Cybernetics and Systems: An International Journal
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/ucbs20

ESTIMATING NULL VALUES IN THE DISTRIBUTED RELATIONAL DATABASES ENVIRONMENT

Shyi-Ming Chen, Hsin-Horng Chen
[a] Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C.
[b] Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.
Published online: 29 Oct 2010.

Cybernetics and Systems: An International Journal, 31:851-871, 2000
Copyright © 2000 Taylor & Francis
0196-9722/00 $12.00 + .00

ESTIMATING NULL VALUES IN THE DISTRIBUTED RELATIONAL DATABASES ENVIRONMENT

SHYI-MING CHEN

Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C.

HSIN-HORNG CHEN

Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

To estimate null values in relational database systems is an important research topic. In Chen and Yeh (1997) a method for estimating null values in relational database systems was presented. In Chen and Chen (1997) a method for fuzzy query translation for information in the distributed relational databases environment was presented. In this article, the works of Chen and Chen (1997) and Chen and Yeh (1997) are extended to propose a method for estimating null values in the distributed relational databases environment. The proposed method provides a useful way to estimate incomplete data when the relations stored in a failed server cannot be accessed in the distributed relational databases environment.

Data processing is an important activity in business processing. There is a lot of information generated when business is running. A database system keeps that information for the enterprise. There are many types

This work was supported in part by the National Science Council, Republic of China, under Grant NSC 87-2213-E-009-011.

Address correspondence to Professor Shyi-Ming Chen, PhD, Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C. E-mail: smchen@et.ntust.edu.tw

of database systems on the commercial market. Relational database systems are most widely used in an enterprise (Date, 1995). Sometimes a company is distributed logically into divisions, where each division may have a database. Thus, a distributed system enables the distributed databases to mirror the structure of the company, where local data can be kept locally and remote data can be accessed when necessary by means of computer networks. It may happen that one of the distributed databases stored in a failed server cannot be accessed when queries are submitted from the end users. In this case, a mechanism is needed to derive the incomplete data before the failed server has recovered to the normal state.

Since Zadeh (1965) proposed the theory of fuzzy sets in 1965, some researchers investigated the application of the fuzzy set theory in relational database systems. In Chen and Jong (1997), the techniques of fuzzy query translation for relational database systems were presented. In Yeh and Chen (1994), a method for fuzzy query processing was presented using automatic clustering techniques. In Chang and Ke (1978) they presented a database skeleton and introduced its application to fuzzy query translation. In Chang and Ke (1979) a method for translation of fuzzy queries of relational database systems was presented. In Bosc et al. (1988) an extension of DBMS querying capabilities was proposed in order to allow fuzzy queries to a usual database. In Bosc and Pivert (1995) a fuzzy query language SQLf for relational database systems was presented. In Nakajima and Senoh (1995) the operations of fuzzy data in fuzzy SQL language was investigated. In Zemankova (1989) he proposed a fuzzy intelligent information system. In Kacprzyk et al. (1987) a ‘‘human-consistent’’ database querying system was developed based on fuzzy logic with linguistic quantifiers.

In Chen and Chen (1997) a method for fuzzy query translation was presented based on Chen and Jong (1997) for information retrieval in the distributed relational databases environment. In Chen and Yeh (1997) a method for estimating null values in relational database systems was presented. In this paper, the works of Chen and Chen (1997) and Chen and Yeh (1997) are extended to propose a method for estimating null values in the distributed relational databases environment, when one of the relations stored in a failed server cannot be accessed in the distributed relational databases.

FUZZY QUERY TRANSLATION IN THE DISTRIBUTED RELATIONAL DATABASES ENVIRONMENT

In Chen and Chen (1997), a method for fuzzy query translation for information retrieval has been presented in the distributed relational databases environment based on Chen and Jong (1997), where the developing tools Delphi and open database connectivity (ODBC) are used to implement the distributed relational databases query environment. In the following, the ODBC architecture and the distributed relational databases query environment from Chen and Chen (1997) is briefly reviewed.

The ODBC interface allows applications to access data in database management systems (DBMS) using structured query language (SQL) as a standard for accessing data.

The ODBC architecture has four components:

1. Application: it performs processing and calls ODBC functions to submit SQL statements and retrieve results.
2. Driver manager: it loads drivers according to the behavior of an application.
3. Driver: it processes ODBC function calls, submits SQL requests to a specific data source, and returns results to the application. If necessary, the driver modifies an application's request such that the request conforms to the syntax supported by the associated DBMS.
4. Data source: it consists of the data in which the user wants to access and its associated operating system, DBMS, and network platform (if any) used to access the DBMS.

