THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY[1]

A METHOD FOR DATABASE RECORD CLUSTERING

Wan P. Chiang and Toby J. Teorey

CRL-TR-5-82

AUGUST 1982

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

[1]Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

ABSTRACT

Record clustering is one of the important problems in physical database design. A heuristic method is proposed for identifying the record types in a database that are to be clustered together so that total access time is minimized for the designed database. A paging environment is assumed. The clustering result, while known to be optimal for hierarchical databases, is shown to be at least near optimal for network databases.

Keywords and Phrases: hierarchical database, network database, CODASYL systems, clustering, physical database design, storage allocation.

CR Categories D.4.3(File System Management), E.2(Data Storage Management), H.2.2(Physical Design)

1. INTRODUCTION

An important cost factor of a physical database design depends on the secondary storage search time that is required by the typical operational workload. The two major components of that search time are the expected number of I/O accesses and the expected distance of the read/write head movement for each access. Thus, one important consideration in the initial physical database design, or a later reorganization of the database, is to map the logical database structure into a secondary storage in such a way so as to minimize both components of the secondary storage search time. One way to minimize the expected number of I/O accesses is to make use of the database usage pattern to group occurences of those record types that are logically related and are frequently accessed together as clusters and store into the same area or page so that inter-page accesses or page faults can be minimized (record clustering). An algorithm for mapping a set of clusters onto secondary storage so as to minimize the expected distance of the read/write head movements will also be presented (storage mapping). Figure 1 illustrates an abstract view of the mapping processes:

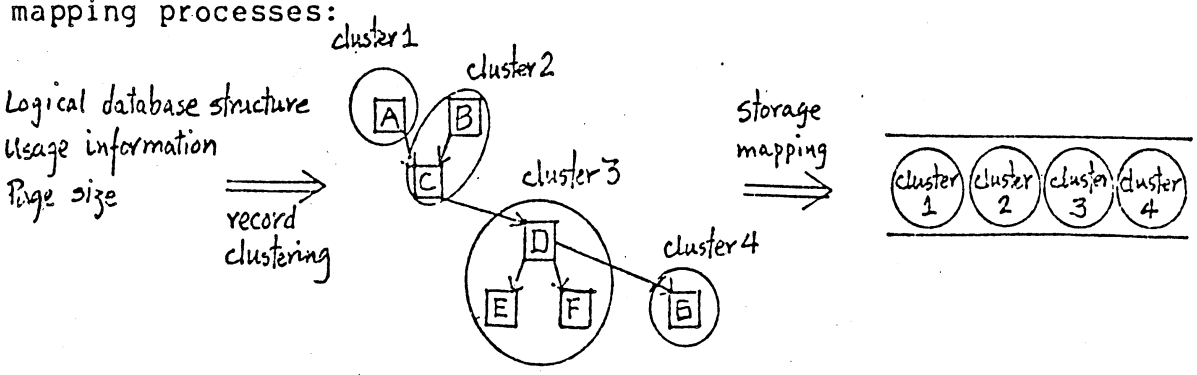


Figure 1 Logical to physical structure mapping process

The physical database record clustering problem has been previously addressed by several authors. Hoffer defines a measure of similarity between pairs of data items (attributes) and uses these measures to group data items into subfiles [Hoff75]. Malmquist proposes a method that uses a direct graph to represent record occurrences and their relationships. Each link in the graph is given a weight of one. The graph is then partitioned into pages so as to have minimal inter-page connections (weights) [Malm78]. Oberlander formalizes the use of a direct graph to represent database structure. His nodes and links correspond to record types and their relationships. He also partitions the graph into pages so as to have minimal inter-page connections by using user supplied weights on each link [Ober79]. Schkolnick uses a cost model instead of weight assignments. Using the cost model and a dynamic programming algorithm, his method produces optimal clustering results for hierarchical database structures [Schk77,78].

The following table summarizes the important features each approach has:

Features	Hoff	Schk	Malm	Ober
1.Generality:				
Tree structured DBMS		X		
Network structured DBMS			X	X
Both types of DBMS	X			
2.Unit of clustering:				
Record occurrence			X	
Attribute	X			
Record type		X		X
3.Clustering result:				
Optimal		X		
Near optimal	X		X	X

The Database Record Clustering Method (DRCM) proposed here extends Schkolnick's method to encompass the following desirable features:

1. It can be applied to any data model DBMS.
2. It uses the database usage pattern, which is fairly easy to gather and output is easy to interpret.
3. It produces optimal solution for tree structured databases and near optimal solution for network structured databases.
4. It can be applied to larger databases since it works on record types instead of record occurrences or attributes.

