INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY (IJCET)

ISSN 0976 – 6367(Print)
ISSN 0976 – 6375(Online)
Volume 4, Issue 6, November - December (2013), pp. 219-231
© IAEME: www.iaeme.com/ijcet.asp
Journal Impact Factor (2013): 6.1302 (Calculated by GISI)
www.jifactor.com

IJCET
©IAEME

A GRAPH BASED APPROACH TO FIND CANDIDATE KEYS IN A RELATIONAL DATABASE SCHEME

Subhrajyoti Bordoloi[1], **Bichitra Kalita**[2]

Dept. of Computer Applications, Assam Engg. College, Guwahati, Assam

ABSTRACT

In this paper, a simple straightforward method to find all the candidate keys of a relational database schemes using graph is presented. First, the FD graph is drawn from the set of functional dependencies in a relational scheme. Thereafter, by using few very simple graph transformations, the FDG is reduced to a graph called the Candidate Graph having only the candidate nodes. From this Candidate Graph, all the candidate keys for the relational scheme can be identified.

Key Words: Candidate Node, Candidate Graph, FD, FDG.

1. INTRODUCTION

Graphs have many applications in the field of knowledge and data engineering. There are many graph based approaches used in database management system, ranging from E-R modeling to database schema design and normalization [1][3][8][10][11][17][18]. Functional dependencies among the entities of a context problem area can be easily represented as a graph called the functional dependence graph (FDG). Apart from database technology, FDGs have many applications in Web data analysis [4], Language processing [5], Data cleaning [9], Network Theory [10] and Data Mining [12]. There are several types of generalized graphs such as FD graph, Implication graph, and Deduction graphs, etc. used to study functional dependencies. Hyper graph is another generalized graph that can be used to study functional dependencies [14]. There are many algorithms for finding all the candidate keys of a relational scheme based on Karnaugh map [6], Boolean matrix [7] and other set theoretic approaches [13]. Hossein Saiedian, et.al, described a method for finding the candidate keys in a relational scheme [2]. In their approach, the functional dependencies among the attributes in a database table can be represented as a graph. Such graphs are also called attribute graphs. They use the attribute graph for analyzing the dependencies to find the candidate keys. Their approach cannot find all the candidate keys of a relational scheme if the FD graph is strongly connected. Their approach needs a minimal set FDs to compute the candidate keys.

In this paper, a generalized graph based method to find all candidate keys in a relational database scheme is presented.

2. FUNCTIONAL DEPENDENCIES AND THEIR GRAPHS

Let $\boldsymbol{\alpha \rightarrow \beta}$ be a functional dependency, where α, β are sets of attributes. Depending on the cardinality of α and β the following graph patterns are obtained for the FD $\boldsymbol{\alpha \rightarrow \beta}$.

2.1. If $\boldsymbol{\alpha}$ and β both are singular, i.e., $\boldsymbol{|\alpha|=|\beta|=1}$ then the graph pattern shown in *fig.-1(a)* is obtained.

2.2. If $|\alpha|=1$ and $|\beta|>1$ and $\beta=\{\beta_1, \beta_2 \ldots, \beta_n\}$,then the graph pattern shown in *fig.-1(b)*is obtained.

2.3. **a)** If $|\alpha|>1$ and $|\beta|=1$ and $\alpha=\{\alpha_1,\alpha_2\ldots,\alpha_n\}$ then the graph pattern shown in *fig.-1(c)* is obtained.

b) If $|\alpha|>1$ and $|\beta|>1$ and $\alpha=\{\alpha_1,\alpha_2 \ldots,\alpha_n\}$ and $\beta=\{\beta_1,\beta_2 \ldots, \beta_n\}$,then the graph pattern shown in *fig.-1(d)* is obtained.

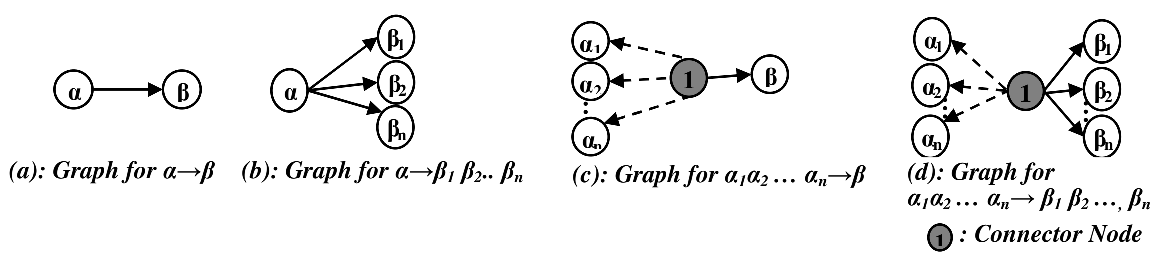

Figure 1: Functional Dependencies and Their Graphs

Note that there are dependencies from a connector node to all components (nodes) of it. In fig.1, the following dependencies are also existed $\mathbf{1\rightarrow (\alpha_1, \alpha_2, .. ,\alpha_n)}$

