Vol. 5, No.3 March 2014 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences
©2009-2014 CIS Journal. All rights reserved.

http://www.cisjournal.org

An Enhanced Method for Generating Composite Candidate Keys Based on N×M Boolean Matrix

Khaled Saleh Maabreh
Asstt. Prof. Computer Information System Department, College of Science and Information Technology, Zarqa University, Jordan

kmaabreh@zu.edu.jo

ABSTRACT

In a relational database, a candidate key is defined as a set of attributes that is distinct the database rows. The determining process of a candidate key could affect the database performance and optimization. In this paper, an algorithm based on N×M Boolean matrix is proposed to automate the process of candidate key generation for both single and composite attributes based on a functional dependency set, as well as it has the ability to add a new functional dependency attributes as required based on a proved theorem. Results showed that the proposed algorithm can be easily implemented and used in practice. It can be easily expanded to solve N-Level composite attributes set.

Keywords: *Relational Database, Candidate Key, Functional Dependency, Attribute Closure, Boolean Matrix*

1. INTRODUCTION

A candidate key is a set of attributes that is distinct the database rows and has not a redundant attributes [1,2]). The determining process of a right candidate key could affect the database design in terms of efficiency and optimization. Many algorithms were provided to solve the candidate key problem by using different approaches, most of them are based on a graph theory [3,4,5]

Duan [6] provides a method for candidate key generation based on N×N Boolean matrix for single attributes. Many expansions have been performed on Duan algorithm and the weakness points are also reported as: First, the algorithm solves only a single attribute candidate key. Second, line three in step 2 of the Duan algorithm must be modified to be "if(M(i,j)=1 and i≠j then" because without the right hand side of the condition "i≠j" an extra useless iteration will be executed, it is already modified in the proposed algorithm. Third, step three may cause an infinite loop because however a matrix is changed, the algorithm executes the step once again.

This paper provides an enhancement and expansion to Duan algorithm [6]for determining a two level of functional dependencies candidate key based on N×M Boolean matrix presented in the paper as well as the theorems, inference rules and the attribute classification that is also presented will be taken into account. We can simply refer to the Duan paper or any other database design reference for seeing such classification and the proof of the related theorems and inference rules.

The rest of this paper is organized as follows: Related theorems are presented in section 2, data processing and the modified algorithm description are explained in section 3, section 4 shows discussion and examples and the conclusion and further works are presented in section 5.

2. RELATED THEOREMS

According to the relation schema R (U,F) where U is a set of attributes and F is a set of functional dependencies defined in attributes, the Armstrong rules [1,2] are:

- Reflexivity rule: If $Y \subseteq X$, then $X \rightarrow Y$
- Augmentation rule: If $X \rightarrow Y$, then $XZ \rightarrow YZ$
- Transitivity rule: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- For X,Y and $Z \subseteq U$

2.1 Classification of Attributes

The functional dependencies attribute can be classified according to their appearance in F to the following groups [1,2,6]:

- Left hand side: Which contains only those attributes appeared at the left side of functional dependencies
- Right hand side: Which contains only those attributes appeared at the right side of functional dependencies
- Both sides: Which contains the attributes appeared on both sides of functional dependencies
- Not appeared: Which contains the attributes that don't appear on both sides of functional dependencies.

3. DATA POPULATION

A relational database schema R(U,F) is given, where U is a set of attributes, U = {a1,a2,…,am}. And F is a set of functional dependencies (constraints) hold on the relational schema. A matrix M(n,m) is constructed where m is the number of attributes in the relation schema R (where m = |R|) and n = m+c (where n = |F| and c is the number of composite attributes in F (such as $ab \rightarrow c$)). When all attribute sets in F are single attribute functional dependencies (i.e. n = m) it becomes a special case in this presented method, in other words the proposed algorithm solves correctly a single attribute set and all examples presented in Duan paper and any other single attribute functional dependency sets are also producing the same results.