DRCM consists of three general phases. The first phase transforms a logical database structure and usage information into a clustering graph (i.e., a directed graph used to represent the logical database structure and associated usage information). The second phase partitions that clustering graph (if it is network structured) into a set of disjoint tree structured clustering graphs. If the graph that is produced by the first phase is already a tree structured graph, then phase two will not change anything. The third phase obtains optimal clustering schemes for each tree structured clustering graphs in the set produced in phase two. Figure 2 illustrates an abstract idea of the three phase process:

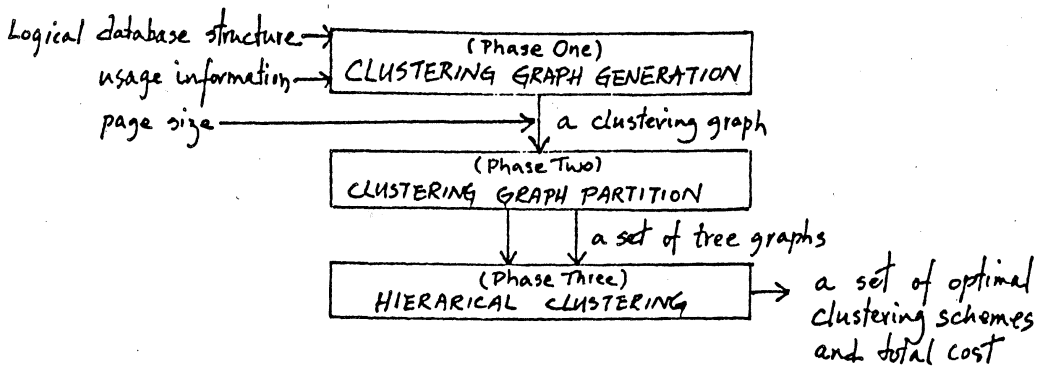


Figure 2 Three phases of DRCM

2. THE COST MODEL

A clustering scheme is an assignment of each member record type in a logical database structure to be stored with one of its respective owner record type via a relationship between them. Therefore, if there are m relationships in the logical database structure that can be used as access paths, then there are 2**m possible clustering schemes. To determine whether a certain clustering scheme is desirable or not, a measurement of its cost in terms of the number of physical block access (PBA), or page faults must be made. To determine the number of PBAs, a cost model which is based on Schkolnick's simplified cost model [Schk78], extended to accommodate network structure databases, is used.

2.1 Cost Model Assumptions

The DRCM cost model makes four sets of assumptions:

Assumption One:

A logical database structure and its related information produced in a logical database design are given. They include:

1) A set of record types. (Three typical elements of it will of it will be denoted by I, J and K.)
2) A set of relationships over record types. (A relationship S between record type I and J will be denoted by (I,S,J).)
3) A set of entry points, E(I).
4) The size of each record type, which is the length of of logical record plus record pointers, record_size(I).
5) The average number of occurrences of each record type, N(I).

Assumption Two:

Any access to a logical database structure is via an entry point. Examples of entry points are root segments for hierarchical databases, system owned record types for network databases, and secondary indexed record types for both types of databases. For relational databases an entry

point could be any relation.

An access traverses the structure in a certain pattern of transition and terminates at a record occurrence. The DRCM cost model takes into account two typical transition patterns. They are:

1) Owner to member (or Parent to child) transition -- An access transition from an owner record type occurrence to the first occurrence of one of its member record types via a relationship.
2) Twin (or sibling) transition -- An access transition from one occurrence to the next occurrence of a record type.

Certain information concerning these two patterns of database usage are also assumed to be available in the DRCM cost model. They are:

1) The frequency of an owner (I) to member (J) transition via a relationship (S), F(I,S,J).
2) The frequency of a twin transition, F(I,I).

Note: This information can be estimated at the initial design stage by the expected frequency of usage of application programs, their access paths and the average number of record occurrences of each record type on the access path. If the clustering is for a database reorganization, then the trace statistics from daily or hourly access activities of the database can be used [Schk78].

Assumption Three:

Only a set of hierarchically related record types (i.e., record types forming a tree structure) can be grouped together as a cluster, and their occurrences will be stored in preorder (hierarchical) sequence on a physical extent (AREA in CODASYL systems, data set group in IMS, and segment in System R). A physcial extent is further divided into pages.

For example, Figure 3 shows the physical extent layout of a cluster which contains record types A, B, and C (TeFr82).

Note: Pointers that exist as part of the record occurrences and are used to represent the logical relationships of the record occurrences are implemented differently by each DBMS. However, regardless of how these pointers are implemented they have nothing to do with clustering whatsoever and thus will not be affected by it. Therefore, the clustering process changes only the physical location of the record occurrences and have absolutely no

affects on their logical relationships.

