On the Automation of Physical Database Design

Sunil Choenni* Henk M. Blanken* Thiel Chang**

University of Twente*
Administration Office:GAK**

Abstract

The design of an *optimal* physical database entails an exponential time complexity. This is the main reason which has shifted the attention of researchers to the problem of determining a *good* physical database design. Two kinds of approaches have been proposed to find a good physical design, an optimizer and a knowledge based approach. An optimizer based approach is characterized by generating physical designs on basis of extracted information from the optimizer. The knowledge based approach is characterized by representing the knowledge of experts, not necessarily knowledge with regard to optimizers, into rules. In this paper we will propose a framework of a tool based on a combination of both approaches which can assist in finding a good physical design. In our approach we deal in a fundamental way with knowledge based aspects such as how to represent knowledge into rules and how to combine different rules. Knowledge based approaches for physical design take ad hoc decisions for these subjects. We believe that these kinds of decisions are essential for a knowledge based approach because the generated solutions are determined by the rules and combination of rules. We will present a technique how to model knowledge in a more proper way into rules. This makes it possible to combine rules in a founded way with the Dempster-Shafer theory, finally leading to a good proposal for physical design.

Keywords: Physical database design, Generating storage schemes, Dempster-Shafer theory, Modelling rules of thumb.

1 Introduction

The design of databases is a difficult and time consuming process. The design process takes place on several levels. One of these levels is the physical level. A typical task on this level is to determine the placement and access structures for relations resulting in a storage scheme. A *storage scheme* contains the *storage configurations* of all relations which are involved in the database. Comer [Come 78] demonstrated that the subproblem of secondary index selection, in determining an optimal storage scheme, is NP-complete. In [ChBC 92a] it is proved that even if the secondary index selection problem will not be relevant, the problem of determining an optimal storage scheme still remains NP-complete.

Because of the exponential complexity which is involved in determining an optimal storage scheme[1], the field of research is shifted to the problem of determining a good storage scheme. A storage scheme is considered as good if a competent human database designer would produce the same or a worse storage scheme with the same available information. Finkelstein et al. [FiST 88] justify the shift also with the argument that the problem specification and the problem that the designer actually wants to solve are usually not the same. The problem specification is in general to handle the workload on a database with minimal costs. To realize this task the designer has to gather informations such as the frequencies of the statements of the workload, statistics of the stored data, etc. The gathered informations are in general approximations of the real values. So, solving the problem with the approximated values will result into an optimal storage scheme belonging to the approximated values and not one to the real values.

Roughly we can distinguish two approaches in selecting a good storage scheme automatically, the optimizer based approach [FiST 88] and the knowledge based approach [RoSh 91,DaJe 88].

The optimizer based approach is characterized by the selection of storage schemes on basis of information extracted from the optimizer. A great advantage of such an approach is that the proposed storage schemes will be used by the optimizer to its full advantage. An-

[1] An optimal storage scheme is the storage scheme which has the lowest cost in processing the workload defined on the database.

ACM-SAC '93/2/93/IN, USA
© 1993 ACM 0-89791-568-2/93/0002/0358...$1.50

other advantage of a tool based on this approach is the independency between tool and optimizer. If the optimizer changes the tool is adapted 'automatically'. One fundamental problem of an optimizer based approach is the entailed exponential complexity. Finkelstein et al. [FiST 88] apply this approach to the optimizer of System R. They developed a tool DBDSGN which aims to select a good set of indices for each relation. Their tool could avoid the exponential complexity problem because they adopt the principles of System R. One of the principles of System R which reduces the exponential complexity considerably is that System R uses at most one index in processing a SELECT FROM WHERE part in a query. In general the optimizer based approach can be applied for relative simple optimizers whose behaviour are predictable. If this is not the case then all possible storage schemes have to be passed on the optimizer.

The knowledge based approach is characterized by representing the knowledge of experts in rules. These rules are then sequentially applied to generate a good storage scheme. A great advantage of such an approach is that it can help to narrow the number of alternative storage schemes which has to be explored in finding a good one. The reason for this is that experts can recognize quickly on the basis of experience and intuition that some storage schemes will be bad while others are eligible as good. However, tools based on such an approach have the following more or less serious drawbacks.

- The generated solutions of the tool are dependent on the knowledge of experts. The better the experts will be, the better the knowledge based system will be.
- The tool must be kept up to date. This means that if experts discover new rules of thumb this must be processed because otherwise experts can generate better storage schemes or generate them more quickly which would make the tool redundant.
- We are not sure whether the proposed storage schemes will be used to its full advantage by the optimizer.

Knowledge based approaches can be found in [RoSh 91, DaJe 88]. The approach discussed in [DaJe 88] takes as input an entity-relationship scheme and a workload. Then a set of rules is applied to the input to generate a number of candidate representations for relationships between entity types. These candidate representations are then further explored by applying another set of rules and more algorithmic processing. The presented approach in [RoSh 91] starts with a more general input. It takes as input a logical database scheme and the workload defined on the scheme. The approach consists of two steps. The first step is based on the intuition that database designers can produce quickly a number of advantageous candidate storage organizations for one single query on basis of heuristics. The second step aims to produce an intermediate between good storage organizations of the individual queries.

