Printing Requisition/Graphic Services

ARTS 569 32 Dept. No. 28434

Title or Description
On optimization of a network model data base

Date December 15/78
Date Required A.S.A.P. PLEASE & THANK YOU
Account 126-6177-41 (D. Rotem)

Signature [signature]
Signing Authority D. Rotem

Department Computer Science
Room 5180
Phone 3143

Delivery
☐ Mail ☐ Via Stores
☐ Pick-up ☐ Other

1. Please complete unshaded areas on form as applicable. (4 part no carbon required).
2. Distribute copies as follows: White, Canary and Pink—Printing, Arts Library or applicable Copy Centre Goldenrod—**Retain.**
3. On completion of order, pink copy will be returned with printed material. Canary copy will be costed and returned to requisitioner, **Retain as a record of your charges.**
4. Please direct enquiries, quoting requisition number, to Printing/Graphic Services, Extension 3451.

Reproduction Requirements
☑ Offset ☐ Signs/Repro's ☐ Xerox

Number of Pages 20
Number of Copies 50

Type of Paper Stock
☑ Bond ☐ Book ☐ Cover ☐ Bristol ☑ Supplied FRONTS & BACKS

Paper Size
☑ 8½ x 11 ☐ 8½ x 14 ☐ 11 x 17

Paper Colour
☑ White ☐ Other

Ink
☐ Black

Printing
☑ 1 Side ☑ 2 Sides

Numbering to

Binding/Finishing Operations
☑ Collating ☑ Corner Stitching ☐ 3 Ring ☐ Tape ☐ Plastic Ring ☐ Perforating

Folding
Finished Size 3 Staples

Cutting
Finished Size

Special Instructions
FRONTS &
BACKS ENCLOSED

Cost: Time/Materials	Fun.	Prod.Un. Cl.	Prod. No.	Opr.	Mins.	Total
Signs/Repro's	1					
Camera	2					
Correcting & Masking Negatives	3					
Platemaking	4					
Printing	5			32	15	
Bindery	6			4	20	
J3						
Sub. Total Time						
Sub. Total Materials						
Prov. Tax						
Total						

Film		Plates	
Qty	Size	Qty	Size & Type

Paper		Plastic Rings	
Qty	Size	Qty	Size

Outside Services

Graphic Services May 68 482-2

ON OPTIMIZATION OF A NETWORK MODEL DATA BASE

by

E. Bretholz

and

D. Rotem

Research Report CS-78-14

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

March 1978

ABSTRACT

After designing and implementing a Data Base, a revision of the design is needed to conform to the gathered statistics on the working Data Base. The problem of optimising the design with respect to the storage requirements and access cost has been formulated as an integer, fixed charge program. For small networks this solution is appropriate. However, the complexity of this solution grows exponentially with the size of the input. In the special case, where the corresponding dependency graph of the network is a tree, a suboptimum algorithm can be applied with bounded error.

Key- words and Phrases

data base, data model, DBTG Report, data base design, fixed charge, integer programming

CR Categories: 4.33, 4.34, 5.32, 5.41

§1 Introduction

Two different approaches to the design of a data model for a Data Base are mentioned in the literature: the existential method and the functional method.

The existential approach "models the enterprize" independently, disregarding the actual use of the Data Base. Using this approach the Data Base Administrator (DBA) designs the data-model, based on the structural organization and interdependencies intrinsic in the enterprize [2,3].

The functional approach views the data as a source of answers to a set of anticipated queries. Therefore the design of the data model is based on the dependencies as expressed in all the anticipated queries [5,7]. Clearly, the functional approach will lead to a data model which answers most efficiently the anticipated queries. However, any modifications in those queries may necessitate a costly redesign.

Both, the above mentioned techniques, result in the schema design. In practice, after the implementation of the Data Base, performance measurements can be used as a feedback into the designing process. At this stage the redesign is performed, taking the functional approach. In this paper, we deal with an efficient use of this feedback information in optimising a given network model by removing possible redundancies.

Under the Network Model [2], the schema can be viewed as a directed graph G, where the directed edges represent set-types, directed from an owner-record type to a member-record type.