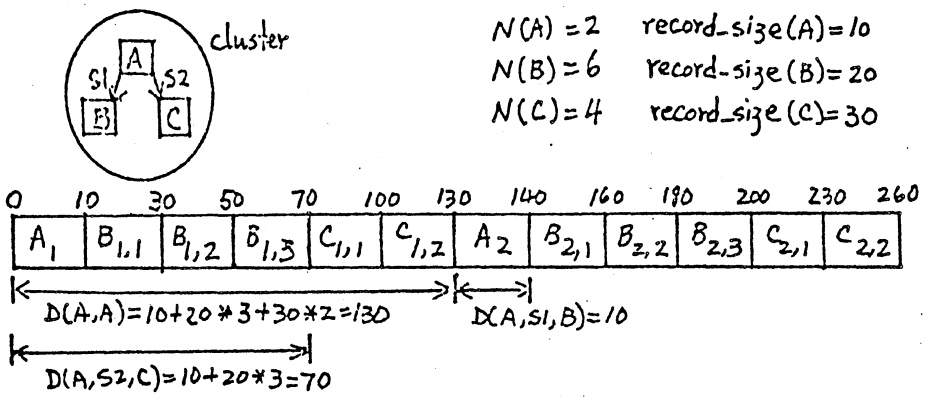


Figure 3 Physical extent layout of a cluster

Assumption Four:

The operating environment of the database is assumed to be a paged memory system. To simplify the cost model, it is also assumed that every access transition is independent of preceding ones and that the buffer used to implement the paging management policy of the DBMS contains only one page. Thus, any access to a record occurrence that is not in current page will result in a physical block access. The page size, denoted by page_size, is assumed to be known.

2.2 Formulation of the Cost Model

Based on the four assumptions above, the cost model can be formulated as follows:

From assumptions one, two and three, the expected distance from an occurrence of record type I to its next occurrence (see Figure 3) is:

$$D(I,I) = record\_size(I) + \sum_{J \in M1} [D(J,J) * AVG(I,S,J)]$$

where M1 is the set of member record types that are clustered together with I, and AVG(I,S,J) = N(J)/N(I).

The expected distance between an occurrence of owner record type I and its first occurrence of member type J in the same cluster (see Figure 3) is:

$$D(I,S,J) = record\_size(I) + \sum_{J \in M2} [D(J,J) * AVG(I,S,J)]$$

where M2 is the set of member record types to the left of J and clustered together with I and J.

Based on assumption four, the number of physical block

accesses (PBAs) from an occurrence of record type I to its next occurrence is:

$$PBA(I,I) = \min \begin{cases} [D(I,I)/page\_size] & \text{within the same cluster} \\ 1 & \text{across different clusters} \end{cases}$$

where page_size is the effective page size which represents the usable portion of a page (excluding space left at load time and page overhead).

Similary, PBA(I,S,J) can be derived by replacing D(I,I) by D(I,S,J) in the above formula.

The total number of page accesses for a particular clustering scheme is therefore the following formula:

$$\text{cost(clustering scheme)} = \sum_{I \in M3} [F(I,I)*PBA(I,I) + \sum_{J \in M4} F(I,S,J)*PBA(I,S,J)]$$

where M3 is the set of all record types I and M4 is the set of all member record types of each I.

This cost equation can be shown to be equivalent to the following top-down decmposition expression:

$$\text{cost(clustering scheme)} = C(ROOT) = F(I,I)*PBA(I,I) + \sum_{J \in M4} [F(I,S,J)*PBA(I,S,J)+C(J)]$$

where C(ROOT) is the cost for the root of the tree [Schk77].

The cost model for a tree structured database therefore becomes the last cost equation given above. For a network structured database, a number of trees will be produced by partitioning the network and the cost model becomes the sum of each tree's clustering cost, that is,

$$\sum_{i=1}^{\text{no. of trees}} \text{cost i(clustering scheme)}$$

The record clustering problem for a given logical database structure and its usage information can now be formulated as follows:

> **Find a clustering scheme out of the 2**m possible clustering schemes which will minimize the above cost expression.**

3. CLUSTERING GRAPH GENERATION

(Phase One of DRCM)

An unified representation needs to be used for DRCM to be applicable to any data model DBMS. The first phase in DRCM requires that the logical database structure and the usage information be converted into a unified representation called a clustering graph. This clustering graph is an augmented directed graph which consists of:

1) Node_set -- A set of nodes where each node represents a record type. Associated with each node I there are a record_size(I), a N(I), a E(I), and a F(I,I).

2) Link_set -- A set of links where each link represents a possible access path between two record types via a relationship. For every link from record type I to record type J via relationship S -- (I,S,J) -- a F(I,S,J) is associated with it.

Figure 4 is a pictorial representation of a clustering graph.

