Mining Full Functional Dependency to Answer Null Queries and Reduce Imprecise Information Based on Fuzzy Object Oriented Databases

Sujoy Dutta
Computer Science and Engineering
Kalinga Institute of Industrial Technology (KIIT University)
Bhubaneswar, India
sujoydutta.356@gmail.com

Dr. Laxman Sahoo
Computer Science and Engineering
Kalinga Institute of Industrial Technology (KIIT University)
Bhubaneswar, India

Debasis Dwibedy
Computer Science and Engineering
Kalinga Institute of Industrial Technology (KIIT University)
Bhubaneswar, India

Abstract—**Discovery of Full functional dependencies from relations has been identified as an important database analysis technique. In order to deal with information inexactness, fuzzy techniques have extensively been integrated with different database models and theories. However, the information is often vague or ambiguous and very difficult to represent in implementing the application software. This problem can be handled by getting secured information to reduce imprecise information. Null queries are queries that elicit a null answer from the database. In fuzzy object oriented database model, a theoretical framework utilizes analogical reasoning and fuzzy functional dependency or full functional dependency to answer null queries. In this paper, the concept of fuzzy functional dependency is extended to full functional dependency on similarity based fuzzy object oriented data model. In addition, a data mining algorithm is proposed to discover all functional dependencies among attributes. From this functional dependency, we shall be able to reach full functional dependency.**

Keywords- database; fuzzy; attributes; tuples; null query; functional dependency; partition; algorithms.

I. INTRODUCTION

Object oriented databases are considered better than the relational and other databases due to increasing demand of new approaches to deal with complex data, complex relationship existing among such data and large data intensive applications. A major goal for database research has been the corporation of additional semantics into the data model. In real-world applications, information is often vague or ambiguous or inexact. Therefore, different kinds of incomplete information have extensively been introduced into relational databases. However, object oriented database model in its extension of imprecise does not satisfy the need of modeling complex objects with imprecision and uncertainly.

Many studies have been carried out on the development of some database models to deal with complex objects and uncertain data together. Fuzzy techniques are used to represent, reclaim and store the complex, imperfect and incomplete information in object oriented databases. It can well represent the complex, imperfect object structures in object oriented data-bases without fragmentation framework of aggregate data and also form complex relationship among attributes. In recent years, extensive studies have been carried out on fuzzy relational and object-oriented databases [1-3, 5-9, 11, 15-18, 20-25]. There are two basic approaches to represent fuzzy information in the data model, i.e. model through similarity relations [3, 5-8, 11, 20] and model through possibility distribution [1, 15-18, 21, 24].

If the conditions specified in a query cannot completely be satisfied due to missing data in the database, explicit null values or no data items, null answers could be obtained. The queries, that attain a null answer from the database, are termed as null queries. Reasoning by analogy is an inference tool in human cognition and AI research. It consists of recognizing certain similarities between a source object tuple and a target object source and derive sealed properties of the target on the basis of its observed similarity with the source. A theoretical model was proposed by Dutta to answer null queries on possibility-based data model [4]. Analogical reasoning and fuzzy functional dependency were used in the model. However, the data with analogy over discrete domain were not considered. In discrete domain, the data are modeled better through similarity relations based on similarity based model [15].

In this study, the concept of functional dependency is extended to fully functional dependency on similarity-based fuzzy object oriented data model. In addition, a mining algorithm is proposed to discover all the full functional dependencies among attributes. The inference mechanism of analogical reasoning to answer null queries is also illustrated.

II. PREVIOUS WORK

It was assumed that the databases may have error and an attempt was made by using approximate functional dependencies to find out both functional and approximate dependencies [12]. The concept of fuzzy functional dependency was extended to approximate dependency on similarity based fuzzy relational data model. An approximate dependency almost holds a functional dependency [13]. The approach developed is a generalization of model of analogy developed for fuzzy relational data models. It utilizes the concept of contexts on domain attributes to deal with the issue of redundancy among object attributes and successfully obtain approximate answers for missing data values in an incomplete database. When there is a natural dependency between attributes, such dependencies arise in many databases. However, some tuples contain errors or represent exceptions to the rule.