The hierarchy of the ODBC architecture is shown in Figure 1 (Chen & Chen, 1997). Figure 2 shows the fuzzy query translation architecture in the distributed relational databases environment (Chen & Chen, 1997). There are four parts of components in Figure 2. These components are described as follows:

1. Fuzzy SQL statements: this component contains the SQL statements that are issued by the users. The statements can be either standard SQL statements or fuzzy SQL statements.
2. Fuzzy SQL translator: this component parses the query statements issued from the component "fuzzy SQL statements" and performs the following operations based on Chen and Jong (1997):

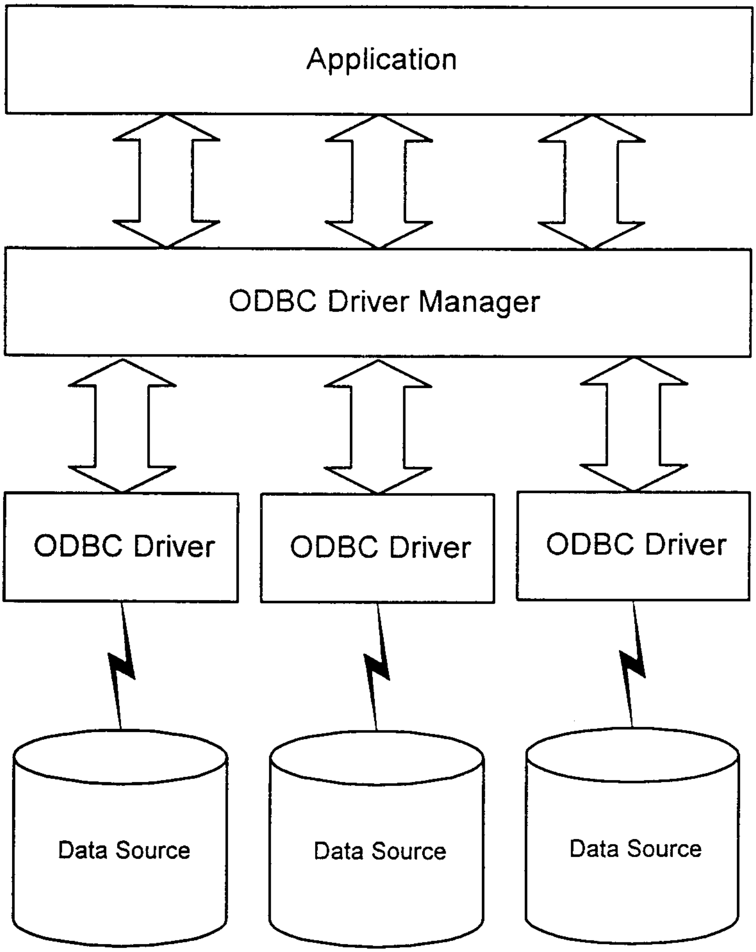

Figure 1. The hierarchy of the ODBC architecture (Chen & Chen, 1997).

a. When the query statement is not a legal SQL statement or fuzzy SQL statement, this component will reject this query statement.
b. If the query statement is a fuzzy SQL statement, this component will translate it into a standard SQL statement. According to the linguistic term corresponding to the attribute in the condition part and the retrieve threshold value written in the query statement, the translator will perform the α-cut operation on the fuzzy term and translate the condition part into the precise

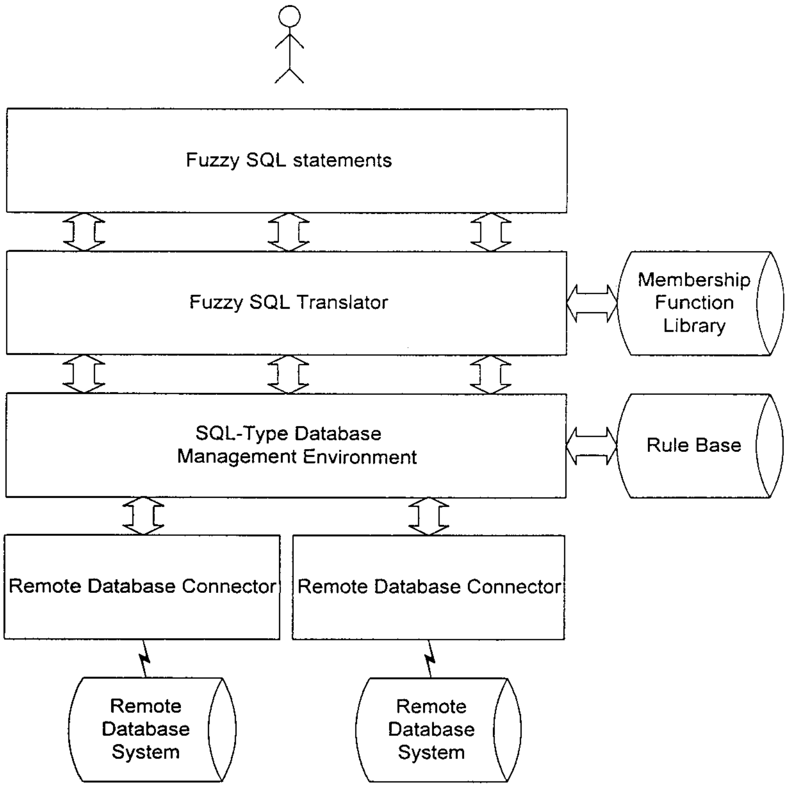
Fuzzy SQL statements
Fuzzy SQL Translator
Membership Function Library
SQL-Type Database Management Environment
Rule Base
Remote Database Connector
Remote Database Connector
Remote Database System
Remote Database System

Figure 2. Fuzzy query translation in the distributed relational databases environment (Chen & Chen, 1997).

