Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 61 (2013) 16946-16949

Elixir ISSN: 2229-712X

Automated methodology to reduce the redundancy in relational database

Ajeet A.Chikkamannur[1] and Shivanand. M. Handigund[2]

[1]Department of Computer Science and Engineering, Sri Venkateshwara College of Engineering, Bangalore 562157, India.

[2]Department of Computer Science and Engineering, Bangalore Institute of Technology, Bangalore 560004, India.

ARTICLE INFO

Article history:
Received: 31 May 2013;
Received in revised form:
24 July 2013;
Accepted: 3 August 2013;

Keywords
Hypergraph,
Functional Dependency,
Incidence matrix,
Cyclic,
Acyclic.

ABSTRACT

Normalization is a correct process for good relational database design of an application but not a complete. To support this, in a Boyce-Codd normal form design, three relations are designed from the two functional dependencies. Clearly, this is a setback in the basic aim of normalization i.e. reduction in database redundancy. The indecision here is that can we reduce the database redundancy further? Secondly, the database designers are creating the relations from the set of attributes and functional dependencies existing among them. The human work in the design process may lead to ambiguity and incorrect relations when the set of attributes and functional dependencies are large. The design of automated process for the human work overcomes this lacuna. The researchers have shown that there is a natural correspondence between the hypergraph and relational database schema. Beeri .et .al introduced a special class of hypergraphs known as 'acyclic', where the properties of relational database are equivalent to 'α -acyclicity'. Another researcher Fagin introduced 'γ -acyclicity, which establishes the condition of unique relationship among the attributes. Hence, the amelioration of hypergraph from cyclic to acyclic satisfies the properties of relational database. Our paper proposes a methodology that takes the functional dependencies, attributes set as an input, and identifies the candidate key attributes. From the candidate key attributes, the hypergraph is redefined for the 'acyclicity' by the isolation of functional dependency (ies) framed by the candidate key attributes.

1.Introduction

In the design of relational database management system the database schema is constructed through the grouping of related attributes. Currently the designers are utilizing the diagrammatic modeling techniques viz., ER or EER for the design but these techniques completely depends on the designer skill, art, interpretation and capability. Further, the complexity is increasing when more number of designers is involved in the design. One of the grounds is the lack of unanimous understanding and viewing of the same application. Pragmatic database design procedure employs either design by synthesis or design by analysis, where the set functional dependencies attributes are joined or set of attributes is decomposed to constitute a relation(s).

The normalization is a process of crystallizing the database in which the grouping of attributes is carried to eliminate the modification anomalies and reduction of redundancy. This layered approach reduces the redundancy by decomposing the set of attributes in to subset(s) as it progresses from one layer to another. To establish this, the set of attributes is decomposed in to subset(s) of attributes based on the functional dependencies.

The process normalization yields the number of relations. This process minimizes the redundancy and avoids the update anomalies of database. We observed that this process is correct but not a complete. The evidence is the design of three relations from two functional dependencies in BCNF [5]. Further, the designed relations must ensure the dependency preservation and lossless join property on decomposition. To ensure manually, this is a herculean task when the decomposition yields the large number of relations. Is it possible to decompose the attributes set to ensure these properties and the reduction of redundancy? Many researchers [1, 2] have shown that there is a natural correspondence between hypergraph and database schema. Among the many properties, one of the properties is that "*the join dependency and loss less join are equivalent to the a-cyclicity of the hypergraph*".

This paper proposes an automated methodology for identifying the cyclicity of the hypergraph. On the existence of cyclicity, the hypergraph is ameliorated to acyclic through the identification of functional

dependency causing the cyclicity. Then the functional dependency is detached from the hypergraph, so that the cyclicity is eliminated and redundancy is reduced by overriding the functional dependency in the relation design.

2. Background

1. Hypergraph: A hypergraph H is a pair (N, E) [4, 9] where N is the set of nodes and E is the set of hyper edges.

2. Cycle: The cycle A in a hypergraph H is a sequence of edges and nodes i.e., E_1, N_1, E_2, N_2, ………………., E_m, N_m, E_{m+1} such that

1. N_1, N_2, ………………., N_m are distinct node.
2. E_1, E_2,………………., E_m are distinct edges and $E_{m+1} = E_1$.
3. $m \geq 3$, that is, there are at least three edges are involved.
4. N_i is in E_i and E_{i+1} $(1 \leq i \leq m)$.