III. OUR CONTRIBUTION

In this paper, an attempt has been made to reduce imprecise information over a database relation. Reduced imprecise information spanning multiple relations forms an interesting extension to previous work. It is assumed that the database is without error as well as exception free and the data within the database are related with Full Functional Dependency. The analogical reasoning is described in basis of full functional dependency to answer the null queries. Partition and equivalence classes are also used to find out the full functional dependency easily and efficiently. It helps to easily identify the erroneous or exceptional rows. In that way, a data base without error is described. Experiments show some algorithms that are efficient in practice and also applicable in much larger datasets.

IV. FUZZY OBJECT ORIENTED DATA MODEL ON SIMILARITY RELATIONS BASED ON FUZZY SIMALARITY BASE MODEL

Fuzzy similarity data base model is the concept of similarity based relation. It has been used as the source to generalize the equality to similarity and allows us to measure proximity of domain element. This consents the representation of indistinctness in data and inheritance. An object algebra is also introduced based on the extension of union, difference, product and selection.

A similarity-relation-based fuzzy object-oriented data model is an extension to the object-oriented data model that permits (1) class/object attribute values to be fuzzy predicates and numbers, and (2) class/object and class/subclass hierarchies to be fuzzy hierarchies.

In database, a class defines properties and behaviour usually for multiple instantiations. It is normally a collection of functional data type. Definition of objects is considered as an instance of classes. For instance, a file is an object: a collection of data and the associated read and write routines. In common speech one refers to the file as a class while the file is the object. A class is fuzzy because it has similar properties. Sometimes, class defined by the objects may be fuzzy. An object is fuzzy because of the lack of information. Sometimes objects, that have at least one attribute whose value is a fuzzy set, are fuzzy objects. May some objects are fuzzy, some objects belong to the class with the membership degree of [0; 1] when a class is intentionally defined.

Similarity-based fuzzy database has been recognized as a model that is most suitable for describing analogical data over discrete domains. Unlike the regular attribute domains, a simple extension is made on these domains. Basic concept of the model is reviewed in the following.

For each domain D**,** a similarity relation [23] **s** is defined over its domain elements, s: $D \times D \rightarrow [0,1]$. A similarity relation is the generalization of an equivalence relation. If $x, y, z \in D$, s can be defined as, reflexive: s(x,x) = 1.0

symmetric: s(x,y) = s(y, x)

transitive: s(x,z) = $Max_{\{y\}}$(Min[Sim(x,y), Sim(y,z)]) for all y∈D

Reflexive, Symmetric, Transitive: implies an equivalence relation. Fig. 1 illustrates the definition of similarity relation.

Sim(x, y)	**A**	**B**	**C**	**D**	**E**
A	1.0	0.8	0.4	0.5	0.8
B	0.8	1.0	0.4	0.5	0.9
C	0.4	0.4	1.0	0.4	0.4
D	0.5	0.5	0.4	1.0	0.5
E	0.8	0.9	0.4	0.5	1.0

reflexive: s(A,A) = 1.0

symmetric: s(A,B) = s(B,A)

transitive: s(A,C) = Max(Min[Sim(A,A), Sim(A,C)], Min[Sim(A,B), Sim(B,C)], Min[Sim(A,C), Sim(C,C)], Min[Sim(A,D), Sim(D,C)], Min[Sim(A,E), Sim(E,C)]) = Max(Min[1.0, 0.4], Min[0.8, 0.4], Min[0.4, 1.0], Min[0.5, 0.4], Min[0.8, 0.4]) = Max(0.4, 0.4, 0.4, 0.4, 0.4) = 0.4

Figure 1. Example of similarity relation.