SQL statement, and then pass this translated query statement to the next component.

c. If the query statement is a standard SQL statement, this component just passes the whole SQL statement to the next component without any modification.

The membership function library is used in this component. It keeps the membership functions which are used when doing the α-cuts operations.

3. SQL-type database management environment: this component takes charge of access data from data sources. Database operations are actually performed by this component. When the fuzzy SQL translator passes the precise SQL statement to this component, it will take

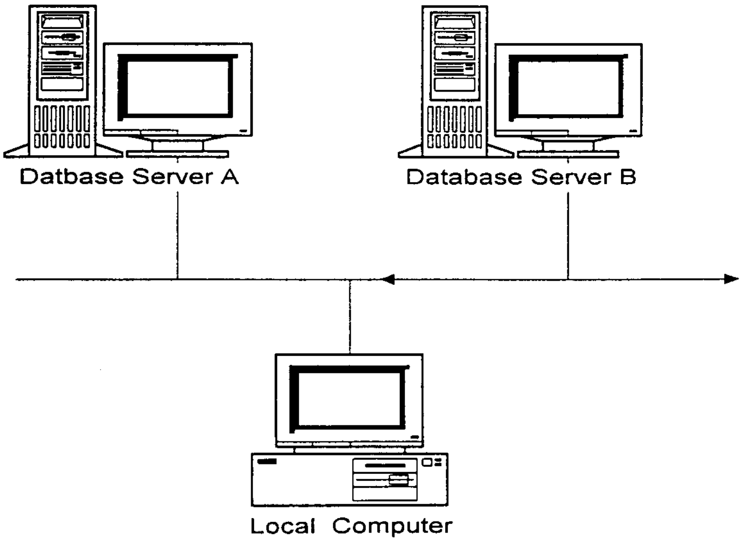

Figure 3. The environment of the implementation (Chen & Chen, 1997).

some necessary operations to deal with this query statement. For example, if there are some databases residing in remote sites, this component maintains the communication links between the remote databases and the local program to ensure the successful access to the databases.

4. Remote database connectors: these components serve as the communication devices between the local program and the remote database servers. These remote database connectors are database dependent, i.e., each type of database has its specific driver. Through these drivers, the program will not necessarily decide what kind of database system it will deal with. Therefore, the methods for accessing databases are totally transparent to the program.

The architecture of this system is based on the client/server database operating environment. This system resides at the user's local site as a client. The databases can be located either in the local site or in the remote site. These two database servers act as the server sites. The distributed relational databases environment presented in Chen and Chen (1997) is shown in Figure 3.

	v_1	v_2	...	v_n
v_1	1	u_{12}	...	u_{1n}
v_2	u_{21}	1	...	u_{2n}
⋮	⋮	⋮	⋮	⋮
v_n	u_{n1}	u_{n2}	...	1

Figure 4. A fuzzy relation matrix for the linguistic variable V.

Estimating Null Values in the Distributed Relational Databases Environment

In Chen and Yeh (1997), a method for estimating null values in relational database systems has been presented. In the following, a method for estimating null values in the distributed relational databases environment is presented.

A distributed relational databases environment is a new trend for business data processing. Sometimes a business distributes its data to each related department. Each department manages its own data. This is for the convenience of speed and management. However, there are some cases that database servers fail when end-users are querying databases. In this situation, a method must be provided to estimate the data stored in the failed server.

In the following, a fuzzy similarity matrix is used to represent a fuzzy relation. Suppose a linguistic variable (Zadeh, 1975a,b,c) V is defined, which contains linguistic terms $v_1, v_2, \ldots, v_n$. Then, a fuzzy relation matrix is made for the linguistic variable V as shown in Figure 4, where u_{ij} is the closeness degree between v_i and v_j, $u_{ij} \in [0, 1]$, $1 \leq i \leq n$, and $1 \leq j \leq n$.

One can say that the closeness degree $CD(v_i, v_j)$, between the fuzzy term v_i and the fuzzy term v_j is u_{ij}, denoted as

$$CD_V(v_i, v_j) = Rv[v_i][v_j] = u_{ij}, \tag{1}$$

where each value of u_{ij} in the fuzzy relation matrix V is defined by an experienced database administrator.

For example, in Table 1, "DEGREE" is a linguistic variable. Suppose this linguistic variable contains fuzzy terms {Ph.D., Master, Bac-

Table 1. A fuzzy relation for the linguistic variable "DEGREE"

	Ph.D.	Master	Bachelor
Ph.D.	1.0	0.7	0.4
Master	0.7	1.0	0.6
Bachelor	0.4	0.6	1.0

Table 2. A fuzzy relation table

X	Y	$CD(X, Y)$
Ph.D.	Ph.D.	1.0
Master	Ph.D.	0.7
Bachelor	Ph.D.	0.4
Ph.D.	Master	0.7
Master	Master	1.0
Bachelor	Master	0.6
Ph.D.	Bachelor	0.4
Master	Bachelor	0.6
Bachelor	Bachelor	1.0

helor}. The fuzzy relation matrix can be constructed for the linguistic variable "DEGREE."