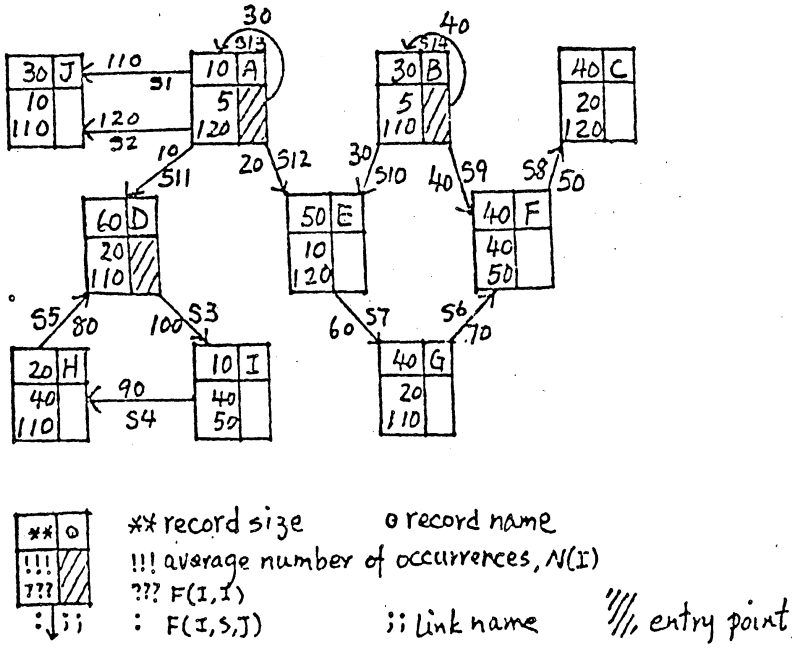


Figure 4 An example clustering graph

3.1 Clustering Graph Generation Algorithm

The generation of a clustering graph for any data model database is a simple process. The following algorithm gives an outline of the process:

ALGORITHM: **Clustering graph generation**

INPUT : 1) logical database structure

: 2) usage information

OUTPUT : a clustering graph

PROCESS : 1) Map each record type(segment or relation) to a node;

2) Map each relationship that can be used as an access path (i.e., $F(I,S,J)>0$) to a link; (In network database, the relationship will be SET type. In IMS, it will be a hierarchical relationship. And in a relational database it will be two relations with matching fields.)

3) Associate the information of $E(I)$, $F(I,I)$, record_size(I) and $N(I)$ to each node;

4) Associate the information of $F(I,S,J)$ to each link;

end;

The time complexity of this algorithm is $O(m+n)$ where m is number of links and n is number of nodes.

3.2 Network and Tree Graphs

Clustering graphs are categorized into two types: tree structured and network structured. The most significant difference between a tree (structured clustering) graph and a network (structured clustering) graph is the types of links that each allows. Links, in general, can be differentiated into five categories (see Figure 5):

1) Loop links -- Those links that join a node to itself.

$$T1 = \{(I,S,I) \mid (I,S,I) \in link\_set\}$$

((A,S1,A) in Figure 5a)

2) Parallel links -- Those links ($\geq$2) which start from the same originating node and terminate at the same ending node.
T2 = {(I,Si,J)|i=1 to k where k$\geq$2}
((A,S1,B) and (A,S2,B) in Figure 5b.
(A,S1,B) and (A,S4,B) in Figure 5c.)

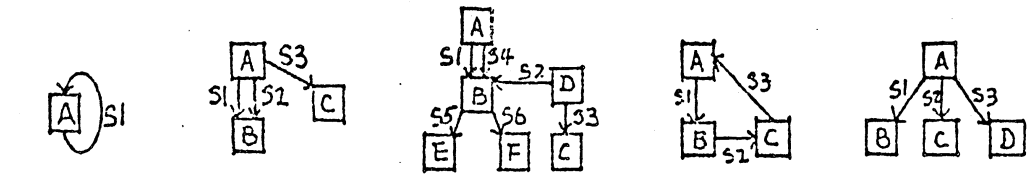
3) End-node sharing links -- Those links ($\geq$2) which start from different originating nodes and terminate at the same ending node.
T3 = {(Ii,Si,J)|i=1 to m where m$\geq$2}
((A,S1,B), (D,S2,B) (A,S4,B) in Figure 5c.)

4) Cycle links -- Those links ($\geq$2) which lie in a cycle in the graph.
T4={(Ii,Si,Ij)|$\exists$(I1,S1,I2),(I2,S2,I3)
,...(Ir,Sr,I1) where r$\geq$2}
((A,S1,B), (B,S2,C) (C,S3,A) in Figure 5d.)

5) Simple links -- Those links which do not have the properties of the above categories.
T5 = link_set - (T1,T2,T3,T4)
((A,S3,C) in Figure 4b. (B,S5,E), (B,S6,F) (D,S3,C) in Figure 5c
(A,S1,B), (A,S2,C) (A,S3,D) in Figure 5e.)

Notes: a) A link can be a combination of cycle, parallel, or end-node sharing links, i.e.,
T2 $\cap$ T3 $\cap$ T4 $\neq \emptyset$.

b) A parallel link can be considered as a special case of the end-node sharing link and a loop link can be defined as a special case of the cycle link. They are divided into separate categories in order to simplify the elimination process in DRCM.



5a loop 5b. parallel 5c. end-node sharing 5d cycle 5e. simple

Figure 5 Categories of links

Tree graphs allow only simple links while network graphs allow all five categories. However, according to assumption three that only hierarchically related nodes can be clustered together, network graphs must be transformed into tree graphs. Such a transformation means that link categories T1, T2, T3 and T4 must be transformed into T5, the simple link category. This transformation is done through elimination. Those links that are eliminated will be called nonessential links and those remaining, essential links.

3.3 Essential and Nonessential Links

To determine which links are essential and which are nonessential, four arguments derived from assumption three are used:

1) Loop links do not apply to record clustering since a node need not be clustered with itself. Therefore all loop links are considered to be nonessential.

