TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
PROCEEDINGS OF TALLINN TECHNICHAL UNIVERSITY
(Data Processing, Compiler Writing, Programming)
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

ISSN 0136-3549, 0320-3409 No. 614 1986
pp. 93 – 105 UDC 681.3.06

P. Vyhandu

The author of this article is Leo Võhandu, not P. Võhandu

New Ideas in Data Base Segmentation

Abstract

In this approach a relational database is designed such that the data more often occurring in the same query belongs to one relation. In this way the need for creating access paths is reduced. Bitmatrices allow connecting different search techniques. Bitmatrices can be interpreted as monotonic systems, rows of bitmatrices as bitmaps of objects and as hashing keys.

1. Introduction

The fast search has become a problem in very large data sets. There many methods for organizing effective search [1,2], but situations always arise where a method becomes ineffective for some reason. It is often useful to use different search techniques in different stages of the search process. But we are faced with matching those techniques.

This paper presents an approach, where Bitmatrices use allows connecting different search techniques. Bitmatrices can be interpreted as monotone systems [3], [1] rows of Bitmatrices as bitmaps of objects [6] and as hashing-keys [7].

[1] Monotone Systems where first introduced by Mullat J.E (1971) in "On the Maximum Principle of Some Set Functions," Proceedings of Tallinn Technical University, *Seria A*, No. 313, pp. 37-44, http://www.datalaundering.com/download/modular.pdf , and later described in extended paper, Mullat J. E, (1976,1977) "Extremal Subsystems of Monotonic Systems, I,II,III," *Automation and Remote Control, v.37*, 758-766, *37*, 1286-1294; v.*38*, 89-96, http://www.datalaundering.com/mono/extremal.htm . Notation belongs to JM.

In section 2 a general description of the proposed method will be given. In section 3 definitions of monotonic systems and their kernels will be introduced. In section 4 a more detailed description of database design and search process will be given.

2. Methods for quick search

In relational databases data is maintained in relations, where each relation contains one record type. [2] The record type usually corresponds to a real world concept. In our approach relational database is designed such that the data more often occurring in the same query belongs into one relation. In this way the need for creating access paths, which is a time-consuming process, is reduced.

To decide which data must be maintained in one relation, an analysis of the data set is made first. After forming relations directories are built for them. Records in the relation are grouped to have similar records in one group. On these groups hierarchical structures, called directories, are built. Using directories enables us to determine the groups, which cannot contain records satisfying the given query in the early stage of search.

In such a database a thorough search on a small part of database is implemented and search time becomes much shorter.

Concepts of the monotonic system and its kernel are used for preceding analysis of the data set and clustering. Bitmatrices serve as so-called upper structures on which the main search is performed.

The structure of the database depends completely on data characteristics and it is impossible to predict the number of relations in the database, which depends of groups in a relation and that of records in one group. For that reason the extendible hash method fits well to find the real records. This hash method has properties, which make it very convenient to use when the number of records in the hash table is not known and the hash keys are in the binary form. The hash method is not dealt with here, but its description can be found in [4, 7].

---

[2] *"The central data description construct in this model is a relation, which can be thought as a set of records*", cit. from Database Management Systems, R. Ramakrishnan & J. Gehrke, Sec. Ed., p.10. Notation belongs to JM.

3. Monotonic systems. Kernel

As it was observed in the previous section data ordering must precede the design of the database. Concepts of monotonic system and its kernel splitting algorithms are used for that [3]. First definitions and kernel splitting algorithms are given. In case of data set analysis and record grouping we have to interpret data matrices as different monotonic systems. Therefore different algorithms for the kernel splitting are introduced.

3.1. Definitions

Let us suppose that there is a system W with a finite number of elements. Each element has a numerical measure of its weight (influence) in the system. Let us suppose further that for every element $\alpha \in W$ there is a feasible discrete operation, which changes the weight of α as well as the weights of any other element β of the system. If the elements on W are independent, then it is natural to suppose that the change in the weight of α does not change the value of any element β. System is called monotonic, if the operation of the weight change of any element $\alpha \in W$ brings about changes on the weights levels of other elements only in the direction in which α itself is changed.