For saving storage, one can, as shown in Table 1, construct the fuzzy relation matrix illustrated in Table 1 by using the fuzzy relation table shown in Table 2.

From Table 2, the closeness degree can be seen between the fuzzy terms "Master" and "Ph.D." is $CD_{\text{degree}}(\text{Master, Ph.D.}) = 0.7$.

A fuzzy term ranking function can also be defined to keep the rank of each fuzzy term in the same linguistic domain. Suppose the linguistic term V contains fuzzy terms $\{v_1, v_2\}$, and fuzzy term v_1 is prior to fuzzy term v_2, then it can be defined as follows:

$$\text{Rank}(v1) > \text{Rank}(v2).$$

A rule base is used for keeping relationships in which one attribute determines other attributes (Yen, 1993). Each rule in the rule base is given by the experts according to their experiences of the data. For example, if the attributes $A_1, A_2, \ldots,$ and A_n determine the attribute AA, then a set of fuzzy rules can be constructed as shown in Table 3.

Table 3. Rules for determining the attribute AA

Rule 1:
IF $A_1 = a_{11}(W = w_{11})$ AND $A_2 = a_{12}(W = w_{12})$ AND
$\cdots$ AND $A_n = a_{1n}(W = w_{1n}$, THEN $AA = t_1$
Rule 2:
IF $A_1 = a_{21}(W = w_{21})$ AND $A_2 = a_{22}(W = w_{22})$ AND
$\cdots$ AND $A_n = a_{2n}(W = w_{2n})$ THEN $AA = t_2$
$\vdots$
Rule m:
IF $A_1 = a_{m1}(W = w_{m1})$ AND $A_2 = a_{m2}(W = w_{m2})$ AND
$\cdots$ AND $A_n = a_{mn}(W = w_{mn})$ THEN $AA = t_m$

Table 4. Rules for determining the attribute "SALARY"

IF DEGREE = Master (W = 0.6) AND EXPERIENCE = 3.6 (W = 0.4) THEN SALARY = 41000
IF DEGREE = Ph.D. (W = 0.6) AND EXPERIENCE = 7.8 (W = 0.4) THEN SALARY = 65000
IF DEGREE = Bachelor (W = 0.65) AND EXPERIENCE = 5 (W = 0.35) THEN SALARY = 36000

In Table 3, one can see that a_{ij} denotes the attribute value, w_{ij} is the weight for each attribute in the antecedent part of the rule, and each w_{ij} is assigned by an experienced expert, where $w_{ij} \in [0, 1]$, $1 \le i \le m$, and $1 \le j \le n$ For example, in a relation EMPLOYEE, one can define that the attribute "DEGREE" and the attribute "EXPERIENCE" determine the attribute "SALARY," and each weight value can be assigned to the attributes "DEGREE" and "EXPERIENCE", then one can get the rules for the attribute "SALARY" as shown in Table 4.

The rule base shown in Table 4 stores important information about estimating null values. It will be used to estimate failed attribute values of a relation in the distributed relational databases environment.

Sometimes a relation may be partitioned into two or more related relations by the method of the relation normalization (Codd, 1971). Moreover, they may be distributed to different locations in the business, just for convenient management of the database. These relations are obtained by vertically partitioning the original relation. It might happen that one of the relations fails when a query is submitted. In such a case, a method is proposed to estimate the failed attribute value of a relation in the distributed relational database environment.

Table 5. Relation EMPLOYEE

ID	Degree	Experience	Salary
S1	Ph.D.	7.2	63000
S2	Master	2.0	37000
S3	Bachelor	7.0	40000
S4	Ph.D.	1.2	47000
S5	Master	7.5	53000
S6	Bachelor	1.5	26000
S7	Bachelor	2.3	29000
S8	Ph.D.	2.0	50000
S9	Ph.D.	3.8	54000
S10	Bachelor	3.5	35000
S11	Master	3.5	40000
S12	Master	3.6	41000
S13	Master	10.0	68000
S14	Ph.D.	5.0	57000
S15	Bachelor	5.0	36000
S16	Master	6.2	50000
S17	Bachelor	0.5	23000
S18	Master	7.2	55000
S19	Master	6.5	51000
S20	Ph.D.	7.8	65000
S21	Master	8.1	64000
S22	Ph.D.	8.5	70000

Consider the relation EMPLOYEE shown in Table 5. This relation can be vertically partitioned into two relations EMP_INFO and EMP_SALARY as shown in Table 6 and Table 7.