It is clear that the rough idea of the first step of the two knowledge based approaches [RoSh 91,DaJe 88] is comparable and we believe that it is a good one. However, the problem is how to model the rules of thumb used by database designers into knowledge rules and how to deal with them. Rules of thumb have always an uncertain character. For example, a database designer can use a heuristic which says that (under some conditions) a certain subset of the available set of storage structures will be good in 80% of all cases. This heuristic says nothing about the remaining 20% which implies ignorance in these cases. So, an adequate modelling of the first step demands to take uncertainty and ignorance into account. However, the approach in [RoSh 91] takes no uncertainty into account while the approach in [DaJe 88] deals with a limited form of uncertainty. In [DaJe 88] a certainty factor (abbreviated as CF) is associated with each rule which lies between [-1,1]. The certainty factor is a numeric measure of the degree to which a fact is believed to be true (or false). If two certainty factors CF_1 and CF_2 are both positive (or negative) and are associated with different rules concluding the same fact, then they are combined using the formula of Bernoulli [GrSt 82] resulting in the new certainty factor $CF_1 + CF_2 * (1 - CF_1)$. If two rules are conflicting then the certainty factor for the consequent (then part) is achieved by subtracting the certainty factors of the involved rules. Besides the fact that the intuition and a formal foundation is missing of this step, this approach is not able to deal with ignorance.

In this paper we adopt the idea of Finkelstein et al. [FiST 88] that the generation of good storage schemes should be made on the basis of information achieved by the optimizer and we also adopt the idea of the authors of the knowledge based approaches that knowledge rules can narrow the search space in finding good storage schemes. As a consequence we will present in Section 2 the architecture of a tool which is based on the integration of the optimizer and knowledge based approach. In an integrated approach the emphasis lies on knowledge aspects such as how to deal with uncertainty, ignorance and the combination of rules. We will present techniques which are able to deal with these issues. These techniques are applications of the Dempster-Shafer theory [Demp 67,Demp 68], which has been mathematically founded by Shafer [Shaf 76]. Despite of the fact that some practical and theoretical objections can be made on this theory [Choe 90,Zade 84] we think that the theory is of practical significance for our purpose. More recently it has been shown in [HaFa 92] that some critiques on the Dempster-Shafer theory stems from a confounding of two different views of the theory. For

the computational aspects of the theory we refer to [Barn 81,Choe 90,ShLo 87].

In Section 3 we will present the form that rules of thumb have as well as how they can be modelled with the Dempster-Shafer theory. The forms are achieved by analysing about 60 rules of thumb that are used by experts at the GAK for designing storage schemes. These rules of thumb can be found in [Walr 90]. Section 4 considers the combination of rules. Some remarks conclude the paper.

2 The Integrated approach

From the introduction it should be clear that the disadvantage of the optimizer based approach (the exponential complexity) is not present in the knowledge based approach and that the last disadvantage of the knowledge based approach (if the proposed storage schemes will be used by the optimizer to its full advantage) is not present in the optimizer based approach. This is the motivation to propose a tool based on a combination of these approaches called the integrated approach. We believe that a tool based on the integrated approach will result in a more powerful tool than one based on pure a knowledge based or an optimizer based approach.

The aim of the tool is to produce proposals for storage schemes given a workload, a logical scheme, the permitted storage structures, possibly other database characteristics such as the cardinality of the involved relations, number of available pages etc, and possibly requirements put by the user. For example, the user can decide to put an index on certain attributes which has to be adopted by the tool. The architecture of the tool is given in Figure 1.

Let us fetch a glance how the tool can be related to its users. After the tool received the cost estimations involved in processing the workload of the candidate storage schemes from the optimizer, it chooses the one with the lowest cost and presents it to the user. The user may possibly change some storage configurations and ask the tool for a cost estimation of the modified storage scheme. In that case the tool will check whether the modified storage scheme has already been evaluated. If this is the case the information will be given to the user otherwise the scheme will be sent to the optimizer. Note, there is also a possibility that the user can change the initial input parameters. This can be useful if the user has made bad decisions with regard to the requirements.

The function of the knowledge system is to produce proposals for storage schemes. When and how storage schemes have to be produced is controlled by the tool. The tool decides which storage schemes proposed by the knowledge system will be sent to the optimizer for an estimation of the involved cost in processing the workload. The tool has also as task to control the invocations of rules and rule groups.