To use the method of monotonic systems we have to meet two conditions:

1) There has to be a function π, which gives a measure (weight) $\pi(W)$ of influence for every element W of the monotonic system W. [3]
2) There has to be rules f [4] to recalculate the influence of the elements of the system in case there occurs a change in the weight of one element.

---

[3] This passage appears to be misleading: "*There has to be a function π, which gives a measure (weight) $\pi(\alpha, H)$ of influence for every element $\alpha \in H$ of the subset H of the monotonic system W*". Note belongs to JM.

[4] An example of influence functions f may be found in "A Study of Infraspecific Groups of the Baltic East Coast Autumn Herring by Two Methods Based on Cluster Analysis," Appendix 1, J. Mullat (1974). Note belongs to JM.

A kernel of a monotonic system W is defined as a subset H^* [5] of its elements on which the global minimum of function

$$F(H) = \max_{\alpha \in H} \pi(\alpha, H),\ H^* = \arg\min_{H \subseteq W} F(H)$$

is reached. [6]

3.2. **Kernel splitting (one-dimensional case)**

Let us have a data matrix $A = \|a_{ij}\|;\ i = 1,...,M;\ j = 1,...,N$. Let us interpret matrix A as a monotonic system, the elements of which are the rows a_i of A. Kernel is a subset H^* of its elements on which the global minimum of function

$$F(H) = \max_{a_i \in H} \pi(a_i, H),\ H^* = \arg\min_{H \subseteq W} F(H)$$

is reached. To measure the influence of an object on the system, we define the function

$$S_i = \sum_j \left(2 \cdot n_{ij}^2 + 3 \cdot n_{ij} + 1\right), \qquad (1)$$

where n_{ij} is the frequency of value a_{ij} in the histogram of the j-th attribute.

To split the kernel of the monotonic system we will find sequentially the elements with the greatest influence and add copies of the to the system. It has been made clear that addition of an element to the system alters the influence of all other elements.

Adding an element k to the system changes the influence for any other element i for

$$\begin{aligned} S(n_{ij}+1) - S(n_{ij}) = \sum_j \delta[\ & 2 \cdot (n_{ij}+1)^2 + 3 \cdot (n_{ij}+1) + \\ & + 1 - 2 \cdot n_{ij}^2 - 3 \cdot n_{ij} - 1\ ] = \sum_j (4 \cdot n_{ij} + 5) \end{aligned}, \qquad (2)$$

---

[5] To be more precise the kernel notification should be $\oplus$ kernel H_+^*, or simply H^*. Correction is made without warning everywhere below as needed. Note added by JM.

[6] In Mullat J. E, (1976) "Extremal Subsystems of Monotonic Systems, I," *Automation and Remote Control, v.37*, 758-766, a subsystem, on which F reaches a global minimum, is called a $\oplus$ kernel of the system W.

where $\delta = \begin{cases} 1, \text{if } a_{kj} = a_{ij} \\ 0, \text{if } a_{kj} \neq a_{ij} \end{cases}$, i.e., we sum the frequencies only for those attributes of object i, when the value of the an attribute matches the value of an attribute of the object k. If objects k and i do not have matching attribute values, addition of the element k to the system does not influence the element i.

Algorithm 1 performs the kernel splitting. [7] In this algorithm elements are not actually added to the system, only the influences are computed accordingly. By an added element we mean a labeled element with the greatest influence.

This algorithm 1 is constructing the so called defining sequence, following J.Mullat

Algorithm 1.

Step 1. For each element a_i, $i = 1,...,M$ compute

$$S_i = \sum_j \left(2 \cdot n_{ij}^2 + 3 \cdot n_{ij} + 1\right).$$

Step 2. Find $k = \arg\max_{i=\overline{1,M}} S_i$, memorize k, S_k.

Step 3. For each element compute $S_i = S_i + P_i$, where $P_i = 4 \cdot T + 5 \cdot L$, and T is the sum of the frequencies of the matching attribute values of elements k and i, and L is the number of matches.

Step 4. Find $\max S_i$ for all elements not added to the system.

Step 5. If $\max S_i < S_k$ then end the algorithm. [8]

Step 6. Memorize $k = \arg\max_{i=\overline{1,M}} S_i$, S_k.

Step 7. In the auxiliary table of frequencies for every value for the element k add 1.