According to the Armstrong axioms and related theorems, the Boolean matrix will be filled as follows: M(i,j) is equal to one if there exists ai→aj functional dependency in F. If i =j then, M(i,j) will be set to 1 according to the reflexivity rule (ai→ai). Each composite attributes in F will be placed in a new row of the matrix. For example aiaj→ak will be placed in the m+1 row of a matrix M. The values of M(i,i), M(i,j) and M(i,k) will be set to one according to the reflexivity rule and the functional dependencies in F. This will be done for every composite attribute in F. The other elements in M are set to be zero. In order to reduce the comparison time, a new column named all-ones column is added to the matrix M after executing step 4 of the proposed algorithm, the new column will set to be one if the row has ones in all elements and zero otherwise.

3.1 The Modified Algorithm

Input:

The initialization matrix M of the given relation schema R(U,F)

Output:

All candidate keys of the given relation schema R(U,F)

Steps:

```
Step 1: /* The initialization matrix M */
Step 2: for (i=1; i<=n; i++)
        {for (j=1; j<=n; j++)
          if (M(i,j) =1 and i≠j) then   /* i≠j is added to the
        original algorithm step */
        for(k=1; k<=n; k++)
        M(i,k)=M(i,k) or M(j,k)}
        /* ai→aj and  aj→ak→ai→ak, that is if M(i,j)=1
        and M(j,k)=1 then M(i,k)=1 */
Step 3: Transitive rule for composite attributes
        for (i=1;i<=m; i++)
        for (j=1;j<=m-1; j++)
        for (k=j+1;k<=m; k++)
        if M(i,j) and M(i,k) =1 then
                for (l=m+1; l++; l<=n)
                if ajak ⊆ F then
                for (x=1; x<=m; x++)
                M(i,x)=M(i,x) or M(l,x)
        /* aiaj→ak and  ak→aL then aiaj →aL, */
Step 4: Changing the last bit to be one if all the row
        elements are ones
        for (i=1; i<=n; i++)
        all_ones=1
        for (j=1; j<=m; j++)
        all_ones=all_ones and M(i,j)
        if all_ones=1 then
        M(i,m+1)=1
Step 5: for all none ones rows
          for (i=1; i<=m-1; i++)
             for (k=i+1; k<=m; k++)
              if aiak ⊄ F then
              for (j=1; j<=m; j++)
              E(j)=M(i,j) or M(k,j)
              for (x=1; x<=m-1; x++)
                 for (y=x+1; y<=m; y++)
              if axay ⊆ F then
                 if E(x) and E(y) =1 then
                    for (l=m+1; l<=n+1; l++)
                       if an+1= aiak then
                       for (d=1; d<=m; d++)
                           M(n+1,d)=E(d) or M(l,d)
Step 6: for ( i=1 ; i<=n ; i++)
       if M(i,m+1)=1 then
       print candidate key ai
       keyfound =1
        /* If all the elements in line i are 1 (i.e. all-ones bit
is one), then ai is a candidate key */

Step 7: If key found = 0, then create a line vector Mi
        which contains the row of not all elements are
        ones and create a vector E which contains ones in
        all elements. Suppose a vector combination
        S=(Mk1,Mk2,…,Mkj) in M, 2≤kj≤m
        1) Mki≠Mkj      (ki≠kj)
        2) Mk1∨Mk2∨…∨Mkj=E
        3) Mk1∨Mk2 ∨ … ∨ Mki-1 ∨ Mki+1
             ∨ …∨ Mkj i≠ E
        If all of these conditions are satisfied, then (ak1,
        ak2, .. ,akj) is a candidate key.
Step 8: Finish
```

4. DISCUSSION AND EXAMPLES

Example 1

Given a relation schema Example in [7] R(A, B, C, D, E) and F ={AB→E, AD→B, B→C, C→D} be a set of functional dependencies hold on R. What are the candidate keys?. Creating a matrix M with initialization values as shown in Table 1, where AB→E means AB→A and AB→B by reflexivity rule and AB→E as F required. So, if ai→ aj then M(i,j) will be set to 1, otherwise it will be set to 0.