A similarity-based fuzzy relational database is defined as a set of relations consists of tuples [3]. Fuzzy Tuple is any member of a fuzzy relation. For interpretation of a fuzzy tuple, it is essential to select any one element from each set of the tuple. The space of interpretations is the cross product ($D_1 \times D_2 \times \ldots \times D_n$). Let t_i represents the i-th tuple of a relation R in the form ($t_{i1}, t_{i2}, \ldots, t_{im}$), where t_{ij} is defined on the domain set D, $1 \leq j \leq m$. Allowing tuple component t_{ij} to be a subset of the domain D_j means that fuzzy information can be represented. This leads to the definition that Fuzzy Relation is essentially the subset of the cross product $P(D_1) \times P(D_2) \times \ldots \times P(D_n)$, where P(D) is the power set of the domain D.

Different degrees of similarity to the elements in each domain are introduced and compared with similarity relation for the representation of "fuzziness" in the fuzzy object-oriented data model based on fuzzy similarity database model. Table I illustrates a simple fuzzy object-oriented database representing NINE EMPLOYERS OF SCHOOL. Fig. 2 shows the similarity relations for domain attributing to JOB-POSITION**,** EXPERIENCE and SALARY respectively.

TABLE I. A FUZZY OBJECT-ORIENTED DATABASE RELATION

EMP#	**JOB-POSITION**	**EXP**	**SALARY**
1	Principal	7	60k
2	Dean	5	60k
3	Clark	2	10k
4	Teacher of Software Engineering	6	42k
5	Teacher of Operating system	3	42k
6	Clark	2	8k
7	Teacher of Software Engineering	3	35k
8	Teacher of Networking	5	?
9	Teacher of Operating system	2	35k

In tuple/row number 8, salary for Teacher of Networking is unknown i.e. null. If we take this data-base, it will lead to approximate functional dependency. So, in the beginning we assumed our data-base will be null or exception and error free. Therefore, we shall omit this tuple to get full functional dependency respective of approximate functional dependency.

	Principal	**Dean**	**Clark**	**Teacher of Software Engineering**	**Teacher of Operating system**	**Teacher of Networking**
Principal	1	0.5	0.7	0.5	0.4	0.2
Dean	0.5	1	0.5	0.7	0.4	0.2
Clark	0.7	0.5	1	0.5	0.4	0.2
Teacher of Software Engineering	0.5	0.7	0.5	1	0.4	0.4
Teacher of Operating system	0.4	0.4	0.4	0.4	1	0.4
Teacher of Networking	0.25	0.7	0.5	0.7	0.4	1

JOB-POSITION = {Principal, Dean, Teacher of Software Engineering, Teacher of Operating system, Teacher of Networking}

	2	3	5	6	7
2	1	0.8	0.6	0.6	0.4
3	0.8	1	0.6	0.6	0.4
5	0.6	0.6	1	0.8	0.6
6	0.6	0.6	0.8	1	0.6
7	0.4	0.4	0.6	0.6	1

EXPERIENCE

	8	10	35	42	60
8	1	0.7	0.5	0.2	0.2
10	0.7	1	0.5	0.2	0.2
35	0.5	0.5	1	0.2	0.2
42	0.2	0.2	0.2	1	0.4
60	0.2	0.2	0.2	0.4	1

SALARY

Figure 2. Similarity relation for attribute domains.

V. ANALOGICAL REASONING TO ANSWER NULL QUERIES

A. *Null query and Analogical Reasoning*

If a null answer is obtained for a query from the database, it is called null query. Let Q be a query imposed to the database R and the answer generated is Q(R). If Q is a null query, Q(R) is an empty set. Such situations occur if the conditions specified in the query cannot be satisfied by any data items or the database contains missing data. Our goal here is to present an approach for obtaining an approximate answer Q'(R) based on analogical reasoning, where Q'(R) is an approximation to Q(R). However, only incomplete similarity-based fuzzy database, i.e., only the cases of explicit null values and missing data in the database are considered here.

If the source object S and target object T have a similar property X and the source object S also has a property Y, It can be suggested from reasoning by analogy that the target object T may also possess the property Y. As for example, consider the relations shown in Table I to illustrate the attaining of approximate answer for null query using analogical reasoning based on similarity relation. In Table I the attribute Salary of tuple Employee#8 is missing and a query, salary(Employee#8)?, is presented to the data base system. A null answer is produced. Table I shows that Employee#4 has most similarity to Employee#8 in terms of JOB-POSITION and EXPERIENCE. Therefore, reasoning by analogy suggests that Employee#8 has similar salary like Employee#4. The most reasonable proximate salary of Employee#8, i.e., answer of the null query should be around 42K.

