Modelling Uncertainty in Physical Database Design

Sunil Choenni[1,2] Henk Blanken[2]
[1]Nat. Aerospace Laboratory, P.O. Box 90502, 1006 BM Amsterdam, the Netherlands
[2]Univ. of Twente, Dept. of Comp. Science, P.O. Box 217, 7500 AE Enschede, the Netherlands
email: {choenni, blanken}@cs.utwente.nl

Abstract

Physical database design can be marked as a crucial step in the overall design process of databases. The outcome of physical database design is a physical schema which describes the storage and access structures of the stored database. The selection of an efficient physical schema is an NP-complete problem. A significant number of efforts has been reported to develop tools that assist in the selection of physical schemas. Most of the efforts implicitly apply a number of heuristics to avoid the evaluation of all schemas. In this paper, we present an approach, based on the Dempster-Shafer theory, that explicitly models a rich set of heuristics —used for the selection of an efficient physical schema— into knowledge rules. These rules may be loaded into a knowledge base, which, in turn, can be embedded in physical database design tools.

1 Introduction

The design of databases takes place on several levels. One of these levels is the so-called physical level, and the design of databases at this level is called *physical database design*. Physical database design aims to achieve efficient physical schemas by organizing data in such way that the operations defined on the data can be quickly processed and with low cost. Typical problems at the physical level are the assignment of efficient storage structures to certain amounts of data and the allocation of secondary indices to attributes. A storage structure may be considered as a file arrangement, whether or not clustered on a certain attribute, providing a way to access data. The clustering attribute is known as the primary index. Secondary indices, also known as access structures, can be regarded as auxiliary files that allow to retrieve parts of the data satisfying a certain selection predicate without having to examine all available data. Updating the database, causes an index to be updated to remain consistent with the new database state. So, an index speeds up retrieval and slows down maintenance.

In general, bad choices for a physical schema will lead to poor performance, and, therefore, the database system may become less valuable. Thus, physical database design can be marked as an important step in the overall design process of databases.

The number of physical schemas among which database designers have to select a schema is enormous. The evaluation of a physical schema is a tedious and error-prone process. One should understand the workings of a particular database management system. Therefore, there is a practical need to develop tools that assist database designers in the selection of physical schemas. In the literature, a significant number of efforts has been reported to develop such tools [2, 4, 6, 7, 8, 11, 18]. Most of the efforts *implicitly* apply a number of heuristics to avoid the evaluation of all schemas. Uncertainty and ignorance, which characterize many of these heuristics, are not taken into account.

In this paper, we present an approach that *explicitly* models a rich set of heuristics —used for the selection of an efficient schema— into production rules to which a measure of uncertainty is attached. These heuristics can be loaded in a knowledge base that might be used in physical database design tools.

We have studied about 60 heuristics that are used by database administrators in various companies in the Netherlands and observed that most of the rules contain a degree of uncertainty, ignorance, and qualitative notions. An example of a typical heuristic is: "A Heap storage structure is in 90% of the cases adequate for small relations". The percentage 90% in this heuristic implies a certain degree of uncertainty and *small* is a qualitative notion. If we asked the database administrators what storage structure is a candidate in the case that a Heap is not chosen for a small relation, it appeared that all possible storage structures might be a choice. This implies that ignorance also plays a role in the field of physical database design.

To capture uncertainty and ignorance in the heuristics, we have used the Dempster-Shafer theory. This theory, also known as the theory of belief functions, has been introduced by Dempster [9] and mathematically founded by Shafer [19]. It can be regarded as a generalization of probability theory and also as a theory for dealing with evidence [12]. For our purpose, we regard to the theory as a theory of evidence.

The theory offers an attractive formalism to represent relevant notions as 'The belief in A on the basis of evidence E'. A central instrument in the Dempster-Shafer theory is the rule of Dempster, which is used to combine several evidences. In Artificial Intelligence, Dempster-Shafer theory has attracted much attention. Variants of the theory have been applied as a tool to handle uncertain information in many applications, see among others [3, 10, 16, 17]. In the field of Database technology, the theory is also receiving attention. In [1, 14, 15], it has been used to extend the relational model. We note that the authors in [1] take the view that their extension of the relational model is based on probability theory and on the so-called concept of 'missing probabilities'. However, they are presumably not aware that their concept of 'missing probabilities' is covered by the Dempster-Shafer theory.

The remainder of this paper is organised as follows. In two consecutive sections, the problem of physical database design and the notion of physical schema will be discussed in more detail. In Section 4, we study a set of heuristics that are used by database administrators to select a physical schema. In Section 5, we model the heuristics into knowledge rules with the Dempster-Shafer theory and propose a Dempster-Shafer approach for physical database design. Finally, Section 6 concludes the paper.

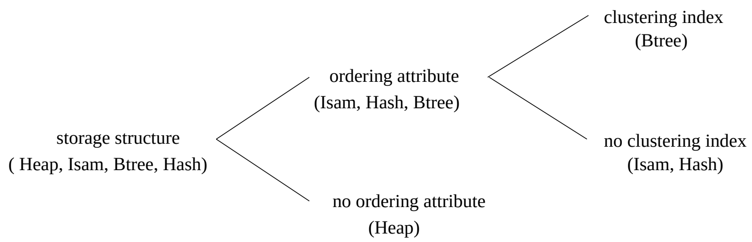

Figure 1: Several kind of storage structures

2 Physical database design

As already noticed, the outcome of physical database design is a physical schema. In the selection of a physical schema, the operations defined on the data, called the *workload*, play a crucial role. A physical schema that may be good or optimal for a certain workload, may be bad for another workload.

