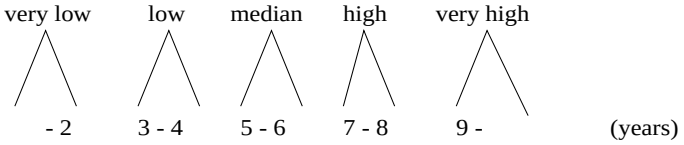

Figure 2: Global domain concept hierarchy for "years-of-experience."

Table 2. relation ASSISTANT

2.2 Semantic associations

Semantic associations are the associations of ordered pairs of elements in different attribute domains on their semantics. The semantic associations may exist between two numerical domains, a numerical domain and a nonnumerical domain or two nonnumerical domains. We use this information to derive dependencies among attributes to be discussed in the next subsection.

A *degree* of *semantic association* for an ordered pair of elements in different attribute domains can be specified. For example, the degree of the semantic association from "black eyes" to "Oriental" can be very high, and that from "blue eyes" to "Oriental" close to zero.

Semantic Association Relation (SAR) is a relation which records the degrees, denoted p, of semantic associations. If the domain of an attribute is a nonnumerical domain, the elements in the SAR are the domain elements of this attribute. Otherwise, the elements in the SAR are concept words specified on the domain of this attribute. The degree of a semantic association is between zero and one, which can be specified by database designers. Two SARs are shown in Tables 3 and 4.

degree	salary	p
PHD	high	0.95
master	high	0.85
bachelor	high	0.7
junior	high	0.1
.	.	.
.	.	.
.	.	.

years-of-experience	salary	p
very high	high	0.9
high	high	0.8
median	high	0.6
low	high	0.4
very low	high	0.2
.	.	.
.	.	.
.	.	.

Table 3. **Table 4.**

Table 3: SAR from "degree" to "salary."
Table 4: SAR from "years-of-experience" to "salary."

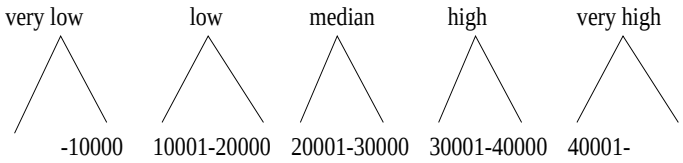

Figure 3: Global domain concept hierarchy for "salary" in relation ASSISTANT.

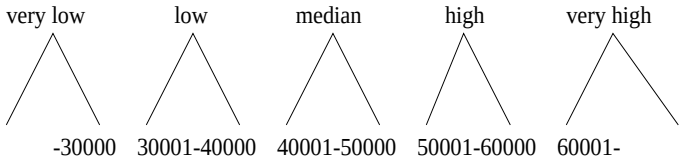

Figure 4: Global domain concept hierarchy for "salary" in relation MANAGER.

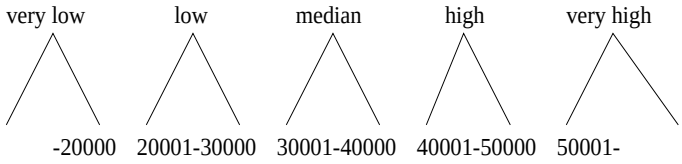

Figure 5: Global domain concept hierarchy for "salary."

2.3 Determinations between attributes

In database relations, there may exist *determinations* between attributes, for example, the higher the "degree" is, the higher the "salary" is, and vice versa. There may also exist determinations between attribute "years-of-experience" and attribute "salary." However, the degrees of these determinations may be different. For example, the degree for "degree" to determine "salary" may be higher than the degree for "years-of-experience" to determine "salary." Also, the degree for "degree" to determine "salary" may be higher than the degree for "salary" to determine "degree."