Any query $q \in Q$, is implemented in this model, by a sequence of FIND statements such that the first FIND directs us to a certain node in G (e.g. using location modes as DIRECT, CALC) [2], and the remaining ones either leading to another node in G (e.g. using location mode via SET) or searching for a record occurence in the last found set-occurence (e.g. FIND NEXT IN <set-name>). Therefore, we assume that the cost of answering a query, the Access Cost (AC) is directly proportional to the number of records processed to get the answer.

On the other hand, there is a cost associated with the creation and maintenance of each set in the Data Base (e.g. storage tied up by pointers on each owner and member record occurences). This cost will be denoted as CC and assumed to be proportional to the average set-size for every set-type. Clearly, there is a trade-of between the two costs CC and AC.

Using the above graph representation, answering a given query requires the traversal of a path in G. However, it is possible that a given query can be answered using several alternative paths.

The goal of the schema designer, the DBA, is to minimize the overall cost for a given application. Mitoma and Irani [7] consider the automatic design of an optimized Data Base schema. However we observe that further cost reduction can be achieved by removing redundant sets and diverting, the queries using them, through alternative paths.

In section 2 some definitions and general concepts of the DBTG model are presented. In section 3, an analytic model is constructed to solve this minimization problem which is formulated as a fixed charge integer problem.

In section 4, an algorithm is presented for a special type of graph, which leads to suboptimal solution with a bounded error.

§2 The Network Model

It is assumed that the reader is familiar with the concepts of the network data model [2]. However, for completeness, the major features of the model are included. Under this model, the SCHEMA is the logical view of the data base. The schema defines a collection of set types; each set type has one owner record type and at least one member record type. A record type may be a member or owner of more than one set type but it cannot be a member and owner of the same set type. It is not allowed to have two set types S_1 and S_2, such that the owner and member record types of S_1 are member and owner type of S_2, respectively. In the data base, to each set type correspond set occurrence s and to each record type, record occurrence s . In what follows, the name 'set' and 'record' will denote set occurrence and record occurrence respectively.

Each set S can be viewed as a circular linked list whose head node is an owner record O(S), and the other nodes are member records M(S).

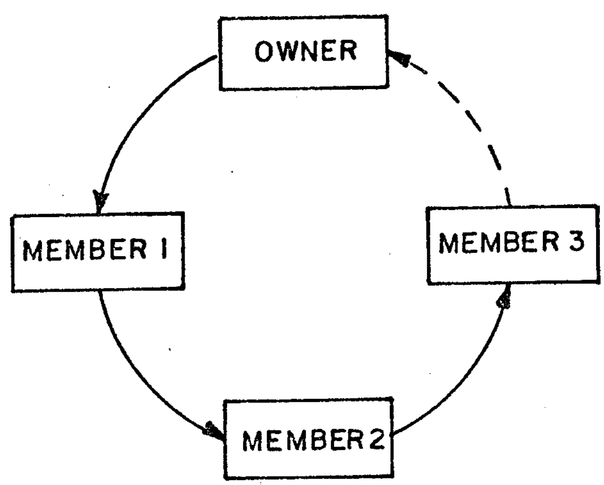

Figure 1 - Set Occurrence

A record occurrence may not be a member of more than one ocurrence of the same set type.

§3 The Theoretical Model

Given the schema φ with the set types $\{S_1,\ldots,S_n\}$ and record types $\{R_1,\ldots,R_m\}$, the directed graph $G = (V,E)$ represents φ if and only if:

(i) $V = \{R_1,\ldots,R_m\}$

(ii) $E = \{S_1,\ldots,S_n\}$

(iii) the edge S_i is directed from R_k to R_ℓ $\Leftrightarrow$ $R_k = O(S_i)$ and $R_\ell = M(S_i)$.

Obviously, from the network model restrictions G will not contain any directed loop of length smaller than three.

The following weights will be assigned to each edge $S_i \in E$.

(1) N_i - the number of traversals of S_i i.e. the number of times $M(S_i)$ has been reached from $O(S_i)$.