Then, these two relations can be joined (i.e., EMP_INFO and EMP_SALARY) into the original relation EMPLOYEE with the following SQL statements:

```
SELECT    ID, DEGREE, EXPERIENCE, SALARY
FROM      EMP_INFO, EMP_SALARY
WHERE     EMP_INFO.ID = EMP_SALARY.ID
WITH      RSV = 0.90,
```

where RSV means the retrieval threshold value (Chen & Chen, 1997). Suppose these two relations are distributed in two different locations. Then, a view called EMP can be created by the following SQL statements:

Table 6. Relation EMP_SALARY

ID	Salary
S1	63000
S2	37000
S3	40000
S4	47000
S5	53000
S6	26000
S7	29000
S8	50000
S9	54000
S10	35000
S11	40000
S12	41000
S13	68000
S14	57000
S15	36000
S16	50000
S17	23000
S18	55000
S19	51000
S20	65000
S21	64000
S22	70000

```
CREATE   VIEW EMP AS
SELECT   ID, DEGREE, EXPERIENCE, SALARY
FROM     EMP_INFO, EMP_SALARY
WHERE    EMP_INFO.ID = EMP_SALARY.ID
WITH     RSV = 0.90.
```

We know that the results of the relation EMPLOYEE and the view EMP will be identical. Now, suppose the relation EMP_SALARY failed to access when the query is submitted as follows:

```
SELECT   *
FROM     EMP
WHERE    ID = 'S1'
WITH     RSV = 0.90.
```

In this case, the query cannot be processed correctly. Thus, a method must be proposed to estimate the attribute values in a failed relation.

Table 7. Relation EMP_INFO

ID	Degree	Experience
S1	Ph.D.	7.2
S2	Master	2.0
S3	Bachelor	7.0
S4	Ph.D.	1.2
S5	Master	7.5
S6	Bachelor	1.5
S7	Bachelor	2.3
S8	Ph.D.	2.0
S9	Ph.D.	3.8
S10	Bachelor	3.5
S11	Master	3.5
S12	Master	3.6
S13	Master	10.0
S14	Ph.D.	5.0
S15	Bachelor	5.0
S16	Master	6.2
S17	Bachelor	0.5
S18	Master	7.2
S19	Master	6.5
S20	Ph.D.	7.8
S21	Master	8.1
S22	Ph.D.	8.5

The basic idea of this method is that the failed tuple compares to each rule in the rule base shown in Table 1 to see which is the closest rule. Then, the failed tuples can be estimated by their closeness degrees.

Assume that relation R_1 contains attributes $A_1, A_2, \ldots, A_m$ and assume that relation R_2 contains attributes B_1 and B_2. Furthermore, assume that attribute A_1 and B_1 are the primary keys of R_1 and R_2, respectively, and A_1 and B_1 are in the same domain. If one performs a joint operation on R_1 and R_2 with a condition "$R_1.A_1 = R_2.B_1$," then one can derive a new relation named R_3, where relation R_3 contains attributes $A_1, A_2, \ldots, A_m$ and B_2. If attribute B_2 is a failed attribute, then attribute B_2 can be estimated by the following method.

Case 1: First, each tuple T_i in relation R_3 is scanned. If attribute B_2 is in a numerical domain, then the closeness degree can be computed between tuple T_i and the chosen rule according to the rules that determine attribute B_2. If rule r_j is the closest to T_i, then that rule is picked as the rule base, where rule r_j is shown as follows:

$$\begin{aligned}&\text{IF } A_1 = A_{j1}(W = w_{j1},\ \text{AND } A_2 = A_{j2}(W = w_{j2})\\ &\quad \text{AND } \cdots \text{ AND } A_n = A_{jn}(W = w_{jn}) \text{ THEN } B_2 = N_j,\end{aligned}$$

where attribute A_k may be either a numerical or a nonnumerical domain, where $1 \le k \le n$. The following two cases are considered:

1. A$_k$ is in a numerical domain.
   If A_k is in a numerical domain, its closeness degree can be directly computed by the following formula:

$$CDv_{ji}(r_j.A_k, T_i.A_k) = \frac{T_i.A_k}{r_j.A_k}. \tag{2}$$

2. A_k is in a nonnumerical domain.
   If A_k is in a nonnumerical domain and the linguistic term rank $\text{Rank}(T_i.A_k) > \text{Rank}(r_j.A_k)$, then we can look up the closeness degree $CDv_{ji}(r_j.A_k, T_i.A_k)$ from the fuzzy relation matrix shown in Table 1, where

$$CDv_{ji}(r_j.A_k, T_i.A_k) = \frac{1}{R_{\text{domain}}[r_j.A_k][T_i.A_k]}. \tag{3}$$