The knowledge system consists of a rule base and a knowledge base management system. The knowledge base management system is responsible for the execution of rules in a proper way. The rule base consists roughly of three rule groups:

1. A group containing optimizer dependent information which will be brought in by human experts or gathered by rule group 3 of the knowledge system. For example, several formulas can be used by an optimizer for the estimation of the number of page accesses involved in retrieving T tuples [ChBC 92b]. The information that optimizer O uses formula F to estimate the number of page accesses belongs typically to this group.
2. A group containing optimizer independent information which will be brought in by human experts. For example, in the selection process of secondary indices the complexity reducing measures, which can be taken, depend on the retrieving formula which is used as illustrated in [ChBC 92b]. The information which complexity reducing measures belongs to which retrieving formula will be stored in this group.
3. A group which is able to gather specific information about optimizers. If this group succeeds in deriving relevant information this will be stored in group 1. This group contains for example information how to decide which formula is used in estimating the number of page accesses involved in retrieving T tuples. How this specific information can be achieved will be demonstrated now.
   Let us consider two well-known formulas proposed by Cardenas [Card 75] and Bernstein [BGWRR 81]. The formula of Cardenas looks as follows: $NPA = p(1-(1-1/p)^T)$, in which NPA is the number of page accesses, p is the number of pages required to store a relation and T is the number of tuples which has to be retrieved.
   The formula of Bernstein looks as follows:

$$NPA = \begin{cases} T & \text{for } T \leq \frac{1}{2}p \\ \frac{1}{3}(T+p) & \text{for } \frac{1}{2}p < T \leq 2p \\ p & \text{for } 2p < T \end{cases}$$

   To determine which formula an optimizer uses can be done for example by formulating a query such that the number of tuples which has to be retrieved is about $2p$ and to offer the optimizer the query to make an estimation about the involved number of pages. If the optimizer estimates that the involved number of pages is about p then the optimizer used probably the formula of Bernstein and if it will be

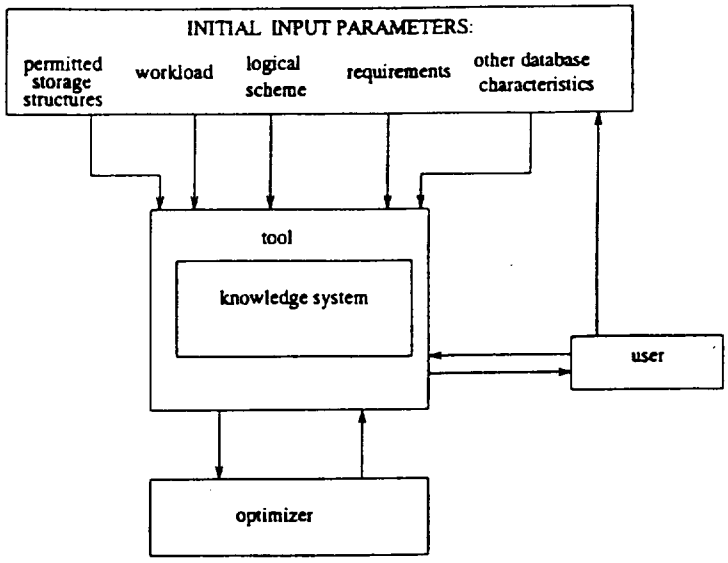

Figure 1: Architecture of a tool for physical database design

about $p(1-(1-p)^{2p})$ then probably the formula of Cardenas is used. Once these kinds of information are derived then they will be stored in rule group 1.

The last rule group is a special one and will be especially used when an optimizer is connected with the tool or the model of the optimizer will be changed. Suppose the tool will be coupled to an optimizer O. If the tool was coupled before to optimizer O then some information will be stored in rule group 1. If this is not the case the tool should invoke the rules of rule group 3 to collect relevant information such that it will be possible to use the information of rule group 2. As already noticed, in the selection process of secondary indices the complexity reducing measures, which can be taken, depend on the retrieving formula which is used. Suppose, for example, that rule group 2 contains the information, which complexity reducing measures have to be taken for each formula. If the tool wants to exploit this information it should find out which retrieving formula is used by the optimizer. For this purpose rule group 3 can be invoked by the tool.

We will now illustrate the interaction between the tool and the knowledge system in more detail. Suppose that the tool invokes all rules of group 1 with the highest 'belief' (belief is a way to model uncertainty and will be discussed in section 3) for a given set of input parameters. Then the knowledge system will give a set of candidate storage schemes. The tool may decide to invoke other rules on basis of the received candidate storage schemes because the time required to let the optimizer make an estimation of the cost of the storage schemes may be too large.

The advantage of the proposed tool is that we are sure that proposed storage schemes will be used to its full advantage and that we have to explore only a restricted number of storage schemes. But, still two drawbacks of the knowledge base remain, the tool must be kept up to date and the generated solutions are dependent on the experts. We can partly take these two drawbacks away by putting a functional requirement on the tool that it has to be extensible, so that it will be easy to add new rules to the tool or to improve existing rules.

The rules in the groups contain a certain *belief* and are ranked on basis of these beliefs. This belief is a way to model the uncertain character of rules of thumb which are used by human designers for physical design. Rules which contain facts will be given full belief. The rules and combinations of rules produce candidate storage schemes and gather information about optimizers. If they will propose bad storage schemes and irrelevant information, the tool will make bad choices. Thus the rules of thumb have to be modelled carefully into rules and the combination of rules is also essential.