3. FUNCTIONAL DEPENDENCY GRAPHS (FDG) FOR A RELATIONAL DATABASE SCHEME

The functional dependency graph (FDG) for a relational scheme is drawn based on the set of functional dependencies available in the relational scheme using the rules ***2.1, 2.2, 2.3.*** *One n*ode is created for each attribute in the relation scheme. A connector-node is created each time the ***Rule 2.3*** is applied, i.e. for each distinct non-singular attribute set which is in the left-hand side of a FD. Therefore, for a relational scheme R with **N** attributes with a set of FDs **F** and having **M** distinct non-singular attribute sets in the left-hand side of the FDs in F there will be **N+M** nodes in the FDG. For example, for a relational scheme R= (A, B, C, D, E) with the set FDs F= {A→ BC, CD →E, B→ D, E →A} we obtain a FDG, as shown in fig-2, with a total of 6 nodes.

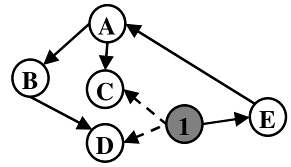

Figure 2: FD Graph for a relational scheme R= (A, B, C, D, E) with the set FDs F= {A→ BC, CD →E, B→ D, E →A}

4. FINDING THE CANDIDATE KEYS USING FDG

To find the candidate keys from the FDG, the following transformations are performed in the FDG. The transformation process has two phases- ***Augmentation*** and ***Reduction.*** After the transformation, the resultant graph contains only the candidate nodes. A candidate node is a node which alone or along with some other candidate nodes, can determine all other nodes in the graph. Therefore the resultant graph is called the ***Candidate Graph*** ***(G^C)***. So, the problem now is to find the candidate graph from an FDG. Therefore the transformation process can be represented as follows: $G \rightarrow G^{+} \rightarrow G^{C}$.

4.1 Augmentation: Finding G^+

In the augmentation phase new connector nodes and edges are added to the FDG by performing the following transformations.

T1: Let there exists a connector-type node which connect the node set C= {$c_{i,}$},1<i≤n, n is the cardinality of the set C, and has an outgoing edge to a node B. If there is a dependency **$c_i \rightarrow c_j$, c_i, c_j ∈ {C}** then remove **c_j** from **{C}**. Delete the connector node from the **G^+** if **{C}** is singular. Assign all edges of **C** to ***$c \subset C$.***

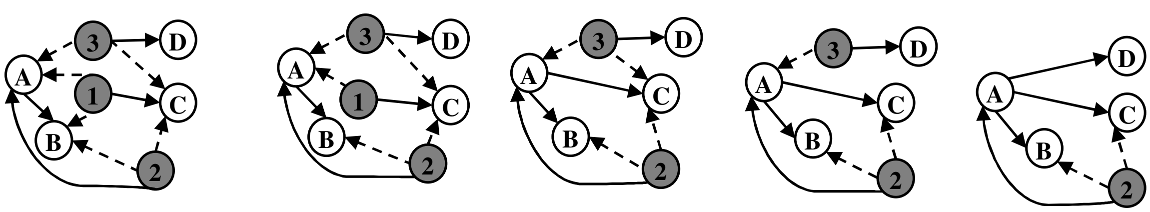

Figure 3: Transformation T1

T2: Let there exists a connector-type node which connect the node set C={$c_{i,}$},1<i≤n, n is the cardinality of the set C, and has an outgoing edge to a node B. Let there exists a node A *(may be a connector node)* in the FDG from which there is a path to any node **c_i** in C.

CASE-I: If ∀i , 1≤ i ≤n there is a path from A to **c_i**, then an edge from A to C is added.

CASE-II: If there is a path from A to a set {**c_k**} ⊂ C, 1<i<n and |c_k|<n where n is the cardinality of C, then for each c_k ,a new connector node C′ is created connecting node A and the nodes in the set **{C- c_k }** and an edge from C′ to C is added.

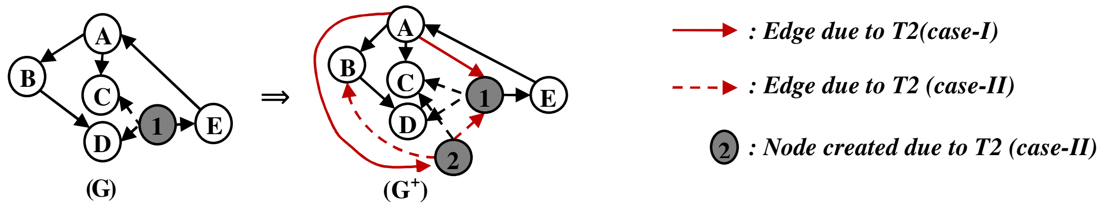

Figure 4: Transformation T2 (Augmentation)

In *fig-4*., node A has a path to both components C and D of the set represented by the connector node **"1",** so an edge from A to the connector node **"1"** is added due to ***case-I*** of the transformation T2. Node B has a path to only one component D of the set represented by the connector node "1", so a new connector node **"2"** is created to connect node B and node C and an edge from **"2"** to the **"1"** is added due to ***case-II*** of the transformation T2.

4.2 Reduction: Finding G^C

In the reduction phase certain nodes and edges are deleted from the augmented FDG by performing the following transformations.