2) For n parallel links between two nodes, only one of those links is needed to indicate the potential for clustering those two nodes. The remaining n-1 links are considered to be nonessential.

3) For n end-node sharing links that exist between n different originating nodes and one ending node, the ending node can only be clustered (without duplication) with one of the originating nodes. The link between the originating node and the ending node will be considered to be the essential link and the other n-1 links will be considered nonessential. Which link to choose as the essential link is a search problem and will be discussed in the next section.

4) For n cycle links that form a cycle, one of those links is nonessential. This occurs because if all such links are treated as essential then any one of the node in the cycle could end up being clustered with itself.

4. CLUSTERING GRAPH PARTITION

(Phase Two of DRCM)

The second phase of the DRCM consists of three major components:

1. Nonessential link elimination
2. Network transformation
3. Hierarchical clustering

Nonessential link elimination and network transformation will be dissucced in the next two subsections while hierarchical clustering will be discussed in section 5.

4.1 Nonessential Link Elimination

The nonessential link elimination component is the most complex component of the DRCM second phase. It makes use of the network transformation and hierarchical clustering components to eliminate the nonessential links, category by category. After all nonessential links are eliminated, the resulting structure is a set of tree graphs.

The following algorithm eliminates every nonessential link by categories:

ALGORITHM: **Nonesssential Link Elimination**

INPUT : a clustering graph

OUTPUT : a set of tree graphs

PROCESS : Step 1. if there are loop links then
call loop_link_elimination;

Step 2. if there are parallel links then
call parallel_link_elimination;

Step 3. if there are end-node sharing links then
1) order the links by F(I,S,J) in descending order on a priority queue;
2) call end-node_sharing_link_elimination;

```
Step 4. if there are cycle links then
            1) if step 3 has not been executed
               then order the links by F(I,S,J) in
               descending order on a priority queue;
            2) call cycle_link_elimination;
end;
```

The PROCESS's four steps are ordered so as to eliminate the most obvious nonessential links at the beginning:

Step 1: Elimination of loop links

Since all loop links are nonessential, they are totally eliminated and each associated F(I,S,I) is added to F(I,I) of node I to preserve the intra-record type access information. An outline of the algorithm is :

```
ALGORITHM: Loop_link_elimination
PROCESS : for each loop link;
                F(I,I)=F(I,I)+F(I,S,I);
                delete the loop link;
          end;
```

The complexity of this algorithm is O(p) where p is the number of loop links. For example, the application of this algorithm to the database structure in Figure 4 will eliminate links (A,S13,A),(B,S14,B) and change F(A,A)=150, F(B,B)=150.

Step 2: Elimination of parallel links

Any n-1 of the parallel links can be considered as nonessential and only one should be retained to indicate there exist a potential of clustering the ending node with the originating node. We will therefore retain one link (I,Si,J) in each group of parallel links which has the

highest F(I,Si,J) and change its associated F(I,Si,J) to the sum of all frequencies of owner to member transition in the same group to preserve the inter-record type access information.

ALGORITHM: Parallel_link_elimination

PROCESS : for each group of parallel links;

1) retain one link (I,Si,J) in the group which has the highest F(I,Si,J);

2) $F(I,Si,J)=F(I,Si,J)+\sum_{i\neq j} F(I,Sj,J)$;

3) delete the rest of links in the group;

end;

The complexity of this algorithm is O(r) where r is the number of parallel links. For example, the application of this algorithm to the graph produced by Step 1 will eliminate (A,S1,J), and change F(A,S2,J) to 230.

Step 3: Elimination of end-node sharing links

This step decides which link in an end-node sharing group (i.e., a set of end-node sharing links that shares a common ending node) is essential and should be included in the resulting tree graph, while all others in the same group are nonessential and should be eliminated. Since the objective of the method is to produce minimal cost clustering schemes, the decision of which one is essential is based on whether retaining it and eliminating all others will produce minimal cost.

It is not practical to exhaustively search for optimal

results because all possible choices of essential links are exponential with respect to the number of end-node sharing groups. Let q be the number of end-node sharing groups and ki be the number of end-node sharing links in group i, then the total possible choices of essential links are:

$$\binom{k1}{1} * \binom{k2}{1} * \ . \ . \ . \ * \binom{kq}{1} = k1*k2*\ . \ .\ .*kq \geq 2^{**}q \ (ki \geq 2)$$

Therefore, a "greedy" strategy will be used to sequentially decide on an essential link for each group. The following algorithm outlines such a strategy:

ALGORITHM: End-node_sharing_link_elimination

```
PROCESS  : /* m is no. of links, n is no. of nodes */
for i=1 to q do; /* q is no. of end-node sharing groups */
    for j=1 to  L do;
    /* L ≤ m  is no. of end-node sharing links remaining */
      1) let (I,Sj,J) be in group r; temporarily retain
         (I,Sj,J) and eliminate all others in group r
         from the priority queue;
      2) call the network transformation and hierarchical
         clustering components to calculate the resulting
         cost;
      3) if cost calculated is the smallest or equal to
         the smallest so far, then record this (I,Sj,J)
         and the cost;
    end;
    permanently retain (I,Sj,J) which corresponds to the
    smallest cost and delete all other links belonging
    to the same group from the priority queue;
    L = L - kr;
end;
```

Note: In 2), the algorithm passes the partially decomposed network structure (some non-essential links eliminated) to the network transformation component which will then

complete the transformation to a set of tree graphs. The hierarchical clustering component is then applied to find the total cost. This transformation process will be used again in Step 4 to decide which nonessential link should be eliminated.

