International Journal of Database Management Systems (IJDMS) Vol.6, No.1, February 2014

ON THE FLY n-wMVD IDENTIFICATION FOR REDUCING DATA REDUNDANCY

Viswanadham Sangeeta and Vatsavayi Valli Kumari

Department of CS & SE, Andhra University, Visakhapatnam, India

ABSTRACT

Data Cleaning helps in reducing redundancy by determining whether two or more records defined differently in database represent the same real world object. If they represent the same, then the values of the records are nested generating a new n-wMVD database. The limitation with this approach is that addition or deletion of data is not done dynamically. To overcome this limitation we propose a dynamic approach that helps in reducing the redundancy, by comparing every incoming record with existing data and allows the modifications to be done if necessary. The operations are performed on the fly on the existing n-wMVD databases. When a record to be inserted, the approach allows a decision on whether to insert or nest with the existing record. If a record needs to be updated it is verified to observe if a new nesting has to be done or existing record can be updated. If a record to be deleted is unique then it is deleted from the database otherwise it is deleted by removing it from nesting. The applicability of our approach was tested and it is found that the results are promising which are discussed in this paper.

KEYWORDS

Data cleaning, data redundancy, n-wMVD (nesting based weak multi valued dependencies).

1. INTRODUCTION

Data warehouses collect large amounts of data from a variety of sources. They load and refresh continuously, so that the probability of some sources containing the redundant information can be eliminated.

Data cleaning is of crucial importance for many industries over a wide variety of applications. Aiming to detect the duplicates or approximately duplicate records that refer the same real world entity, duplicate record elimination is a very important data cleaning task. The naïve method is to pair-wisely compare all record pairs in the database in order to detect duplicate records. But, it is infeasible to implement due to its intolerable complexity. Because of high complexity alternate techniques are introduced in which duplicates are identified by defining integrity constraints. Inconsistencies (duplicates) are identified by the violation of integrity constraints. (But, traditional FD's identify inconsistency at schema design.)[6] CFDs aim at capturing the consistency of data by incorporating bindings of semantically related values. Conditional Functional Dependencies (CFDs) can identify inconsistencies by forming appropriate application specific integrity constraints. Though CFDs can eliminate redundancy by forming integrity constraints on single valued attributes(FDs) it fails in identifying redundancy when multi valued dependencies(MVDs) are exiting. An extension to CFDs referred as weak Multi-Valued Dependencies (wMVD)[8], deals with multi valued attribute dependencies. [2] n-wMVDs captures inconsistency and reduce redundancy by applying nesting on wMVDs. Automated and manual data cleaning is not handled by any of the existing techniques i.e. addition or deletion of data is not done dynamically.

DOI : 10.5121/ijdms.2014.6106

This paper introduces an extension to n-wMVD which deals with on the fly cleaning of data in reducing redundancy. It is done by comparing every incoming record with existing data and allows the modifications to be done if necessary. The remaining paper is organized into sections, where section 2 discusses related work, section 3 introduces how to reduce redundancy, section 4 explains methodology to reduce redundancy, section 5 deals with experimental analysis and section 6 concludes the paper.

2. RELATED WORK

[6] has proposed the CFDs(conditional functional dependencies), serves the basis for removing redundancy by forming application specific integrity constraints. A first step for data cleaning is the efficient detection of constraint violations in the data. Given an instance I of a relation schema R and a set Σ of CFDs on R, it is to find all the inconsistent tuples in I, i.e., the tuples that (perhaps together with other tuples in I) violate some CFD in Σ. Violated tuples(redundant) are deleted from the database. But it can handle single valued attribute dependencies but not multi-valued attribute dependencies. To deal with multi-valued attribute dependencies weak multi valued dependencies(wMVD)[8] have been defined. Let R be a relation schema and X, Y are sets of attributes, where X, Y $\subseteq$ R. A weak multivalued dependency (wMVD) is an expression X-w->Y and if the relation R satisfy the wMVD X-w->Y, then we can observe the reduction in the redundancy. Further reduction in redundancy on the wMVD satisfied relation can be achieved by applying nesting[7]. In nesting tuples that have matching values on some fixed attribute set can be represented as a single nested tuple by collecting set of the different tuple values on the remaining attributes. Application of nesting can be observed by applying on Table 1 as input and result is shown in Table 2(n - wMVD).