(2) AC_i - the number of records processed in the above N_i traversals

(3) CC_i - the storage cost of set type S_i.

Note that N_i, AC_i and CC_i are the sum of the respective values of all individual set occurences within set type S_i.

The resulting weighted graph is denoted by

$G_W = (V,E,f)$ where $f(S_i) = (N_i,AC_i,CC_i)$.

A key concept in the process of removing redundancies is transitivity which we now describe. Given three set types S_1, S_2 and S_3 and three record types R_1,R_2,R_3 such that

$R_1 = O\ (S_1)$ $\qquad$ $R_2 = M\ (S_1)$

$R_2 = O\ (S_2)$ $\qquad$ $R_3 = M\ (S_2)$

$R_1 = O\ (S_3)$ $\qquad$ $R_3 = M\ (S_3)$

Let $M_i(x)$ denote the set of members of the set occurence of type S_i defined by the owner occurence x. We say that the subgraph $G^* = (V^*,E^*)$ where

$V^* = \{R_1,R_2,R_3\}$ and

$E^* = \{S_1,S_2,S_3\}$

is 2-level transitive (2-transitive) if for all $a \in R_1$ we have

$$M_3(a) = M_2(M_1(a)). \qquad (2)$$

Intuitively, 2-transitivity, indicates that queries which use the set S_3, may be directed through the alternative path S_1 and S_2.

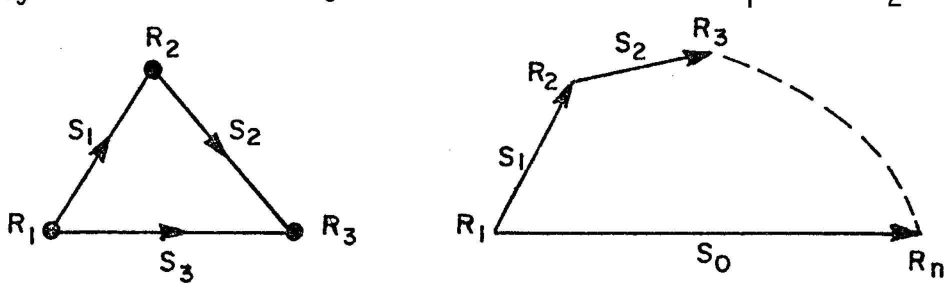

Figure 2. 2(a) 2-transitivity 2(b) n-transitivity

We can easily extend the above n-transitivity definition to n-level transitivity, as shown in figure (2b).

Note that equation (2) => (1)

but (1) $\not\Rightarrow$ (2)

as shown in the following example:

Example 1

Given the following three record types R_1 = DEPARTMENT; R_2 = ADVISOR; R_3 = STUDENT and the following set types.

S_1 = DEP - ADV such that $O(S_1) = R_1,\ M(S_1) = R_2$

S_2 = ADV - STU such that $O(S_2) = R_2,\ M(S_2) = R_3$

S_3 = DEP - STU such that $O(S_3) = R_1,\ M(S_3) = R_3$.

Consider, a particular student r_3 in department r_1; r_3 may have an advisor r_2 who is a member of department r_1' such that $r_1' \neq r_1$. This contradicts (2) while (1) still holds, i.e.

$$r_3 \in M_3(r_1) \quad \text{and} \quad r_3 \notin M_2(M_1(r_1)) \ . \qquad \square$$

The following example shows a case of transitivity.

Example 2

Given:

R_1 = DIVISION; R_2 = DEPARTMENT; R_3 = EMPLOYEE

and

S_1 = DIV - DEP where $O(S_1) = R_1,\ M(S_1) = R_2$

S_2 = DEP - EMP where $O(S_2) = R_2,\ M(S_2) = R_3$

S_3 = DIV - EMP where $O(S_3) = R_1,\ M(S_3) = R_3$

Clearly $\forall r_3 \in R_3$

if $r_3 \in M_2(r_2)$ and $r_2 \in M_1(r_1)$

then $r_3 \in M_3(r_1)$ and vice versa.

$$\therefore \qquad M_3(r_1) = M_2(M_1(r_1))$$

In other words, the department to which an employee belongs uniquely determines the division.

