GRAPHS AND THE THIRD NORMAL FORM FOR RELATIONAL DATABASE

Jože Nemec and Janez Grad*

University of Maribor, College of Agriculture, 62000 Maribor, Vrbanska 30, Slovenia
Phone: +386 62 226 611, Fax: +386 62 23 363
E-mail: joze.nemec@uni-mb.si

*University of Ljubljana, Faculty of Economics, 61109 Ljubljana, Slovenia
Phone: +386 61 1683 333, Fax +386 61 301 110

Keywords: relational database, relations, normal forms, normalization process, DB graphs, matrices

Edited by: Rudi Murn
Received: July 12, 1993 **Revised:** May 30, 1994 **Accepted:** June 9, 1994

Determination of higher order normal forms for a relational database (RDB) is frequently a time-consuming process. We can solve this problem by applying graph theory. In the paper the necessary characteristics of graphs that represent a given RDB are analyzed. The connectivity matrices for these graphs and the properties they must satisfy are also introduced and discussed. The established RDB graphs and the corresponding relationship matrices form an important basis of the algorithm for designing RDB with no redundant relations.

1 Introduction

The RDB application to data handling demands a careful design of RDB structure. The designing process consists of several stages. First the conceptual model of a DB has to be defined in which logical data structures of the information system (IS) are determined. An adequate conceptual model is an important prerequisite for a successful DB design. The RDB consists of a number of attributes that change during the time. Within the process of the RDB design the relations must be normalized. The characteristics of different normal forms are described in (Stout, Woodworth, 1983), (Elmasri, Navathe 1989) etc.

The normalization process becomes quite demanding in case of a large number of interrelated attributes. This asks for an experienced model designer. It is desirable to predetermine the process as much as possible in order to exploit the computer for performing the most tedious part of the task. Such procedures have already been developed in the past. In (Vetter, Maddison, 1981) a DB design procedure was described where the resulting model was obtained by ascertaining the redundant relationships. Another algorithm was presented in (Salzberg, 1986), and its modifications were published in (Atkins, 1988) and (Diederich, 1991),respectively. In the following paper we develop an RDB by means of graphs and state the basic characteristics of the graphs corresponding to the normalized RDB.

2 Graphs and relations

Let us present the relations in the form of graphs. Each attribute becomes a node within the graph. We denote the nodes as $x_1, x_2, \ldots, x_n$. A possible relation between two attributes is presented by means of a directed arc. For example, if x_1 stands for an employee's social security number (SSN) and x_2 stands for the employee's salary, we may express this relationship as illustrated in Figure 1.

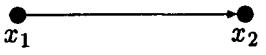

Figure 1: Directed relationship between nodes (attributes) x_1 and x_2

The directed arc begins in x_1 and ends in x_2. This is understandable because salary is defined by employee's social security number (SSN). The starting node represents the key attribute, while the ending node represents the non-key attribute of the relation R(x_1,x_2).

Let us briefly discuss the type of nodes that appear in graphs. We ascertain two groups of nodes that represent

(i) the simple attributes, and
(ii) the composite attributes, respectively.

A simple attribute represents some property of the entity, for example the colour of an article, birth date, etc. Two or more simple attributes may compose a composite attribute which serves as a key of some relation. By introducing composite attributes, we assure that to each value of the attribute, in which the arc starts, there belongs only one value of the attribute in which the arc ends. The composite attribute x_s defined by the simple attributes x_1, x_2, ..., x_k is shown in Figure 2. The arcs start in nodes that compose a composite key, and end in the composite key, which in turn is the key of the relation R(x_s,x_n). An arc starting in a simple attribute and ending in a composite attribute is a trivial arc. We shall mark trivial arcs by a broken line.

Suppose that we have defined the necessary data of a company in consideration. Since this data collection comprises the needs of different users (departments, services) it consists of a number of attributes and relationships between them. On the basis of the collected data we can draw a graph which consists of a set of nodes (attributes) and directed arcs (relationships) between them.