Table 1: (Initialized M)

	A	B	C	D	E
A	1	0	0	0	0
B	0	1	1	0	0
C	0	0	1	1	0
D	0	0	0	1	0
E	0	0	0	0	1
AB	1	1	0	0	1
AD	1	1	0	1	0

Table 2: (The changed M)

	A	B	C	D	E	All-ones
A	1	0	0	0	0	0
B	0	1	1	1	0	0
C	0	0	1	1	0	0
D	0	0	0	1	0	0
E	0	0	0	0	1	0
AB	1	1	1	1	1	1
AD	1	1	1	1	0	0

Table 3: (Changed M using step5)

	A	B	C	D	E
A	1	0	0	0	0
B	0	1	1	1	0
C	0	0	1	1	0
D	0	0	0	1	0
E	0	0	0	0	1
AB	1	1	1	1	1
AD	1	1	1	1	1
AC	1	1	1	1	1

Table 4: (Initialized M)

	A	B	C	D	E
A	1	1	1	0	0
B	0	1	0	1	0
C	0	0	1	0	0
D	0	0	0	1	0
E	1	0	0	0	1
CD	0	0	1	1	1

Table 5: (The changed M)

	A	B	C	D	E
A	1	1	1	1	1
B	0	1	0	1	0
C	0	0	1	0	0
D	0	0	0	1	0
E	1	1	1	1	1
CD	1	1	1	1	1

Table 6: (Changed M using step5)

	A	B	C	D	E
A	1	1	1	1	1
B	0	1	0	1	0
C	0	0	1	0	0
D	0	0	0	1	0
E	1	1	1	1	1
CD	1	1	1	1	1
BC	1	1	1	1	1

Table 7: (Initialized M)

	A	B	C	G	H	I
A	1	1	1	0	0	0
B	0	1	0	0	1	0
C	0	0	1	0	0	0
G	0	0	0	1	0	0
H	0	0	0	0	1	0
I	0	0	0	0	0	1
CG	0	0	1	1	1	1

Table 8: (The changed M by using step 5)

	A	B	C	G	H	I
A	1	1	1	0	1	0
B	0	1	0	0	1	0
C	0	0	1	0	0	0
G	0	0	0	1	0	0
H	0	0	0	0	1	0
I	0	0	0	0	0	1
CG	0	0	1	1	1	1
AG	1	1	1	1	1	1

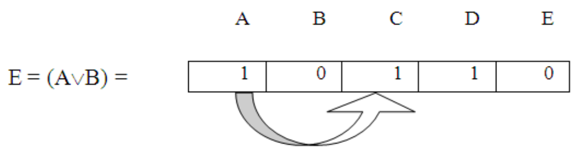

Fig 1: (A row combination example)

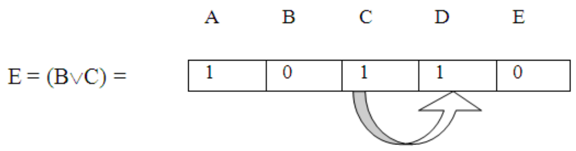

Fig 2: (A row combination example)

After executing the algorithm by using M as input, the matrix M will set to be as shown in **Table 2**.

In order to reduce the comparison time, a new column named all-ones will be added into the matrix M, which sets to be one for the rows that having one in all elements and zero otherwise. After executing step 5 by using the new matrix shown in **Table 2**, every two rows that not having ones in all elements are joined together by using OR operator. Then, the result will be placed in a temporary vector E. We can simply determine a none ones rows by reading the all-ones bit, which is M(i,m+1) where $1 \leq i \leq n$. For example, a vector E which contains (A∨B) will be set as shown in **Fig. 1**.

Then, the vector E is scanned by AND operator to determine if E contains an existing composite functional dependency in F or not? We can notice that A∨B→AC. Whenever AB⊆F, then, Join the temporary vector E with AB by using OR operator and then adds a new row into the matrix M named AC as shown in **Table 3**.

After executing the final step in the algorithm, the candidate keys will be: AC, AB and AD. Which are the same results as obtained in [7]

Example 2

Given a relation schema Example in [8] R(A,B,C,D,E) and F={A→BC, CD→E, B→D, E→A} is a set of functional dependencies hold on R. What are the candidate keys?