We provide a method to find out the degrees of the determinations between attributes in a database. Assume that A_1 and A_2 are attribute names. Let D(A_1) be the domain of A_1 if it is a nonnumerical domain, or the set of all concept words specified on the domain of A_1 if it is a numerical domain. Also, let p(a,b) denote the degree of the semantic association from element a to element b, which can be found in SARs. Let D(A_2) be {e_1, e_2, ..., e_n}. Then the degree for A_2 to determine A_1, denoted D(A_2,A_1), can be computed as follows:

$$D(A_2, A_1) = \frac{\sum_{i=1}^{n} \max_{e \in D(A_1)} p(e_i, e)}{n} \qquad (1)$$

2.4 Rule base

Rules are constructed for each attribute A which is determined by other attributes to estimate the values of A. Let DT(A) be the set of attributes which determine attribute A. For all database attribute A_i, DT(A) can be computed as follows:

$$DT(A) = \{A_i | D(A_i, A) > \alpha_1, 0 \leq \alpha_1 \leq 1\}, \qquad (2)$$

where α_1 is a threshold of the degree of determinations from A_i to A.

Let DT(A) = {A_1, A_2, ..., A_m}. The *weight* for attribute A_i to determine attribute A, denoted W_i, is computed as follows:

$$W_i = \frac{D(A_i, A)}{\sum_{j=1}^{m} D(A_j, A)} \qquad (3)$$

Given that attribute A_1, A_2, ..., A_m determine attribute A. If the domain of attribute A is a numerical domain, the rules about attribute A can be constructed as follows:

Rules (Attribute A)

	W = W_1		W = W_2				W = W_m			
IF	$A_1 = a_{11}$	AND	$A_2 = a_{12}$	AND	...	AND	$A_m = a_{1m}$	THEN	A = r_1	(κ_1)
	.		.		.		.		.	
	.		.		.		.		.	
	.		.		.		.		.	
IF	$A_1 = a_{q1}$	AND	$A_2 = a_{q2}$	AND	...	AND	$A_m = a_{qm}$	THEN	A = r_q	(κ_q)

For each rule, the IF clause is called the condition part of the rule and the THEN clause is called the action part of the rule. The values of a_{11}, ..., a_{1m}, ..., a_{q1}, ..., a_{qm} are specified by the database designer. κ_1, ..., κ_q are concept words to which numbers r_1, ..., r_q correspond, respectively. W_1, W_2, ..., W_m can be obtained from expression (3).

If the domain of attribute A is a nonnumerical domain, the rules about attribute A are constructed as follows:

Rules (Attribute A)

	W = W_1		W = W_2				W = W_m		
IF	$A_1 = a_{11}$	AND	$A_2 = a_{12}$	AND	...	AND	$A_m = a_{1m}$	THEN	A = w_1
	.		.		.		.		.
	.		.		.		.		.
	.		.		.		.		.
IF	$A_1 = a_{q1}$	AND	$A_2 = a_{q2}$	AND	...	AND	$A_m = a_{qm}$	THEN	A = w_q

where w_1, w_2, ..., w_q are domain elements of attribute A.

For example, in relation MANAGER, we can find that attributes "degree" and "years-of-experience" determine attribute "salary" by expressions (1) and (2). Suppose the weight for attribute "degree" to determine attribute "salary" and for attribute "years-of-experience" to determine attribute "salary" are 0.6 and 0.4, respectively. Then the rules about attribute "salary" are as follows:

Rules (Attribute salary)

	W = 0.6		W = 0.4			
IF	degree = PHD	AND	years-of-experience = 9	THEN	salary = 61000	(very high)
IF	degree = master	AND	years-of-experience = 5	THEN	salary = 52000	(high)
IF	degree = bachelor	AND	years-of-experience = 8	THEN	salary = 45000	(median)
IF	degree = bachelor	AND	years-of-experience = 3	THEN	salary = 38000	(low)
IF	degree = senior	AND	years-of-experience = 5	THEN	salary = 26000	(very low)

3 Estimating Attribute Values

3.1 Relative similarity

We construct a table, named *relative similarity table*, to represent the relative similarity for each element in a domain. This table has two attributes denoting an element and its relative similarity degree which is between zero and one. An element is assigned the degree one, and is called a *base element*. The *relative similarity degree* for an element represents the similarity degree between this element and the base element. These relative similarity degrees are specified by database designers.

element	s
PHD	1
master	0.8
bachelor	0.6
senior	0.3
junior	0.1

Table 5.

element	s
very high	1
high	0.8
median	0.5
low	0.3
very low	0.1

Table6.