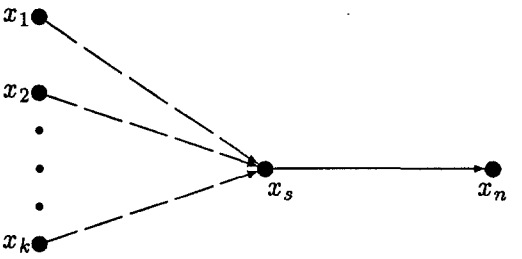

Figure 2: Composite attribute x_s defined by simple attributes x_1, x_2, ..., x_k

For each node x_i we can define the degree d(x_i). This is the number of arcs which either start or end in the node. We denote by $d^+(x_i)$ the outdegree of a node, i.e. the number of arcs which start in x_i, and by $d^-(x_i)$ the indegree of a node, i.e. the number of arcs that end in x_i. A simple attribute has $d^-(x_i)$ equal to one or zero, while a composite attribute has $d^-(x_i)$ greater than one.

If there exists a sequence of arcs between two nodes and the terminal node of the previous arc coincides with the starting node of the following arc, then the sequence forms a path between the two nodes. The path is called a closed path when the initial node and the terminal node of the path coincide, otherwise, the path is an open path. The path is called a cycle when all its arcs are different and any two nodes of it are different except for the initial one and the terminal one that coincide. The length of a path is the number of arcs that compose the path.

3 Basic characteristics of DB graphs

The process of a DB design results in a number of attributes and relations between them that determine the corresponding DB graph. In general, this graph represents a DB being in the first normal form (1NF). We want to transform 1NF into higher order normal forms, the graphs of which have some special properties. In the following paragraphs these properties will be discussed more in detail.

Theorem 1: *Suppose that the set of simple attributes X=(x_1, x_2, ...x_i, x_{i+1}, ...,x_n) defines the composite node x_s so that x_s and the set of simple attributes Y=(y_1, y_2, ...,y_{m-1}, y_m) form the relation R(X,Y), where X is the key of R. Suppose also that X'=(x_1, x_2, ...x_i) is a subset of X, and Y' is a subset of Y. Then no such Y' exists that R(X',Y'), where X' is the key of R.*

Suppose we have the graph as shown in Figure 3. x_1,x_2,...,x_n are simple attributes which compose the key x_s, and y_1,y_2,...,y_m are simple attributes dependent on x_s.

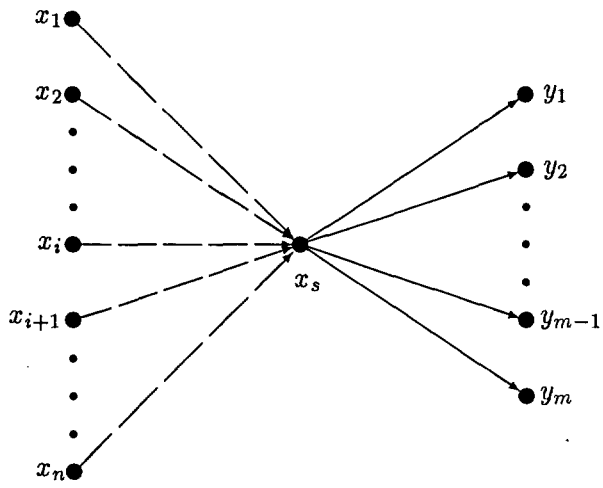

Figure 3: R(x_s,Y) graph of the composite attribute x_s

Let there be the attribute y_m which functionally depends upon the key composed of x_{i+1},...,x_n. The case is shown in Figure 4. x_s is composed of two subsets of attributes composing x_{s1} and x_{s2}. As y_m functionally depends upon the attributes of x_{s2}, we can replace the arc between x_s and y_m by the arc between x_{s2} and y_m. In this case y_m does not functionally depend upon attributes that compose x_s in Figure 3 and the corresponding relation is not in the second normal form (2NF).