T3: Let **C** be a connector node in $\mathbf{G^+}$. If there is a node **A** in $\mathbf{G^+}$ which has a direct edge to node **C,** then delete all edges from A to the first node in the paths to any $\mathbf{c_i \subseteq C}$.

T4: Delete a node N from the augmented FDG G^+ if its in-degree >0 and out-degree =0.
In *fig-5(b),* first node D is deleted because its in-degree is non zero and out-degree is zero. Deletion of this node causes the deletion of the nodes B and C from the graph.

T5: Delete a cycle K from G^+ if in-degree (K) >0 and out-degree (K) =0. *Fig-5(c),* deletion of cycles is shown.

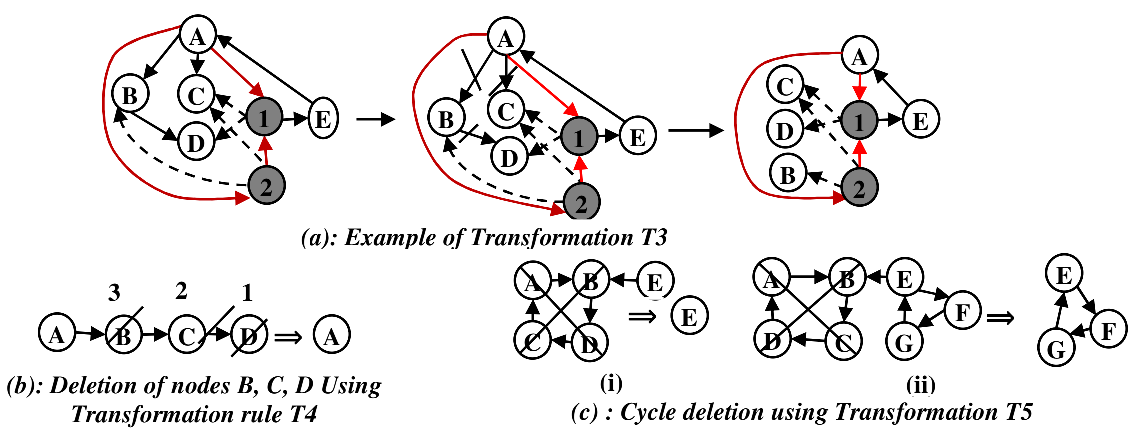

Figure 5: Transformations in Reduction Process

The following example in *fig-6* explains the transformation process. Let R= (A, B, C, D, E) be a relational scheme with the set FDs F= {A→ BC, CD →E, B→ D, E →A}. With this set of FDs F, the FD graph G is obtained by using rule *2.1, 2.2 and 2.3* discussed in section-2 of this paper. After the transformation process the candidate graph G^C is obtained. From G^C it is clear that there is cycle having four nodes in the graph. The candidate nodes are A, E, 1 and 2. Therefore R= (A, B, C, D, E) with the set of FDs F= {A→ BC, CD →E, B→ D, E →A} has four candidate keys- *A, E, BC, and CD.*

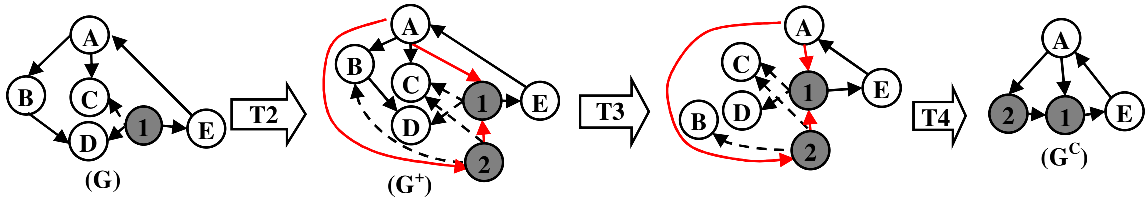

Figure 6: The Transformation Process to find the candidate keys of relational scheme R= (A, B, C, D, E) with the set FDs F= {A→ BC, CD →E, B→ D, E →A}

The following definition and theorems are presented with respect to the *Candidate graph* G^C.

4.3 ***Definition 1:*** *A node in a cycle in a candidate graph* G^C *of a relational scheme R is either a candidate key or a part of the candidate keys of R.*

4.4 Theorems

Theorem 1: *$G=G^C$ iff one of the following conditions is satisfied.*

i. $F=\phi$

ii. *G has a cycle having all the nodes in the cycle and there is no connector node.*

Proof: Let R= {α1, α_2, α_n) be a relational scheme with a set of FDs, F. Let G be the FD graph for R.

Case (i): If F=ϕ, $\forall i,j,\ i\neq j,\ \alpha_i \rightarrow \alpha_j \notin F$, so we get that G has *n* isolated nodes and G remains unchanged during the Augmentation process and the Reduction process.

Therefore, $\mathbf{G=G^+}$ **----------- (1)**

$\mathbf{G^+=G^C}$ **----------- (2)**

Hence $\mathbf{G=G^C}$ *(from 1 and 2).*

Case (ii): Now if F≠ ϕ and we have a chain $\forall i,j,\ i\neq j,\ \alpha_i \rightarrow \alpha_j \rightarrow \alpha_i$ then no new connector nodes can be added during the Augmentation process, which gives.