Suppose that attribute A_2 determines attribute A_1, the domain of attribute A_2 contains nonnumerical elements or concept words e_1, e_2, ..., e_n and the domain of attribute A_1 is a numerical domain. If the numbers in attribute A_1 increases with the ordered elements e_1, e_2, ..., e_n, then the element e_n is designated the base element for attribute A_2 in the relative similarity table. If the numbers in attribute A_1 decreases with the ordered elements e_1, e_2, ..., e_n, then the element e_1 is designated the base element.

For example, suppose the salary increases with the degree "junior," "senior," "bachelor," "master," and "PHD." Then, the element "PHD" is the base element for attribute "degree." The relative similarity tables for "degree" and "years-of-experience" which determine "salary" are shown in Tables 5 and 6, respectively.

If the domain of attribute A_1 is a nonnumerical domain, then we can choose any element e_i as the base element for attribute A_2 and construct the table based on the semantic similarity of the other elements and the base element in A_2. Similarly, a relative similarity table is constructed for A_1.

Let s(v) denote the relative similarity degree for element v. The *semantic distance* $d(a,b)$ between element a and element b in an attribute can be defined as follows:

$$d(a,b) = |s(a) - s(b)| \qquad (4)$$

The semantic distance d is the distance of arbitrary pairs of the elements in the same attribute domain.

3.2 Estimation methods

Suppose a relation contains attributes A_1, A_2, ..., A_m and A, and a tuple t in the relation has a null attribute value for attribute A. To estimate the null value, we need to compute the closeness degree between tuple t and the condition part of each rule for attribute A, and choose a rule which is the closest to tuple t to estimate the null value. Let A(t) be the attribute value for a tuple t, and Range(A) the domain range of attribute A.

3.2.1 estimating a number

If the domain of attribute A is a numerical domain, we compute the closeness degree between tuple t and the condition part of the jth rule in Rules(Attribute A). The jth rule is shown as follows:

Rule (Attribute A)

	W = W_1		W = W_2		...		W = W_m			
IF	$A_1 = a_{j1}$	AND	$A_2 = a_{j2}$	AND	...	AND	$A_m = a_{jm}$	THEN	A = r_j	(κ_j)

If the domain of attribute A_i ($1 \leq$ i $\leq$ m) is a nonnumerical domain, then the closeness degree between a_{ji} and A_i(t) is:

$$CD_{ji}(t) = \frac{s(A_i(t))}{s(a_{ji})} \qquad (5)$$

If the domain of attribute A_i ($1 \leq$ i $\leq$ m) is a numerical domain, then there are two methods to find the closeness degree between a_{ji} and A_i(t):

Method 1. Convert a_{ji} into concept word a'_{ji}. If A_i(t) is a number, then convert A_i(t) into concept word $A_i(t)'$. If A_i(t) is a concept word, then let A_i(t) be $A_i(t)'$. The closeness degree between a_{ji} and A_i(t) is:

$$CD_{ji}(t) = \frac{s(A_i(t)')}{s(a'_{ji})} \qquad (6)$$

Method 2. If A_i(t) is a concept word, then convert A_i(t) into a number N(A_i(t)). The N(A_i(t)) is the median number of the range of the concept word A_i(t). If A_i(t) is a number, then let A_i(t) be N(A_i(t)). Apply the following expression to find the closeness degree between a_{ji} and A_i(t):

$$CD_{ji}(t) = \frac{N(A_i(t))}{a_{ji}} \qquad (7)$$

The closeness degree between the rule and the tuple t can then be computed as:

$$CD_i(t) = \sum_{i=1}^{m} CD_{ji}(t) \times W_i \qquad (8)$$

The difference between tuple t and the jth rule for attribute A is:

$$Diff_j(t) = |1 - CD_j(t)| \qquad (9)$$

Choose a rule which has the minimal difference with tuple t. Suppose we choose the following rule:

Rule (Attribute A)

	W = W_1		W = W_2		...		W = W_m			
IF	$A_1 = a_1$	AND	$A_2 = a_2$	AND	...	AND	$A_m = a_m$	THEN	A = r	(κ)