3 Modelling rules of thumb into knowledge rules

In this section we shall take a closer look at the heuristics which are used by designers of physical databases as well as how to model them into knowledge rules. We believe that in our approach (and also in pure knowledge based approaches) an adequate modelling of heuristics is essential because the generated storage schemes are determined by these rules.

We start this section by taking a look at the heuristics

Heuristic 1:
IF physical tables are small (< 15pages)
THEN
choose an ISAM storage structure above
a B-tree storage structure

Heuristic 2:
IF more than 10% of the workload change the value of an attribute
THEN
this attribute is not an index candidate

Figure 2: Examples of some heuristics as used at the GAK

presented in [Walr 90]. Then we will define techniques to represent these heuristics into rules.

Walraven [Walr 90] has gathered about 60 rules of thumb which are used by physical design experts at the GAK office for the design of Ingres databases. After a rough inspection of these heuristics we can make the following two observations:

- The experts have apparently no difficulties to translate qualitative notions into quantitative measures because in most heuristics where a qualitative notion is used, a quantification is given between brackets. An example is given by heuristic 1 of Figure 2.
- Two types of heuristics can be discerned. The first type can be characterized in the following way:

  IF condition THEN conclusion(s) (1)

  For an example of this type of heuristic we refer again to heuristic 1 of Figure 2.
  Let $C(\alpha)$ be the extent (in %) to which a condition has been satisfied. Then, the second type heuristic can be characterized as:

  IF $C(\alpha) \geq (<)x\%$ THEN conclusion(s) (2)

  In this heuristic x represents a constant value. For an example of this type of heuristic we refer to heuristic 2 of Figure 2. The value of x in this heuristic is 10.

It is clear that the first observation makes the modelling of the heuristics into knowledge rules easier because in general the representation of qualitative notions into quantitative measures is a tough problem.

We will now concentrate on how to model the two types of heuristics into knowledge rules. The first type of heuristic (1) has almost the form of a knowledge rule only the belief committed to the heuristic is missing.

Before we propose how to commit belief to a heuristic we shall first introduce some definitions originating from the Dempster-Shafer theory [Shaf 76].

Definition 3.1: Let SC be a hypothetical set consisting of all permitted storage configurations for the relations and let $I\!P(SC)$ be the power set of SC, then a function $m : I\!P(SC) \rightarrow [0,1]$ is called a *basic probability assignment (bpa)* whenever

$$m(\emptyset) = 0 \quad (3)$$

$$\sum_{S \subseteq SC} m(S) = 1 \quad (4)$$

The quantity $m(S)$ is called S's basic probability *number* and it is understood to be the measure of the belief that is committed exactly to the set of storage configuration S. The total belief in S ($Bel(S)$) is the sum of the basic probability numbers of all subsets SS of S. The following definition describes the relation between belief and basic probability assignment in a formal way.

Definition 3.2: A function Bel is called a *belief function* over SC if it is given by (5) for some bpa $m : I\!P(SC) \rightarrow [0,1]$.

$$Bel(S) = \sum_{SS \subseteq S} m(SS) \quad (5)$$

Note, a basic probability assignment induces a belief function and conversely.

Example 3.1 Suppose $SC = \{S_1, S_2, ..., S_n\}$ and we have a rule which provides the following bpa 's: $m(\{S_1, S_2, S_3\}) = 0.5$, $m(\{S_2, S_3\}) = 0.4$ and $m(SC) = 0.1$. Then the following belief function can be constructed: $Bel(\{S_2, S_3\}) = 0.4$, $Bel(\{S_1, S_2, S_3\}) = 0.9$ and $Bel(SC) = 1.0$ □

In the following two definitions we will introduce the notions of plausibility and ignorance.

Definition 3.3: The plausibility of a set of storage configuration S, $Pl(S)$, is defined as:

$$Pl(S) = 1 - Bel(\neg S) \quad (6)$$

Definition 3.4: The degree of ignorance with regard to a set of storage configurations S, $Ig(S)$, is:

$$Ig(S) = Pl(S) - Bel(S) \quad (7)$$

Example 3.2 Consider the following bpa defined over $SC = \{S_1, S_2, S_3\}$.
$m(\{S_1\}) = 0.2$, $m(\{S_2\}) = 0.2$, $m(\{S_2, S_3\}) = 0.4$ and $m(SC) = 0.2$. Then the corresponding belief function is: $Bel(\{S_1\}) = 0.2$, $Bel(\{S_2\}) =$

0.2, $Bel(\{S_2, S_3\}) = 0.6$ and $Bel(SC) = 1.0$.
To compute the plausibility of a set of storage configuration S we will make use of the following formula:

$$Pl(S) = \sum_{S \cap S_j \neq \emptyset, S_j \subseteq SC} m(S_j)$$

The derivation of this formula from definition 3.3 is straightforward.
Then, $Pl(\{S_1\}) = 0.4$, $Pl(\{S_2\}) = 0.8$, $Pl(\{S_2, S_3\}) = 0.8$ and $Pl(SC) = 1.0$ and the corresponding degrees of ignorance are: $Ig(\{S_1\}) = 0.2$, $Ig(\{S_2\}) = 0.6$, $Ig(\{S_2, S_3\}) = 0.2$ and $Ig(SC) = 0.8$ □

We continue with illustrating how this theory can be applied in modelling the rules of thumb. Applying definition 3.1 to the heuristics of type (1) means that for each heuristic experts have to indicate which storage configurations are supported by the heuristic and the belief they commit to it.