$\mathbf{G=G^+}$ **------------ (3)**

Similarly, no node of the graph G^+ is deleted during the reduction process, because there exists no node $\alpha_i, 1 \leq i \leq n$ such that *in-degree(*α_i*)>0 and out-degree(*α_i*)=0*. Hence

$\mathbf{G^+=G^C}$ **----------- (4)**

Therefore, we get that $\mathbf{G=G^C}$ *(from 3 & 4). //*

Theorem 2*: In a candidate graph* G^C*, a candidate node cannot have edges if the node does not belong to a cycle.*

Proof: Let A be the only candidate node in the candidate graph, G^C. So, A is an isolated node. Therefore we get that,

In-degree (A) =Out-degree (A) =0 ------ (5)

Now, let there are two candidate nodes A and B in G^C. Here, we get two possibilities:

a) Node A and node B both may be isolated or
b) A and B are connected.

In case (b) we get that, if in-degree(B)=1 and out-degree(B)=0 then node B cannot be survived in G^C due to transformation T4. Similarly, if in-degree(A)=1 and out-degree(A)=0 then node A cannot survive in G^C due to transformation T4. Therefore, both nodes A and B can survive in G^C if they form a cycle.

$$\textbf{in-degree (A)=out-degree(A)= in-degree(B)= out-degree(B)=1} \quad \text{------ (6)}$$

Therefore, nodes A and B can have edges if they are in a cycle. Otherwise they are isolated nodes. //

***Theorem 3**: If the Candidate graph, G^C, of a relational scheme R has only one cycle and if all the nodes of G are in the cycle, then every attribute of R is a candidate key.*

Proof: Let $\mathbf{V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow \ldots V_{n-2} \rightarrow V_{n-1} \rightarrow V_n \rightarrow V_1}$ is the cycle in the ***Candidate graph G^C*** with n vertices. In this chain, from any node $\mathbf{V_i}$ there is a path to any node $\mathbf{V_j}$, $j \neq i$ and $1 \leq j \leq n$ by transitivity rule. Therefore, from any $\mathbf{V_i}$, $1 \leq i \leq n$, all other nodes $\mathbf{V_j}$, $1 \leq j \leq n$ and $j \neq i$ are reachable, where n is the degree of R.

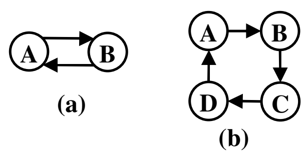

Figure 7: Cyclic Candidate Graphs having only one cycle

In *fig-7(a)*, nodes A, B, both are candidate nodes. From node A, node B is reachable and from node B is reachable. In *fig-7(b)*, nodes A, B, C, and D all are candidate keys. From node A, nodes B, C, D are reachable i.e., node A can functionally determine B, C, and D. So, node A is a candidate key. Similarly, from node B, nodes A, C, D are reachable i.e., node B can functionally determine A, C, D. So, node B is also a candidate key. //

***Theorem 4**: If the Candidate graph G^C has disconnected components, then the Cartesian product of the candidate nodes of each component is the set of all candidate keys.*

Proof: Let us consider a relation scheme R= {α1, α2 …, α_n) with a set of FDs, F=ϕ. Since F=ϕ, there are ***n*** disjoint components in the G^C for R *(by theorem 2)*. The candidate key **k** for R is the Cartesian product all attributes of the relational scheme.

Therefore, $\mathbf{k=\alpha_1 \times \alpha_2 \times \ldots \times \alpha_n}$ **----------- (7)**

Now, let F≠ϕ, then we get the candidate graph $\mathbf{G^C}$ for R. Let C be the set of *m<n (since F≠ϕ, some nodes will be deleted or there will be at least one cycle in G^C)* disjoint components in G^C such that $\mathbf{C=\{\{c_1\},\{c_2\},\ldots,\{c_m\}\}}$. The component c_i can be considered as a set.

By Theorem-2, if the candidate graph contains disjoint components then they must be either isolated nodes or cycles. If we consider these disjoint components as nodes, then the set of functional dependencies F available among these components, is null, i.e. F=ϕ. Therefore, by equation-7, we get that, the set of candidate keys K= $\mathbf{\{\{c_1\} \times \{c_2\} \times \ldots \times \{c_m\}\}}$. //

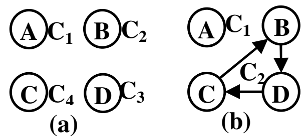

Figure 8: Candidate graphs with Disjoint Components

In *fig-8*, an FDG with disjoint components is shown. In *fig-8(a)* {{A} × {B} ×{C} × {D}} = {ABCD} is the candidate key. But in *fig-8(b)*, {{A} × {B, C, D}} = {{AB}, {AC}, {AD}} is the set of candidate keys.