Basically, we may distinguish four kinds of database operations on a relational schema[1] namely, insertions, deletions, updates, and queries. Insertions are used to insert new tuples in a relation, deletions are used to delete tuples, updates are used to change the values of some attributes, and queries are used to derive a relation .

In general, a number of operations of each type are defined on a relational schema. To each operation a weight is assigned, which is based on the frequency and the importance of the operation. A high weight implies that once an operation is started to be processed, this should be done fast, while a low weight implies that there are hardly any conditions for the processing time.

Based on the relational schema, the workload, and some other database characteristics, such as the cardinality of a relation, length of a tuple, number of pages to store a relation, etc., a storage structure and a set of indices should be selected for each relation. A storage structure determines the order of the tuples of a relation on disk. If this order is determined by an attribute, this attribute is called the *ordering* attribute. An index is a set of pairs (key value, TID-list). The key values are a subset of the domain of the indexed attribute, and a tuple identifier (TID) in the TID-list identifies a tuple possessing the key value.

An index on an ordering attribute is called a *clustering* index and an index on a non-ordering attribute is called a *secondary* index. We note that a storage structure is also associated with each index.

In Figure 1, we depict how the notions storage structure, ordering attribute, and clustering index are related. Furthermore, for a number of storage structures, we have indicated between brackets whether they have an ordering attribute or not. If a storage structure has an ordering attribute, we have indicated whether the ordering attribute is indexed or not. For example, the storage structure *Heap* does not have an ordering attribute, and, therefore, it is unordered. Storage structures that have an

[1]A relational schema is a set of relations. A relation R is defined over some attributes $\alpha_1, \alpha_2, ..., \alpha_n$, and is a subset of the Cartesian product $dom(\alpha_1) \times dom(\alpha_2) \times ... \times dom(\alpha_n)$, in which $dom(\alpha_j)$ is the domain of attribute α_j.

ordering attribute are Isam, Btree, and Hashing [13].

In general, the way a clustering index is organized depends on the storage structure to which the clustering index is related. For example, the storage structure Btree in Ingres allocates an index to the ordering attribute (resulting into a clustering index), and to this clustering index a pre-defined storage structure is assigned

In the remainder of this paper, we focus on the selection of a storage structure and a set of indices for each relation, and refer to it as physical database design. Although our description does not cover the overall problem of physical design, it covers the most difficult and crucial parts [6].

3 Physical schema

In this section, we formalize the notion of physical schema. First, we outline the assumptions on which the definition of a physical schema is based.

We assume that either a secondary index or a clustering index can be allocated on an attribute (but not both). The way a clustering index is stored is assumed to be fixed.

Exactly one storage structure can be assigned to a relation. The storage structures that are considered are Heap, Isam, Btree, and Hashing. Since these storage structures are concerned with the arrangement of tuples of a *single* relation on disk, we do *not* consider the possibility to absorb a relation in another relation. As a consequence, we assume that a page contains tuples of exactly one relation.

A last assumption is that indices and ordering attributes concern *single* attributes. A physical schema for a single relation is now defined as follows.

Def. 1 Let R be a relation with attributes $\alpha_1, \alpha_2, ..., \alpha_n$. A *physical schema* p^R corresponding to R is an element of P^R, in which

$$\begin{aligned} P^R &= \{(x_0(A_0), \{x_i(A_i) \mid i = 1, 2, .., m\}) \mid m \in I\!N; \\ &\quad \forall i, j \in \{0, 1, 2, ...m\} : i \neq j \Rightarrow A_i \cap A_j = \emptyset; \\ &\quad |A_0| \leq 1; \forall i > 0 : |A_i| = 1; \forall i \geq 0 : A_i \subset \{\alpha_1, \alpha_2, ..., \alpha_n\}; \\ &\quad x_0, x_i \text{ are storage structures}; |A_0| = 0 \Rightarrow x_0 = \text{Heap}\} \end{aligned}$$

The expression $x_0(A_0)$ means that a relation is stored as x_0 and ordered on the set of attributes A_0. We note that if a relation is stored as Heap, then A_0 is the empty set, else A_0 contains exactly one element. The expression $x_i(A_i)$ represents that a secondary index is allocated to the set of attributes A_i and is stored as x_i. Note, A_i consists of exactly one element, since we restrict ourselves to single attribute indices. So, extension of Def. 1 by multi-attribute indices is straightforward.

An (overall) physical schema for a set of relations is defined as the union of the selected physical schema for each relation.

We note that the selection of physical schemas per relation is justified in [20, 6].

The following example illustrates the notion of physical schemas. Since the number of elements of a set $A_i, i \geq 0$, is zero or one, we write $x_i()$ and $x_i(\alpha_i)$ instead of

Heuristic 1:
IF *a relation is small (< 6 pages)*
THEN *Heap is often (90%) an adequate storage structure*

Heuristic 2:
IF *the more the percentage of operations that changes the value of an attribute in a workload exceeds 10%*
THEN *the more this attribute is not an index candidate*

Figure 2: Examples of some heuristics

$x_i(\{\})$ and $x_i(\{\alpha_i\})$ respectively. For convenience's sake, the set brackets are omitted.