There are two issues to be resolved in this inference process. One is the mechanism of measuring the similarity where property (attribute) X can determine the unknown property (attribute) Y, e.g. X is {JOB-POSITION, EXPERINCE} and Y is SALARY. This is referred to as the problem of fuzzy functional dependency or proximate dependency between attributes. The other one is that, with respect to attribute X, which tuple S (e.g. Employee#4) is "most similar" to the tuple T (e.g. Enployee#8) with unknown attribute Y? This is referred to the problem of measuring the similarity between two tuple attribute values. Based on this analogical reasoning in the following sub section, we will define a similarity measure on similarity-based fuzzy object oriented data model to measure the similarity between two tuple components.

B. *Measuring Concept Similarities*

The similarity between two concepts is measured as the similarity shown by their super tuples. The super tuples contain bags of keywords for each attribute in the relation. The similarity between two super tuples is determined using Jaccard Coefficient [14, 19] with bag semantics.

The Jaccard Coefficient (Sim_J) is calculated as,

$$Sim_J(A,B) = \frac{|A \cap B|}{|A \cup B|}. \qquad (1)$$

The similarity between two fuzzy sets can be described as a similarity measure of two fuzzy sets in the same domain of universe. In the context of fuzzy object oriented data model based on similarity relations in similarity based model, a similarity measure for the object attributes U_{ij} and V_{kj} defined as follows:

Let U_{ij} and V_{kj} be object attributes defined on domain D_j with given similarity relation. The similarity Measure $\mu(U_{ij}, V_{kj})$ is defined as, $\mu(U_{ij}, V_{kj}) = \max S_j(X, Y)$ for all $X \in U_{ij}$ and $Y \in V_{kj}$.

VI. MINING FULL FUNCTIONAL DEPENDENCY

A. *Partitions and Full Functional dependencies*

A functional dependency $S \rightarrow A$ is minimal (in r) if A is not functionally dependent on any proper subset of S, i.e., if $P \rightarrow A$ does not hold in r for any P (proper subset) of S. The dependency $S \rightarrow A$ is trivial if $A \rightarrow S$.

Our approach to the discovery of dependencies is based on considering sets of tuples that agree on some set of attributes determining whether a dependency holds or not. It can be done by checking whether the tuples agree on the right-hand side whenever they agree on the left-hand side. The minimal cover of dependencies is also computed. Thus the approach extends naturally to full functional dependencies. The idea is described more formally by applying equivalence classes and partitions on relations.

If u [A] = v [A] for all **A** in **S,** the two tuples, u and v, are equivalent with respect to a given set **S** of attributes. Any attribute set **S** partitions the tuples of the relation into equivalence classes. Formally, for all pairs of tuples u and v ∈ R, the dependency holds or is valid in a given relation R over r. The set $\pi_S=\{[u]_S \mid u \in r\}$ of equivalence classes is a *partition* of r under S, i.e., π_S is a collection of disjoint sets (equivalence classes) of tuples such that each set has a unique value for the attribute set S and the union of the sets equals the relation r. The *rank* $|\pi|$ of a partition π is the number of equivalence classes in π.

Example 1: Consider the relation in Table II. Attribute A has value 1 only on tuples u_1 and u_2, so they form an equivalence class $[u_1]_{\{A\}} = [u_2]_{\{A\}} = \{1,2\}$. The whole partition with respect to A is $\pi_{\{A\}}=\{\{1, 2\}, \{3, 4, 5\}, \{6, 7, 8\}\}$, B is $\pi_{\{B\}}=\{\{1\}, \{2, 3, 4\}, \{5, 6\}, \{7, 8\}\}$, C is $\pi_{\{C\}}=\{\{1,3,4,6\}, \{2, 5, 7\}, \{8\}\}$, D is $\pi_{\{D\}}=\{\{1, 4, 7\},\{2\}, \{3\},\{5\},\{6\},\{8\}\}$, AB is $\pi_{\{AB\}}=\{\{1\},\{2\}, \{3, 4\}, \{5\}, \{6\}, \{7, 8\}\}$. The partition with respect to BC is $\pi_{\{BC\}}=\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\},\{6\},\{7\},\{8\}\}$.