Creating a matrix M with initialization values as shown in **Table 4.** to be as input for executing the algorithm. After executing the algorithm, the matrix M will be changed to be as shown in **Table 5.**

After executing step 5 of the proposed algorithm and after joining a combination of every two none ones rows, we can notice that the rows B and C are joined together by OR operator which produces a new vector E as shown in **Fig. 2**. Then, every two elements in E are combined together by using AND operator to see if the two rows are exist in F or not. If the two elements are exist in F then, the original rows which are B and C in our case are added into the changed matrix M. Because B∨C→CD and CD exists in F, so BC is joined by using the OR operator with the CD and then BC is added to the matrix M as shown in **Table 6**.

The candidate keys are A,E, CD and BC as produced in [8].

Example 3

Given a relation schema page 244 in [9] R(A,B,C,G,H,I) and F ={A→B, A→C, B→H, CG→H, CG→I} be a functional dependencies set holds on R. What are the candidate key?

Creating a matrix M with initialization values as shown in **Table 7**. **Table 8** shows the changed M after executing step 5.

The candidate key is AG which is the same result as produced in [9]

The algorithm used in this paper is based on a proved theorem. So, steps 2 and 3 are based on transitivity rule (ai→aj and aj→ak→ai→ak) which is one of the Armstrong axioms. This inference rule can be used to solve a combination of attributes (ai→bici and bici→di then ai→di) where ai is a single attribute functional dependency and bici is a composite attribute placed in a new row in the matrix M. After changing M and if a_i→b_ic_i and b_ic_i exists in F then, a transitive rule is implemented between a_i and b_ic_i.

5. CONCLUSION

A modified algorithm based on N×M Boolean matrix is presented in this paper. The new algorithm can be used for candidate key generation for single and composite attributes of a functional dependency set. It has the ability to add a new functional dependency attribute as required by applying the theorems presented in this paper. Results showed that all composite attributes functional dependency of two levels are solved correctly. The algorithm can be easily expanded to solve N-Level composite attributes set.

ACKNOWLEDGEMENT

This research is funded by the Deanship of Research and Graduate Studies in Zarqa University /Jordan

REFERENCES

[1] Elmasri, R. and S.B. Navathe, 2010. Fundamentals of Database Systems. 6th Edn., Pearson Education, Delhi.

[2] Silberschatz, A., H.F. Korth and S. Sudarshan, 2010. Database System Concepts. 5th Edn., McGraw-Hill Higher Education, Boston.

[3] Li, H. and L. Zhou, 2008. Use closure of relevant sets of attributes to efficiently find candidate keys. International Conference on Computer Science and Software Engineering, Dec. 12-14, IEEE Xplore Press, Wuhan, Hubei, pp: 237-242. DOI: 10.1109/CSSE.2008.441

[4] Qin Z., HU H., FENG J. and CAI G., 2004. Seeking for Relation Scheme Candidate Key by Using Functional Depending Graph. Journal of Anqing Teachers College(Natural Science Edition), China.

[5] Saiedian, H. and T. Spencer, 1996. An efficient algorithm to compute the candidate keys of a relational database schema. Comput. J., 39: 124-132. DOI: 10.1093/comjnl/39.2.124

[6] Duan, W., 2009. Candidate keys solving method based on Boolean matrix for single attribute functional dependency sets. Proceedings of the WRI World Congress on Software Engineering, May 19-21, IEEE Xplore Press, Xiamen, pp: 265-267. DOI: 10.1109/WCSE.2009.126

[7] Chen J., (n.d). Introduction to Database Systems. http://csc.lsu.edu/~jianhua/jianhua.html. Accessed on March, 5, 2013.

[8] Vijayakumar G, Closure of Attribute Sets http://www.indiessoft.com/doeaccalumni/qp/closure.doc. doeaccalumni.org, Accessed on March, 5, 2013

[9] Connolly, T. and Begg C., 2009. Database systems: A practical approach to design, implemenation and management. 5th Edn., Addison-Wesley.