Volume 6, Issue 4, April 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Functional Dependencies Discovery in RDBMS

Akshay Kulkarni, Sachin Batule, Manoj Kumar Lanke, Adityakumar Gupta
Computer Department, VIIT, SPPU University,
Maharashtra, India

Abstract: Functional dependencies are important metadata used for schema normalization, data cleansing and many other task. The efficient discovery of functional dependencies in schemas is a well known challenge in database research and has seen several approaches. The discovery of functional dependencies from relations is an important analysis technique. We present TANE, a proficient algorithm for finding functional dependencies from larger databases. TANE is based on partitioning the sets of rows with respect to their attribute values which makes testing the validity of functional dependency fast even for big databases. The results have shown that the algorithm is faster in use. It is observed that for benchmark databases the running times have improved.

Keywords: TANE,

I. INTRODUCTION

Functional dependencies represent the relation between two set of attributes in a given database or it is unique relationship between the value of one attribute determined by the value of other attribute for example in the relation V the set of attribute r is functionally determined by the value of set of attribute y also in relation v,(r->y) if and only if the each value of r is associate with at mast one value of y. Functional dependency express relationship between attributes of a database relations. If V is a relation with attributes A and B ,a functional dependency between the attributes as A->B which specifies B is functional dependent on A.

The problem is to find all possible functional dependencies among attributes in a relational database [1] .The previous for discovering functional dependencies is based on repeated sorting and comparing of tuples to determine whether or not these tuples meet FD definition. This approach does not utilise the discovered FD's as knowledge to obtain new knowledge. As a result this approach is very responsive to the number of tuples and attributes, it is not practical for large databases._There are several methods that represent the functional dependencies from the relational databases, we describe the newer method that represent or discover functional and approximate dependencies.

We represent the TANE algorithm that carries new methods and ideas to determine whether the functional dependencies hold or not between two set of attributes. This method is based on representing the equivalence class partitioning of the set of tuples. This algorithm is efficient to discover and detect the functional dependencies from the thousand number of tuple in fraction of seconds [1].The proposed system consists of the following steps:

1) Partitioning
2) Pruning
3) Functional Dependency

II. LITERATURE SURVEY

Data mining is the process of examining larger pre-existing databases in order to discover knowledge from database. It is used to extract useful knowledge and information from the database. It uses some tools and techniques to analysis the information for extracting the patterns and relationship in larger databases that could be use.

We can use data mining algorithm and techniques for discovering functional dependency in larger databases to extract the useful information. There are several techniques we can use to discover the functional dependencies such as Apriori algorithm.

The problem is to find out all possible functional dependencies among the attribute in the relation database. The early method used for discovering functional dependency is based on repeatedly sorting and comparing the tuple to determine whether functional dependency meet by tuple or not. So this approach does not utilize the discovered functional dependency as knowledge to obtain new knowledge. As a result this approach is highly sensitive to the number of tuple and attributes, it is not practical for large database. TANE algorithms, by using these algorithms, there is no need to sort on any attribute or compare any value.

III. THE PROPOSED METHODOLOGY

Partition:

TANE algorithm works on partitions that represent the set of tuple values of each specific attribute in a given relational database. It works to partition the tuple in such way that the redundancies of tuple value of each attribute can easily determined and contributed. In a level wise manner the value of tuple of each attribute are computed and redundancies of each item in the attributes are represented in the appropriate subsets of each attributes through the redundancies of each item in the specific attribute are easily which will help to discover the functional dependencies from the databases [1].

For example [1] consider the relation in figure that represent the table in which the attribute A has value 1 in the tuple 1,tuple 2 so they form the equivalence class $[1]_{\{A\}}= [2]_{\{A\}} = \{1,2\}$ so we use identifier to each tuple to identify the each tuple, the whole partition with respect to A is $_A=\{\{1,2\},\{3,4,5\},\{6\},\{7\}\}$

Tuple Id	A	B	C	D
1	1	a	$	flower
2	1	b	$	lily
3	2	b	@	tulip
4	2	c	@	flower
5	2	d	#	orchid
6	3	d	#	rose
7	4	e	#	orchid

The remaining partitions with respect to attributes B, C, D are as follows

$\pi_B = \{\{1\}, \{2, 3\}, \{4\}, \{5, 6\}, \{7\}\}$

$\pi_C = \{\{1, 2\}, \{3, 4\}, \{5, 6, 7\}\}$

$\pi_D = \{\{1, 4\}, \{2\}, \{3\}, \{5, 7\}, \{6\}\}$

The above partitions represent the frequent occurrence of item sets in each attributes.

Partition refinement:

The concept of partition directly related to functional dependencies. A partition pie refers to another partition π, if every equivalence class in π is the subset of another equivalence class of π. A functional dependencies r->y holds if and only if $\pi_{\{r\}}$ refines $\pi_{\{y\}}$.

There is a test for whether r->y holds or not if $|\pi_r| = |\pi_{r\,U\,\{y\}}|$. If pie$_r$ refines π_y then $\pi_{r\,U\,\{y\}}$ equals $\pi_{r.}$ On the other hand $\pi_{r\,U\,\{y\}}$ always refine π_r cannot have equal number of equivalences class as pie$_r$ unless $\pi_{r\,U\,\{y\}}$ are equal. A functional dependencies r->y holds if and only if $\pi_{\{r\}}$ refines $\pi_{\{r\,U\,\{y\}\}}$.