Example 3.3 Consider heuristic 1 of Figure 2. Suppose that the belief in an ISAM storage structure for small tables is 0.5 and the belief in a B-tree is 0.4. This heuristic can be modelled then as follows:

IF physical tables are small (< 15pages)
THEN

ISAM storage structure;	$m(\{\text{ISAM}\}) = 0.5$
B-tree storage structure;	$m(\{\text{B-tree}\}) = 0.4$
SC;	$m(SC) = 0.1$

We shall explain the meaning of this rule with the following example. Suppose we have a logical scheme consisting of two relations R_1 and R_2, in which R_1 is a small relation. Then the rule supports all storage schemes which have an ISAM storage configuration for relation R_1 with a bpa of 0.5 whatever the configuration of R_2 will be. Storage schemes with a configuration of a B-tree for relation R_1 are supported by a bpa of 0.4. The meaning of $m(SC) = 0.1$ is that no statement can be made for the remaining belief of 0.1, so we assign it to all possibilities. The assignment of the remained belief to all possibilities means that we do not know anything about the remaining possibilities. In this way we provide a technique to model *total ignorance*.

Note, the plausibility and the degree of ignorance with regard to a set of storage configurations can be computed straightforward. □

Note that the Dempster-Shafer theory provides an intuitive and flexible manner to model heuristics because if experts can not make choices between good storage configurations it is possible to model this. This theory provides the possibility to model exactly what you know and what you do not know.
Let us analyse the heuristic of type 2. Suppose that α

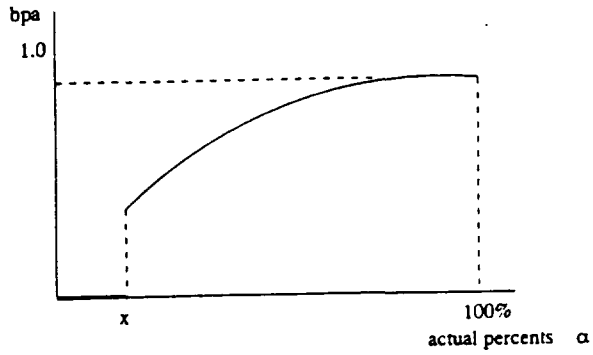

Figure 3: A possible function between the fraction satisfying a condition and the bpa

is the actual percentage that satisfies the condition and x the required percentage which has to be satisfied for the assignment of a non-zero belief to some conclusions. Then the heuristic of this type implicates that the belief in a conclusion depends on α. In general, the larger $\alpha - x$, the stronger the belief in the conclusion will be. Thus, the bpa in modelling heuristics of type 2 will be a function of α. In general, the grow of bpa as function of α will have the form of Figure 3.

Example 3.4 Consider Heuristic 2 of Figure 2. Let $IC(a)$ represent that a is an index candidate, $C_W(a)$ the percentage of changes on a by workload W and $f(.)$ is a function like the one in Figure 3. Then heuristic 2 can be modelled as follows:

IF $C_W(a) > 10\%$
THEN

$\neg(IC(a))$;	$m(\{\neg(IC(a))\}) = f(\alpha)$
SC;	$m(SC) = 1 - f(\alpha)$

To clarify the meaning of the $\neg$ operator we will consider the secondary index selection problem. Suppose we have a relation R consisting of attributes $\{a, b, c, ..., n\}$ and the placement of R is already determined. We have the task to select the most advantageous set of indices for R.

So, in this case secondary indices are the only permitted access structures. We will model SC now as the set containing all possible subsets of $\{a, b, ..., n\}$, because these are the sets of candidate index sets. Thus SC will contain 2^n elements.

SC = {(a), (b), ..., (n), (a,b), (a,c) ...., (a,n), ...,
(n-1,n),(a,b,c),..., (a,b,c, ...,n)}

Applying heuristic 2 of Figure 2 on SC (under the assumption that the condition is satisfied) means that all index sets are supported with $f(\alpha)$ *except* the sets which contain an index on attribute a because

an index on a is not supported. Thus, $\neg(IC(a))$ is equivalent with all index sets which do not contain a. So, $m(\{\neg(IC(a))\}) = f(\alpha)$ can be replaced by $m(\{(b),(c),...,(n),(b,c),...,(b,n),...,$
$(n-1,n),(b,c,d),....,(b,c,...,n)\}) = f(\alpha)$
Note, the set $\neg(IC(a))$ contains 2^{n-1} elements.