Theorem 5: *A relational scheme R has multiple candidate keys if and only if the Candidate graph G^C for R contains at least one cycle.*

Proof: Let G^C be the candidate graph for a relational scheme R, with *n* components $C_1, C_2, \ldots C_n$. Let K be the set of candidate keys of R. Each component C_i can be considered as a set of nodes. By theorem3,

$$K = \{C1 \times C_2 \times \ldots \times C_n\} \ \ldots\ldots\ldots\ldots \qquad \textbf{(7)}$$

Now by definition2 we get that,

$$|C_k| > 1 \text{ if } C_k \text{ is a cycle. } \ldots\ldots\ldots \qquad \textbf{(8)}$$

From equation 3,

$$|K| = 1 \text{ if } \forall i,\ 1 \le i \le n,\ |C_i| = 1 \ \ldots. \qquad \textbf{(9)}$$

$$|K| > 1 \text{ if } \exists k,\ 1 \le k \le n \mid C_k \mid > 1 \ \ldots \qquad \textbf{(10)}$$

From Equation-9 and Equation-10, it is clear that K is non- singular if and only if there exists at least one cyclic component $C_{k,}$ $1 \le k \le n$, in G^C.//

Fig-9 shows all possible cyclic candidate graphs for a relational scheme R=(A,B,C,D).

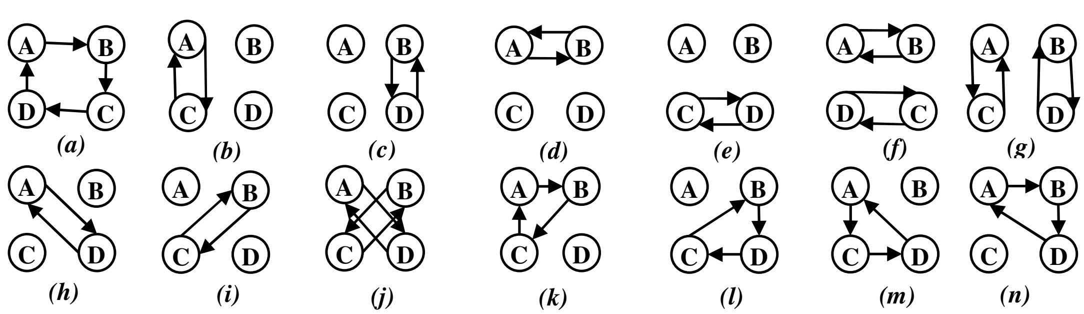

Figure 9: Candidate Graphs for R=(A,B,C,D) having Cycles

5. ANALYSIS OF THE APPROACH

In the approach, from an FDG, G we get transformed graph G^C by using the transformation rules described here. But G cannot be obtained from G^C since all functional dependencies except the dependencies among the candidate nodes are deleted, i.e., G^C does not include all dependencies which imply the original set of FDs F. Though G^C cannot be obtained from G, the transformations applied in the FDG drawn from the set of FDs F do not generate any extra dependencies that are not members of the closure of F, F^+. Moreover, in this graph based approach, it is not required to calculate F^+, the closure of F and F^C, the canonical cover of F. The approach can be implemented by the following algorithms. The outline of the approach is shown in *fig.10.*

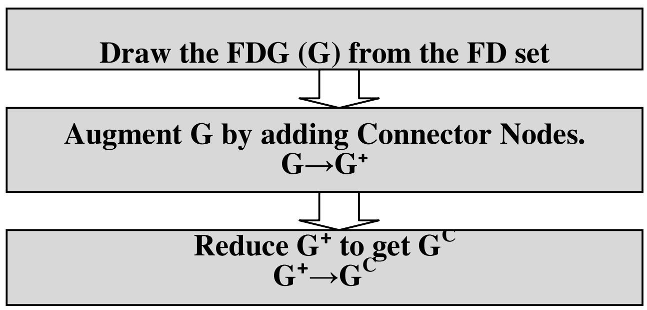

Figure 10: Outline of the Approach

5.1 Procedure *Make-FDG=(R(α₁, α₂ …, αₙ),F,G)*

```
{
 G=ϕ;
  For each α ЄR create a node for in G;
  If ( F≠ϕ ) Then
  {
     For all (α→β) Є F
     { If (│α│=1 AND │β│=1) Then
          Add an edge from node α to β in G;
       If (│α│=1 AND │ β│≥1) Then
          For all b Є β add an edge from node α to b in G
            { If (│α │>1 AND │β │≥1) Then
               { Create a new connector node C to connect all a Є {α};
               For all a Є {α} Put an undirected edge from all a to C;
               For all b Є{β} Add an edge from node C to each b Є{β};
               Delete the functional dependency α→β from F ;}
            }
     }
  }
}
```

Figure 11: Algorithm to construct the FDG from the set of FDs F

5.2 Procedure Augment-G ***(graph G, graph G^+)***

```
{
  G+=G;
  For each connector node C ∈ G+
  {
    If (there is a dependency ci→cj, ci, cj ∈ {C}) Then   {Remove cj from {C}. }
    If (│{C}│=1) Then { Delete node C from G+; Assign all edges of C to c ⊂{C};}
  }
  For all node A∈ G+, A∉ {C}
  {
     If there is a path from A to all nodes ci ∈ {C} Then Add an edge from A to C;
  }
  Repeat
   { For all node A∈ G+, A∉ {C} and for all nodes ci⊂ {C}
      If there is a path from A to ci Then Create a connector node C'
        to group node A and nodes {C-{ci}} add an edge from C' to C;
   }  Until no more connector node can be created;
  For all node A∈ G+, A∉ {C}
  {
     If there is a path from A to all nodes ci ∈ {C} Then Add an edge from A to C;
  }
}
```