Definition

An edge S_0 in G_w from R_1 to R_n, is called redundant if there exists a path $P_0 = (S_1, \ldots S_n)$ from R_1 to R_n such that the subgraph $G^* = S_0 \cup P_0$ is n-transitive (figure 2b).

The decision whether or not to remove a redundant edge of G_w, depends on the net change incurred to the total system cost.

Let us compare the cost of answering a given query in G^* (figure 2b) using each of the two possible alternatives.

Let K_i be the average number of records processed for answering N_i queries using edge S_i in G^*.

Clearly:

$$K_i = \frac{AC_i}{N_i} \qquad i = 0, 1, 2, \ldots, n. \tag{3}$$

Removing the edge S_0 from G^*, leads to a redistribution of FIND statements resulting in a change in the Access Cost for each edge in the alternative path P_0.

The sets and records in the data base which correspond to G^* can be viewed as a structure composed of tree 1 and tree 2 (figure 3) where $|S_i|$ is the average number of member records in a set of type S_i.

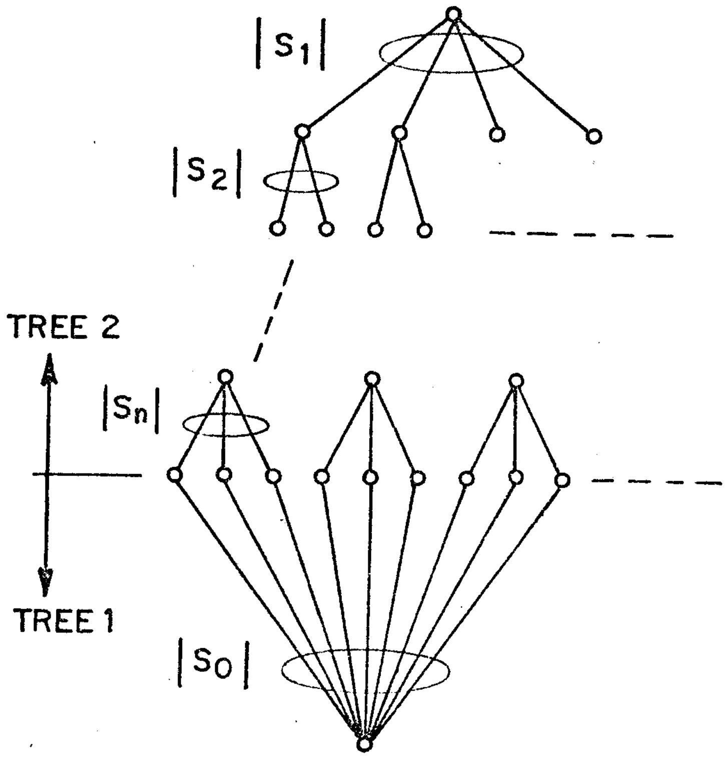

Figure 3. The Data Base of G^*.

Processing a query q_1 of N_0 using edge S_0 is equivalent to processing K_0 leaves of tree 1, whereas the removal of S_0 implies that q must be answered using tree 2. Since we have to process K_0 leaves in both cases, using tree 2, requires the processing of

$$K_{n-1} = \frac{K_0}{|S_n|} \text{ of their direct ancestors.}$$

By using this argument recursively we get

$$K_{n-j} = \begin{cases} K_0 & \text{for } j = 0 \\ \dfrac{K_0}{\prod_{i=0}^{j-1} |S_{n-i}|} & \text{for } n-1 \geq j \leq 1 \end{cases} \tag{4}$$

Noting that $|S_0| = |S_1| * |S_2| * \ldots * |S_n|$ the above equation (4) can be simplified to

$$K_j = \frac{K_0}{|S_0|} \prod_{i=1}^{j} |S_i| \quad \text{for } 1 \le j \le n \tag{5}$$

A MILP formulation

We assume that all alternatives of answering a given query from R_1 to R_n are known, i.e. the set of all transitive paths between R_1 to R_n. A relaxation of this assumption, will require the application of an algorithm for finding all paths from R_1 to R_n [6] as well as checking for the transitivity property for each such path.