While designing a DB we must analyze each attribute dependent on a composite key to find out whether it depends upon some subset of the key's attributes. If so we can decompose the key into a subset of keys where some attributes functionally depend upon one key only. In this case the DB is not in 2NF.

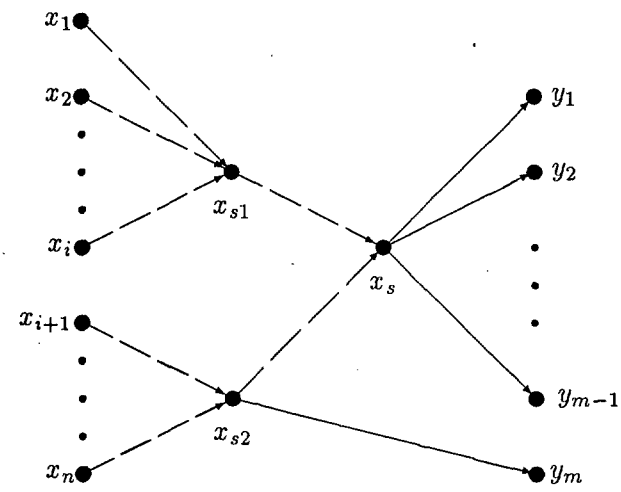

Figure 4: The key composed of two subsets of attributes

For the composite nodes some additional requirements must be met. For cycles the following theorem holds:

Theorem 2: *A cycle within a graph includes no node which is the composite key of some relation.*

Suppose such a cycle exists. Further, it is assumed that the initial (and therefore also the terminal) node is one of the attributes which form the composite key of some relation. Now we assume that we have the following sequence of arcs

$$x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow \ldots. \rightarrow x_n \rightarrow x_1$$

Let x_2 be the composite key of R(x_2,x_3), and x_1 one of the attributes that form the composite key. If we substitute all the arcs between nodes x_3 and x_1 with the transitive arc, we obtain the sequence

$$x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow x_1$$

x_2 is the composite key of a relation according to the supposition. If we now replace the arcs

$$x_1 \longrightarrow x_2 \text{ and } x_2 \longrightarrow x_3$$

by the transitive arc

$$x_1 \longrightarrow x_3$$

we see that x_3 depends upon x_1 only. The obtained case is the same as the case shown in Figures 3 and 4. The relation R(x_2,x_3) is not in 2NF what completes the proof of the theorem.

Now we can prove that for any DB graph the following theorem holds:

Theorem 3: *When a graph has the arc starting in the node x_1 and ending in the node x_2 then it has no other path starting in x_1 and ending in x_2.*

Suppose that Theorem 3 is not true and that there are an arc and some other path both starting in x_i and ending in x_j. Then this arc is a transitive arc for the path, and x_j is transitively dependent on x_i. The corresponding DB is not in the third normal form (3NF) and the following corollary can be stated:

Corollary 3.1: *There is only one path between two nodes.*

In the opposite case, i.e. when there are two paths, we can replace one of them by an adequate transitive arc.

Consider now also the notion of the graph connection.

The directed graph is said to be

(i) a strongly connected graph, when for any two different nodes x_i and x_j there exist the directed paths from x_i to x_j and x_j to x_i.

(ii) a one-way connected graph, when for any two different nodes there exists the directed path starting in one and ending in the second node. The directed path in the opposite direction may also exist.

(iii) a weakly connected graph when any two different nodes are connected by a set of optionally directed arcs.

(iv) a disconnected graph when it is not a weakly connected graph.

Let us consider how the graph connection reflects the DB graphs. For the graph of a DB in 3NF the following theorem holds:

Theorem 4: *The graph of a DB in 3NF is a weakly connected graph.*

The statement is based on the fact that all the nodes within the strongly connected graph are connected in both directions. Therefore, they are, reciprocally dependent, which is not allowed within 3NF. Further, in the case of a disconnected graph we can divide the nodes into at least two subsets where the nodes of one subset are disconnected from the nodes of the other. Therefore, there is no relationship between any two nodes of different subsets, which means that we deal with two different databases.