To express that the index set (a) is the only one which is not supported but for example the set (a,b) is supported we use the notation $m(\{\neg(IC([a]))\})$. □

An adequate modelling of heuristics into knowledge rules requires in our approach for each heuristic an associated belief function. If experts can not estimate a belief function for an heuristic then the heuristic is probably not sufficiently crystallized out.

Note, facts can be modelled by assigning a bpa with the value 1.0 to the fact.

Briefly, we can say that each knowledge rule will propose a number of storage configurations for one or more tables. In general, the proposals generated by different rules can support each other or can be conflicting. How to combine the proposals will be treated in section 4.

4 Combining of belief functions

In section 3 we have modelled heuristics into rules containing belief functions. For combining the belief functions we will make use of the combination rule of Dempster. This rule is most accessible when it is expressed in terms of the basic probability numbers, and especially when these basic probability numbers are depicted geometrically.

To make the discussion about the combination rule easier we introduce the notion of focal storage configuration. A storage configuration S is called a focal storage configuration if $m(S) > 0$.

Suppose m_1 is the bpa for a belief function Bel_1 and m_2 the bpa for a belief function Bel_2, both defined over SC. The focal storage configurations of Bel_1 are represented by $SO_i, i = 1,2,...,k$ and the focal storage configurations of Bel_2 are represented by $ST_j, j = 1,2,...,l$. A graphical representation of both belief functions is given in Figure 4 in which the bpa's of the focal storage configurations are depicted as segments of a line segment of length one. Figure 5 shows how m_1 and m_2 can be orthogonally combined to obtain a square.

The total surface of the square is one. The surface of a subsquare is the bpa assigned to the intersection of the focal storage configurations SO_i and ST_j and is achieved by multiplying the values of $m(SO_i)$ and $m(ST_j)$. If the intersection between two focal storage configurations is empty we want to assign the value zero to the bpa according definition 3.1 equation (3). This is realized by discarding all the subsquares corresponding to an empty intersection and normalizing the remaining surfaces of the subsquares such that the sum of the surfaces of these subsquares is one. This process is realized by the following combination rule of Dempster:

$$m(S) = K^{-1} \sum_{\substack{i,j \\ SO_i \cap ST_j = S}} m_1(SO_i)m_2(ST_j)$$

in which S is a non empty set and

$$K = \sum_{\substack{i,j \\ SO_i \cap ST_j \neq \emptyset}} m_1(SO_i)m_2(ST_j)$$

For a proof of the formula we refer to [Shaf 76]. K is called the normalization constant and m is called the sum of Bel_1 and Bel_2.

How to apply the rule of Dempster in the context of physical database design will be explained with the following example.

Example 4.1 Suppose we have a relation R with the attributes $\{a,b,c,d,e,f\}$ and the permitted storage structures for R are hashing, secondary indexing and vertical fragmentation. Assume further that there are two knowledge rules proposing storage configurations for R with belief functions Bel_1 and Bel_2 with bpa m_1 and m_2 respectively.

Let SC_1 be the possibilities for hashing, SC_2 the possibilities of secondary index sets and SC_3 the possibilities of vertical fragments. Note, the possibilities of secondary index sets and the possibilities of vertical fragments are all subsets of R (thus the power set of R).
Then $SC_1 = \{hash[a], hash[b], ..., hash[f]\}$, in which $hash[x]$ means hashing on attribute x, $SC_2 = \{IS[a], IS[b], ..., IS[f], IS[a,b], ..., IS[a,f], ..., IS[e,f], IS[a,b,c], ..., IS[a,b,c,d,e,f]\}$, in which $IS[x]$ means that x is an candidate index set, and $SC_3 = \{v[a], v[b], ..., v[f], v[a,b], ..., v[a,f], ..., v[e,f], v[a,b,c], ..., v[a,b,c,d,e,f]\}$, in which $v[x]$ means that x is a partition. SC is now defined as the union of SC_1, SC_2 and SC_3.
By $IC(x)$ we mean that attribute x is an index candidate. This implies all index sets which contain x. Some other notation that will be used in the following are $hash(*)$, $IS(*)$ and $v(*)$. By the statement "$hash(*)$ is supported" we mean all elements of SC_1 are supported. Analogous, the meaning of "$IS(*)$ is supported" and "v(*) is supported" is that all elements of SC_2 respectively all elements of SC_3 are supported.