The following notations are used in the model

Q - the number of query types

q_i - the number of alternatives of query of type i

N_i - number of type i queries

ν_{ij} - a vector of length M (number of edges) where

$$\text{for } k = 1,2\ldots M, \nu_{ij}^{(k)} = \begin{cases} 1 - \text{if } S_K \text{ is used by alternative } j \text{ in query type } i \\ 0 - \text{otherwise.} \end{cases}$$

a_{ij} - the number of records processed for answering one query of type i by alternative j.

$$a_{ij} = \sum_{\ell=1}^{M} K_\ell \, \nu_{ij}^{(\ell)}, \quad \text{where } K_\ell \text{ is defined in (5)}$$

$$x_{ij} = \begin{cases} 1 \text{ - if alternative } j \text{ is chosen for query } i \\ 0 \text{ - otherwise} \end{cases}$$

c_k - the cost of maintaining a set of type k, $k = 1,2...M$.

$$y_k = \begin{cases} 1 \text{ - if set of type } k \text{ is to be implemented} \\ 0 \text{ - otherwise} \end{cases}$$

ST - the total available storage space

Under this notation, the objective function to be minimized is the total cost

$$C = \sum_{i=1}^{Q} \sum_{j=1}^{q_i} N_i \, a_{ij} \, x_{ij} + \sum_{k=1}^{M} c_k \, y_k \tag{6}$$

subject to the following constraints

$$\sum_{j=1}^{q_i} x_{ij} = 1 \quad \text{for all } i = 1,...Q \tag{7}$$

i.e. exactly one alternative is chosen for each query type.

$$\sum_{k=1}^{M} c_k \, y_k \leq ST \tag{8}$$

i.e. the total used storage does not exceed the total available storage

Let

$$X^{(k)} = \sum_{i=1}^{Q} \sum_{j=1}^{q_i} x_{ij} \, \nu_{ij}^{(k)} \tag{9}$$

Clearly, $X^{(k)} = 0$ if S_k has not been used by any chosen alternative and $X^{(k)} > 0$ otherwise.

Therefore we have the following constraints

$$(1 - x^{(k)})\ y_k = 0 \tag{9a}$$

$$x^{(k)}\ y_k = x^{(k)} \tag{9b}$$

(9a) together with (9b) imply that

if $x^{(k)} = 0$ then $y_k = 0$

if $x^{(k)} > 0$ then $y_k = 1$

i.e. if an edge S_k is chosen by at least one alternative, a fixed charge c_k will be added only once to C.

This problem is called the "fixed charge problem" [4] , and is a hard integer linear programming problem, with complexity growing exponentially in the network size.

Therefore a suboptimum algorithm is proposed for special network topology commonly found in practical applications.

§4 A Special Case

For each redundant edge S_i in G_W we define Δ_i to be the net cost change achieved by removing S_i from the network, assuming that the queries N_i will be directed via the cheapest alternative to S_i.

Clearly $$\Delta_i = c_i - N_i \sum_{i=1}^{M} K_m \nu_{ij}^{(m)} \tag{10}$$

where alternative j is the cheapest of the q_i possible alternatives.

We say that edge S_i from X to Y is dependent on edge S_j, if S_j is on a transitive path from X to Y. Removing a redundant edge S_k, will result in a decrease in the total network cost by Δ_k. However for each of its dependent edges S_i, we have to update the Δ_i.

Our algorithm has to find an optimal order of deletion among the redundant edges. Clearly, only the order of deletion among mutually dependent edges is significant.

<u>Lemma 1</u>: Let Δ_i' be the new net cost change for edge S_i after the removal of a redundant edge S_k. Then

$$\Delta_i' \leq \Delta_i \qquad i = 1,\ldots,M, \quad i = k \tag{11}$$

<u>Proof</u>:

(1) If S_i and S_k are mutually independent then $\Delta_i' = \Delta_i$.

(2) If S_i depends on S_k then two cases are possible.