The size of cycle depends on the m distinct edges and the least size of cycle is with three edges. This least and most number of edges are resulting in two kinds of configuration shown in the figure 1.

Tele:
E-mail addresses: ac.ajeet@gmail.com

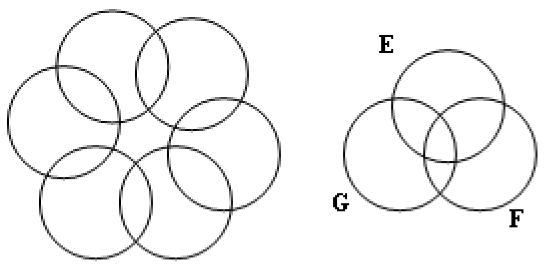

Figure 1. Forbidden Configurations

3. FD graph definition: A functional dependency graph (FD-graph) [3, 9] is a labeled graph with two kinds of nodes and edges and the definition is : For a hypergraph H = (N, E), let S_M = {Z | there exists a hyper arc (Z, i) $\in$ H and | Z | > 1}. The FD-graph of H is the labeled graph G (H) = ($N_s \cup N_c$, $E_f \cup E_d$), where:

$N_s \equiv N$ is the set of simple nodes;

N_c is the set of compound nodes, which is in bijective relationship with S_M. If Z $\in S_M$ is a source set then z will denote the corresponding compound node, and any simple node z_i in the source set Z will be called a compound node of the compound node z;

$E_f \subseteq (N_c \times N_s) \cup (N_s \times N_s) = \{(z, x) \mid (Z, x) \in H\}$ is the set edges referred to as full edges in bijective relationship with H;

$E_d \subseteq N_c \times N_s = \{(z_i, z) \mid z \in N_c \text{ and } z_i \in Z\}$ is the set of edges referred to as dotted edges connecting any compound node to its components.

4. Model of Hypergraph: A set of functional dependencies represented by hypergraph [7, 8] H = (N, E) over an attribute set A with number of vertices N = A, and edge with following:

$$E = \{(X, Y): F (X, Y) \in F \text{ and } Y \not\subset X\} \qquad (1)$$

Where F (X, Y), with X and Y are subsets of A, uniquely defines the value of attributes in Y if attribute value of X is given.

5. Dependency Matrix: The hypergraph contains the directed hyper edges [7, 8]. The directed hyperedge represented as E = (T (E), H (E)), where T (E) is set of tail vertices and H (E) is set of head vertices. From the equation 1, the set of tail vertices T (E) = X and the set of head vertices H (E) = Y. The dependency matrix of hypergraph H is V $\times$ E matrix and a_{ij} defined as follows

$$a_{ij} = \begin{cases} -1 & \text{if } v_i \in T(E) \\ 1 & \text{if } v_i \in H(E) \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

3. Methodology

Our design utilizes the methodology designed and developed by the researcher [6] to extract the attributes and functional dependencies from an application. The systematic approach to renovate the hypergraph that is in the form of dependency matrix constructed by a set of functional dependencies and attributes is given below:

Input: Matrix of Minimal Covered Functional Dependencies [7, 8]

Output: a-cyclic Matrix of Hypergraph

Step1: Create the additional row at the bottom of the matrix with all columns' value as 0. This row is called as a "key row".

Step 2: For each column of the matrix, readthrough the entire column element's value for with the value of -1. On a true condition, the key row column value corresponding to the readthrough column is stored as -1.

Step 3: The columns' attributes having -1 in column of key row together forms a candidate key. Then count the number of -1 in the row. This count is termed as "keycount"

Step 4: for a row and each column, if there is a (-1, -1) or (-1, 1) values corresponding to row, keycount row respectively then count such values. This count is termed as "row count"

Step 5: If row count is equal to the keycount then separate that row.

Step 6: Repeat step 4 and 5 until all rows of incidence matrix are traced.