Example 1 Let us consider the following relational schema:
Person(per#, first_name, last_name, birth_date, city),
Vehicle(veh#, model, color, doors, body, manufacturer)
Owns(per#, veh#, money_paid)

Two overall physical schemas for the above-mentioned relational schema are given below.

In the first overall physical schema, the relation *Person* is stored as Heap and two secondary indices, both stored as Btrees, to the attributes *city* and *last_name* respectively are allocated. The relation *Vehicle* is hashed on the attribute *veh#*. The relation *Owns* is stored as Heap.

In the second overall physical schema, the relations *Person* and *Vehicle* are stored as Heap. The relation *Owns* is stored as Btree, ordered on attribute *per#*, and a secondary index, stored as Btree, to *money_paid* is allocated.

1. ((Heap(),{Btree(*city*), Btree(*last_name*)}), (Hash(*veh#*), {}), (Heap(),{}))
2. ((Heap(),{}), (Heap(),{}), (Btree(*per#*), {Btree(*money_paid*)})) □

4 Heuristics

After analysing about 60 heuristics used by experts for physical database design, we have observed the following. First, the heuristics consist of a condition and conclusion part (see Figure 2). Second, experts have apparently no difficulties to translate qualitative notions into quantitative measures. In general this is a tough task. In Heuristic 1 of Figure 2, a quantification of the notion small is given between brackets. Third, heuristics have an uncertain character. A heuristic works well in many cases but not in all cases. Database administrators are able to estimate in how many percent of the cases a heuristic may be successfully applied. For example, applying Heuristic 1 of Figure 2 results in 90% of the cases into Heap as storage structure. The heuristic says nothing about the remaining 10% implying *ignorance* in these cases. We note that the latter information is not explicitly captured in Heuristic 1. Fourth, we may distinguish two types of heuristics.

1. The belief in the conclusion(s) is based on the fact whether the condition part is true or not. For example, in Heuristic 1 of Figure 2, the belief that a Heap storage structure is chosen for a small relation is independent of how small the relation is.

2. The belief in the conclusion(s) is dependent of the extent to which a condition part is satisfied. For example, the idea behind Heuristic 2 of Figure 2 is that if the number of operations in a workload that changes an attribute α_h increases, then the belief that α_h is not an index candidate grows. To represent this uncertain character, it is not sufficient to represent heuristics only with a condition and conclusion part.

For the time being, we represent the heuristics of type 2, thus for which holds that the belief in the conclusion increases (or decreases) if the extent to which the conditions are satisfied increases (decreases), as follows.

$$\underline{\text{IF}}\ (\textit{conditions}(y\%)) \wedge (y \geq y_0)\ \underline{\text{THEN}}\ \textit{conclusion with belief}\ f(y - y_0) \qquad (1)$$

We note that y is the actual percentage to which the conditions are satisfied, y_0 is the minimal required percentage in order to draw *conclusion*, and $f(y)$ is a function of y. The belief in *conclusion* increases (or decreases) if the value of $y - y_0$ becomes higher (smaller).

In the next section, we discuss the Dempster-Shafer theory to capture the uncertain character of both type of heuristics in order to achieve knowledge rules.

5 A Dempster-Shafer approach to select physical schemas

We feel that the Dempster-Shafer theory is a suitable theory to capture the uncertainty contained in the heuristics used by database designers. Before illustrating this, we give a brief description of the theory in the context of physical database design. We start with defining what should be understood by all permitted overall physical schemas for a relational schema in which r relations are involved.

Def. 2 Let P^R (see Def. 1)be the set of all permitted physical schemas corresponding to a relation R having attributes $\alpha_1, \alpha_2, ..., \alpha_n$. The *set of permitted overall physical schemas* for a relational schema in which r relations, $R_1, R_2, ..., R_r$, are involved, called the frame of discernment, is $P^{R_1,R_2,...,R_r} = P^{R_1} \times P^{R_2} \times ... \times P^{R_r}$ In the following, $P^{R_1,R_2,...,R_r}$ is abbreviated as P^{DB}.

The following example lists all permitted physical schemas corresponding to a single relation.

Example 2 Consider the relation *Owner(per#, veh#, money_paid)*, which has been introduced in Example 1. We assume that a relation is either stored as a Heap or hashed on a single attribute. A secondary index is stored as a Btree.

In the following, we write p_i for the i-th physical schema of relation *Owner* instead of p_i^{Owner}. The set of all permitted physical schemas for *Owner*, is $P^{Owner} =$

p_1	=	(Heap(), {})
p_2	=	(Heap(), {Btree(per#)})
p_3	=	(Heap(), {Btree(veh#)})
p_4	=	(Heap(), {Btree(money_paid)})
p_5	=	(Heap(), {Btree(per#), Btree(veh#)})
p_6	=	(Heap(), {Btree(per#), Btree(money_paid)})
p_7	=	(Heap(), {Btree(veh#), Btree(money_paid)})
p_8	=	(Heap(), {Btree(per#), Btree(veh#), Btree(money_paid)})
p_9	=	(Hash(per#), {})
p_{10}	=	(Hash(per#), {Btree(veh#)})
p_{11}	=	(Hash(per#), {Btree(money_paid)})
p_{12}	=	(Hash(per#), {Btree(veh#), Btree(money_paid) })
p_{13}	=	(Hash(veh#), {})
p_{14}	=	(Hash(veh#), {Btree(per#)})
p_{15}	=	(Hash(veh#), {Btree(money_paid)})
p_{16}	=	(Hash(veh#), {Btree(per#), Btree(money_paid) })
p_{17}	=	(Hash(money_paid), {})
p_{18}	=	(Hash(money_paid), {Btree(veh#)})
p_{19}	=	(Hash(money_paid), {Btree(money_paid)})
p_{20}	=	(Hash(money_paid), {Btree(per#), Btree(veh#)})