   If A_k is in a nonnumerical domain and the linguistic term priority $\text{Rank} T_i.A_k) \le \text{Rank}(r_j.A_k)$, then one can look up the closeness degree $CDv_{ji}(r_j.A_k, T_i.A_k)$ from the fuzzy relation matrix shown in Table 1, where

$$CDv_{ji}(r_j.A_k, T_i.A_k) = R_{\text{domain}}[r_j.A_k][T_i.A_j]. \tag{4}$$

   The total closeness degree between tuple T_i and Rule r_j is

$$CD(r_j.T_i) = \sum_{k=1}^{n} CDv_{ji}(r_j.A_k, T_i.A_k) * w_{jk}. \tag{5}$$

   After all closeness degrees are computed between tuple T_i and all rules r_j, the rule closest to T_i is chosen. Suppose the following rule is the closest one to tuple T_i:

$$\text{IF } A_1 = A_{j1}(W = w_{j1}) \text{ AND } A_2 = A_{j2}(W = w_{j2}) \\ \text{AND } \cdots \text{ AND } A_n = A_{jn}(W = w_{jn}) \text{ THEN } B_2 = N_j,$$

then the attribute value of B_2 in tuple T_i can be estimated by the following calculation:

$$T_i.B_2 = CD(r_j, T_i) * N_j \tag{6}$$

Repeat each tuple T_i until all failed attributes have been estimated.

Case 2: If the failed attribute B_2 is in a nonnumerical domain, the closeness degree can also be computed between tuple T_i and the chosen rule according to the rules that determine attribute B_2. If rule r_j is the closest to T_i, then that rule is picked as the base rule, where rule r_j is shown as follows:

$$\text{IF } A_1 = A_{j1}(W = w_{j1}) \text{ AND } A_2 = A_{j2}(W = w_{j2}) \\ \text{AND } \cdots \text{ AND } A_n = A_{jn}(W = w_{jn}) \text{ THEN } B_2 = W_j,$$

where attribute A_k may be either a numerical or a non-numerical domain, where $1 \le k \le n$. We consider the following two cases:

1. A_k is in a numerical domain.
   If A_k is in a numerical domain, we can directly compute its closeness degree by the following formula:

$$CDv_{ji}(r_j.A_k, T_i.A_k) = \frac{T_i.A_k}{r_j.A_k}. \tag{7}$$

2. A$_k$ is in a non-numerical domain.
   If A_k is in a non-numerical domain and the linguistic term rank $\text{Rank}(r_j.A_k) > \text{Rank}(r_j.A_k)$, then we can look up the closeness degree $CDv_{ji}(r_j.A_k, T_i.A_k)$ from the fuzzy relation matrix shown in Table 1, where

$$CDv_{ji}(r_j.A_k, T_i.A_k) = \frac{1}{R_{\text{domain}}[r_j.A_k][T_i.A_k]}. \tag{8}$$

If A_k is in a nonnumerical domain and the linguistic term priority $\text{Rank}(T_i.A_k) < \text{Rank}(r_j.A_k)$, then we can look up the closeness degree from the fuzzy relation matrix:

$$CD_{ji}(r_j.A_k, T_i.A_k) = R_{\text{domain}}[r_j.A_k]\,[T_i.A_k]. \tag{9}$$

The total closeness degree between tuple T$_i$ and Rule r$_j$ is

$$CD(r_j, T_i) = \sum_{k=1}^{n} CD_{ji}(r_j.A_k, T_k.A_k) * w_{jk}. \tag{10}$$

After all the closeness degrees are computed between tuple T_i and all rules, the rule closet to T_i is chosen. Suppose the following rule is the closest one to tuple T_i:

IF $A_1 = A_{j1}(W = w_{j1})$ AND $A_2 = A_{j2}(W = w_{j2})$
AND $\cdots$ AND $A_n = A_{jn}(W = w_{jn})$ THEN $B_2 = W_j$,

then the estimated attribute value of B_2 is W_j, where

$$T_i.B_2 = W_j. \tag{11}$$

Repeat each tuple T_i until all failed attributes have been estimated.

AN EXAMPLE

In the following, an example is used to illustrate the null values estimation process in the distributed relational databases environment.

Example: Consider the relation EMPLOYEE shown in Table 5. This relation can be vertically partitioned into two relations EMP_INFO and EMP_SALARY as shown in Tables 6 and 7.

Then, these two relations can be joined into the original relation EMPLOYEE with the following SQL statements:

```
SELECT    ID, DEGREE, EXPERIENCE, SALARY
FROM      EMP_INFO, EMP_SALARY
WHERE     EMP_INFO.ID = EMP_SLARY.ID
WITH      RSV = 0.90.
```

Suppose these two relations are distributed in two different locations. Then, a view can be created called EMP by the following SQL statements:

```
CREATE    VIEW EMP AS
SELECT    ID, DEGREE, EXPERIENCE, SALARY
FROM      EMP_INFO, EMP_SALARY
WHERE     EMP_INFO.ID = EMP_SALARY.ID
WITH      RSV = 0.90.
```

One can see that the results of the relation EMPLOYEE and the view EMP are identical. Now suppose the relation EMP_SALARY failed to access when a query is submitted shown as follows:

```
SELECT    *
FROM      EMP
WHERE     ID = S1
WITH      RSV = 0.90.
```

In this case, the query cannot be processed correctly. Thus, the attributes must be estimated in the failed relation. According to the problem described above, when the database system that stores the relation EMP_SALARY failed, the query submitted by the user cannot be executed. In this case, the join result may be like the failed result shown in Table 8.