Suppose we have Flat relation schema with the attributes Course(C), Lady(L) and Gentleman(G). The purpose of this database is to record the information of who partners up with whom and in which course. This can be observed in Table 1

Table 1. Flat database.

T-Id	Course(C)	Lady(L)	Gentleman (G)
1	Latin	Racquel	Jose
2	Latin	Lorena	Flavio
3	Latin	Flor	Ruy
4	Latin	Flor	Flavio
5	Latin	Lorena	Ruy
6	Swing	Theresa	Quixote
7	Swing	Dulcinea	Quixote
8	Swing	Theresa	Sancho
9	Swing	Beatrice	Dante
10	Street	Eve,Queen L	Tupac,DMX

Table 2. n-wMVD database.

T-Id	Course(C)	Lady(L)	Gentleman(G)
1	Latin	{Racquel}	{Jose}
2	Latin	{Lorena,Flor}	{Flavio,Ruy}
3	Swing	{Dulcinea,Theresa}	{Quixote,Sancho}
4	Swing	{Beatrice}	{Dante}
5	Street	{Eve,Queen L}	{Tupac,DMX}

Nesting reduces redundancy in static wMVD databases. If the database is dynamic then automated data cleaning is required i.e. whenever addition, deletion or updataion of record is required redundancy verification is done automatically.

3. REDUCING REDUNDANCY

This paper introduces an extension of n-wMVD[1], on the fly approach to deal with addition or deletion of data dynamically on a n-wMVD database. The applicability of our approach requires the database to be converted into n-wMVD database. Our contributions in this paper are in three folds. In the first one we handle dynamic insertion of a new tuple into the existing database. The second deals with deletion of an existing tuple and the third handles the possibility of updating the tuple according to end user requirement.

The overall architecture of our system is shown in Figure 1.