TABLE II. AN EXAMPLE RELATION

Row Id	A	B	C	D
1	1	f	#	car
2	1	h	@	flower
3	2	h	#	fruit
4	2	h	#	car
5	2	f	@	bird
6	3	f	#	shirt
7	3	p	@	car
8	3	p	$	pant

The concept of partition refinement gives almost directly the functional dependencies. A partition π is a refinement of another partition π' if every equivalence class in π is a subset of some equivalence class of π'. In another word, a functional dependency $S \rightarrow A$ holds if and only if π_S refines $\pi_{\{A\}}$. The dependencies for full functional dependency will also be obtained using the minimal cover.

Example 2: Continuing Example 1 to find out whether the dependency {B, C}⟶A holds, we can compare the partitions $\pi_{\{B,C\}}$ and $\pi_{\{A\}}$ and check whether $\pi_{\{B,C\}}$ refines $\pi_{\{A\}}$. In the relation of Table II, the dependency holds since each equivalence class in $\pi_{\{B,C\}}$ is totally contained by some equivalence class in $\pi_{\{A\}}$.

The dependency {A}⟶B does not hold in the Table II. The equivalence class $[u_3]_{\{A\}}=\{3,4,5\}$, for instance, is not contained in any equivalence class in $\pi_{\{B\}}=\{\{1\}, \{2, 3, 4\}, \{5, 6\}, \{7, 8\}\}$.

There is an even simpler test for whether S ⟶ A holds or not. If π_S refines $\pi_{\{A\}}$, adding A to S does not break any equivalence classes of $\pi_{\{S\}}$**;** thus $\pi_{S\cup\{A\}}$ equals $\pi_{\{S\}}$. On the other hand, since $\pi_{S\cup\{A\}}$ always refines $\pi_{\{S\}}$**,** $\pi_{S\cup\{A\}}$ cannot have the same number of equivalence classes as $\pi_{\{S\}}$ unless $\pi_{S\cup\{A\}}$ and $\pi_{\{S\}}$ are equal. In another word, functional dependency S→ π holds if and only if $|\pi_S| = |\pi_{S\cup\{A\}}|$. Furthermore, the dependencies for full functional dependency will be obtained using the minimal cover.

B. Searching for full functional Dependency

A functional dependency S→A is minimal (in r) if A is not functionally dependent on any proper subset of S, i.e., if P→A does not hold in r for any P (proper subset) of S. The dependency S→A is trivial if A→S. In searching for full functional dependency, it is required to find all minimal non-trivial full functional dependencies that hold in a given relation R. To find all minimal non-trivial dependencies S→A, we search through the space of non-trivial dependencies and test the validity and minimality of each dependency. Searching is started from singleton sets of attributes and works its way to larger attribute sets through the set containment lattice level by level until the minimal dependencies that hold are found. It is required to compute partitions efficiently and test minimality. The collection of all possible left-hand sides of dependencies is the collection of all attribute sets. They constitute a set containment lattice as shown in Fig. 3 [10, 27]. Using a level wise algorithm, such lattices have been searched successfully in many data mining applications [27]. During this level wise search, false dependencies are eliminated as early as possible with some pruning techniques in order to reduce the search space. All possible right-hand sides are considered with a single breadth-first or level wise pass through the lattice. There is a one-to-one correspondence between the edges of the lattice and the non-trivial dependencies: we view an edge between sets S and S∪{A} as representing the non-trivial dependency S→A. The efficiency of the level wise algorithms depends on reduction of computation in each level by using results from previous levels. Now, it is described how to use previous levels to solve efficiently the tasks of computing partitions and testing minimality.