The number for A(t) can be estimated as follows:

$$V(A(t)) = CD(t) \times r \qquad (10)$$

Example 1. Consider the first tuple t1 in relation MANAGER. To estimate the null value in attribute "salary," suppose we choose the following rule:

Rules (Attribute salary)

	W = 0.6		W = 0.4			
IF	degree = master	AND	years-of-experience = 5	THEN	salary = 52000	(high)

For attribute "degree", the attribute value for tuple t1 is "PHD" and the condition part of the rule contains the condition "degree = master." Hence, by the expression (5), the closeness degree between "master" and "PHD" is:

$$CD_{degree}(t1) = \frac{s(PHD)}{s(master)} = \frac{1}{0.8} = 1.25 \qquad (11)$$

The relative similarity degrees for elements "PHD" and "master" can be found from Table 5.

For attribute "years-of-experience," the attribute value for tuple t1 is 4 and the condition part of the rule contains the condition "years-of-experience = 5." We use Method 2 in the expression (7) to find the closeness degree between 5 and 4:

$$CD_{years-of-experience}(t1) = \frac{4}{5} = 0.8 \qquad (12)$$

From the Rules(Attribute salary), the weight for "degree" to determine "salary" and the weight for "years-of-experience" to determine "salary" are 0.6 and 0.4, respectively. By the expression (8), the closeness degree between the rule and tuple t1 is:

$$CD(t1) = 1.25 \times 0.6 + 0.8 \times 0.4 = 1.07 \qquad (13)$$

By the expression (9), the estimated number for the attribute value salary(t1) is:

$$V(salary(t1)) = CD(t1) \times r = 1.07 \times 52000 = 55640 \qquad (14)$$

3.2.2 estimating a nonnumerical value

If the domain of A is a nonnumerical domain, we compute the closeness degree between tuple t and the condition part of the rules in Rules(Attribute A). The jth rule is shown as follows:

Rule (Attribute A)

	W = W_1		W = W_2		...		W = W_m		
IF	$A_1 = a_{j1}$	AND	$A_2 = a_{j2}$	AND	...	AND	$A_m = a_{jm}$	THEN	A = w_j

If the domain of attribute A_i ($1 \leq$ i $\leq$ m) is a nonnumerical domain, then the closeness degree between a_{ji} and A_i(t) is :

$$CD_{ji}(t) = 1 - |s(a_{ji}) - s(A_i(t))| \qquad (15)$$

If the domain of attribute A_i ($1 \leq$ i $\leq$ m) is a numerical domain, then there are two methods to find the closeness degree between a_{ji} and A_i(t):

Method 1. Convert a_{ji} into concept word a'_i. If A_i(t) is a number, then convert A_i(t) into concept word $A_i(t)'$. If A_i(t) is a concept word, then let A_i(t) be $A_i(t)'$. The closeness degree between a_{ji} and A_i(t) is:

$$CD_{ji}(t) = 1 - |s(a_{ji})' - s(A_i(t)')| \qquad (16)$$

Method 2. If A_i(t) is a concept word, then convert A_i(t) into a number N(A_i(t)). The N(A_i(t)) is the median number of the range of the concept word A_i(t). If A_i(t) is a number, then let A_i(t) be N(A_i(t)). The closeness degree between a_{ji} and A_i(t) is:

$$CD_{ji}(t) = 1 - \frac{|N(A_i(t)) - a_{ji}|}{Range(A_i)} \qquad (17)$$

The closeness degree $CD_j(t)$ between the condition part of the rule and tuple t can be computed as expression (8).

The difference between tuple t and the condition part of the rule is:

$$Diff_j(t) = 1 - CD_j(t) \qquad (18)$$

Choose a rule which has the minimal difference with tuple t. Suppose we choose the following rule:

Rule (Attribute A)

	W = W_1		W = W_2		...		W = W_m		
IF	$A_1 = a_1$	AND	$A_2 = a_2$	AND	...	AND	$A_m = a_m$	THEN	A = w

The estimated value for the null value is "w."