Step 8. Go to step 3.

---

[7] Actually the algorithm does not split the $\oplus$ kernel but it finds somewhat estimate for $\ominus$ kernel.

[8] Here is the major mistake why the algorithm does not split the kernel. The algorithm must continue once again, this time from step 2, memorizing the lower $\max S_i$ value as usual and continuing with Step 3 over again, as usual. The algorithm ends when all elements in table A are processed (added) – not just one pass through step 5. Only the last pass through the step 5 splits the $\oplus$ kernel from the table. Fortunately, for the author, it might be quite probable that an estimate for $\ominus$ kernel from the A table splits away by this algorithm, see the duality theorem in J. Mullat,"Extremal Subsystems of Monotonic Systems, II," *Automation and Remote Control*, v.37, 1286-1294. On the conceptual level the $\ominus$ kernel estimate spitted by current algorithm, fits well and serves the purposes of current investigation. Unfortunately no formal theoretical properties of the estimate been investigated. Notice belongs to JM.

On step 5 reaching the extremum of the function π is controlled. All the elements memorized before reaching the extremum of function π, belong to the kernel of the monotonic system. [9]

3.3. Kernel splitting (two-dimensional case)

Let us have a data matrix $A = \|a_{ij}\|$, $i = 1,...,M$, $j = 1,...,N$. Let us interpret matrix A as a monotonic system, the elements of which are rows a_i and columns a_j, $a_i, a_j \in A$. The kernel is a subset H^* of its elements on which the global minimum of function

$$F(H) = \max_{\substack{a_i \in H \\ a_j \in H}} \pi(a_i, a_j, H), \; H^* = \arg\min_{H \subseteq W} F(H),$$

is reached.

Let us have for every element a_i a number of its incidences n_i. Then the number of incidences having the j-th attribute is equal to

$$h_j = \sum_i n_i \cdot \delta, \tag{3}$$

where $\delta = \begin{cases} 1, \text{if } a_{ij} = 1 \\ 0, \text{if } a_{ij} = 0 \end{cases}$.

Initial influences can be computed as follows

$$g_i = n_i \cdot \sum_j \left(2 \cdot h_j^2 + 3 \cdot h_j + 1\right) \cdot \delta \tag{4}$$

for the element a_i and

$$g_j = h_j \cdot \left(2 \cdot h_j^2 + 3 \cdot h_j + 1\right) \tag{5}$$

for the elements a_j.

---

[9] Obviously the author means that the local extremum has been controlled, and all elements memorized constitute ⊖ kernel estimate. In the next section the same type of algorithm performs, but this time upon two-dimensional table. All just said about the one-dimensional algorithm is valid in two-dimensional case. We are not going to make similar notation in this direction any more, JM.

To split the kernel we will add the elements with the greatest influence to the system. Adding an element a_j to the system, the influence of every other element a_r, $r = 1,...,M$ grows by

$$d_r = n_r \cdot (2 \cdot h_j^2 + 3 \cdot h_j + 1) \cdot \delta . \tag{6}$$

Adding an element a_i to the system, influence of elements a_j can be computed by a formula

$$g_j = \kappa \cdot (2 \cdot k^2 + 3 \cdot k + 1) \cdot \delta ,$$

where

$$k = h_j + n_i \tag{7}$$

and the influence of the element a_r, $r = 1,...M$ grows by

$$\begin{aligned} \Delta r &= [\ (2 \cdot k^2 + 3 \cdot k + 1) - (2 \cdot h_j^2 + 3 \cdot h_j + 1)\ ] \cdot n_r = \\ &= n_r \cdot [\ 2 \cdot (h_j + n_i)^2 + 3 \cdot (h_j + n_i) + 1 - 2 \cdot h_j^2 - 3 \cdot h_j - 1\ ] = \\ &= n_r \cdot (4 \cdot h_j \cdot n_i + 2 \cdot n_i^2 - 3 \cdot n_i) = n_r \cdot n_i \cdot [\ 2 \cdot (n_i + 2 \cdot h_j) - 3\ ]. \end{aligned} \tag{8}$$

To add elements to the monotonic system up to the kernel splitting, we use:

Algorithm 2. **Once again, the algorithm 2 is constructing the so called defining sequence, following J.Mullat**

Step 1. For each j, $j = 1,...,N$ compute the number of incidences h_j by formula (3).