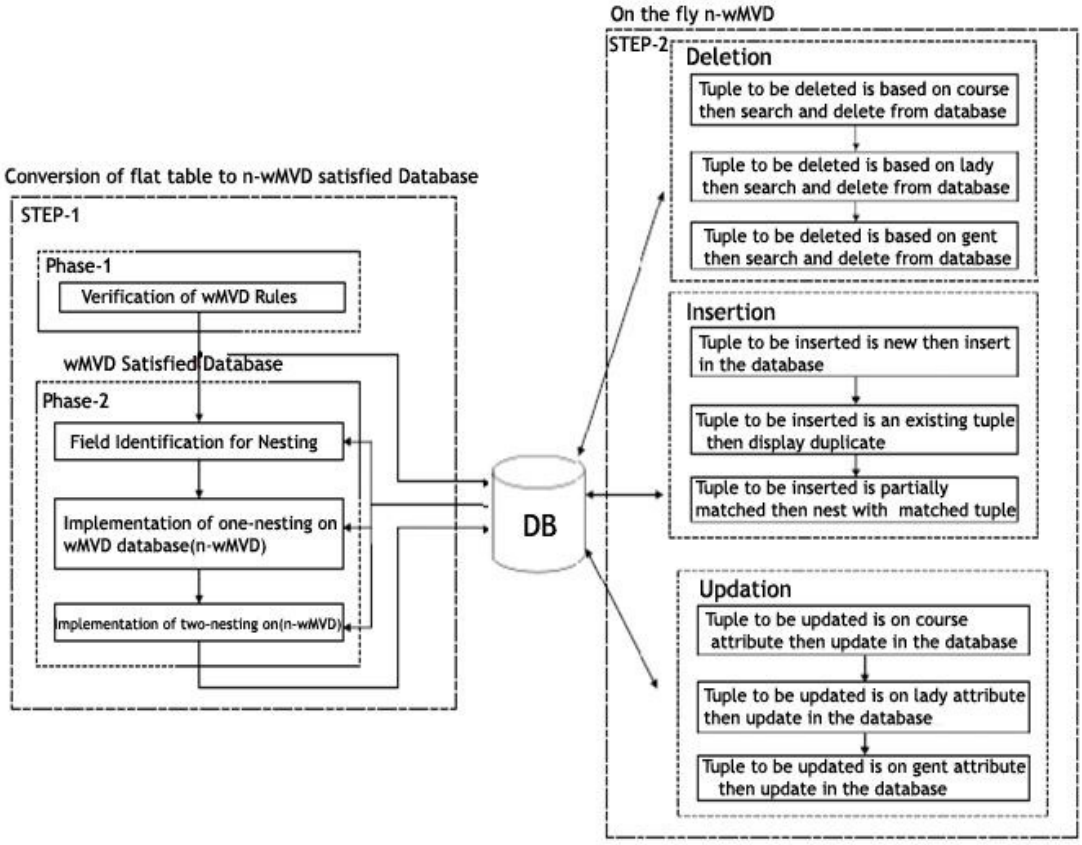

Figure 1. A model for implementation of on the fly approach

In the architecture it is clearly mentioned that on-the-fly approach can be done in 2 steps.

Step 1: Conversion of flat table to n-wMVD satisfied database[1].

Step 2: Implementation of dynamic database operations (on the fly)

In step 1 we have 2 phases. In the first phase we accept a flat database and convert into wMVD satisfied database. In the second phase nesting is applied on the wMVD database to make it as n-wMVD database [1]. Step 1 is a prerequisite for step2.

In step2 we deal with on the fly modification of the database by considering three operations insert, delete and update. We have three possibilities in the on the fly approach to handle three respective operations. If a tuple to be inserted in static mode operation of the database it needs the database to be down and needs to call the n-wMVD conversion. Such things are required for every operation on the database. The new tuple can accommodated without disturbing the database by using the on the fly approach. In case of updating a tuple on static database it is not allowed as it needs un-nesting the attribute. On the fly approach allows automatic updating of the tuple.

4. METHODOLOGY

This section includes 3 algorithms to perform operations on the fly(dynamic) on the existing nesting based weak multi valued database.

Algorithm 1: to insert a tuple to the existing n-wMVD[1] database.

Algorithm 2: to perform deletion from the given n-wMVD satisfied database.

Algorithm 3: modification of the existing tuple by updating values on the database.

4.1. PROCEDURE FOR INSERTION :

A record to insert into the database can be managed in 3 different possibilities

(i) If the tuple to be inserted is matched with an existing tuple (duplicate) then such tuple is not allowed to insert in the database.

(ii) If completely a new tuple i.e. no match is encountered with the existing data then the tuple is inserted as a new individual tuple.

(iii) If there is a partial match of the tuple values then it is handled by the following possibilities

(a) If course and Gent are matched , then nesting on Lady is done.

(b) If course and Lady are matched, then nesting on Gent is done.

Algorithm 1: For Inserting a new record into a n-wMVD database

Input : A tuple for insertion from the end user into n-wMVD Database (DB)

Output : Database table which includes the given tuple

Assumptions: Cursor (CR) Variable C

DB Attributes C1,C2 (Course) , L1,L2 (Lady) , G1,G2 (Gent)

„ N" represents total no of records in the DB

Method : Entire Database is scanned to identify to see whether new tuple can be paired with existing tuples or results in new insertion