Pruning:

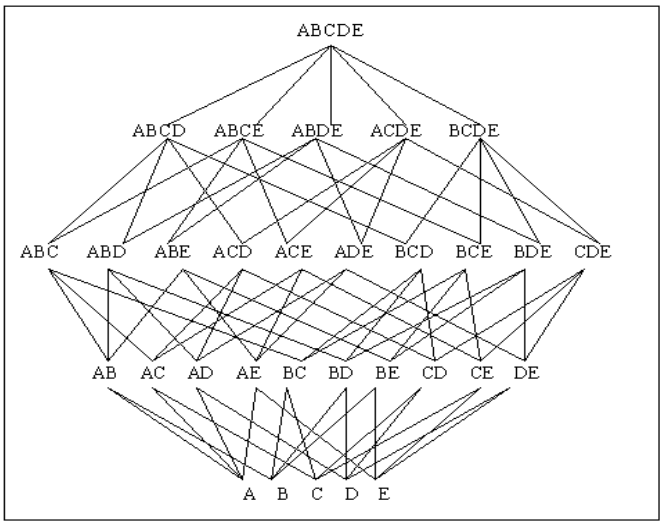
Pruning is a technique which reduces the size of decision trees by removing sections of the tree that provide powers to classify attributes. Pruning reduces the complexity of the classifier and hence improves predictive accuracy by the minimization of over fitting.

Pruning is a part of the Tane algorithm which discovers the functional dependencies among the attributes of the relation. Early methods for discovering of FDs were totally based on repeatedly sorting and comparing tuples to determine whether or not these tuples meet the FD definition [3]. For example[3], in table the tuples are firstly sorted on attribute A, then each pair of tuples that have the same value on attribute A is compared with attributes B, C, D, and E, in turn, to decide whether or not A->B,A->C,A->D or A->E holds. Then the tuples are sorted on attribute B and examined to decide whether or not B->A,B->C,B->D or B->E holds. This process is repeated continuously for attributes C, D, E, AB, AC, AD, and so on. After the last candidate BCDE has been checked, all FDs will have been discovered. The disadvantage of this approach is that it does not utilize the discovered FDs as knowledge to obtain new knowledge.

Pruning determines all the functional dependencies depend on the partitions technique which was described earlier in Tane algorithm. It determines whether or not a FD holds on a given dataset can be identified by comparing the number of the groups among partitions for various attributes consider the example [3].

	A	B	C	D	E
t1	0	0	0	1	0
t2	0	1	0	1	0
t3	0	2	0	1	2
t4	0	3	1	1	0
t5	4	1	1	2	4
t6	4	3	1	2	2
t7	0	0	1	0	0

For the dataset r, shown in table, the partition for attribute A can be represented as ΠA(r) ={{t1, t2, t3, t4, t7},{t5,t6}because the values of tuples t1, t2, t3, t4, and t7 on attribute A are all the same, they are assigned to the same group. Similarly, because the values of t5 and t6 are the same, they are placed in another group. The partition for the attribute combination AD for is ΠAD(r) = {{t1, t2, t3, t4, t7}, {t5, t6}}. The *cardinality of the partition* |ΠA(r)|, which is the number of groups in partition ΠA, is 2, and |ΠAD(r)| is 2 too. Because |ΠA(r)| is equal to |ΠAD(r)|, A->D can be obtained.


ABCDE
ABCD ABCE ABDE ACDE BCDE
ABC ABD ABE ACD ACE ADE BCD BCE BDE CDE
AB AC AD AE BC BD BE CD CE DE
A B C D E

Functional Dependency: Functional dependency is based on the attributes values to determine whether they are functional dependent are not. In this step we will continuously compare the attributes and discover functional dependencies based on their key-value pair as explained in previous steps If they satisfy the constraints they are functional dependent on each other.

TANE algorithm:

Step 1: TANE searches the set containment lattice in a level wise manner .A level LI is the Collection of attributes sets of size I.

Step 2: TANE starts with L1 = A | A R, and computes L2 from L1, L3 from L2 and so on According to the information obtained during the algorithm.

Step 3: To computing partitions; in the beginning, partitions with respect to the singleton Attribute sets are computed straight from relation r.

Step 4: A partition ÃcLRA, where ÃcLR denoted partition, is computed from the column r[A] as follows. First, the values of the column are replaced with integers 1, 2, 3 ÃcAe so the same values are replaced by same integers and different values with Different integers.

Step 5: Then the value t[A] is the identifier of the equivalence class [t]A of ÃcLRA, and ÃcLR A is then easy to construct.

IV. CONCLUSION

Functional dependencies are important metadata which can used to gain knowledge. Discovery of functional dependency can help in removing inconsistent and redundant data we propose a algorithm, TANE, for the discovery of functional and approximate dependencies from relations. The approach is based on deriving dependencies from partitions and searching for it in level-wise manner.

REFERENCES

[1] Huhtala, Y., Karkkainen, J., Porkka, P., and Toivonen, H., (1999), *TANE: An Efficient Algorithm for discovering Functional and Approximate Dependencies*, Computing Journal, V.42, No.20, pp.100-107.

[2] Bitton,D.,Millman,j.(1989), *A feasibility and performance study of dependency inference* , IEEE Computer Society Press.

[3] FD_Mine: Discovering Functional Dependencies in a Database Using Equivalences: Hong Yao, Howard J.Hamilton, and Cory J.Butz

[4] Mannila, H., Y., Karkkainen, J (1992) on the complexity of inferring dependencies, Discrete Applied mathematics.

[5] Savnik, I.,and Flach, P., (1993), Bottom-up induction of functional dependencies from relations, Computing Journal, V.42, No.20, pp.100-107.