Step 2. By formula (4) and (5) compute initial influences g_i and g_j.

Step 3. Find the element with the greatest influence, i.e., $p = \max\limits_{g \in g_i \cup g_j} g$.

Step 4. Memorize $k = \begin{cases} i, \text{if } g \in g_i \\ j, \text{if } g \in g_j \end{cases}$ and $S_k = p$.

Step 5. If $\max g \in g_i$, go to step 8.

Step 6. For each r, $r = 1,...,M$ compute d_r by formula (6) and add it to the influence g_r.

Step 7. Go to step 10.

Step 8. For each j, $j = 1,...,M$ compute influences g_j by formula (7).

Step 9. For each r, $r = 1,...,M$ compute $g_r = g_r + \Delta r$, where Δr is computed by formula (8).

Step 10. Find $p = \max\limits_{g \in g_i \cup g_j} g$. If $S_k < p$ go to step 4.

Step 11. End of algorithm.

4. Database design

Database design is divided into two stages: analysis of the data set for partitioning data items into relations and build-up of directories.

4.1. **Data set analysis**

For a relational database we have the following definitions [5].

Attributes are identifiers taken from a finite set $A_1, A_2, ..., A_n$. Each attribute A_i is associated with its domain, denoted by $DOM(A_i)$, which is a set of possible values for that attribute. We use the letter $A, B, ...$ for single attribute and the letter $X, Y, ...$ for sets of attributes.

A relation on the set of attributes $\{A_1, A_2, ..., A_n\}$ is a subset of the Cartesian product $DOM(A_1) \times DOM(A_2) \times ... \times DOM(A_n)$. The elements of the relation are called tuples. A relation R on $\{A_1, A_2, ..., A_n\}$ is denoted by $R(A_1, A_2, ..., A_n)$.

Relational algebra as a data manipulation language is introduced. There are two basic operations of interest for us: projection and natural join.

The projection of a relation $R(X, Y, Z)$ over the attributes in X will be denoted $R[X]$, and defined by $R[X] = \{x \mid \exists y \exists z : (x, y, z) \in R\}$.

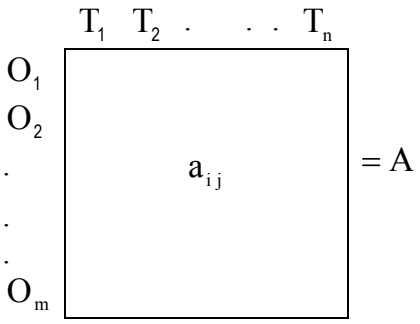
The natural join operation is used to make a connection between attributes that appear in different relations. Let $R(x, y)$ and $S(x, z)$ be two relations; then the natural join $R * S$ is a relation defined over the attributes $\{X, Y, Z\}$.

Let $R(X, Y, Z)$ be a relation; we shall say that R is decomposable if there exist two relations S and T, such as:

a) S and T are the projections of R: $S = R[x, y]$, $T = R[y, z]$.

b) the natural join of S and T is R: $R = S * T$.

Using the natural join operation all the relations in the database can be put into one relation U. A model matrix A can be put into correspondence with the relation U



where $\Omega = \{ O_i \}$, $i = 1,...,n$ is the set of objects types, $\tau = \{ T_j \}$, $j = 1,...,m$ is the set of attributes and $a_{ij} = \begin{cases} 1, \text{if an object type } O_i \text{ has an attribute } T_j, \\ 0, \text{otherwise.} \end{cases}$

For each object type the number of its incidences is also given.

Such a model can be interpreted as a monotonic system described in section 3.3. Using Algorithm 2 all kernels of the system are separated.