1. **Select** all tuples from the DB into Cursor C
2. **Begin**
3. **Open** Cursor C
4. **Fetch** a record from the Cursor C into C1,L1,G1
5. **Enter** data to be inserted and store in C2,L2,G2
6. **For** I in 1 to N do
7. **For** each tuple in C in DB do
8. **If** (C1 = C2 and L1= L2 and G1= G2) then
9. **Display** "Duplicate Tuple"
10. Continue to select next record from C1
11. **Else If** (C1<>C2 and L1<>L2 and G1<>G2 then
12. **Store** the attribute values of C2,L2,G2 in the DB
13. **Else If** (C1= C2 and L1<> L2 and G1= G2) then
14. j:=L1||','||L2(L1 is concatenated with L2)

```
15.   Delete from DB where L=L1
16.  Store the attribute values of C2,j,G2 into the DB
17.  Else If (C1= C2 and L1=L2 and G1<> G2) then
18.  j: = G1||','||G2
19.  Delete from DB where G=G1
20.  Store the attribute values of C2,L2,j into the DB
21.  End If
22.                    End If
23.                End If
24.            End If
25.            Fetch the next record from the Cursor C
26.        End For
27.      Close C
28.     End Begin
```

This algorithm takes Table 2 as input and if we insert tuples{Latin,Raquel,josh}, {Swing,satvika,Dante}, {Summer, Rithika,Raju} the resultant can be observed in Table 3.

Table 3. n-wMVD database after insertion

T-Id	Course(C)	Lady(L)	Gentleman(G)
1	Latin	{Racquel}	{Jose}
2	Latin	{Lorena,Flor}	{Flavio,Ruy}
3	Swing	{Dulcinea,Theresa}	{Quixote,Sancho}
4	Swing	{Beatrice}	{Dante}
5	Street	{Eve,Queen L}	{Tupac,DMX}
6	Summer	{ Rithika}	{Raju}

We can observe that with insertion of 3 tuples the database size is increased with 1 tuple where as other 2 tuples are nested with the existing tuples.

4.2. PROCEDURE FOR DELETION

If user wants to delete a value from the existing data then we need to know which attribute's value i.e whether nested attribute or normal attribute(un-nested attribute) C,G, or L to be deleted.

(i) If user choice is to delete a record for a normal attribute then it is deleted from the database.

(ii) If 'L or G' i.e any nested attribute have to be deleted then we have to check whether it is nested with other values or not. If it is nested then it is simply deleted from nesting otherwise its corresponding record is deleted from the database.

Algorithm 2:For Deleting a record from a n-wMVD database

Input : Any attribute value for deletion from the n-wMVD Database (DB)
Output : Modified Database table
Assumptions : DB Attributes C1,C2 (Course) L1,L2 (Lady) , G1,G2 (Gent). N represents total no of records in the DB. CHO character variable

Method **:** Entire Database is scanned to identify to see whether the tuple to be deleted is existing

```
1.  Begin
2.  For each tuple in DB do
3. Accept Value be Deleted in DB and Store in C2.
4. Accept Value of choice of attribute to be
        Deleted in DB and Store it in CHO.
5.        If (CHO="c") then
6.              Delete from  DB  where C = C2
7.        Else If (CHO = „l") then
8.            Delete from  DB  where L = C2
9.            C1:=concat(C2,',')
10.       Update DB =>set L = replace(DB.L,C1,'')
11.           C1:=concat(',',C2)
12.       Update DB =>set L = replace(DB.L,C1,'')
13.       Else If (CHO = „g") then
14.             Delete from  DB  where G = C2
15.               C1:=concat(C2,',')
16.       Update DB =>set G = replace(DB.G,C1,'')
17.               C1:=concat(',',C2)
18.       Update DB =>set G = replace(DB.G,C1,'')
19.           End If
20.       End If
21.     End If
22.   End For
23.  End Begin
```

This algorithm takes Table 3 as input. Suppose we want to delete a tuple with values {Latin,Raquel,Jose}. We can observe that the required tuple is in nested format then it is simply removed from nesting. The result can be observed in Table 4.