Figure 12: Algorithm for the Augmentation Process of FD Graph G

5.3 Procedure ***Reduce-G (graph G^+, graph G^C)***

```
{
      GC =G+;
      For each connector node C ∈ GC
         { If the node c ∈ {C} has no edges except the connecting edge Then
                  Delete node c from GC;
                 Else
                       Delete the connecting edge between c and C;
         }
      For all node A which has a direct edge to C in GC
          { Delete the edge from A to the path to ci ∈{C}; }
      For all Cycle K∈GC
         { Remove all redundant edges from K;/Removal does not break the cycle/
}
      While ((∃ node A ∈ GC) AND (in-degree(A)>0 AND out- degree(A)=0))
                  Delete node A from GC;
      For all Cycle K∈GC
          { If (in-degree(K)>0 AND out- degree(K)=0) Then
                Delete cycle K from GC ;}
}
```

Figure 13: Algorithm for Reduction Process of the Augmented FDG G^+ to G^C

6. EXPLANATORY EXAMPLES

6.1 Example1

Problem: *Find the candidate keys of the relation schema R= (A, B, C, G, H, I) with the set of functional dependencies F= {A→ B, A→C, CG→ H, CG→ I, B→H}.*

Solution: From the problem statement we know that the FDG G has 7 vertices *A, B, C, G, H, I and CG* as shown in *fig-14*. From *fig.-14* it is found that the candidate node for R= *(A, B, C, G, H, I) with the set of functional dependencies F= {A→ B, A→C, CG→ H, CG→ I, B→H}* is the connector node 2. Therefore the candidate key of R is ***{2} = (AG)***.

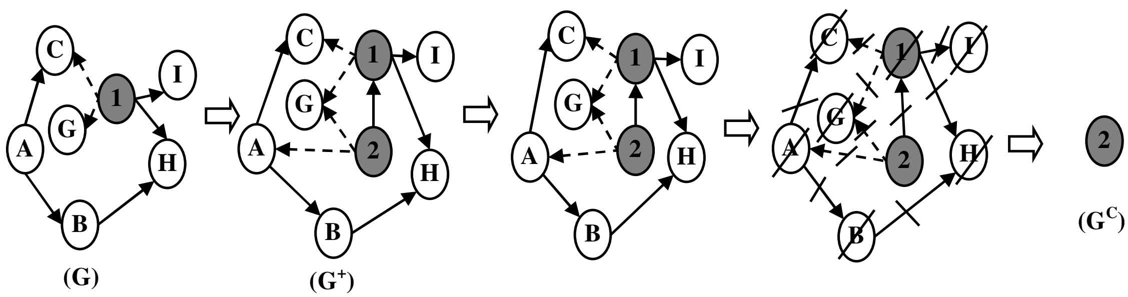

Figure 14: Finding candidate nodes for R= (A, B, C, G, H, I) with the set of functional dependencies F= {A→ B, A→C, CG→ H, CG→ I, B→H}

6.2 Example2

Problem: *Find the candidate keys of the relation scheme R= (A, B, C, D, E, F, G) with the set of functional dependencies F= {A→ B, AB→C BC→A, AC→ D, DE→F, EF→G, AG→E}*

Solution: From the problem statement we know that the FDG **G** has 13 vertices *A, B, C, G, E, F, G and AB,AC,BC,DE,EF,AG* as shown in *fig-15*. There are following six connector nodes in the FDG: *1={AB}, 2={BC}, 3={AC}, 4={DE}, 5={EF}, 6={AG}.*

For node 1: Since A→B so B is removed from **1**.

For node 3: Since A→C so C is removed from **3**.

Since ***|1|=1*** *and* ***|3|=1,*** connector nodes 1 and 3 are deleted from the graph G^+.

For node 2: Since A→B and A→C so an edge from A to node **2** is added.

For node 4, the connector nodes ***7={AE}*** and ***8={BCE}*** are added to the graph.

For node 6, the node ***9={BCG}*** is added to the graph.