Let rule 1 produce the following bpa:

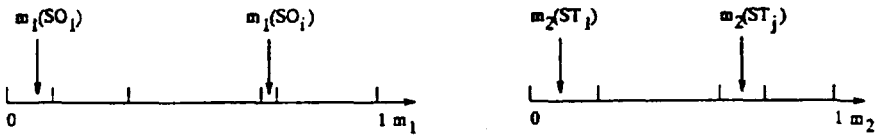

Figure 4: Graphical representation of the belief functions Bel_1 and Bel_2

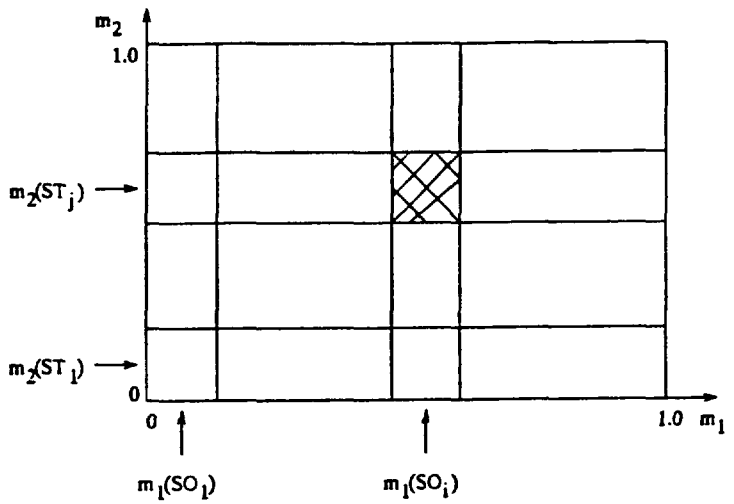

Figure 5: Combination of two belief functions Bel_1 and Bel_2

$m_1(\{hash[b], IC(c,d)\}) = 0.5$,
$m_1(\{\neg IC(e,f), \neg v(*)\}) = 0.3$ and $m_1(SC) = 0.2$
Let rule 2 produce the following bpa:
$m_2(\{hash[b]\}) = 0.4$, $m_2(\{IS[c,e,a]\}) = 0.4$ and
$m_2(\{v[a,b]\}) = 0.2$

Let us take a closer look at rule 1. The expression $IC(c,d)$ in $m_1(\{hash[b], IC(c,d)\}) = 0.5$ means that all index sets which contain an index on c or d as well as hashing on attribute b are supported which a bpa of 0.5. The number of index sets which contains an index on c or d is $2^5 + 2^5 - 2^4 = 48$.

The expression $m_1(\{\neg IC(e,f), \neg v(*)\}) = 0.3$ means that there is no support at all for any vertical partition and that there is no support for the index sets which contain an index on attribute e or f with a bpa of 0.3. This implies support for hashing and for all index sets which are a subset of $\{a,b,c,d\}$ with a bpa of 0.3.

Combining the two belief function leads to Figure 6. In this figure $m_1(\{\neg IC(e,f), \neg v(*)\})$ is replaced by its equivalent $m_1(\{hash(*), \neg IC(e,f)\})$.

The normalization constant is achieved by summing the bpa's of the subsquares leading to a non empty intersection. Thus, the normalization constant is 0.72. Then, the combined bpa is: $m(\{hash(b)\}) = \frac{0.20+0.12+0.08}{0.72} = 0.56$, $m(\{IC[c,e,a]\}) = 0.39$, and $m(\{v[a,b]\}) = 0.06$.

On basis of the combination of the two rules we can conclude that the storage of the relation by hashing on attribute b and the index set (a,c,e) has reasonable support. The division of the relation in two subrelations (a,b) and (c,d,e,f) is weakly supported.

The tool has now the choice to send several storage configurations to the optimizer for an estimation of the cost in processing the workload defined on the database. Then, the tool will choose the one with the lowest cost. For example, the tool may decide to send all storage configurations with a belief greater than 0.8 to the optimizer. An example of a storage configuration which satisfies this condition is the storage configuration which has hashing on b as placement and the index set $[c,e,a]$ as access structures. Note, this storage configuration has a belief of 0.95. □

The combination rule provides a mechanism to combine several rules so that it is possible to select a good storage scheme whether the rules are conflicting or supporting each other. In the example above it is illustrated how to deal with conflicting rules. For example, rule 1 does not support any vertical partition while rule 2 supports the partition (a,b).

5 Conclusions

We propose a framework of a tool which can help a database designer in the selection of good storage schemes. The tool is based on the combination of an