Let as suppose that the number of separated kernels is equal to p. The kernel is denoted by K_s, $s = 1,...,p$. Using the projection operation, relation U can be decomposed in the following way:

a) $R_1 = U[X_1]$, $R_2 = U[X_2]$,..., $R_n = U[X_n]$,

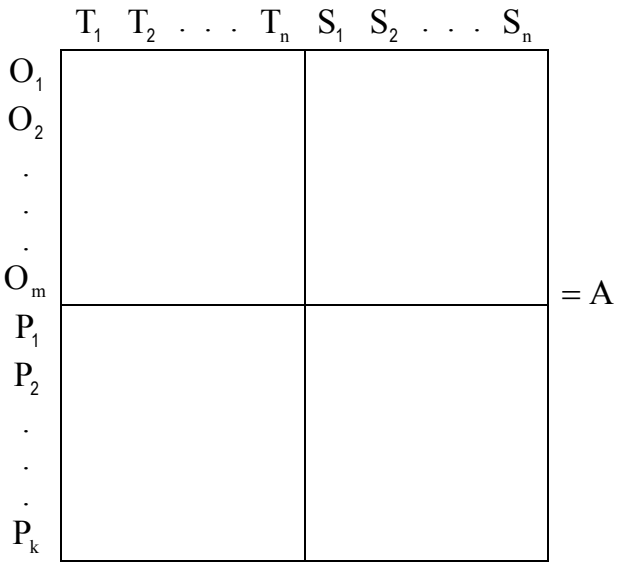
b) $U = R_1 * R_2 * ... * R_p$,

where $X_S = \{ a_j \}$, $a_j \in K_s$ and tuples of relation R_i are incidences of the object types corresponding to the elements $a_i \in K_S$.

The relations created contain similar items.

But it might to be more effective, if we could store data that occurs in the same query in one relation. Then the need to use operations of relational algebra is reduced and therefore search time decreases.

To achieve that, queries and connection determined by the queries must be considered on the model matrix A in addition to object types and attributes. The model matrix A is presented in the following form:



where $\Omega = \{ O_i \}$, $i = 1,...,m$ denotes the set of objects types,

$\tau = \{ T_j \}$, $j = 1,...,n$ denotes the set of attributes,

$\pi = \{ P_i \}$, $i = 1,...,k$ denotes the set of queries,

$\Sigma = \{ S_j \}$, $j = 1,...,\ell$ denotes the set of connections,

and $$a_{ij} = \begin{cases} 1, \text{ if an object type } O_i \text{ has an attribute } T_j, \\ 0, \text{ otherwise, } i = 1,...,m,\ j = 1,...,n; \end{cases}$$

$$a_{ij} = \begin{cases} 1, \text{ if a query } P_i \text{ contains an attribute } T_j, \\ 0, \text{ otherwise, } i = m+1,...,m+k,\ j = n+1,...,n+k; \end{cases}$$

$$a_{ij} = \begin{cases} 1, \text{ if a query } P_i \text{ determines a connection } S_j, \\ 0, \text{ otherwise, } i = m+1,...,m+k,\ j = n+1,...,n+k; \end{cases}$$

$$a_{ij} = \begin{cases} 1, \text{ if a connection } S_j \text{ contains an object type } O_i, \\ 0, \text{ otherwise, } i = 1,...,m,\ j = n+1,...,n+\ell. \end{cases}$$

For each object type the number of its instances and for each query the frequency of occurrence is also given. Analogously to the preceding model, matrix A is interpreted as a monotonic system, all the kernels are separated and new relations are created. Record types in relations are structures where data items are not only similar to each other, but are closely connected through occurrence in the same queries.

4.2. **Bitmaps and directories**.

Let us have in one relation m records and n attributes, where j-th attribute $(j = 1,...,n)$ has N_j different values. Then the bitvector of length $d = \sum_{j=1}^{n} N_j$ can be created for each record.

The elements of the bitvector are filled as follows:

$$a_j^k = \begin{cases} 1, \text{if record has value } k, \\ 0, \text{otherwise}, j = 1,...,N_j. \end{cases}$$

This bitvector is called bitmap. [10] In [6] the approach of directories built of the bitmaps is given.

Let us interpret the set of bitmaps as a monotonic system. Using Algorithm 1 from section 3.2 all kernels are separated. Using disjunction operator one superbitmap is formed of bitmaps belonging to one kernel. This superbitmap is called an address.

Out of the formed addresses a new monotonic system can be made and the process can be repeated. Using that process recursively a hierarchical structure, called directory, is formed.