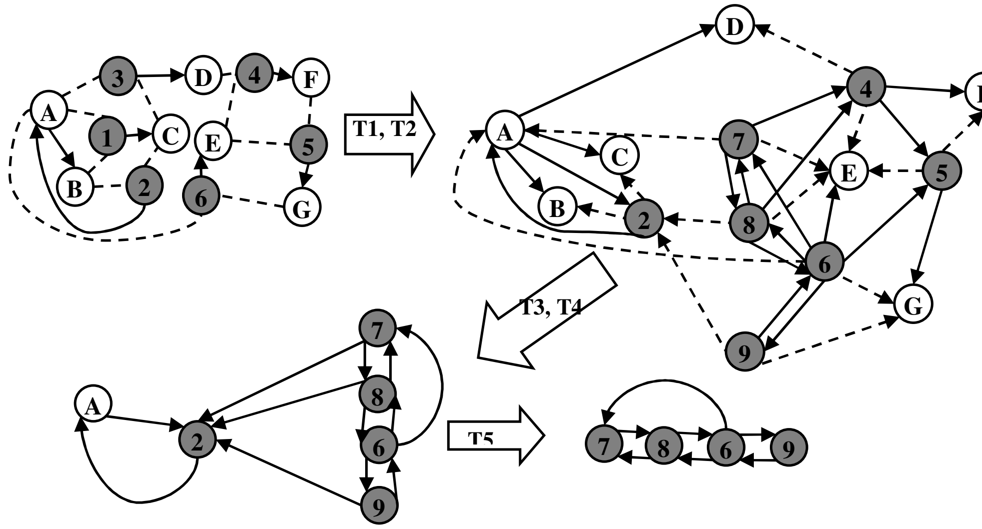

Figure 15: Finding the candidate keys of the relation scheme R= (A, B, C, D, E, F, G) with the set of FDs F= {A→ B, AB→C BC→A, AC→ D, DE→F, EF→G, AG→E}

By doing so, the augmented graph G^+ is obtained. After the reduction process the Candidate graph G^C is obtained from which it is decided that the relation scheme R=(A,B,C,D,E,F,G) with the set of FDs F={A→B,AB→C,BC→A,AC→D,DE→F, EF→G,AG→E} has four candidate keys- ***AG,BCE,BCG and AE.***

6.3 Example3

Problem: *Find the candidate keys of the relation scheme R= (A, B, C, D, E, F) with the set of functional dependencies F={AB→C, CD→E,CD→-B,EF→-C}.*

Solution: From the problem statement we know that the FDG **G** has 9 vertices *A, B, C, G, E, F and AB,CD, EF,* as shown in *fig-16*. There are three six connector nodes in the FDG: ***1={AB}, 2={CD}, 3={EF}.***

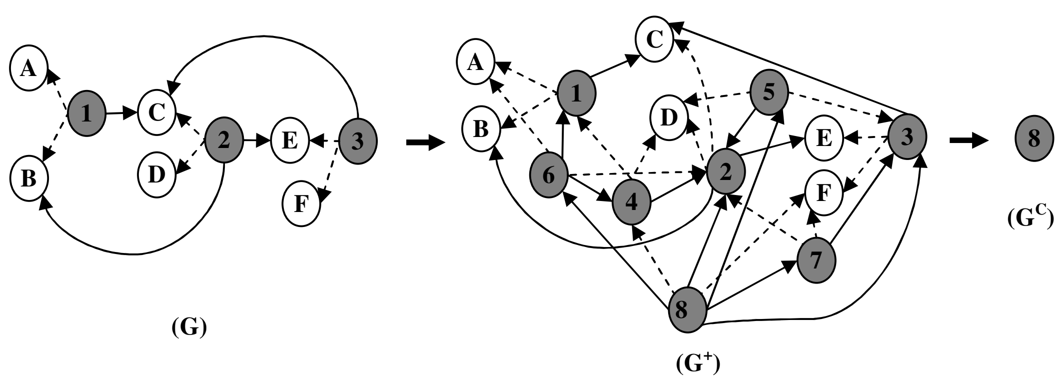

Figure 16: Finding the candidate keys of the relation scheme R= (A, B, C, D, E, F) with the set of functional dependencies F={AB→C, CD→E,CD→-B,EF→-C}

For node 1: Since **2→B** the connector node ***6= {ACD)*** is obtained.

For node 2: Since **1→C** and **3→C,** the connector nodes ***4= {ABD}*** and ***5= {DEF***} is added.

For node 3: Since **2→E** the connector node ***7= {CDF}*** is added. Now because **4→2→E** node ***8= {ABDF}*** is added.

Now,

- **6** has a path to each of the components of node **4** so the edge **6→4** is added.
- **8** has a path to each of the components of node **2** so the edge **8→2** is added.
- **8** has a path to each of the components of node **3** so the edge **8→3** is added.
- **8** has a path to each of the components of node **5** so the edge **8→5** is added.

After the reduction process all other nodes except node **8** are deleted from the Augmented Graph $\mathbf{G^+}$ due to *Transformation T4*. The Candidate Graph $\mathbf{G^C}$ contains only node **8**. Therefore, the candidate key of the relation scheme R= (A, B, C, D, E, F) with the set of functional dependencies F= {AB→C, CD→E, CD→-B, EF→-C} is ***ABDF.***

7. CONCLUSION

In this paper, a simple graph based method to find candidate keys of a relation scheme is described. This method takes an FDG, drawn based on the set of FDs available in a relational scheme, as an input and using five very simple graph transformations the input FDG is converted into a candidate graph (Transformed FDG). This candidate graph gives all the candidate keys of the relation scheme. This approach does not require computation of F^+, F^c and α^+, the closure, canonical cover of F and closure of an attribute set α. This is a purely graph based approach. This approach can also be used to find the candidate factors in the problem areas like medical diagnosis, social problems where there are many factors with cause-effect dependencies among them.