(a) S_k is on the minimal cost alternative of S_i. Therefore the removal of S_k increases the cost of the minimal alternative, decreasing the net cost change i.e., $\Delta_i' < \Delta_i$.

(b) S_k is not on the cheapest alternative. In this case it's removal will not affect Δ_i.

(3) If S_k depends on S_i and S_i is on the cheapest alternative, all queries N_k will be diverted through S_i causing $\Delta_i' < \Delta_i$ by (10). □

Let us construct the dependency weighted directed graph D_w from G_w as follows:

- each redundant edge $S_k \in G_w$ is represented by a vertex k, with weight Δ_k.
- $\langle k,j \rangle$ is a directed edge in D_w iff S_k depends on S_j.

Theorem 1: Given a network with initial cost C_0, let Δ_i be the net cost gain on the redundant edge S_i $i = 1,2,\ldots,\ell$. Then the optimum cost C_{opt} satisfies

$$C_{opt} \geq C_0 - \sum_{i=1}^{\ell} \Delta_i = C^* \tag{12}$$

Proof:

Since an optimal network, is obtained by removing redundant edges in a certain order, it follows from Lemma 1, that removing S_k will decrease the cost to $C_0 - \Delta_k$ and $\Delta_i' \leq \Delta_i$ for the remaining redundant edges.

At most, all redundant edges will be removed before achieving the optimal cost network. Hence the net decrease in network cost can not exceed $\sum_{i=1}^{\ell} \Delta_i$.

Therefore $C_{opt} \geq C_0 - \sum_{i=1}^{\ell} \Delta_i$. □

In the special case, when the dependency graph D_w is a tree, a suboptimum solution with bounded error is achieved by deleting the edges of G_w corresponding to a maximum weighted independent set of D_w.

Theorem 2: Let D_w be the dependency graph of G_w and D_w is a tree. Let $L = \{S_1, S_2, \ldots, S_k\}$ be the edges in G_w corresponding to the vertices of a maximum weighted independent set of D_w. Then

$$C(G_w - L) - C^* \leq \tfrac{1}{2} \sum_{i=1}^{\ell} \Delta_i. \qquad (13)$$

where $C(G_w - L)$ represents the total cost of the resulting network after removing the set of edges L from G_w.

Proof:

The total weight of the vertices of D_w is equal to

$$\sum_{i=1}^{\ell} \Delta_i.$$

Consider the independent sets D_1 and D_2 in D_w constructed by choosing the nodes in alternating levels of the tree, D_1 levels 0,2,4,... and D_2 levels 1,3,5... . Clearly $\max(w(D_1), w(D_2)) \geq \tfrac{1}{2} \sum_{i=1}^{\ell} \Delta_i$ where $w(D_1)$ is the total weight of D_i. Therefore, any choice of a maximum independent set L satisfies (13). □

An efficient linear algorithm, for finding a maximum weighted independent set of a tree has been developed by Cockayne and Hedetniemi [1].

It follows from theorem 2, that in this case using this algorithm will result in an error which is bounded by $\frac{1}{2} \sum_{i=1}^{\ell} \Delta_i$.

Note that, in many cases, where D_w is not a tree we can easily modify the initial network G_w so that cycles in D_w are removed. This can be done by merging nodes in G_w.

For example, G_1 in figure 4 has the dependency graph D_1 which contains a loop; we remove this loop by merging node 3 and 4 as shown in G_2, and the resulting D_2 is a tree.