Since the time complexity of 2) is O(n*e) (see Section 4.2 and 5), and the number of end-node sharing links L remaining in the inner loop is always less than or equal to the total number of links, the time complexity for this algorithm is O(n*e*m*q), where m is the number of links, n is the number of nodes, e is the number of tree graphs generated and q is the number of end-node sharing groups. For example, the application of this algorithm to the resulting structure produced by Step 2 will produce the following sequence of processes ((I,Si,J) is abbreviated to Si):

q=3

i	j	temporarily retain	temp. eliminate	resulting total PBA
1	1	S5	(S11)	152.0
	2	S6	(S9)	162.0
	3	S9	(S6)	157.2
	4	S10	(S12)	162.0
	5	S11	(S5)	152.0
	6	S12	(S10)	162.0
		permanently retain S12 and eliminate S10;		
2	1	S5	(S11)	142.0
	2	S6	(S9)	152.0
	3	S9	(S6)	147.2
	4	S11	(S5)	152.0
		permanently retain S5 and eliminate S11;		
3	1	S6	(S9)	142.0
	2	S9	(S6)	137.2
		permanently retain S9 and eliminate S6;		

Figure 6 Example data after elimination of end-node sharing links

Step 4: Elimination of cycle link

The elimination of nonessential cycle links may also appear to be of exponential time complexity, but not if the following proposition property is observed:

PROPOSITION:

> If the nonessential links of all the previous link categories are eliminated and if there still remain cycles in the network graph, then each of them can be traversed in linear time (with respect to the number of cycle links in the cycle).

PROOF:

> See [ChTe82].

Based on the property given in the proposition, the elimination of nonessential cycle links is as follows:

```
ALGORITHM: Cycle_link_elimination

PROCESS :

  for each cycle i remaining in the priority queue do;

      for each link (I,Sj,J) in the same cycle i do;
      /* q is no. of links in cycle i */

          1) temporary eliminate (I,Sj,J) in cycle i
             from the priority queue;

          2) call the network transformation and
             hierarchical clustering components to
             calculate the resulting cost;

          3) if cost calculated is the smallest so far,
             record this (I,Sj,J) and the cost;

      end;

      permanently delete the link (I,Sj,J) from the
      priority queue which corresponds to the least cost;

  end;
```

The time complexity of this algorithm is O(m*n*e), where m is the number of links, n is the number of nodes and e is

the number of tree graphs generated. As an example, after applying this algorithm to the graph produced after end-node_sharing_link_elimination, Figure 7 shows the sequence of processes performed and tree graphs produced.

i	j	temporarily eliminate	resulting cost(PBAs)
1	1	S3	1112.4
1	2	S4	1112.4
1	3	S5	137.2

Permanently eliminate S5; total cost=133.9

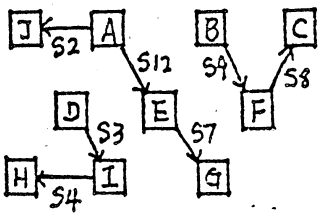


Figure 7 Three tree graphs.

Since this nonessential link elimination component eliminates some links (within a group in Steps 3 and 4) temporarily and leaves the rest of the graph un-partitioned, a second component, the network transformation, is used to partition the rest of the graph in the order of high F(I,S,J)'s in order to arrive at a cost. When a minimun cost is found, the associated nonessential link is permanently deleted. This process repeats until the entire graph is partitioned into a set of tree graphs.

4.2 Network Transformation

There are many ways to transform the network graph into a set of "disjoint" tree graphs. The decompositive approach used in the nonessential link elimination component is a

locally optimal cost searching process. Another approach, using a synthetic view, is simpler and faster but does not always produce or even try to search for an optimal result in terms of cost. However, this component is used only to get a set of disjoint tree graphs from a network graph for temporary purposes, and thus optimality is not very important. This approach views each node in the network graph as an independent tree graph. It then sequentially selects the links from the priority queue one at a time and start forming composite tree graphs based on the following rule:

Given a new link (I,S,J) do one of the following cases:

Case 1. If both the originationg node I and ending node J of the link (I,S,J) are singleton types (i.e., no other link is related to it) form a new tree graph where I is the originating node and J is the ending node (See Figure 8.).

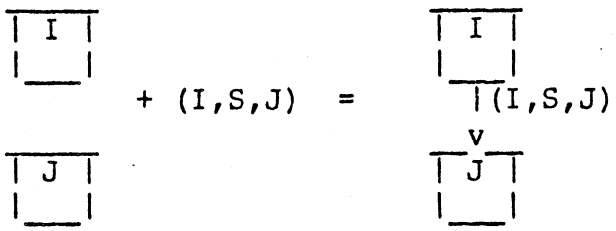


FIGURE 8 Case 1

Case 2. If the originating node I of the new link (I,Si,J) is a singleton type while the ending node J is not, and at the same time J is not an ending node of any other link in the existing graph, then combine the originating node, the link, and the tree graph to form a new tree graph.