Table 1: All permitted physical schemas for the relation *Owner*

$\{p_1, p_2, p_3, ..., p_{20}\}$. The schemas $p_1, p_2, p_3, ..., p_{20}$ are listed in Table 1. The physical schema p_1 means that *Owner* is stored as Heap and no secondary indices are allocated, while p_{12} means that *Owner* is hashed on the attribute *per#* and secondary indices —both stored as Btrees— are allocated to attributes *veh#* and *money_paid*. □

Def. 3 Let P^{DB} be the set of all permitted overall physical schemas for a relational schema. Let $I\!P(P^{\mathrm{DB}})$ be the power set of P^{DB}, then a function $m: I\!P(P^{\mathrm{DB}}) \rightarrow [0,1]$ is called a *basic probability assignment (bpa)* whenever

$$m(\emptyset) = 0 \quad \text{and} \quad \sum_{P \subseteq P^{\mathrm{DB}}} m(P) = 1$$

The quantity $m(P)$ is called P's *basic probability number* and it is understood to be the measure of *belief* that is exactly committed to the set of overall physical schemas P. The *total* belief in P, $(Bel(P))$, is the sum of the basic probability numbers of all subsets PP of P. The following definition describes the relation between belief and basic probability assignment in a formal way. We note that in the definition for P holds that $P \subseteq I\!P(P^{\mathrm{DB}})$.

Def. 4 A function *Bel* is called a *belief function* over P^{DB} if it is given by the following equation for some bpa $m: I\!P(P^{\mathrm{DB}}) \rightarrow [0,1]$.

$$Bel(P) = \sum_{PP \subseteq P} m(PP) \tag{2}$$

We note that a basic probability assignment induces a belief function and conversely. In the following, we illustrate how to compute a belief function from a given bpa.

Example 3 Assume that the following bpa is defined on the set of physical schemas listed in Table 1: $m(\{p_1\}) = 0.2$, $m(\{p_{10}\}) = 0.2$, $m(\{p_6, p_{10}, p_{18}\}) = 0.4$,$m(P^{Owner}) = 0.2$, and $m(P) = 0$ otherwise. Therefore, the corresponding belief function is: $Bel(\{p_1\}) = 0.2$, $Bel(\{p_{10}\}) = 0.2$, $Bel(\{p_6, p_{10}, p_{18}\}) = 0.6$, and $Bel(P^{Owner}) = 1$. Note that the expression $Bel(\{p_6, p_{10}, p_{18}\}) = 0.6$ means that the total belief in the set of schemas $\{p_6, p_{10}, p_{18}\}$ is 0.6. However, we are not able to distribute this belief among the schemas in the set. □

Two other notions that are related with a belief function are plausibility and ignorance. The plausibility in a set of physical schemas P expresses the maximal belief in this set, and is defined as $Pl(P) = 1 - Bel(P^C)$, in which P^C is the complement of P relative to P^{DB}. The ignorance with regard to a set of overall physical schemas P, is defined as $Ig(P) = Pl(P) - Bel(P)$.

In Section 5.1, we illustrate how the two types of heuristics discussed in Section 4 may be modelled as knowledge rules. Then, in section 5.2, we discuss how these rules may contribute in solving the problem of physical database design.

5.1 Knowledge rules

In Section 4, it was noted that database experts use heuristics that contain uncertainty and ignorance for the design of physical schemas. We continue by illustrating how to model the heuristics into knowledge rules. A knowledge rule has an antecedent and a consequent. With the consequent, a bpa is associated that expresses the belief that is committed to the consequent.

Since the conclusion(s) of both types of heuristics of Section 4 actually support a number of overall physical schemas, the consequent part of a knowledge rule should support this property. In the following example, we derive the knowledge rule corresponding to Heuristic 1 of Figure 2.

Heuristic 1 Suppose that the belief in a Heap storage structure for small tables is 0.9 and let $P^{R_l}_{\text{Heap}}$ be the set of all permitted physical schemas storing relation R_l as Heap whatever the set of secondary indices —and their storage structures— is. We note that $P^{R_l}_{\text{Heap}}$ formally means:

$$\begin{array}{l}\{(\text{Heap}(), \{x_i(A_i) \mid i = 1, 2, .., m\}) \mid m \in I\!N; \\ \forall i, j \in \{0, 1, 2, ...m\} : i \neq j \Rightarrow A_i \cap A_j = \emptyset; \\ \forall i \geq 1 : (A_i \in \{\alpha_1, \alpha_2, ..., \alpha_n\} \wedge |A_i| = 1); \\ \;x_i \text{ is a storage structure}\}\end{array}$$

We note that $\alpha_1, \alpha_2, ..., \alpha_n$ are attributes of relation R_l.