Now, all the data will be estimated in the relation EMP_SALARY. Every tuple is scanned in the relation EMP_INFO. First, the first tuple in the relation EMP_INFO is picked. From the rules for the attribute "SALARY," one knows that the attribute "DEGREE" and the attribute "EXPERIENCE" determine the attribute "SALARY." Thus, the closeness degree is calculated between the first tuple of the relation EMP_SALARY to each rule for the attribute "SALARY." For example, the first rule (i.e., Rule 1) is chosen in the rule base shown in Table 4 as follows:

$$\text{IF DEGREE} = \text{Master}(W = 0.6) \text{ AND EXPERIENCE} = 3.6(W = 0.4)$$
$$\text{THEN SALARY} = 41000.$$

In the condition part of this rule, one can see that "DEGREE = Master"

Table 8. A failed join result

ID	Degree	Experience	Salary
S1	Ph.D.	7.2	?????

is a nonnumerical domain. One cannot directly calculate the closeness degree between "Master" and "Ph.D." One can see that Rank(Ph.D.) > Rank(Master). Therefore, their closeness degree is computed using formula (3):

$$\begin{aligned} CD_{\text{degree}}(\text{Master},\ \text{Ph.D.}) &= \frac{1}{R_{\text{degree}}[\text{Master}][\text{Ph.D.}]} \\ &= \frac{1}{0.7} \\ &= 1.428571428571\,. \end{aligned}$$

For the attribute "EXPERIENCE," one can see that it is a numerical domain attribute. According to formula (2), their closeness degree can be directly calculated:

$$CD_{\text{experience}}(3.6, 7.2) = \frac{7.2}{3.6} = 2.$$

From Rule 1 shown in Table 4, one can see that the weights for the attributes "DEGREE" and "EXPERIENCE" are 0.6 and 0.4, respectively. Then, the total closeness degree is calculated between the first tuple of the relation EMP_INFO and Rule 1 using formula (5):

$$\begin{aligned} CD(\text{Rule1},\ \text{EMP}.T_1) &= 1.428571428571 \times 0.6 + 2 \times 0.4 \\ &= 1.657142857143\,. \end{aligned}$$

Again, the closeness degree is calculated between the first tuple of the relation EMP_INFO and the rest rules in the rule base for the attribute "SALARY." The same method described above is used to calculate each CD value:

$$\begin{aligned} CD(\text{Rule 2},\ \text{EMP}.T_1) &= 1 \times 0.6 + 0.9230769230769 \times 0.4 \\ &= 0.969230769231\,, \\ CD(\text{Rule 3},\ \text{EMP}.T_1) &= 2.5 \times 0.65 + 1.44 \times 0.35 = 2.129. \end{aligned}$$

In this case, the closest rule is chosen to estimate the value of the attribute "SALARY," say, Rule 2. By formula (6), one can estimate the first tuple of the relation of the failed attribute "SALARY" as follows:

$$\text{EMP}.T_1.\text{SALARY} = 0.96923069231 \times 65000 = 63000.$$

Table 9. An estimated relation for the relation EMP_INFO

ID	Degree	Experience	Salary (estimated)
S1	Ph.D.	7.2	63000
S2	Master	2.0	33711
S3	Bachelor	7.0	46648
S4	Ph.D.	1.2	36216
S5	Master	7.5	56200
S6	Bachelor	1.5	27179
S7	Bachelor	2.3	29195
S8	Ph.D.	2.0	39861
S9	Ph.D.	3.8	48061
S10	Bachelor	3.5	32219
S11	Master	3.5	40544
S12	Master	3.6	41000
S13	Master	10.0	64533
S14	Ph.D.	5.0	55666
S15	Bachelor	5.0	35999
S16	Master	6.2	51866
S17	Bachelor	0.5	24659
S18	Master	7.2	55200
S19	Master	6.5	52866
S20	Ph.D.	7.8	65000
S21	Master	8.1	58200
S22	Ph.D.	8.5	67333

In the same way, after scanning the rest of the tuples in the relation to estimate the values for the failed attribute "SALARY," one can get the estimated relation shown in Table 9.

Compared to the original EMPLOYEE relation shown in Table 5, the estimated salaries of the employees and the estimation errors are shown in Table 10, where the estimated error is calculated as follows:

$$\text{Estimated Error} = \frac{\text{Estimated Value} - \text{Original Value}}{\text{Original Value}}. \tag{12}$$

In this paper, the estimated errors are represented using three digits of significant numbers.

From Table 10, one can see that the average estimated error is 0.061. That is, the forecasting accuracy of the example is 93.9%.