4 Query Processing

4.1 Conceptual query language

In order to deal with fuzzy query condition and neighborhood query condition, we develop a conceptual query language. Let R_i (i = 1,2,...) denote relations, and A_i (i = 1,2,...) attributes in relations. The conceptual query language has the following form:

SELECT $\{A_1$**[**,A_2**,...]**;**ALL**$\}$
FROM R_1**[**,R_2**,...]**
<**select clause**>

The general form of the <**select clause**> is:

$$< \textbf{select clause} > ::= \{\textbf{conj}\} < \textbf{condition} > (\{\textbf{AND}; \textbf{OR}\}\{\textbf{conj}\} < \textbf{condition} >)^{\mathbf{k}} \quad \mathbf{k} \geq \mathbf{0}$$

Where **conj** is chosen from {SUCH THAT; WHERE; WHO; WHOSE; WHOM; WHICH}. Let A be an attribute, κ be a fuzzy term in the domain of attribute A, w be an element in a nonnumerical domain, c be an element in a numerical domain, Θ be a fuzzy comparison operator chosen from {GREATER THAN, LESS THEN} and θ be an exact comparison operator chosen from $\{\leq, <, =, >, \geq\}$. The <**condition**> has the following three forms:

<**condition**>:

I. neighborhood query conditions

A IS w
A IS c
A IS Θ **c**
A_1 **IS** A_2
A_1 **IS** Θ A_2
R_1**.A IS** R_2**.A**
R_1**.A IS** Θ R_2**.A**

II. fuzzy query conditions

A IS κ
IS κ

III. exact query conditions

A = w
A θ **c**

In the rest of this section, we explain the query processing for neighborhood query conditions and fuzzy query conditions in select operations and join operations.

4.2 Neighborhood query condition

To judge whether a tuple is a neighborhood answer to the query condition and to find the neighborhood degree of a tuple which is a neighborhood answer, we consider the following cases:

Case I. The domain of attribute A in a neighborhood query condition is a numerical domain:

Consider the neighborhood query condition q_1**: A IS c** and c is a number. We process this condition in the following steps:

1. Convert the number c into concept word κ by referencing the domain concept hierarchy of attribute A.

2. To judge whether tuple t is a neighborhood answer, we explain the process in the following cases. If A(t) is a null value, according to the concept word κ , choose the rule whose action part corresponds to the concept word κ and estimate a number for the null value. Convert the estimated number into concept word $A(t)'$. If A(t) is a number, convert the number into concept word $A(t)'$. If A(t) is a concept word, then let $A(t)'$ be A(t). If $|s(\kappa) - s(A(t)')| \leq \alpha_2$, then the tuple t is a neighborhood answer to the query condition q_1, where α_2 is a threshold of the neighborhood distance.

3. Compute the neighborhood degree for tuple t which is a neighborhood answer. If A(t) is a concept word, according to the concept word, choose the rule whose action part corresponds to the concept word, and estimate a value V(A(t)) for A(t). If it is null, also estimate a value V(A(t)) for A(t). Let A(t) be V(A(t)) if it is a number.

   The neighborhood degree for tuple t to satisfy the query condition q_1 is:

$$N(t) = 1 - \frac{|V(A(t)) - c|}{Range(A)} \qquad (19)$$

Consider the following query conditions:

q_2: **A IS GREATER THAN c**
q_3: **A IS LESS THAN c**

The query processing for query conditions q_2 and q_3 is the same as that for query condition q_1 except that in step 3, if V(A(t)) for A(t) is less than or equal to c, then tuple t is not a neighborhood answer to the query condition q_2; if V(A(t)) is greater than or equal to c, then tuple t is not a neighborhood answer to the query condition q_3.