The knowledge rule (k_1) corresponding to Heuristic 1 is given below. In this rule $n^{R_l}_{pag}$ represents the number of pages required to store relation R_l.

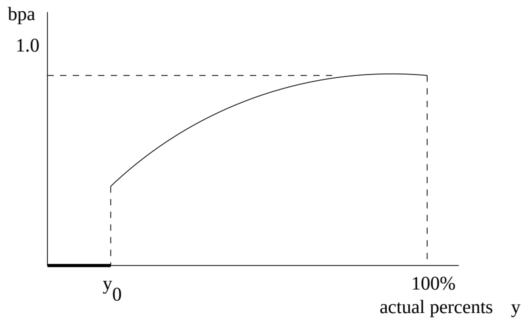

Figure 3: A possible function between the fraction satisfying a condition and the bpa

k_1: IF $n_{pag}^{R_l} < 6$ pages
THEN

$$P_{\text{Heap}}^{R_l} \times P^{\text{DB}\backslash R_l}; \quad m(P_{\text{Heap}}^{R_l} \times P^{\text{DB}\backslash R_l}) = 0.9$$

$$P^{\text{DB}}; \quad m(P^{\text{DB}}) = 0.1$$

We note that $P_{\text{Heap}}^{R_l} \times P^{\text{DB}\backslash R_l}$ is an abbreviation for:
$P^{R_1} \times P^{R_2} \times ... \times P^{R_{l-1}} \times P_{\text{Heap}}^{R_l} \times P^{R_{l+1}} \times ... \times P^{R_r}$

Let us explain the belief value committed to P^{DB}. Heuristic 1 of Figure 2 tells us that if a physical table is small we choose to store relation R_l as Heap with a belief of 0.9. However, no statement is made for the remaining belief of 0.1. In this case, no preference is given to any overall physical schema. Therefore, this belief is committed to the whole frame of discernment. In this way *ignorance* is modelled. □

Let us recall the meaning of the heuristic of type 2 in Section 4, before giving the corresponding knowledge rule. Suppose that y is the actual percentage that satisfies the condition and y_0 the required percentage that has to be satisfied for committing a non-zero belief to a set of overall physical schemas. Then, the heuristic of this type implies that the belief in a set of overall physical schemas depends on y. In general, the larger $y - y_0$, the stronger the belief in this set of overall physical schemas. Thus, the bpa in modelling heuristics of type 2 will be a function of y, which generally have the form of Figure 3. In the following, we model Heuristic 2 of Figure 2.

Heuristic 2 Let $Ch_W(\alpha_h)$ be the percentage of changes on an attribute α_h of relation R_l by workload W and $f(Ch_W(\alpha_h))$ is a function like the one in Figure 3. Then, Heuristic 2 can be modelled as knowledge rule k_2:

k_2: IF $Ch_W(\alpha_h) > 10\%$
THEN

$$P_{\neg\alpha_h}^{R_l} \times P^{\text{DB}\backslash R_l}; \quad m(P_{\neg\alpha_h}^{R_l} \times P^{\text{DB}\backslash R_l}) = f(Ch_W(\alpha_h))$$

$$P^{\text{DB}}; \quad m(P^{\text{DB}}) = 1 - f(Ch_W(\alpha_h))$$

The expression $P^{R_l}_{\neg\alpha_h}$ is a shorthand for:

$$\{(x_0(A_0), \{x_i(A_i) \mid i = 1, 2, .., m\}) \mid m \in I\!N;$$
$$\forall i, j \in \{0, 1, 2, ...m\} : i \neq j \Rightarrow A_i \cap A_j = \emptyset;$$
$$|A_0| \leq 1; \forall i > 0 : |A_i| = 1; \forall i \geq 0 : A_i \subset \{\alpha_1, \alpha_2, ..., \alpha_n\} \setminus \{\alpha_h\};$$
$$x_0, x_i \text{ are storage structures}; |A_0| = 0 \Rightarrow x_0 = \text{Heap}\}$$

We note that the expression $P^{R_l}_{\neg\alpha_h} \times P^{\text{DB}\setminus R_l}$ represents the set of physical schemas storing relation R_l in such a way that R_l is neither ordered on attribute α_h nor a secondary index is allocated on α_h. □

Modelling mathematical properties in knowledge rules is straightforward. Since mathematical properties are exact under certain conditions, these properties should be given full belief. This means that a knowledge rule that represents a mathematical property will be associated with a bpa having the value 1.0. An example is given below.