Table 10. An estimated relation compared to the original relation

ID	Degree	Experience	Salary (original)	Salary (estimated)	Estimated Error
S1	Ph.D.	7.2	63000	63000	+ 0.000
S2	Master	2.0	37000	33711	−0.089
S3	Bachelor	7.0	40000	46648	+ 0.166
S4	Ph.D.	1.2	47000	36216	−0.229
S5	Master	7.5	53000	56200	+ 0.060
S6	Bachelor	1.5	26000	27179	+ 0.045
S7	Bachelor	2.3	29000	29195	+ 0.007
S8	Ph.D.	2.0	50000	39861	−0.203
S9	Ph.D.	3.8	54000	48061	−0.110
S10	Bachelor	3.5	35000	32219	−0.079
S11	Master	3.5	40000	40544	+ 0.014
S12	Master	3.6	41000	41000	+ 0.000
S13	Master	10.0	68000	64533	−0.051
S14	Ph.D.	5.0	57000	55666	−0.023
S15	Bachelor	5.0	36000	35999	−0.000
S16	Master	6.2	50000	51866	+ 0.037
S17	Bachelor	0.5	23000	24659	+ 0.072
S18	Master	7.2	55000	55200	+ 0.000
S19	Master	6.5	51000	52866	+ 0.037
S20	Ph.D.	7.8	65000	65000	+ 0.000
S21	Master	8.1	64000	58200	−0.091
S22	Ph.D.	8.5	70000	67333	−0.038

CONCLUSIONS

In this paper, the works of Chen and Chen (1997) and Chen and Yeh (1997) have been extended to present a method for estimating null values in the distributed relational databases environment. The proposed method provides a useful way to estimate incomplete data when the relations stored in a failed server cannot be accessed in the distributed relational databases environment.

REFERENCES

Bosc, P., M. Galibourg, and G. Hamon. 1988. Fuzzy querying with SQL: Extensions and implementation aspects. *Fuzzy Sets and Systems* 28(3):333–349.

Bosc, P. and O. Pivert. 1995. SQLf: A relational database language for fuzzy querying. *IEEE Trans. Fuzzy Systems* 3(1):1–17.

Chang, S. K. and J. S. Ke. 1978. Database skeleton and its application to fuzzy query translation. *IEEE Trans. Software Engineering* 4(1):31–43.

Chang, S. K. and J. S. Ke. 1979. Translation of fuzzy queries for relational database systems. *IEEE Trans. Pattern Analysis and Machine Intelligence* PAMI-1(3):281–294.

Chen, H. H. and S. M. Chen. 1997. Fuzzy query translation for information retrieval in the distributed relational database environment. In *Proc. 6th National Conference on Science and Technology of National Defense*, Taoyuan, Taiwan, Republic of China. Vol. 2, 433–439.

Chen, S. M. and W. T. Jong. 1997. Fuzzy query translation for relational database systems. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics* 27(4):714–721.

Chen, S. M., J. S. Ke, and J. F. Chang. 1986. Techniques of fuzzy query translation for database systems. In *Proceedings of 1986 International Computer Symposium*, Tainan, Taiwan, Republic of China, Vol. 3, 1281–1290.

Chen, S. M. and M. S. Yeh. 1997. Generating fuzzy rules from relational database systems for estimating null values. *Cybernetics and Systems: An International Journal* 28(2):695–723.

Codd, E. F. 1971. Normalized data base structure: A brief tutorial. In *Proc. 1971 ACM SIGFIDET Workshop on DATA Description, Access, and Control*, San Diego, California.

Date, C. J. 1995. *An introduction to database systems*. Reading, MA: Addison–Wesley.

Kacprzyk, J., S. Zadrozny, and A. Ziokkowski. 1987. FQUERY III+: A "human-consistent" database querying system based on fuzzy logic with linguistic quantifiers. In *Proc. Second International Fuzzy Systems Association Congress*, Tokyo, Japan, 443–453.

Kaufmann, A. and M. M. Gupta. 1988. *Fuzzy mathematical models in engineering and management science*. The Netherlands: North–Holland.

Nakajima, H. and Y. Senoh. 1995. Development of fuzzy add-in macros with spread sheet. In *Proc. FUZZ-IEEE/IFES'95 Workshop on Fuzzy Database Systems and Information Retrieval*, Yokohama, Japan, 61–66.

Yeh, M. S. and S. M. Chen. 1994. A new method for fuzzy query processing using automatic clustering techniques. *Journal of Computers* 6(1):1–10.

Yen, S. J. 1993. Neighborhood/conceptual query answering with imprecise incomplete data. Master thesis, Institute of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, R. O. C.

Zadeh, L. A. 1965. Fuzzy sets. *Inform. Control* 8:338–353.

Zadeh, L. A. 1975b. The concept of a linguistic variable and its application to approximate reasoning—Part II. *Information Sciences* 8(4):301–357.

Zadeh, L. A. 1975c. The concept of a linguistic variable and its application to approximate reasoning—Part III. *Information Sciences* 9(1):43–80.

Zemankova, M. 1989. FIIS: A fuzzy intelligent information system. *Data Engineering* 11(2):11–20.

Zimmermann, H. J. 1991. *Fuzzy set theory and its applications*. Boston: Kluwer Academic Publishers.