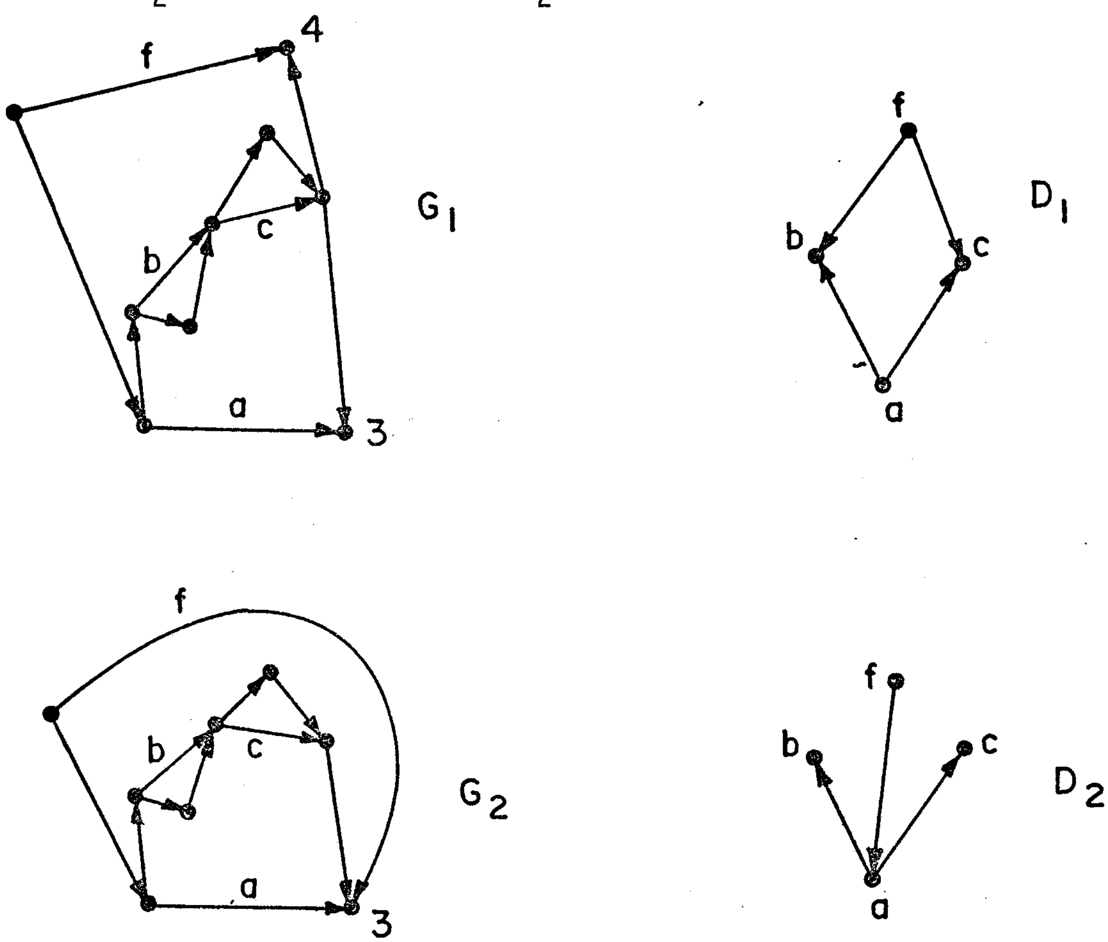

Figure 4: Removing a loop from the dependency graph.

Conclusions

The problem of minimizing the total cost of a Data Base operation has been investigated. The concept of transitivity in a network was introduced and used in the formulation of a model which reduces the above problem to a "fixed charge problem". For the special case of a tree dpendency graph, a suboptimal solution was obtained.

It remains an open problem whether this technique can be applied to a larger class of networks.

References

[1] Cockayne, E. and Hedetniemi, S.
A linear algorithm for the maximum weight of an independent set in a tree. Proceedings of the Seventh Southeastern Conference on Cambinatorics Graph Theory and Computing. Louisiana State University, Baton Rouge, 1976.

[2] Codasyl, Data Base Task Group,
April 1971 report, ACM, New York.

[3] Date, C.
An introduction to database systems.

[4] Garfunkel, R.S. and Newhauser, G.L.
Integer Programming
Willey, New York, 1972.

[5] Gerritsen, R.
A Preliminary System for the Design of DBTG data structures. Comm. ACM - Oct. 1975, pp.551-557, Vol. 18 No. 10.

[6] Horowitz, E. and Sahni, S.
Fundamentals of Data Structures, Computer Science Press, 1976.

[7] Mitoma, M.F. and Irami, K.B.
Automatic database schema design and optimization,
Proc. of the International Conference on Very Large Databases, ACM, New York, pp.286-321.