Example 2. Consider the relation MANAGER in Table 1. The query Q_1 is given in the following:

SELECT ALL
FROM MANAGER
WHOSE salary IS 53000

The query condition in query Q_1 is a neighborhood query condition on the select operation. Consider the first tuple t1 in relation MANAGER. To judge whether t1 is a neighborhood answer to the query Q_1, we explain the query processing in the following steps:

1. Convert the number 53000 into concept word "high" by referencing the domain concept hierarchy of attribute "salary" in Figure 4.
2. Because the attribute value salary(t1) is a null value in relation MANAGER, choose the second rule from the rules on attribute "salary" in Rules(Attribute salary) according to the concept word "high." By Example 1, we have obtained that the estimated number for the null value is 55640.
3. Convert the estimated number V(salary(t1)) into concept word "high" by referencing the domain concept hierarchy of attribute "salary" in Figure 4. Assume that the threshold α_2 of the neighborhood distance is 0.2. The concept word in the query condition is "high" and the relative similarity degree for concept word "high" can be found from the relative similarity table of attribute "salary." Because $|s(high) - s(high)| = 0$ (≤ 0.2), tuple t1 is a neighborhood answer to the query Q_1.
4. By expression (19), the *neighborhood degree* for tuple t1 to satisfy the query condition in the query Q_1 is as follows, assuming that the domain range for "salary" is 70000:

$$N(t1) = 1 - \frac{|55640 - 53000|}{70000} = 0.96 \qquad (20)$$

From the above process, we can see that the first tuple in relation MANAGER is a neighborhood answer to the query Q_1 and its neighborhood degree is 0.96. We can process the other tuples in relation MANAGER similarly.

Case II. The domain of attribute A in a neighborhood query condition is a nonnumerical domain.

Consider the query condition q_4: **A IS w** and w is an element in the domain of attribute A.

If A(t) is not a null value, then if $|s(w) - s(A(t))| \leq \alpha_2$, then tuple t is a neighborhood answer and the neighborhood degree for tuple t to satisfy the query condition q_4 is:

$$N(t) = 1 - |s(w) - s(A(t))| \qquad (21)$$

Otherwise, tuple t is not a neighborhood answer. If A(t) is a null value, then choose the rule whose action part corresponds to w and compute the closeness degree CD(t) between the condition part of the rule and tuple t. If CD(t) is greater than or equal to a threshold α_3, then the tuple t is a neighborhood answer to the query condition q_4 and the neighborhood degree for tuple t to satisfy the query condition is CD(t). Otherwise, tuple t is not a neighborhood answer.

4.3 Fuzzy query condition

Fuzzy query condition is a query condition containing a fuzzy term. Hence, the domain of the attribute in fuzzy query condition must be a numerical domain.

Let κ be a fuzzy term, i.e., concept word specified in the domain concept hierarchy of attribute A.

Consider the fuzzy query condition q_5: **A IS** κ

To judge whether tuple t is an answer to the fuzzy query condition q_5, we explain the process in the following cases:

1. A(t) is a concept word.

   If A(t) matches κ in the fuzzy query condition q_5, then tuple t is an answer to the fuzzy query condition. Otherwise, tuple t is not an answer to the fuzzy query condition.

2. A(t) is a number.

   Convert the number A(t) into concept word $A(t)'$ and judge whether tuple t is an answer to the fuzzy query condition q_5 using the same process in case 1.

3. A(t) is a null value.

   According to the fuzzy term κ, choose the suitable rule from the rules on attribute A, and estimate the number V(A(t)) for the null value A(t). Convert V(A(t)) into concept word $V(A(t))'$ and judge whether tuple t is an answer to the fuzzy query condition q_5 using the same process in case 1.

Example 3. Consider relation MANAGER in Table 1. The query Q_2 is given in the following:

SELECT ALL
FROM MANAGER
WHOSE salary IS high

The query condition in query Q_2 is a fuzzy query condition. Consider the first tuple t1 in relation MANAGER. Because salary(t1) is a null value, we estimate salary(t1) and get 55640 as done in Example 1. Since the concept word for salary 55640 is "high," tuple t1 is an answer to the query Q_2.

4.4 Neighborhood join

Since the attribute values in relations may contain fuzzy terms or null values whose values can be estimated, when two tuples each from a relation are joined, the system gives the join result a *join degree*.

Consider the following query condition q_6: R_1**.A IS** R_2**.A**. To find a neighborhood answer to the query condition q_6 and the join degree, we explain the process by the following two cases:

Case I. The domain of the join attribute A is a numerical domain.