Table 4. n-wMVD database after deletion

T-Id	Course(C)	Lady(L)	Gentleman(G)
1	Latin	{Racquel}	{Josh}
2	Latin	{Lorena,Flor}	{Flavio,Ruy}
3	Swing	{Dulcinea,Theresa}	{Quixote,Sancho}
4	Swing	{Beatrice}	{Dante}
5	Street	{Eve,Queen L}	{Tupac,DMX}
6	Summer	{ Rithika}	{Raju}

4.3. PROCEDURE FOR UPDATING :

Updating an existing tuple may be possible i.e., if the user requests for updating. We need to know to which column i.e nested(L,G) or un-nested columns C the updating is required.

(i). If an unnested column 'C' is the choice from the user, then it is directly updated since it is not nested.

(ii). If nested columns 'L or G' have to be updated then we have to replace the existing data with data given by the user.

Algorithm 3:For Updating a record in a n-wMVD database

Input : Any attribute value for updating a tuple in the n-wMVD Database (DB)

Output : Updated Database table

Assumptions : DB Attributes C1,C2 (Course) , L1,L2 (Lady) , G1,G2 (Gent)
„ N‟ represents total no of records in the DB
„CHO‟ character variable

Method : Entire Database is scanned to identify to see whether the required tuple to be updated is existing

1. **Begin**
2. **For** each tuple in DB do
3. **Accept** new Value to be updated in DB and store it in C2.
4. **Accept** old Value to be replaced by new value and store it in C1
5. **Accept** value of choice of attribute to be Updated in DB and Store it in CHO.
6. **If** (CHO = „c‟) then
7. **Update** DB =>set C = C2 where C = C1
8. **Else If** (CHO = „l‟) then
9. **Update** DB =>set L= replace(DB.L,C1,C2)
10. **Else If** (CHO = „g‟) then
11. **Update** DB =>set G= replace(DB.G,C1,C2)
12. **End If**
13. **End If**
14. **End If**
15. **End For**
16. **End Begin**

This algorithm takes Table 4 as input and after updating it produces the result into Table 5. Suppose the database to be updated in the gent field with the value ‘Jose’ in the tuple {Latin ,Raquel ,Josh}. The result can be observed in Table 5.

Table 5. n-wMVD database after updating

T-Id	Course(C)	Lady(L)	Gentleman(G)
1	Latin	{Racquel}	{Jose}
2	Latin	{Lorena,Flor}	{Flavio,Ruy}
3	Swing	{Dulcinea,Theresa}	{Quixote,Sancho}
4	Swing	{Beatrice}	{Dante}
5	Street	{Eve,Queen L}	{Tupac,DMX}
6	Summer	{ Rithika}	{Raju}

Updating is not possible on the static databases, because the updating converts the database into non n-wMVD format. Calling of n-wMVD conversion for every operation on the database is mandatory. This leads to be more complex and more time consuming and occupies more space which is proved through comparison with flat database.

5.EXPERIMENTAL ANALYSIS

In this section, we present our findings about the performance of our scheme for reducing redundancy over static n-wMVD databases.

Setup: For the experiments, we used postgres SQL on Windows XP with 1.86 GHz Power PC dual CPU with 3GB of RAM.

Data: Our experiments used Adult Database (UCI) with a few synthetic attributes addition.

Size of the Relation: In this experiment we investigated how the redundancy is reduced by considering the space occupied by the static and on the fly relations on the disk by varying number of tuples can be seen in Table 6 and Figure. 2

Table 6. Size of the Relation

No.of Tuples	Static	On-the-fly
100	16	10
200	26	11
300	37	12.5
400	43	14
500	49	18
600	58	23
700	63	28

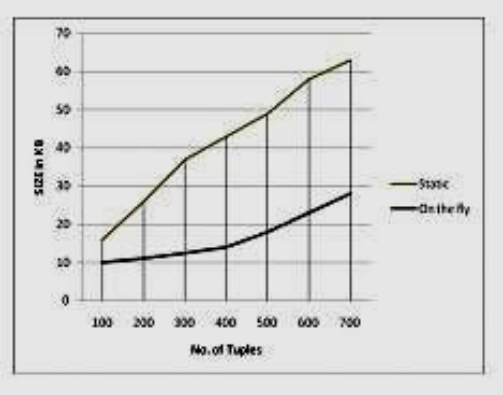

Figure 2. Graph representing size of the Relation

6. CONCLUSION

We presented, the on the fly approach on n-wMVD (nesting based wMVD's) proved to be good in reducing redundancy in generalized static databases. This work shows better performance if the database contains more repetitive data. We considered only simple operations like insert, delete and update and showed the experiments resulting in space saving of static n-wMVD and dynamic n-wMVD. There is naturally much more to be done. First, we can extend it to work with more complex operations. Second, on the fly application on unstructured and semi structured database.