5.0 Case Study

5.1 Employee System

To illustrate the methodology consider the data items and functional dependences for the Employee Information Database [5]. The data items are emp_id (A), dept_name (B), skill_id (C), emp_name (D), dept_phone (E), skill_name (F), emp_phone (G), dept_mgrname (H), skill_date (I), skill_lvl (J). The functional dependencies are 1) A$\rightarrow$ BDG, 2) B $\rightarrow$ EH, 3) C $\rightarrow$ F, 4) AC $\rightarrow$ IJ. The systematic procedure to illustrate our designed methodology is given below.

Step 1. The dependency matrix is constructed from the data items i.e. attributes and the functional dependencies, which is shown below

	A	B	C	D	E	F	G	H	I	J
1	-1	1	0	1	0	0	1	0	0	0
2	0	-1	0	0	1	0	0	1	0	0
3	0	0	-1	0	0	1	0	0	0	0
4	-1	0	-1	0	0	0	0	0	1	1

Step 2. The candidate key attributes are identified by the presence of -1 in the column. The attributes A, B and C have the -1 value in their respective columns. Hence they are taken as candidate key attributes. They are shown as -1 value in the last row.

	A	B	C	D	E	F	G	H	I	J
1	-1	1	0	1	0	0	1	0	0	0
2	0	-1	0	0	1	0	0	1	0	0
3	0	0	-1	0	0	1	0	0	0	0
4	-1	0	-1	0	0	0	0	0	1	1
	-1	-1	-1	0	0	0	0	0	0	0

Step 3. The number of -1's in last is 3. Hence the row count value is taken as 3. The row 1 have the row count value as 2 because of the attributes A, B columns have the value (-1, -1), (1, -1) with respect to key attribute row respectively. The value of key count 3 is not equal to the row 1 count value 2. Hence, the row is not discarded. The rows 2, 3, and 4 are retained since their row count values are 1, 1, 2 respectively. The resulting matrix is shown below

	A	B	C	D	E	F	G	H	I	J	
1	-1	1	0	1	0	0	1	0	0	0	2
2	0	-1	0	0	1	0	0	1	0	0	1
3	0	0	-1	0	0	1	0	0	0	0	1
4	-1	0	-1	0	0	0	0	0	1	1	2
	-1	-1	-1	0	0	0	0	0	0	0	**3**

Step 4. Group the attributes present in the each row for a relation.

5.2 Project Part Supplier System

Let us consider six attributes SUPPLIER (A), PART (B), PROJECT (C), COUNT (D), DATE (E) and COST (F). The functional dependencies are BC$\rightarrow$D, AC$\rightarrow$E, AB$\rightarrow$ F and AC$\rightarrow$ B [2].

The systematic procedure of eliminating cyclicity is as follows

Step 1. The attributes and functional dependencies are represented as incidence matrix, which is shown below

	A	B	C	D	E	F
1	0	-1	-1	1	0	0
2	-1	0	-1	0	1	0
3	-1	-1	0	0	0	1
4	-1	1	-1	0	0	0

Step 2. The attributes A, B and C have the elements value as -1 in their respective columns. Hence the attributes A, B and C becomes the candidate key attributes which are depicted as -1 value in last row of respective column. Further the key count is 3 since the last row consist of three -1 element's value. The matrix is shown below.

	A	B	C	D	E	F
1	0	-1	-1	1	0	0
2	-1	0	-1	0	1	0
3	-1	-1	0	0	0	1
4	-1	1	-1	0	0	0
	-1	-1	-1	0	0	0

Step 3. The row 1 have the -1 value corresponding to columns B, C respectively. Hence, the row 1 count have the value of 2. Similarly, the rows 2, 3 have the row count values 2, 2 respectively. The row 4 have the count value as 3 because of the A, B, C columns have the values (-1, -1), (-1, 1), (-1, -1) in correspondence with (key row, row). Hence, the row count value results as a 3. The matrix with count values is shown below

	A	B	C	D	E	F	
1	0	-1	-1	1	0	0	2
2	-1	0	-1	0	1	0	2
3	-1	-1	0	0	0	1	2
4	-1	1	-1	0	0	0	3
	-1	-1	-1	0	0	0	3

Step 4. The row 4 count and the key count values are same. Hence, the row 4 is detached from the matrix and resulting matrix is shown below.