Example 4 Suppose we have derived the following property under some conditions *Con* for physical schemas consisting of *single* relations: 'If the addition of a secondary index on attribute α_h to a physical schema with regard to relation R_l decreases the cost (in handling the workload defined on the schema), then this index should be added to the set of secondary indices [5]'. Let us assume that secondary indices are stored as Btree. Then this property can be modelled as follows:

k_3: IF $\quad Con$ AND $C(p^{R_l}_{\neg\alpha_h \cup \alpha_h}) < C(p^{R_l}_{\neg\alpha_h})$
THEN
$\quad P^{R_l}_{\neg\alpha_h \cup \alpha_h, \text{Btree}}; \qquad m(P^{R_l}_{\neg\alpha_h \cup \alpha_h, \text{Btree}}) = 1.0$

We note that $C(.)$ is a cost function that computes the cost of a physical schema, $p^{R_l}_{\neg\alpha_h}$ is a physical schema that neither has α_h as ordering attribute nor as index, and $p^{R_l}_{\neg\alpha_h \cup \alpha_h}$ is the physical schema $p^{R_l}_{\neg\alpha_h}$ to which α_h is added as secondary index. For $p^{R_l}_{\neg\alpha_h}$ and $p^{R_l}_{\neg\alpha_h \cup \alpha_h}$ holds, $p^{R_l}_{\neg\alpha_h} \in P^{R_l}_{\neg\alpha_h, \text{Btree}}$ and $p^{R_l}_{\neg\alpha_h \cup \alpha_h} \in P^{R_l}_{\neg\alpha_h \cup \alpha_h, \text{Btree}}$, in which the sets $P^{R_l}_{\neg\alpha_h, \text{Btree}}$ and $P^{R_l}_{\neg\alpha_h \cup \alpha_h, \text{Btree}}$ represent the physical schemas

$$\{(x_0(A_0), \{\text{Btree}(A_i) \mid i = 1, 2, .., m\}) \mid m \in I\!N;$$
$$\forall i, j \in \{0, 1, 2, ...m\} : i \neq j \Rightarrow A_i \cap A_j = \emptyset;$$
$$|A_0| \leq 1; \forall i > 0 : |A_i| = 1; \forall i \geq 0 : A_i \subset \{\alpha_1, \alpha_2, ..., \alpha_n\} \setminus \{\alpha_h\};$$
$$x_0 \text{ is a storage structure}; |A_0| = 0 \Rightarrow x_0 = \text{Heap}\}$$

and

$$\{(x_0(A_0), \{\text{Btree}(A_i) \cup \text{Btree}(A_h) \mid i = 1, 2, .., n\}) \mid m \in I\!N;$$
$$\forall i, j \in \{0, 1, 2, ...m\} : i \neq j \Rightarrow A_i \cap A_j = \emptyset;$$
$$A_h = \{\alpha_h\}; |A_0| \leq 1; \forall i > 0 : |A_i| = 1; \forall i \geq 0 : A_i \subset \{\alpha_1, \alpha_2, ..., \alpha_n\};$$
$$x_0 \text{ is a storage structure}; |A_0| = 0 \Rightarrow x_0 = \text{Heap}\}$$

respectively. □

We assume that experts are able to give a reliable belief function for a knowledge

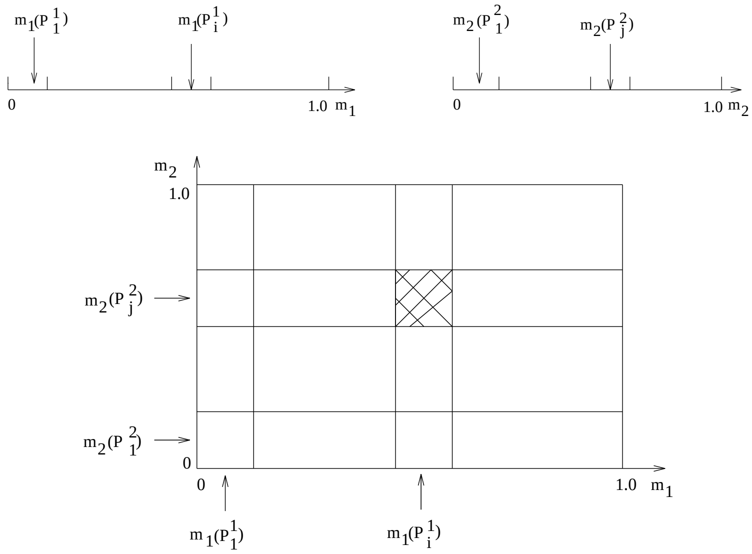

Figure 4: Combination of m_1 and m_2

rule. If experts are not able to estimate a belief function for a knowledge rule corresponding with a heuristic that is used by them, then the heuristic probably has not taken shape yet. Such a rule might be better omitted from a knowledge base.

We are aware of the fact that a belief function proposed by an expert will be an approximation rather than an exact function. For example, it is not unlikely that in knowledge rule k_1 the belief in $P_{\text{Heap}}^{R_l} \times P^{\text{DB}\setminus R_l}$ should be 0.86 or 0.93 instead of 0.9. Consequences of variations in belief functions is a topic for further research.

5.2 Combining knowledge rules

Each knowledge rule supports or rejects a set of overall physical schemas with a certain belief. Intuitively, if two rules support the same set of overall physical schemas P, then the combination of these rules should result into a higher belief for P, while if one of the rules supports P and the other rule rejects P, then this should result into a lower belief for P. The combination rule of Dempster possesses these properties. We discuss this combination rule and illustrate how it may be applied.

The rule of Dempster 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 overall physical schemas. A set of overall physical schemas P is called a *focal* set if $m(P) > 0$.

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 a set P^{DB}. The focal sets of m_1 are represented by $P_i^1, i = 1, 2, ..., k$ and the focal sets of m_2 are represented by $P_j^2, j = 1, 2, ..., l$. In Figure 4, a graphical representation of both bpa's is given. The bpa's of the focal sets are depicted as segments of a line segment of length one and it is shown how m_1 and m_2 can be orthogonally combined to obtain a square.