1) Computing partitions: It is not required to compute the partitions from scratch for every set of attributes considered. Indeed, the product of two earlier partitions in the previous level can be the partitions of the next level in the lattice. The product of two partitions π' and π'' denoted as $\pi'\pi''$ is the least refined partition π that refines both π' and π''. Therefore, for all S, P ⊆ R, $\pi_S . \pi_P = \pi_{S\cup P}$. For each A∈ R, the partitions π{A} are computed directly from the database. For |S| ≥ 2, partitions πS are computed as a product of partitions with respect to two subsets of S.

2) Testing minimality: When the algorithm is processing n set S, it will test dependencies of the form S\{A}→A, where A∈S. To test minimality of S\{A}→A, it is required to identify whether P\{A}→A holds for some proper subset S of P. This information is stored in the set C(S\{A}) of right hand side candidates of S\{A} for all A.

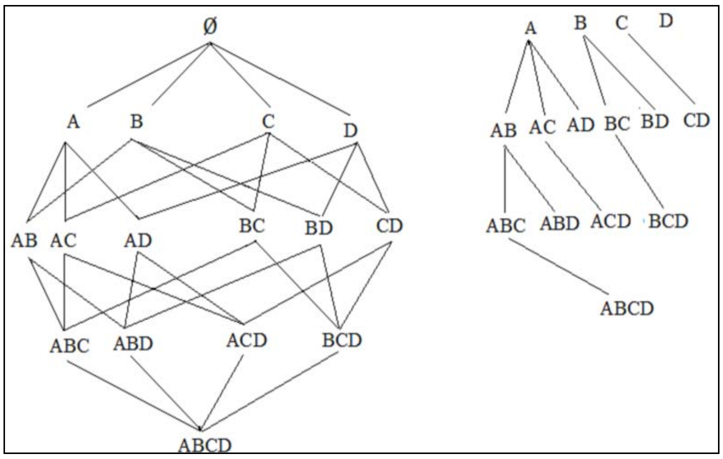

Figure 3. Set containment lattice into a tree structure For{A, B, C, D} representing the search space of all possible left-hand sides.

3) Pruning the search space: Pruning the search space means reducing the number of dependencies needed to be considered.

a) RHS candidate pruning: TANE works through the lattice until the minimal dependencies that hold are found. To test the minimality of a potential dependency S\{A}→A, it is required to know whether P\{A}→A holds for some proper subset P of S. The information is stored in the set Ҁ(P) of right-hand side candidates of Y.

It is found that **A** does not depend on any proper subset of a given set S for A∈Ҁ(S). More specifically, the collection of initial RHS candidates of a set S ⊆ R is Ҁ(S) = R\Ҁ(S), where Ҁ(S) = {A∈S|S\(A)⟶ A holds}.

To find minimal dependencies, it is sufficient to test dependencies S\{A}⟶ A, where A∈S and A∈(X\(B)) for all B ∈ S.

Pruning the search space in TANE is based on the fact that Ҁ(P) = Ø for all supersets P of S if Ҁ(S) = Ø. No dependency of the form P\{A}⟶ A can be minimal and the set P need not proceed at all. This information can be used effectively by the breadh-first search in the set containment lattice as illustrated in Fig. 2.

b) RHS$^+$ candidates: The minimality of the discovered dependencies can be guaranteed by the initial RHS candidates. However, in order to prune the search space more effectively, the improved RHS$^+$ candidate C$^+$(S) can be used. C$^+$(S) ={A∈R|∀B∈S:S\{A,B}→{B} does not hold}

VII. ALGORITHMS

The set containment lattice is searched in a level wise manner to find all valid minimal non-trivial dependencies. A Level L_l is the collection of attribute sets of size *l* in a set containment lattice such that the sets in L_l can potentially be used to construct based on dependencies. We start with L_1= {{A} |A∈R} and compute L_2 from L_1, L_3 from L_2 and so on. As for example,

L_1= {JOB-POSITION, EXPERIENCE, SALARY}

L_2 = {JOB POSITION, EXPERIENCE}, {JOB-POSITION, SALARY}, {EXPERIENCE, SALARY}.