Suppose that DB is a one-way connected graph. Let x_0 be the starting node, and x_1 and x_2 the terminal nodes of two different arcs. Accordingly, as the graph is a one-way connected graph, there exists at least one path between x_1 and x_2 starting in one node, say x_1, and ending in the other one. Then there exists the transitive dependency between x_0 and x_2, which must not be the case in 3NF. A DB graph may, therefore, not be a one-way connected graph.

According to the statement above, a DB in 3NF can only be a weakly connected graph and, therefore, $d(x_i) \neq 0$ for any node x_i. This means that each node in a graph is either the starting node or the terminal node of at least one arc.

Let us now focus our attention to the properties of the DB subgraphs. Suppose we have a DB subgraph with all its nodes being strongly connected. We call such a subgraph a strong graph component of the given graph. Each graph may have several strong graph components. Since such a subgraph may represent a subdatabase, which is not in 2NF, the following theorem holds:

Theorem 5: *A DB graph may not involve strong graph components.*

When dealing with a large organization, we trace the relationships in the corresponding DB by setting up an initial model, which as a rule involves some redundant relationships. The database administrator task is to minimize the number of relationships. The problem can be solved by determining the minimal cover of the DB. The searching process is not uniform, as there may exist more minimal covers for the same DB. In order to obtain one, we must determine the or-

der in which to remove the redundant arcs. We can set up different removal criterions, like the longest transitive arcs, the arcs with the minimum access value, etc. By setting up an appropriate criterion, the process can become automatic.

4 Relations and matrices

As already stated, a RDB can be presented and described as a graph with nodes representing attributes and directed arcs representing relations between the attributes. We want to present and describe such a graph by means of adequate matrices. We show that the matrices which describe a DB also have some particular properties.

Let us first define the node relationship matrix and state its properties.

Definition 1: *For each DB graph we can build a square node relationship matrix* $P = \| p_{ij} \|$ *so that* p_{ij}, *where* $p_{ij} \geq 0$, *denotes the number of arcs starting in i-th and terminating in j-th node.*

Now we compute matrices P^2, P^3,... We prove the following theorems:

Theorem 6: *The element* r_{ij} *of R, where* $R = P^n$, *equals the number of the directed paths from the i-th node to the j-th node with the length of n.*

We prove the theorem by means of the total induction. It holds for n = 1 by the definition of P. Suppose that it holds also for P^n. Now we compute $P^{n+1} = P^n$ - P the (i,j) element of which, say c_{ij}, is obtained from

$$c_{ij} = \sum_{k=1}^{m} r_{ik} p_{kj}$$

According to the supposition, r_{ik} equals the number of the directed paths from the i-th node to the k-th node with the length of n. The element p_{kj} = p $\neq$ 0 if there exist p arcs starting in the k-th and ending in the j-th node. Therefore the product r_{ik}-p_{kj} equals the number of the paths with the length of n+1, the last arc starting in the k-th node. By computing and adding together $r_{ik}p_{kj}$ for all k = 1,..., m, where m is the number of the nodes, we obtain c_{ij}. The value of c_{ij} equals the number of directed paths heading from the i-th node to the j-th node with the length of n+1. This is just what we had to prove.

The following corollary also can be proved:

Corollary 6.1: *The elements* r_{ij} *of R, where* $R = P^n$ *and P is the relationship matrix, can only take the values 0 or 1.*

Suppose this is not true and there exist two directed paths between nodes i and j. We can replace one of the two by a transitive arc between nodes i and j. This arc, however, represents also a transitive arc for the second path, and therefore this is not the graph of a DB in 3NF. This contradicts the demand for a DB being in 3NF.

Theorem 7: *If an element* r_{ij} *of* P^n *is equal to 1, then all the corresponding elements* a_{ij} *of matrices P,* P^2, P^3, ... P^{n-1} *are equal to zero.*

Suppose $r_{ij} \neq 0$ and one of the corresponding $a_{ij} \neq 0$. Then two directed paths exist between nodes i and j, and the DB is not in 3NF what we proved earlier.