	A	B	C	D	E	F	
1	0	-1	-1	1	0	0	2
2	-1	0	-1	0	1	0	2
3	-1	-1	0	0	0	1	2
	-1	-1	-1	0	0	0	3

6. Results

In this section, we are illustrating the correctness of our methodology by considering the data corresponding to the relations designed by the project, part supplied system utilized in the previous section. Without the application of our methodology, the following relations are designed by the pragmatic design methodologies [5]. The example data, which is stored in each relation, is shown below. For simplicity the example data is not entered in the first three relations

Part	**Project**	Count

Supplier	**Part**	Cost

Supplier	**Project**	Date

Supplier	**Project**	Part
Dell	1	Monitor

The process of updating is straight forward for existence of a table. We encounter the challenge that How the tuple 'dell', '1', 'monitor' is inserted and retrieved on the non-existence of table in a schema. Presuming that the INSERT operation is updating the number of relations' primarily key attributes the value is inserted in the database. To evidence this, the following relations with example tuple are considered.

Part	**Project**	Count
Monitor	1	NULL

Supplier	**Part**	Cost
Dell	Monitor	NULL

Supplier	**Project**	Date
Dell	1	NULL

We have proven that instead of designing an exclusive relation for supplier, project, part attributes, the tuple is inserted in the other relations corresponding with these attributes as key attributes. Since, non-key attribute's values of relations are unknown; currently their values are stored as NULL. When the relations are updated depending on the primary key value, the NULL value can be ameliorated to the fact value.

On storage of the tuple 'Dell', '1', 'Monitor' in relations, the application of relationally complete operators on these three relations retrieves the tuple i.e. apply the join and project on attributes Supplier, Project and Part. Hence, it is proved that any functional dependency constituted by the candidate key attributes becomes the redundant functional dependency. This functional dependency is isolated from the relation design to reduce the redundancy.

7. Conclusion

This paper attempts to provide an automated methodology to identify the candidate key attributes from the given set of the attributes and functional dependencies. Then functional dependencies constituted by the candidate key attributes are identified by the designed automated methodology. i.e. the attributes causing the cyclicity is determined. The isolation of this functional dependency (framed by the candidate key attributes) eliminates the cyclicity of the hypergraph. Hence, the reduction of number of functional dependency (ies) avoids the redundant relation(s) design. This is shown by the two case studies and discussion in result section.

In feature, the work is to be extended for identification and elimination of redundancy completely so that the basic aim of normalization i.e. elimination of complete redundancy is to be fulfilled. On other way, the isolated functional dependencies can be studied for the super class design in the Object Technology. The iterative application may yield the super-sub class hierarchy.

References

[1] Beeri. C, Fagin. R, Maier, Yannakakis, "On the desirability of the database schemes" ACM, vol 30, No. 3, 1983

[2] Fagin R. "Degrees of acyclicity for hypergraphs and relational database schemes" ACM Vol 30, No 3 1983

[3] G. Ausiello, G. Italiano, U. Nanni, " Optimal traversal of directed hypergraphs" TR-92-073, September 1992.

[4] Yun-zhou Zhu "Line graph of gamma-acyclic database sechemes and its recognition algorithm", VLDB, Singapore, August 1984.

[5] Patric O'Neil, Elizabeth O'Neil, "Database principles, programming and performance" second edition, Harcourt Asia Pte Limited, 2001.

[6] S M. Handigund, "Reverse Engineering of Legacy COBOL systems", Ph. D. thesis Indian Institute of Technology Bombay, 2001

[7] Chikkamannur A. A. Handigund S. M. "Categorization of Functional Dependencies for a Minimal Cover", ICSTC, San Diego USA. page 213-217, 2008.

[8] Chikkamannur A. A. Handigund S. M. "An efficient Methodology for Determining a Minimal Cover of Functional Dependencies", ICISTM 2008, Dubai. 2008. Unpublished

[9] Chikkamannur A. A. Handigund S. M. "A methodology for renovation of γ-acyclicity" VTU and ISTE sponsored, National Conference on Information Technology, BIT Bellary, 2010