For example L_0={Ø} and L_1={{A},{B},{C},{D}} in Fig. 3. They contain all the possible left hand side candidates of a potential functional dependency.

A. *Algorithm: level wise search of dependencies*

```
1. $L_0$ := {Ø}
2. $C^+(Ø)$ := R
3. $L_1$ := {{A} |A∈R}
4. $\ell$ :=1
5. While $L_\ell \neq Ø$
6.     COMPUTE-DEPENDENCIES($L_\ell$)
7.     PRUNE ($L_\ell$)
8.     $L_{l+1}$ := GENERATE-NEXT-LEVEL($L_l$)
9.     $\ell := \ell + 1$
```

B. *Algorithm: Pruning the search space*

```
Procedure PRUNE ($L_\ell$)
1. for each  S ∈ $L_\ell$ do
2. $C^+(S)$ := Ø do
3. delete S from $L_\ell$
4. if S is a (super)key do
5. for each A ∈ $C^+(S)$\S do
6. if $A \in \cap_{B \in S} C^+(S \cup \{A\} \setminus \{B\})$ then
7. output S→A
8. delete S from $L_\ell$
```

Procedure PRUNE implements the two pruning rules described above. By the first rule, X is deleted if $C^+(S) = Ø$. By the second rule, S is deleted if S is a key. In the later case, the algorithm may also output some dependencies. A dependency $S \rightarrow A$ is output on step 7 of the procedure PRUNE if and only if S is a key, $A \in C^+(S) \setminus S$ and $A \in C^+(\{S \cup A\} \setminus \{B\})$, for $B \in X$. It can be shown that the pruning does not cause the algorithm to miss any dependencies.

VIII. CONCLUSION

The major objective of this paper is to reduce imprecise information over databases. Different degrees of similarity to the elements in each domain are introduced and compared with similarity relation for the representation of "fuzziness" in the fuzzy object-oriented data model based on fuzzy similarity database model. An attempt has been made to answer null queries using analogical reasoning in basis of full functional dependency on similarity based fuzzy object-oriented data model. An algorithm to find out full functional dependencies from semantic relations has been provided. The approach is based on considering partitions of the relation and deriving valid dependencies from the partitions. The algorithm searches for dependencies in a level wise manner. We showed how the search space can be pruned effectively, and how the partitions and dependencies can be computed efficiently. Partition and equivalence classes are also used to find out the full functional dependency easily and efficiently. It helps to easily identify the erroneous or exceptional rows. In that way, a data base without error is described. This kind of facility will certainly improve the cooperative nature of objected-oriented databases and enhance the user friendliness of the database systems.

REFERENCES

[1] M. Anvari and G. F. Rose, "Fuzzy relational databases," Analysis of Fuzzy Information, in Bezdek, Ed., vol. I, CRC Press, Boca Raton, FL, 1987.

[2] P. Bosc and O. Pivert, "Fuzzy querying in conventional databases. In Fuzzy Logic for the Management of Uncertainty," Zadeh, L. and Kacprzyk, J. Eds, John Wiley, New York, 1992, pp.645-671.

[3] B. P. Buckles, and F. E. Petry, "A fuzzy representation of data for relational databases," Fuzzy Sets and Systems, vol. 7, 1982, pp.213-226.

[4] S. Dutta, "Approximate reasoning by analogy to answer null queries," Int. J. Appr. Reasoning, vol. 5, 1991, pp.373-398.

[5] B. P. Buckles and F. E. Petry, "Information-theoretic characterization of fuzzy relational databases," IEEE Trans. Systems Man Cybernetics, vol. 13, 1983, pp.74-77.

[6] B. P. Buckles and F. E. Petry, "Extending the fuzzy database with fuzzy numbers," Inform. Sci., vol. 34, 1984, pp.145-155.

[7] B. P. Buckles and F. E. Petry, "Query languages for fuzzy databases, in Management Decision Support Systems using Fuzzy Sets and Possibility Theory," Kacprzyk, J. and Yager, R.R., Eds., Verlag TUV Rheinland, Cologne, 1985, pp.241-152.