Two tuples can be joined if their join attribute values correspond to the same concept word or neighboring concept words. We partition the process into the following steps:

1. If a join attribute value is null, then estimate a number for this null value.

2. Determine which tuples in relations R_1 and R_2 can be joined.

   Convert the numbers for the join attribute into concept words. Let the concept words for tuples t1 and t2 be $A(t1)'$ and $A(t2)'$, respectively. If $|s(A(t1)') - s(A(t2)')|$ is less than or equal to a threshold α_4, then the two tuples t1 and t2 can be joined. Otherwise, they cannot be joined.

3. Find the join degree for the join result.

   Suppose tuples t1 and t2 can be joined. If a join attribute value is a concept word, estimate a number for the attribute value. Let the number or estimated number for the attribute values A(t1) and A(t2) be denoted as V(A(t1)) and V(A(t2)), respectively. The join degree J(t1,t2) between tuples t1 and t2 is:

$$J(t1,t2) = 1 - \frac{|V(A(t1)) - V(A(t2))|}{max(Range(R_1.A), Range(R_2.A))} \qquad (22)$$

Case II. The domain of the join attribute A is a nonnumerical domain.

We partition the query processing into the following two steps:

1. Estimate values for the null join attribute values.

2. Determine which tuples in relations R_1 and R_2 can be joined and find the associated join degree.

   Let the value or the estimated value for the attribute values A(t1) and A(t2) be denoted as V(A(t1)) and V(A(t2)), respectively. If $d = |s(V(A(t1))) - s(V(A(t2)))|$ is less than or equal to a threshold α_5, then t1 and t2 can be joined and the join degree is d. Otherwise, they cannot be joined.

5 Another Application of Our Estimation Techniques

In order to improve reliability and query response time of a distributed database system, relations are usually partitioned into fragments which are replicated and stored at networked sites [3][14]. Because the replication of fragments needs communication and processing overheads to ensure consistency among the replicas, fragments are usually not fully replicated at all sites. When communication links or sites fail such that a network partitioning is caused, certain fragments may become inaccessible. Based on our approach, the values in the inaccessible fragments can be regarded as null values, and be estimated. We give an example below to show the application of our estimation techniques to the fault tolerant distributed query processing.

Example 4. Assume relation MANAGER is vertically partitioned into three fragments in a distributed database: MANAGER1(id, salary), MANAGER2(name, id, years-of-experience) and MANAGER3(name, age, degree), which are stored at sites A, B, and C, respectively. The following query Q_3 posed from, say, site B, can be answered by our approach although the values for "salary" are inaccessible (assume site A is isolated during a network partitioning).

SELECT Name
FROM MANAGER
WHOSE salary IS 53000

We process the query processing as follows:

1. Find the attributes which determine attribute "salary" from the rules on attribute "salary." The attribute "salary" can be determined by attributes "degree" and "years-of-experience" which are in fragments MANAGER2 and MANAGER3, respectively. Join the two relations MANAGER2 and MANAGER3 into a temporary relation which contains attributes name, id, age, years-of-experience and degree.
2. According to the query condition, choose a rule from the rules on attribute "salary" and estimate all inaccessible attribute values for attribute salary.
3. Process the neighborhood query condition and obtain query answers.

6 Conclusion and Future Work

We have derived semantic information including domain concept hierarchies, determinations between attributes and relative similarities for elements in the same domain to handle incomplete and imprecise information specified in a database. Also, the query processing for the neighborhood query conditions and fuzzy query conditions which can be specified in our conceptual query language has been discussed. By the same approach, a fault tolerant distributed query processing mechanism was also proposed.

In this paper, we have assumed that the given rules are from statistics. The rules can also come from the database itself. Also, more complicate determinations between attributes can be derived to estimate values more accurately. Finally, based on various situations on network partitioning, partitioning of relations, and allocations of fragments to network sites, different degrees of fault tolerant distributed query processing may be achieved, which needs further study.

References

[1] Buckles, B.P. and Petry, F.E., A Fuzzy Representation of Data for Relational Databases, *Fuzzy Set and Systems*, 1982, pp.213-226.

[2] Buckles, B.P. and Petry, F.E., Fuzzy Databases and their Applications, *Fuzzy Information and Decision Processes*, 1982, pp.361-371.