[10] R. Ramakrishnan & J. Gehrke, call the exact notion by bitmaps indexes in their Database Management Systems monograph, Sec. Ed., – "*First, they allow the use of efficient bit operations to answer the queries…Second, bitmap indexes can be much more compact than the traditional* $B+$ *tree index and are very amenable to use of compression techniques*." p.691.

4.3. Search process

If the database is created using the methods described in this paper, a relation and its directory will be in the form shown in Figure 1.

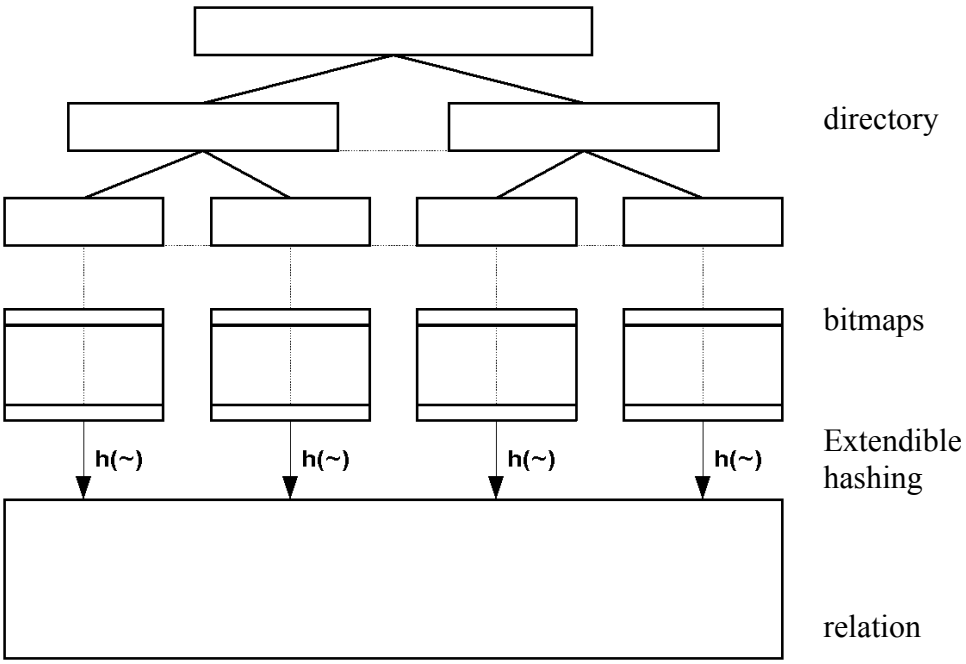


Figure 1

The database consists of relations. A set of bitmaps corresponds to each relation on which a directory is built. Records in relations are connected with their bitmaps via extendible hashing. Bitmaps and directories are called upper structures of the database.

The search process runs as follows.

According to a given query we have first to determine in which relations the needed data is stored. After that directories of these relations are examined. If we found out that the group the given address is representing cannot contain the needed data, we do not investigate it any further. A sequential search is performed on the set of bitmaps, which may meet query conditions. Using extendible hashing the needed records are quickly located.

The search time is reduced because

(1) of a great probability that the needed data is stored in one relation. The need for relational algebra operations is decreased.

(2) even in the realm of one relation the sieving process cuts off the number of objects on which the full search is performed.

References

1. Martin J., Computer database organization, Prentice-Hall, 1977.
2. Salton G., Dynamic information and library processing, Prentice-Hall, 1975.
3. Mullat J., Vyhandu L., "Monotonic systems in scene analysis", Symposium, *Mathematical Processing of Cartographic Data*, Tallinn, 1979, pp.63-66.
4. Fagin R., Nievergelt J., Pippenger N., Strong H.R., "Extendible hashing, a fast access method for dynamic files," ACM TODS, 1979, Vol. 4, No. 3, pp.315-344.
5. Dolobel C,. Pichat E., "The design of relational information system according to different kinds of dependencies," *COMPSAC 78, Proc. IEEE*, Chicago, 1978.
6. Выханду Л.К., Выханду П.Л., Быстрый поиск на битматрицах, Тр. Таллинск. Политех. Ин-та, 1983, № 554, с. 49.60.
7. Выханду Л.К., Выханду П.Л., Синтез метода адресных книг и расширяющегося хэширования, Тр. Таллинск. Политех. Ин-та, 1984, № 568.