[8] B. P. Buckles, R. George and F. E. Petry, "Towards a fuzzy object-oriented model," Proc. of the NAFIPS-91 Workshop on Uncertainty Modeling in the 90's, 1991, pp.73-77.

[9] D. Dubois, H. Prade and J. P. Rossazza, "Vagueness, Typicality, and Uncertainty in Class Hierarchies," International Journal of Intelligent Systems, Vol. 6, 167-183, 1991.

[10] Y. Huhtala, J. Karkkainen, P. Porkka and H. Toivonen, "Eficient discovery of functional and approximate dependencies using partitions," Proceedings of IEEE Conference on Data Engineering, 1998, pp. 392-410.

[11] R. George, B. P. Buckles and F. E. Petry, "Modeling class hierarchies in the fuzzy object-oriented data Model," Fuzzy Sets and Systems, vol. 60, 1993, pp.259-272.

[12] Y. Huhtala, J. Karkkainen, P. Porkka and H. Toivonen, "TANE:An Efficient Algorithm for Discovering Functional and Approximate Dependencies," The Computer Jurnal, vol. 42, No. 2, 1999.

[13] J. Kivinen and H. Mannila, "Approximate dependency inference from relations," Theoretical Computer Science, vol. 149, No. 1, 1995, pp.129–149.

[14] T. Haveliwala, A. Gionis, D. Klein and P. Indyk, "Evaluating strategies for similarity search on the web," Proceedings of WWW, Hawai, USA, May 2002.

[15] J. M. Medina, M. A. Vila, J. C. Cubero and O. Pons, "Towards the implementation of a generalized fuzzy relational database model," Fuzzy Sets and Systems, vol. 75, 1995, pp.273-289.

[16] S. Parsons, "Current Approaches to Handling Imperfect Information in Data and Knowledge Bases," IEEE Transc. on Knowledge and Data Engineering, vol. 8, No. 3, June 1996.

[17] H. Prade, "Lipski's approach to incomplete information databases restated and generalized in the setting of Zadeh's possibility theory," Inform. Systems, vol. 9, 1984, pp.27-42.

[18] H. Prade and C. Testmale, "Generalizing database relational algebra for the treatment of incomplete or uncertain information and vague queries," Inform. Sci., vol. 34, 1984, pp.115-143.

[19] R. Baeza-Yates and B. Ribiero-Neto, "Modern Information Retrieval," Addison Wesley Longman Publishing, 1999.

[20] S. Shenoi and A. Melton, "Proximity relations in the fuzzy relational database model," Fuzzy Sets and System, vol. 31, 1989, pp.285-296.

[21] M. Umano, "Freedom-0: A fuzzy database system," in Gupta-Sanchez, Ed., Fuzzy Information and Decision Processes, North-Holland, Amsterdam, 1982.

[22] S. L. Wang and T. J. Hwang, "Answering Null Queries in Similarity-Relation-Based Fuzzy Relational Databases," International Conference of Industrial & Engineering Applications of AI and Expert Systems, 1997.

[23] L. A. Zadeh, "Similarity relations and fuzzy orderings," Inform Sci., vol 3, No. 1, March 1971, pp.177-200.

[24] M. Zemankova-Leech, and A. Kandel, “Fuzzy Relational Databases - A Key to Expert Systems,” Verlag TUV Rheinland, Cologne, 1985.

[25] R. Zicari, “Incomplete Information in Object-Oriented Databases,” SIGMOD RECORD, vol. 19, No. 3, 1990, pp.5-16.

[26] Y. Huhtala, J. Karkkainen, P. Porkka, and H. Toivonen, “Eficient discovery of functional and approximate dependencies using partitions,” Proceedings of IEEE Conference on Data Engineering, 1998, pp.392-410.

[27] H. Mannila and H. Toivonen, “Levelwise search and borders of theories in knowledge discovery,” Data Mining and Knowledge Discovery, vol. 1, No. 3, 1997, pp.241–258.