REFERENCES

[1] Sangeeta Viswanadham and Valli Kumari V, An Efficient Approach to Identify n-wMVD for Eliminating Data Redundancy in 'CUBE 2012 INTERNATIONAL IT CONFERENCE EXHIBITION, pp. 464 -469.

[2] Sangeeta Viswanadham and Valli Kumari V, An Approach to Identify n-wMVD for Eliminating Data Redundancy in 'International Symposium on Intelligent Informatics (ISI' 12), pp.89 – 97.

[3] V.Sangeeta and V. Valli Kumari, Eliminating Data Redundancy using Nesting based wMVD in '2012 4th International Conference on Electronics Computer Technology (ICECT 2012), pp.285 – 289.

[4] Rahm, E., Hong, H.D.: Data Cleaning: Problems and Current Approaches. In: IEEE Techn. Bulletin on Data Engineering, University of Leipzig, Germany (2000)
[5] Ezeife, C.I., Ohanekwu, T.E.: Use of Smart Tokens in Cleaning Integrated Warehouse Data. International Journal of Data Warehousing & Mining (April-June 2005)
[6] Fan, W., Kementsietsidis, A.: Conditional Functional Dependencies for Capturing Data Inconsistencies. ACM Transactions on Data Base Systems (TODS) 33(2) (June 2008)
[7] Hartmann, S., Link, S.: On Inferences of Weak Multivalued Dependencies. Fundamenta Informaticae 92, 83–102 (2009), doi:10.3233/FI-2009-0067
[8] Fischer, P., Van Gucht, D.: Weak Multivalued Dependencies. In: PoDS Conference. ACM (1984)
[9] Korth, H., Roth, M.: Query Languages for Nested Relational Databases. In: Abiteboul,S., Schek, H.-J., Fischer, P.C. (eds.) NF2 1987. LNCS, vol. 361, pp. 190–204. Springer, Heidelberg (1989)
[10] Fagin, R.: Multivalued Dependencies and a New Normal Form for Relational Databases. Trans. ACM Database Syst. (1977)
[11] Hartmann, S., Link, S.: Characterising nested database dependencies by fragments of propositional logic. Annals of Pure and Applied Logic Journal of Science Direct 152, 84–106 (2008)
[12] S. Abiteboul, N. Bidoit. \Non _rst normal form relations: An algebra allowing restructuring". Journal of Computer and System Sciences, 33(3): 361-390, 1986.
[13] C. Beeri, M.V. Vardi. \Formal systems for tuple and equality generating dependencies". SIAM Journal of Computing, 13(1):76-98, February 1984.
[14] P. Buneman, W. Fan, S. Weinstein. \Path Constraints on Semistructured and Structured Data". In Proceedings of the Seventeenth Symposium on Principles of Database Systems, 1998.
[15] Z.M. Ozsoyoglu, L.-Y. Yuan. \A new normal form for nested relations". ACM Transactions on Database Systems, 12(1):111-136, March 1987.

AUTHORS

V.Sangeeta received the M.Tech degree from Andhra University and is pursuing Ph.D from Andhra University in the area of Data Cleaning.Currently working as Associate Professor in the Dept.of Computer Science Engineering since 9 years at Pydah College of Engineering and Technology,Visakhapatnam.

Prof.V. Valli Kumari received the Ph.D degree from Andhra University and is working as Professor in the department of Computer Science and Systems Engineering, Andhra University. She is also a member of IEEE.