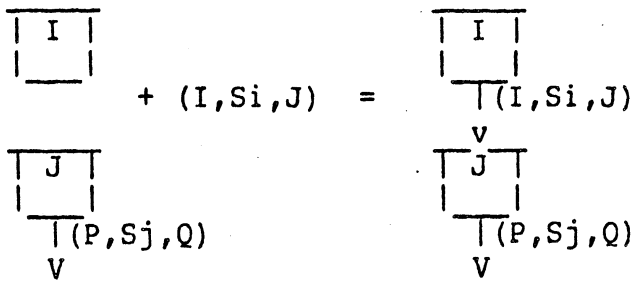


FIGURE 9 Case 2

Case 3. If the originating node I is not a singleton type in the new link (I,Si,J) while the ending node J is, then include (I,Si,J) and the new J as an ending node in the existing tree graph to form a new tree graph.

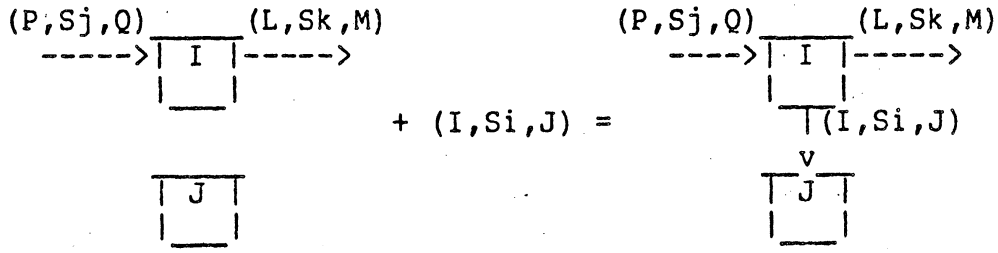


FIGURE 10 Case 3

Case 4. If both the originating node I and ending node J in the new link (I,Si,J) are not singleton types, then if they are in two tree graphs and if the ending node J is not an ending node of any other link in the tree graph it belongs to, combine these two existing tree graphs into one with the new link (I,Si,J) added.

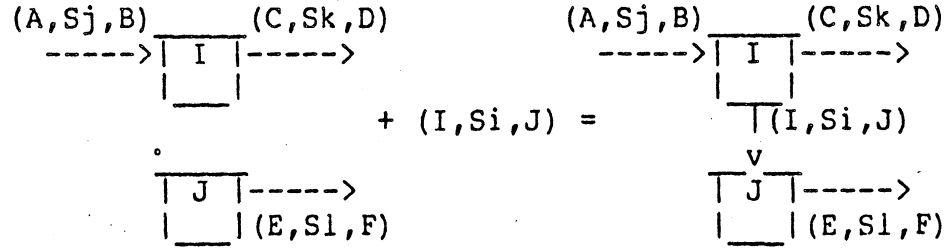


FIGURE 11 Case 4

Case 5. If none of the above cases occur, then the new (I,S,J) is ignored.

This synthetic approach uses the highest F(I,S,J) link whenever possible to synthesize a set of tree graphs. The algorithm for the approach is as follows:

ALGORITHM: Network Transformation

INPUT : 1) node_set
2) priority queue

OUTPUT : A set of tree graphs

PROCESS : For all links do one of the following cases:

case 1;
case 2;
case 3;
case 4;
case 5;

end;

The complexity of this algorithm is O(m) where m is the number links. A proof that it always generates a set of tree graphs from the given input node_set and priority queue is as follows:

PROPOSITION:

The network transformation algorithm always generates tree graphs.

PROOF:

See [ChTe82].

To give an example of the network transformation algorithm, when it is called by the end-node sharing link elimination algorithm (step 3) for i=1 and j=1, the following three tree graphs are produced:

```
A-------->J
 (A,S1,J)

D--------->I--------->H
 (D,S3,I)  (I,S4,H)

B--------->E-------->G-------->F-------->C .
 (B,S10,E)  (E,S7,G) (G,S6,F) (F,S8,C)
```

5. HIERARCHICAL CLUSTERING

(Phase Three of DRCM)

The hierarchical clustering phase serves two functions in DRCM:

1. It is a component of the second phase of DRCM, the clustering graph partition, for calculating the cost of each set of temporarily generated tree graphs. This cost is then the deciding factor of whether a link is essential or nonessential.

2. It is phase three of DRCM and is used for finding a final optimal clustering scheme for each tree graph produced by the clustering graph partition phase.