REFERENCES

[1] Paris C. Kanellakis, ”Elements of Relational Database Theory” Technical Report No. CS-8939, October, 1989, Brown University.

[2] Hossein Saiedian and Thomas Spencer “An Efficient Algorithm to Compute The Candidate Keys of A Relational Database Schema” The Computer Journal, Vol. 39, No. 2, 1996.

[3] Hua Chang,”Graph Algorithms for Database Schema Design”, Master Thesis, Concordia University, (C) National Library of Canada, ,ISBN 0-315-97701-9, September, 1986 .

[4] Yang Yu, et. al. “Extending Functional Dependency to Detect Abnormal Data in RDF Graphs”. Proc. 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Part-I, LNCS 7031, © Springer-Verlag Berlin Heidelberg, 2011, pp. 794-809.

[5] Joakim Nivre, “Algorithms for Deterministic Incremental Dependency Parsing” Computational Linguistics, Volume 34, Number 4, (C) 2008 Association for Computational Linguistics, pp-523-553.

[6] Y. Zhang, “Determining all candidate keys based on Karnaugh map.”, Proc. IEEE International Conference on Information Management, Innovation Management and Industrial Engineering - Volume 04, 2009, pp. 226–229.

[7] Wenxiu Duan, "Candidate keys Solving Method Based on Boolean Matrix for Single Attribute Functional Dependency Sets", World Congress on Software Engineering, IEEE Computer Society, 2009, pp. 265-267.

[8] Vitalie COTELEA "Problem Decomposition Method to Compute an Optimal Cover for a Set of Functional Dependencies" Database Systems Journal vol. II, no. 4, 2011 pp. 17-30.

[9] D.Raghu, "Comparison of Conditional Functional Dependencies using Fast CFD and CTANE Algorithms", International Journal of Computer Science and Technology Vol. 3, Issue 1,Jan. -March 2012 ISSN: 0976-8491 (Online) | ISSN: 2229-4333 (Print), pp. 176-178.

[10] X Xu, et al.," Reduced Functional Dependence Graphs and Their Applications", Proc. IEEE International Symposium on Network Coding (NetCod), 29-30 June 2012, pp. 61-66.

[11] Moussa Demba, "Algorithm for Relational Database Normalization Up To 3NF", International Journal of Database Management Systems (IJDMS) Vol.5, No.3, June 2013, pp. 39-51.

[12] Catharine Wyss, Chris Giannella, and Edward, Robertson, " FastFDs: A Heuristic-Driven, Depth-First Algorithm for Mining Functional Dependencies from Relation Instances" DaWaK, Lecture Notes in Computer Science, Volume 2114, 2001,© Springer-Verlag Berlin Heidelberg,2001 pp 101-110.

[13] Raymond Fadous, John Forsyth "Finding candidate keys for relational databases", Proceeding SIGMOD '75 Proc. ACM SIGMOD International conference on Management of Data, New York, NY, USA ©1975, pp. 203-210.

[14] Giorgio Gallo1, "Directed Hyper graphs and Applications**",** Discrete Applied Mathematics**,** Elsevier, Volume 42, Issues 2–3, April 1993, pp. 177-201.

[15] Anupama A Chavan, Vijay Kumar Verma , "Functional Dependency Mining form Relational Database: A Survey ",International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-2, Issue-6, August 2013,pp-208-210.

[16] P. Cordero, M. Enciso, A. Mora "Automated Reasoning to Infer all Minimal Keys", Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, pp. 817-823.

[17] Subhrajyoti Bordoloi, Bichitra Kalita, "E-R Model to an Abstract Mathematical Model for Database Schema using Reference Graph", International Journal of Engineering Research And Development, e-ISSN:2278-067X,p-ISSN:2278-800X, Volume 6, Issue 4, March, 2013, pp. 51-60.

[18] Subhrajyoti Bordoloi, Bichitra Kalita, "Designing Graph Database Models From Existing Relational Databases" International Journal of Computer Applications (0975 – 8887) Volume 74 No.1, July, 2013, pp. 25-31.

[19] M.Siva Parvathi and B.Maheswari, "Minimal Dominating Functions of Corona Product Graph Of A Cycle with a Complete Graph", International Journal of Computer Engineering & Technology (IJCET), Volume 4, Issue 4, 2013, pp. 248 - 256, ISSN Print: 0976 – 6367, ISSN Online: 0976 – 6375.

[20] Shameem Akthar, Dr. D Rajaylakshmi and Dr. Syed Abdul Sattar, "A Modified Pso Based Graph Cut Algorithm for the Selection of Optimal Regularizing Parameter in Image Segmentation", International Journal of Advanced Research in Engineering & Technology (IJARET), Volume 4, Issue 3, 2013, pp. 273 - 279, ISSN Print: 0976-6480, ISSN Online: 0976-6499.

[21] Jayanta Kr. Choudhury, Anupam Dutta and Bichitra Kalita, "Decomposition of Complete Graphs into Circulant Graphs and its Application", International Journal of Computer Engineering & Technology (IJCET), Volume 4, Issue 6, 2013, pp. 25 - 47, ISSN Print: 0976 – 6367, ISSN Online: 0976 – 6375.