[3] Ceri, S. and Pelagatti, G., Distributed Database Principles and Systems, *New York, NY: McGraw-Hill*, 1984.

[4] Chu, W. and Chen, Q., A Structured Approach for Cooperative Query Answering, *To Appear in IEEE Trans. on Knowledge and Data Engineering*, 1993.

[5] Chu, W. and Chen, Q., Neighborhood and Associative Query Answering, *To Appear in Journal of Intelligent Information Systems*, 1993.

[6] Chu, W., Chen, Q. and Page, T., Cobase: Cooperative Distributed Databases, *Invited Paper, Proc. of the 6th Brazilian Symposium on Databases*, Brazil, 1991.

[7] Chu, W., Chen, Q. and Lee, R., Cooperative Query Answering via Type Abstraction Hierarchy, *In Cooperating Knowledge Based Systems,1990, S.M. Deen eds, Elsevier Science Publishing Co. Inc*, 1991.

[8] Chu, W., et al., Design Considerations of a Fault Tolerant Distributed Database System by Inference Technique, *Extended Abstract, Proceedings of PARBASE-90*, March 6-8, 1990, Miami.

[9] Chu, Wesley and Hwang, Andy, Inference Techniques for a Fault Tolerant Distributed Database System, *Extended Abstract, Proceedings of PARBASE-90*, March 6-8, 1990, Miami.

[10] Chu, W., Hwang, A., Chen, Q. and Lee, R.C., An Inference Technique for Distributed Query Processing in a Partitioned Network, *Technical Report, CSD-900005*, February, 1990, UCLA.

[11] Chu, W., Lee, R.C. and Chen, Q., Fault tolerant distributed database system via data inference, *Proceedings of the Ninth Symposium on Reliable Distributed Systems*, October, 1990.

[12] Chu, W., Lee R.C. and Chiang, K., Capture database semantics by rule induction, *Technical Report CSD-900013*, May, 1990.

[13] Codd, E.F., Extending the database relational model to capture more meaning, *ACM Trans. Database Systems*, 4(4), 1979, pp.397-434.

[14] El Abbadi, Skeen, D. and Criistian, F., An efficient fault-tolerant protocol for replicated data management, *in Proc. 4th ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, 1985, pp.215-229.

[15] Garcia-Molina, H. and Abbott, R.K., Reliable Distributed Database Management, *Proc. of the IEEE*, May, 1987, pp. 601-620.

[16] Grant, J., Null values in a Relational Data Base, *Inform. Process. Lett.*, 6, 1977, pp.156-157.

[17] Ichikawa, T. and Hirakawa, M., ARES: A Relational Database with the Capability of Performing Flexible Interpretation of Queries,*IEEE Trans. Softw. Eng.* SE-12, 5,May 1986, pp.624-634.

[18] Imielinski, T. and Lipski, W., Incomplete Information in Relational Databases, *JACM*, Vol.31(4), 1984, pp.761-791.

[19] Motro, A., Supporting Goal Queries in Relational Databases, *Proc. 1st International Conference on Expert Database Systems*, 1986, pp.85-96.

[20] Motro, A., VAGUE: A User Interface to Relational Databases that Permits Vague Queries, *ACM Trans. on Office Information Systems*, 1988, pp.187-214.

[21] Prad, H., Lipski's Approach to Incomplete Information Databases Restated and Generalized in the Setting of Zadeh's Possibility Theory, *Information Systems*, 9(1), 1984, pp.27-42.

[22] Prad, H., The Connection between Lipski's Approach to Incomplete Information Data Base and Zadeh's Possibility Theory, *Proceedings of the International Conference on Systems Methodology*, 5-9, Jan. 1982, pp.402-408.

[23] Prad, H. and Testemale, C., Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries, *Information Sciences*, 1984, pp.115-143.

[24] Takahashi, T., A Fuzzy Query Language for Relational Databases, *IEEE Transactions on Systems, Man and Cybernetics*, Vol.21, No.6, November/December, 1991.

[25] Zemankava, M. and Kandel, A., Implementing Imprecision in Information Systems, *Information Sciences*,1985, pp.107-141.