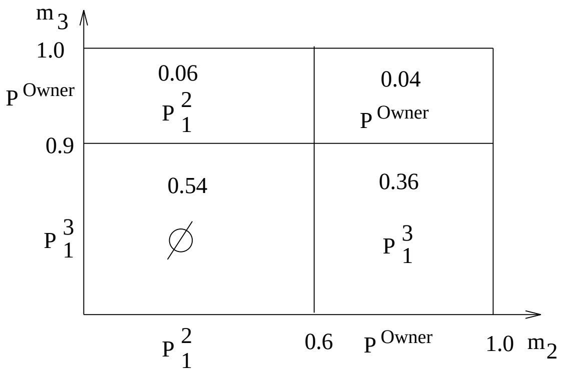

Figure 5: Combining the bpa's m_2 and m_3

The total surface of the square is one. The surface of a subsquare is the bpa assigned to the intersection of the focal sets P_i^1 and P_j^2. If the intersection between two focal sets is empty, the value zero should be assigned to the bpa according to Def. 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 [19], in which K is called the normalization constant.

$$m_1 \oplus m_2(P) = K^{-1} \sum_{\substack{i,j \\ P_i^1 \cap P_j^2 = P}} m_1(P_i^1) m_2(P_j^2)$$

in which P is a non empty set and

$$K = \sum_{\substack{i,j \\ P_i^1 \cap P_j^2 \neq \emptyset}} m_1(P_i^1) m_2(P_j^2)$$

We write $Bel_1 \oplus Bel_2$ for the belief function induced by $m_1 \oplus m_2$. The following example illustrates the use of the combination rule.

Example 5 Consider a relational schema consisting of the relation *Owner(per#, veh#, money_paid)* (introduced in Example 1). We note that all physical schemas for relation *Owner*, $P^{Owner} = \{p_1, p_2, p_3, ..., p_{20}\}$ are listed in Table 1.

Let us assume that the workload W defined on the schema is such that the percentage of modifications on *veh#*, $Ch_W(veh\#)$, is 15%. This fact induces the execution of rule k_2 (see Section 5), which neither supports a secondary index on *veh#* nor an ordering on *veh#*. Suppose this results in the following bpa:
$m_2(\{p_1, p_2, p_4, p_6, p_9, p_{11}, p_{17}, p_{19}\}) = m_2(P_1^2) = 0.6$ and $m_2(P^{Owner}) = 0.4$.

Suppose that another rule, e.g., k_3, results in (a secondary index on *veh#*):
$m_3(\{p_3, p_5, p_7, p_8, p_{10}, p_{12}, p_{18}, p_{20}\}) = m_3(P_1^3) = 0.9$ and $m_3(P^{Owner}) = 0.1$.

A third rule, e.g., k_4, supports hashing on the attributes *veh#* an d *per#* with the following bpa: $m_4(\{p_9, p_{10}, p_{11}, p_{12}\}) = m_4(P_1^4) = 0.6$, $m_4(\{p_{13}, p_{14}, p_{15}, p_{16}\}) =$

$m_4(P_2^4) = 0.3$, and $m_4(P^{Owner}) = 0.1$.

Combining rules k_2 and k_3, which are conflicting, results in Figure 5. The normalization constant is 0.46 and the combined bpa is: $m_2 \oplus m_3(P_1^3) = 0.36/0.46 = 0.78$ $m_2 \oplus m_3(P_1^2) = 0.06/0.46 = 0.13$ $m_2 \oplus m_3(P^{Owner}) = 0.04/0.46 = 0.09$

The combination of $m_2 \oplus m_3$ with m_4 can be carried out in the same way, and gives the following results.

$m_2 \oplus m_3 \oplus m_4(\{p_{10}, p_{12}\}) = 0.64$	$Bel(\{p_{10}, p_{12}\}) = 0.64$	$Pl(\{p_{10}, p_{12}\}) = 0.83$
$m_2 \oplus m_3 \oplus m_4(\{p_9, p_{11}\}) = 0.11$	$Bel(\{p_9, p_{11}\}) = 0.11$	$Pl(\{p_9, p_{11}\}) = 0.20$
$m_2 \oplus m_3 \oplus m_4(P_1^3) = 0.11$	$Bel(P_1^3) = 0.75$	$Pl(P_1^3) = 0.83$
$m_2 \oplus m_3 \oplus m_4(P_1^4) = 0.07$	$Bel(P_1^4) = 0.71$	$Pl(P_1^4) = 0.95$
$m_2 \oplus m_3 \oplus m_4(P_2^4) = 0.04$	$Bel(P_2^4) = 0.04$	$Pl(P_2^4) = 0.05$
$m_2 \oplus m_3 \oplus m_4(P_1^2) = 0.01$	$Bel(P_1^2) = 0.12$	$Pl(P_1^2) = 0.20$
$m_2 \oplus m_3 \oplus m_4(P^{Owner}) = 0.01$	$Bel(P^{Owner}) = 0.99$	$Pl(P^{Owner}) = 0.99$

We note that the normalization constant is 0.73 and due to roundings $Bel(P^{Owner})$ and $Pl(P^{Owner})$ take the value 0.99 in stead of 1.0.