Let us now form a sequence of matrices D_1, D_2, ..., D_i, where the elements d_{ij}^k of D_k are defined as follows:

$d_{ij}^k = 0$, if $a_{ij} = 0$ for P, P^2, ..., P^k, and

$d_{ij}^k = 1$, if at least one corresponding element $a_{ij} \neq 0$.

Matrix D_k tells us that $d_{ij}^k = 1$, if there exists at least one path starting in the i-th node and ending in the j-th node with the length not exceeding k.

Theorem 8: *There always exists such integer m, that* $D = D_m = D_{m+k}$, *for k = 1, 2, 3,...*

Proof. The sequence of matrices D_k determines the node relationships within the paths composed of no more than k arcs. Suppose that the longest path in the graph which includes no cycles comprises m arcs. The value of m is less than or equal to the number of arcs in the graph. An optional element d_{ij}^m of D_m is not equal to zero if a path exists starting in the i-th node and ending in the j-th node, otherwise $d_{ij}^m = 0$.

Let there exist some additional matrices D_{m+k} which satisfy the conditions stated above. Let element d_{ij}^s of D_s be equal to zero for some s, where s ¿ m, and the corresponding element of

D_{s+1} be equal to one. Then a path of length s+1 exists between nodes i and j which includes no cycle. This contradicts the statement that the longest path in the graph which includes no cycles comprises m arcs. □□

We prove now the following corollary related to Theorem 8.

Corollary 8.1: *Matrix D defines the transitive closure of a DB.*

Transitive closure incorporates all elementary relations and also the relations obtained by all possible transitive arcs. We can introduce a transitive arc between two nodes only when there exists a path between the initial node and the terminal node. The path exists only then when the corresponding element d_{ij} of matrix D equals one.

Theorem 9: *If D represents a normalized DB then $d_{ij} = 0$ if $d_{ji} = 1$.*

Proof. Suppose the statement is not true and that we can write D in a form

$$\mathrm{D} = D_s + D_u$$

where D_s is a symmetric matrix and D_u is an asymmetric matrix such that

$$d_{ijs} = 1 \text{ and } d_{iju} = 0 \text{ if } d_{ij} = d_{ji} = 1$$

and

$$d_{ijs} = 0 \text{ and } d_{iju} = d_{ij} \text{ if } d_{ij} \neq d_{ji}.$$

This means that (i) $d_{ijs} = 1$ only when there is a path from node i to node j and a path from node j to node i, and (ii) $d_{iju} = 1$ only when there is a path from node i to node j and no path in the opposite way.

Consider first the matrix D_s. Suppose there exist at least two elements, say d_{ijs} and d_{jis}, not equal to zero. There may in addition exist some more elements in the i-th row and the j-th column which are not equal to zero. Observe the relations between the set of nodes defined by the non-zero elements in the i-th row. There exist paths from the i-th node to each of them and also in the opposite directions. We can, therefore, reciprocally connect any two nodes of this set. A DB composed of these nodes is a strongly connected graph. This is contrary to our statement that RDB can only be a weakly connected graph. Therefore, the initial statement must hold, stating that $d_{ij} = 0$ if $d_{ji} = 1$.

The theorem leads to the following properties of the elements of P.

Corollary 9.1: *If $p_{ij} = 1$, then $p_{ji} = 0$, for all $i \neq j$. All diagonal elements $p_{ii} = 0$.*

The statement holds because otherwise we would have the case where $d_{ij} = d_{ji} = 1$. Since all diagonal elements $p_{ii} = 0$, the graph must not include loops.

5 Algorithm for finding DB in the third normal form

Suppose that we have found all possible relations among the attributes. These relations are not in 3NF, so we carry out the normalization process. The process consists of several steps as shown in Table 1.