This hierarchical clustering phase is basically an algorithm to generate a set of candidate clustering schemes and their cost for each tree graph. The minimum cost scheme will then be chosen as the clustering scheme for that tree graph. The following is an outline of the algorithm:

ALGORITHM: Hierarchical Clustering

INPUT : 1) a set of tree graphs

2) page_size

OUTPUT : a set of clustering schemes and a total cost

PROCESS : For each tree graph do;

1) apply Schkolnick's algorithm to produce candidate clustering schemes and costs[Schk77];

2) choose the clustering scheme that has the minimum cost as the resulting scheme;

3) if the root node is not an entry point, then total cost = total cost + MAX
else total cost = total cost + minimum cost

end;

Note that: a. Cost is the cost(clustering scheme) discussed in Section 1.2.

b. MAX is an arbitrarily large number for cost which is used to avoid generating a tree graph with a root node that is not an entry point of the original network graph.

c. The run time of Schkolnick's algorithm is proportional to the number of nodes n. The complete algorithm is given in [Schk77].

The whole algorithm has time complexity O(n*e) where n is number of nodes and e is number of tree graphs. Figure 12 shows the clustering schemes generated by this algorithm for the three tree graphs produced by step 4 of phase two. It assumes page_size = 1000 bytes and MAX=999.

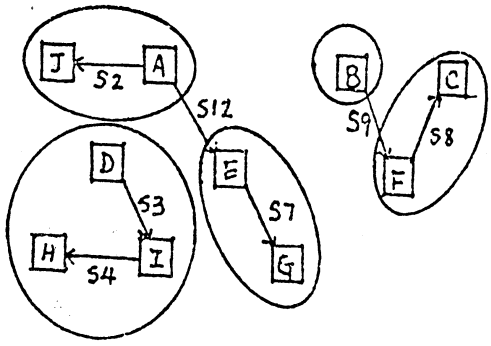


Figure 12 The resulting clustering schemes

This concludes the whole process of DRCM. It can be shown [ChTe82] that DRCM produces the same optimal clustering result as Schkolnick's algorithm for tree structured database in O(m+n) time and near optimal clustering result for network structured database in O(n*e*m*q) time.

6. POSSIBLE EXTENSIONS

A number of extensions to this method can be made. It is possible to incorporate more access transition patterns such as those in [Schk77,Ober79] in the cost model of the DRCM. This will then allow a more accurate modelling of access patterns. Special features of the DBMS (page replacement strategy, multiple page buffer) can also be taken into account in the DRCM to produce a database with better performance. Duplication of physical record types in the clustering schemes perhaps can also add to the efficiency of record access. Another extension that has been mentioned at the begining of this paper is to map the resulting clustering schemes into a linear storage space. To do that, one can treat the produced clustering schemes for each tree graph as a unit and order them on a linear storage space in such a way that the distance of the read/write head movement can be minimized for those frequently used tree graphs [Wong80]. The following is an outline of such an algorithm.

ALGORITHM: **Storage Mapping**
INPUT : a set of clustered tree graphs
OUTPUT : a list of ordered clusters
PROCESS : 1) compute the average access frequency for each clustered tree graphs:
$[\sum_{I,J} (F(I,S,J)+F(I,I))] \; / \; \sum_{I} N(I)$;
2) sort the set of clustered tree graphs by their average access frequencies in descending order;
3) start with the highest accessed clustered tree graph and place the second and third and so forth on the right and left side of the first one alternatively;
end;

The time complexity of this algorithm is O(m+n), where m is the number of links and n is the number of nodes.

REFERENCES

ChTe82 Chiang, W.P. and Teorey, T.J., "A Method for Database Clustering", Technical Report 82 DE 23, Information Systems Research Group, Graduate School of Business Administration, The University of Michigan, 1982.

Hoff75 Hoffer, J.A., "A Clustering Approach to the Generation of Subfiles for the Design of a Computer Data Base", Ph.D. dissertation, Cornell University, Ithaca, 1975.

HoSe75 Hoffer, J.A. and Severance, D.G., "The Use of Cluster Analysis in Physical Data Base Design", Proceedings of the First International Conference on Very Large Data Bases, ACM, N.Y., 1975, p.69-86.

Malm78 Malmquist, J.P., Gudes, E. and Robertson, E.L., "Storage Allocation for Access Path Minimization in Network Structured Data Bases", Proceedings of The International Conference on Databases: Improving Usability and Responsiveness, 1978.

Ober79 Oberlander, L.B., "Physical Design of Database Structure", Ph.D. thesis, Dept. of Comp. and Comm. Sci., The University of Michigan, 1979.

Schk77 Schkolnick, M., "A Clustering Algorithm for Hierarchical Structures", ACM Transactions on Database Systems 2,1 (1977), pp. 27-44.

Schk78 Schkolnick, M., "Physical Database Design Techniques", NYU Symposium on Database Design, NYU, May 1978, pp. 89-110.

TeFr82 Teorey, T.J. and Fry, J.P., Design of Database Structures, Prentice-Hall, Inc., Englewood Cliffs, N.J., chapter 11, 1982.

Wong80 Wong, C.K., "Minimizing Expected Head Movement in One-Dimensional and Two-Dimensional Mass Storage Systems", ACM Comp. Surveys 12,2 (June 1980), pp. 167-178.