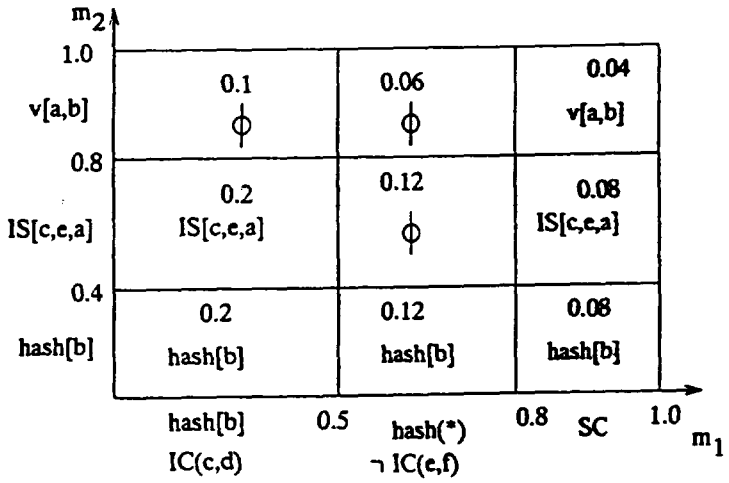

Figure 6: Combining two belief functions with bpa m_1 and m_2 respectively

optimizer and a knowledge based approach. In this way we profit from the advantages of both approaches.. In a combined approach the emphasis lies on the aspects of the knowledge based approach because the knowledge rules and the combination of these rules are responsible for the generation of candidate storage schemes. This means that we have to model rules and to perform the combination of rules carefully. We have analysed about 60 rules of thumb which are used by physical design experts at the GAK. We discovered that these heuristics can be classified into two types and we provide techniques how to model these heuristics in a proper way into knowledge rules taking *uncertainty* and *ignorance* into account. Then we propose to combine these rules with the combination rule of Dempster which has a mathematical foundation. The combination of rules which are conflicting and rules which are supporting each other can be treated in the same way. So, the foundation of a tool for physical database design has been achieved.

Such a tool is more powerful because it may be connected to more advanced optimizers than the tool presented in [FiST 88]. The knowledge system of the tool contains information which takes care of the exponential complexity. This information is achieved by modelling the knowledge and experience of experts in a founded way which fits better in practice. So, the knowledge system of the tool may simulate the knowledge and experience of experts better than for example in the approaches of [DaJe 88,RoSh 91]. In general experts are able to generate a number of good storage schemes on basis of their knowledge and experience for a given workload and a logical scheme. The limited number of storage schemes proposed by the knowledge system will be passed on the optimizer to determine the one with the lowest cost.

The foundation of the tool has as advantage that proposed storage schemes can be better understood and possibly maintenance of the tool is easier.

Topics for further research are the implementation of the tool, the connection of the tool with Ingres and the application of the tool in designing large databases.

References

[Barn 81] Barnett, J.A., Computational Methods for a Mathematical Theory of Evidence, in Proc. IJCAI-81, pp. 868-875 (1981).

[BGWRR 81] Bernstein, P.A., Goodman, N., Wong, E., Reeve, C.L., Rothnie, J.B., Query Processing in a System for Distributed Databases (SDD-1), in ACM TODS, Vol. 6, No. 4, pp.602-625 (1981).

[Card 75] Cardenas, A.F., Analysis and Performance of Inverted Database Structures, in Comm. ACM, 18 (1975), pp. 253-263.

[Choe 90] Choenni, R., The Dempster-Shafer theory and applications of the Dempster-Shafer theory with regard to the multi-radar tracking problem, Master thesis 1990, Delft University of Technology, 107p (1990).

[ChBC 92a] Choenni, R., Blanken, H.M., Chang, S.C., Physical Design for Network Databases, Memorandum 92-26, University of Twente, 27p. (1992).

[ChBC 92b] Choenni, R., Blanken, H.M., Chang, S.C., Reducing the Complexity in the Selection of an Optimal Set of Secondary Indices, to appear as memorandum, University of Twente.

[Come 78] Comer, D., The Difficulty of Optimum Index Selection, in ACM Transactions on Database Systems, Vol. 3, No. 4, pp. 440-445 (1978).

[DaJe 88] Dabrowski, C.E., Jefferson, K.J., A Knowledge Based System for Physical Database Design, in NBS Special Publication 500-151, 51p. (1988).

[Demp 67] Dempster, A.P.,Upper and Lower Probabilities Induced by a Multi-Valued Mapping, in Annals Math. Stat. 38, pp.325-339 (1967).

[Demp 68] Dempster, A.P., A Generalisation of Bayesian Inference, in Journal of the Royal Statistical Society, Series_B 30, pp. 205-247 (1968).

[FiST 88] Finkelstein, S., Schkolnick, M., Tiberio, P., Physical Database Design for Relational Databases, in ACM TODS, Vol. 13, No. 1, pp. 91-128 (1988).

[GrSt 82] Grimmet, G.R., Stirzaker, D.R., Probability and Random Processes, Oxford Science Publications, 354p. (1982).

[HaFa 92] Halpern, J.Y., Fagin, R., Two views of belief: belief as generalized probability and belief as evidence, in Artificial Intelligence 54, pp. 275-317 (1992).

[RoSh 91] Rozen, S., Shasha, D., A framework for automating database design, in Proc. of the International Conference on Very Large Databases, pp. 401-411 (1991).

[Shaf 76] Shafer, G., A Mathematical Theory of Evidence, Princeton University Press, 297p. (1976).

[ShLo 87] Shafer, G., Logan, R., Implementing Dempster's rule for Hierarchical Evidence, in AI 33, pp. 271-298 (1987).

[Walr 90] Walraven, H.G., A knowledge system for supporting physical database design, Master thesis, University of Twente, 88p. (1990).

[Zade 84] Zadeh, L.A., Review of Shafer's A Mathematical theory of Evidence, in AI-Magazine 5, pp. 81-83 (1984).