(1)	Determining the relationship matrix P
(2)	Determining the strong components of the graph
(3)	Removing the arcs from the cycles
(4)	Finding and removing the transitive arcs
(5)	Determining the relations in the data base

Table 1: The necessary steps for obtaining data base in 3NF

Let us explain the steps more in detail.

(1) The relationship matrix P is established which includes all possible arcs (nontrivial as well as trivial). Put D^1 = P.

(2) Matrices P^k, for k=2,3,.. are calculated, and the elements d_{ij}^k of D^k are determined by comparing the corresponding elements p_{ij} of P^k and d_{ij}^{k-1} of D^{k-1} . If one of these elements is equal to or greater than one, then $d_{ij}^k = 1$, otherwise $d_{ij}^k = 0$. The process ends when $D^{k-1} = D^k$. Put D = D^k and D = $D_s + D_a$, where D_s is symmetric and D_a is asymmetric. The strong components of the graph are determined by the non-zero elements of D_s: all the non-zero elements (attributes) within a row or column define a strong component.

(3) Suppose there exists at least one strong component forming a subgraph with all the nodes (attributes) being connected via a path in both

directions. This subgraph has cycles and at least one cycle is broken by removing an arc out of it. The process can be made fully automatic by applying some weight to the arcs (for instance the probabilities of their use) and removing the arc related with the lowest value of the weight. Otherwise it is up to a DB administrator to determine which arc should be removed.

The subgraph obtained by removing an arc is analyzed in the same way as the whole graph. The process ends after removing the arcs from all cycles. Matrix D becomes asymmetric.

Here we must point out that no composite attribute is a part of a strong component. If some strong component has a node which represents a composite element, this composite element is not defined properly.

(4) All transitive arcs are found and removed. Suppose that the element a_{ij} of P^k is equal to n (n¿1). In this case n paths can exist between the i-th node and the j-th node. The ending arcs of these paths can be found by comparing the i-th row of P and the j-th column of D^{k-1}. If the m-th elements of the corresponding row and column are equal to one, then the arc connecting the m-th node and the j-th node is an ending arc of the transitive path.

Transitive arc may also be found when the element a_{ij} of P^k and the element d_{ij} of D^{k-1} are equal to one. The ending arc corresponding to a_{ij} was discussed earlier. The ending arc corresponding to d_{ij} can be obtained by comparing the elements of D^1, D^2,..,D^{k-2}. Assume that the first non-zero (i,j) element is in D^t. The (i,j) element in P^t is also equal to one and defines an ending arc of the transitive path.

A path involving a transitive arc does not contain a composite node. We can prove this fact by applying the matrix D^{k-1}. Each trivial arc ends in the composite attribute. Assume that there exist two paths between the i-th node and the j-th node and that one of the two contains a composite node x_s. Then the i-th node is one of the attributes which define the composite attribute x_s. The path between x_i and x_j is longer than the path between x_s and x_j. Therefore the element d_{sj} of D^{k-1} equals one. In this case the composite node is not determined properly.

In order to remove the transitive arcs, the same criterion can be applied as for removing the arcs from cycles. After removing them, a data base in 3NF can be determined. First, all trivial arcs are removed and the remaining arcs are used to construct matrix P. Some columns of P contain only the zero elements. Suppose that the i-th column is one of them. The i-th row contains at least one non-zero element a_{ij}. This element defines the first arc of the path with x_i being the starting node and x_j being the second node. Now put a_{ij}=0 and observe the j-th row of P. The possible non-zero element a_{jk} defines the second arc of the path. The third arc is found by putting a_{jk}=0 and observing the k-th row. The process repeats until a row with all zero elements is found. When all elements of P are equal to zero all the paths in a graph have been found.

Suppose that the nodes in these paths are as follows:

Path 1: $x_{1,1}, x_{1,2}, x_{1,3}, \ldots x_{1,m1}$
Path 2: $x_{2,1}, x_{2,2}, x_{2,3}, \ldots x_{2,m2}$
Path 3: $x_{3,1}, x_{3,2}, x_{3,3}, \ldots x_{3,m3}$
Path 4: $x_{4,1}, x_{4,2}, x_{4,3}, \ldots x_{4,m4}$

.
.
.