The highest belief, after combining the three rules, is assigned to the set of schemas $\{p_{10}, p_{12}\}$. We note that the high *total* belief in the sets P_1^3, P_1^4 and P^{Owner} is due to the fact that these sets contain the schemas p_{10} and p_{12}. □

The combination of three rules has resulted into the support of several physical schemas with different belief values. If the bpa's assigned to the three knowledge rules are the real bpa values, there is a high belief that a good physical schema is among the schemas p_{10} and p_{12}. By passing both schemas to the optimizer, we may decide which physical schema of two is the best one.

6 Conclusions & further research

Since the selection of efficient physical schemas is a tough process, there is a practical need for tools that assist database administrators in this process. A significant number of research has been reported to develop such tools. Most of the efforts implicitly apply a few heuristics to avoid the evaluation of all schemas, while database administrators in real-life apply a rich set of heuristics to select physical schemas. Our goal is to exploit this rich set of heuristics in tools for physical database design. Therefore, we have analysed about 60 heuristics used by database administrators in real-life. These heuristics contain a degree of uncertainty and ignorance. We have proposed an approach to model *explicitly* these heuristics into knowledge rules by using the Dempster-Shafer theory, which appeared to be a suitable theory for our purposes. These knowledge rules may be loaded in a knowledge base, which, in turn, can be embedded in physical database design tools as has been demonstrated in [6, 7]. On the basis of our approach, we have implemented a prototype tool [6, 7] and we have compared our results with other approaches. The results obtained by our tool are promising [6, 7].

References

[1] D. Barbara, H. Garcia-Molina, D. Porter, A Probabilistic Relational Data Model, In: *Proc. Int. Conf. on Extending Database Technology, 1990*, 60-74.

[2] F. Bonfatti, D. Maio, P. Tiberio, A Separability-Based Method for Secondary Index Selection in Physical Database Design. *Methodology and Tools for Data Base Design,* S. Ceri (Ed.) North-Holland Publishing Company, the Netherlands, 1983, 149-160.

[3] P.L. Bogler, Shafer-Dempster Reasoning with Applications to Multisensor Target Identification Systems. *IEEE Trans. on Systems, Man, Cybernetics SMC-17(6),* 968-977.

[4] S. Chaudhuri, V. Narasayya, AutoAdmin 'What-if' Index Analysis Utility. In: *Proc. ACM/SIGMOD Int. Conf. on Management of Data, 1998*, 367-378

[5] R. Choenni, H.M. Blanken, S.C. Chang. On the Selection of Secondary Indices in Relational Databases. *Data & Knowledge Engineering 11(3),* 207-233.

[6] R. Choenni, H. Wagterveld, H.M. Blanken, S.C. Chang, ToPhyDe: A Tool for Physical Database Design. In: *Proc. DEXA, Int. Conf. on Databases, Expert Systems, and Applications, 1995*, LNCS 978, 502-511.

[7] R. Choenni, H.M. Blanken, H. Wagterveld, Automating Physical Database Design. In: *Handbook of Data Management 1998,* B. Thuraisingham (Ed.), Auerbach Pulications, Boston, USA, 295-311.

[8] C.E. Dabrowski, K.J. Jefferson, A Knowledge-Based System for Physical Database Design. NBS Special Publication 500-151, Gaithersburg, USA, February 1988.

[9] A.P. Dempster, Upper and Lower Probabilities Induced by a Multi-Valued Mapping. *Annals Math. Stat. 38,* 1967, 325-339

[10] D. Dubois, H. Prade, An Approach to Approximate Reasoning Based on the Dempster-Shafer Rule of Combination. *Int. J. of Expert Systems 1(1),* 1987, 67-85.

[11] S. Finkelstein, M. Schkolnick, P. Tiberio, Physical Database Design for Relational Databases. *ACM Trans. on Database Systems 13(1),* 91-128.

[12] J.Y. Halpern, R.Fagin, Two Views of Belief: Belief as Generalized Probability and Belief as Evidence. *Artificial Intelligence 54(3),* 275-317.

[13] D.E. Knuth, *The Art of Computer Programming Vol. 3, Sorting and Searching.* Addison Wesley Publishing Company, Kent, Great Britain, 1973.

[14] S.K. Lee, Imprecise and Uncertain Information in Databases. In: *Proc. of the 8th Int. Conf. on Data Engineering, 1992,* 328-335.

[15] S.K. Lee, An Extended Relational Database Model for Uncertain and Imprecise Information. In: *Proc. of the 18th Int. Conf. on Very Large Database, 1992,* 211-349

[16] Z. Li, L. Uhr, Evidential Reasoning in a Computer Vision System. *Uncertainty in Artificial Intelligence 2,* J.F. Lemmer, L.N. Kanal (Eds), North-Holland Publishing Company, the Netherlands, 403-412.

[17] J.R. Quinlan, INFERNO: A Catious Approach to Uncertain Inference. *The Computer Journal 26(3),* 255-269.

[18] S. Rozen, D. Shasha, A Framework for Automating Database Design, In: *Proc. of the 17th Int. Conf. on Very Large Database, 1991,* 401-411.

[19] G. Shafer, *A Mathematical Theory of Evidence,* Princeton University Press, Princeton, USA, 1976.

[20] Whang, K., Wiederhold, G., Sagalowicz, D., Separability - An approach to physical database design, *Proc. Int. Conf. on Very Large Databases, 1981.* 487-500.