Path k: $x_{k,1}, x_{k,2}, x_{k,3}, \ldots x_{k,mk}$

Each two neighboring elements in a row define a relation. The first element in each row is a key of at least one relation.

(5) We can now determine the set, say M, of the connected nodes. The necessary statements and tasks are as follows:

(i) All the nodes in the first path are in M.

(ii) If any node in M is one of the nodes which define the composite node x_s, then x_s is in M.

(iii) Find out whether there is a path with at least one node in M.

(iv) Add to M all the nodes in this path which are not in M.

(v) Repeat steps (iii) and (iv) until all the connected nodes are found.

Suppose that after this process all the nodes are in M. Then all the nodes are part of a DB. If not,

then there are at least two unconnected components and the graph of the DB is disconnected. This but contradicts the theorem that DB is a weakly connected graph. In a case like this we must determine the relations between the nodes of different components.

The relations in a DB can be found by following and executing the following procedure:

(i) Consider all different first elements x_1 (key attributes) in the rows.

(ii) The second elements in the rows represent the key depending attributes.

(iii) Remove the first element of each row, and eliminate the rows with only one element left.

(iv) Repeat steps (i) to (iii) until all the rows are eliminated.

The programming process is relatively simple and so is the data preparation. Only matrix multiplications are needed in the algorithm. All the redundant arcs can be determined by comparing the corresponding elements in matrices. We must define the connected attributes and for each connection the number (probability) of its appearance. In the beginning the probabilities can only be estimated, and determined more accurately within later stages of the process. In this way an optimum organization can be obtained by iterative reorganizations of the DB.

The process of finding the 3NF of a DB based on the graph theory can be carried out entirely automatically. In a large organization, the attributes and relations form a large connectivity matrix P. The number of operations within the solution process is high, too. This number can be reduced if some columns and rows of P need no processing. If all elements in the i-th row of P are equal to zero and only one element in the i-th column is equal to one, then this node represents the depending attribute in one relation only. This node has no effect on the elimination process and therefore the corresponding row and column can be removed. Similarly, if all elements in the i-th column of P are equal to zero and only one element in the i-th row is equal to one, then this node represents a key attribute in one relation only, and the corresponding row and column can be removed. By eliminating all such rows and columns, we get a reduced matrix P'. Only P' is needed in the normalization process. The size of this matrix is usually much smaller than the size of P. So the number of operations is considerably reduced and the third normal form can be obtained quickly.

6 Conclusion

In the paper we point out some particular properties which graphs must satisfy in order to represent a normalized DB. Matrices which describe these graphs must also comply with some requirements. This all helps us in exerting control over possible redundancies in the DB. By applying matrices P, D and higher powers of P, we can ascertain the redundant relationships. Matrix properties help in describing a DB and can also be exploited in designing an algorithm for the DB construction process.

By introducing the criterions for establishing and deleting the redundant arcs within different stages of the solution process, we can make the process of building up a DB in 3NF fully automatic.

7 References

Atkins John, A Note on Minimal Covers, SIGMOD Record, Vol. 17, No. 4, December 1988, pp. 16-21.

Diederich Jim, Minimal Cover Revisited: Correct and Efficient Algorithms, SIGMOD Record, Vol. 20, No. 1, March 1991, pp. 12-13.

Elmasri Ramez, Navathe B. Shamkant, Fundamentals of Database Systems, The Benjamin Cummings Publishing Company, Inc., Redwood City, California.... Reading, Massachusetts. New York ..., Bonn... 1989.

Salzberg Betty, Third Normal Form Made Easy, SIGMOD Record, Vol. 15, No. 4, December 1986, pp. 2-18.

Stout F. Quentin, Woodworth A. Patricia, Relational Databases, The American Mathematical Monthly, Vol. 90, 1983, pp. 101-118.

Vetter M., Maddison R.N., Database Design Methodology, Prentice Hall, Englewood Cliffs, 1981.