Reihe Informatik
2 / 1996

Indexing a Fuzzy Database Using the Technique of Superimposed Coding – Cost Models and Measurements

Birgit Boss Sven Helmer

Indexing a Fuzzy Database Using the Technique of Superimposed Coding – Cost Models and Measurements

Birgit Boss[1] [2]

Forschungszentrum Informatik (FZI)
Haid-und-Neu-Str. 10-14
76131 Karlsruhe
Germany

http://www.fzi.de

Sven Helmer

Universität Mannheim
Lehrstuhl für Praktische Informatik III
68131 Mannheim
Germany

helmer@pi3.informatik.uni-mannheim.de

Abstract

Recently, new applications have emerged that require database management systems with uncertainty capabilities. Many of the existing approaches to modelling uncertainty in database management systems are based on the theory of fuzzy sets. High performance is a necessary precondition for the acceptance of such systems by end users. However, performance issues have been quite neglected in research on fuzzy database management systems so far. In this article they are addressed explicitly. We propose new index structures for fuzzy database management systems based on the well known technique of superimposed coding together with detailed cost models. The correctness of the cost models as well as the efficiency of the index structures proposed is validated by a number of measurements on experimental fuzzy databases.

Keywords

Information Storage and Retrieval, Fuzzy Database Management Systems, Fuzzy Object Oriented Database Management Systems, Index Structures, Cost Analysis, Superimposed Coding

1 Introduction

Recently, new applications have emerged that require database management systems with uncertainty capabilities, e.g. heterogeneous database environments, multimedia databases, imputation and knowledge discovery, knowledge base management systems [14], and design environments with an underlying database management system (DBMS) [1]. Many of these applications assume an object-oriented DBMS (ooDBMS) as a basis due to its rich modelling capabilities. Although we assume an object-oriented system, our results are not limited to ooDBMSs.

[1]Part of the work has been funded by the EC within the ESPRIT project 7364 JESSI-Common-Frame

[2]New address from February 1996: Robert Bosch, Forschung und Vorausentwicklung (FV/SLD), Kleyerstr. 94, 60326 Frankfurt, Germany, e-mail: boss@da0245.fr.bosch.de or birgit.boss@fr.bosch.de

Many of the existing approaches to modelling uncertainty in DBMSs are based on the theory of fuzzy sets. See [4, 5] and [10] for recent overviews on fuzzy DBMSs (fDBMSs). They mainly address fuzzy relational DBMSs since fuzzy ooDBMSs are still a current topic of research. Although the need for handling uncertainty in DBMSs has already been recognized earlier, commercial DBMSs are slowly – if at all – incorporating corresponding capabilities. The main reason lies in the fact, that, on one hand, high performance is a necessary precondition for the acceptance of such systems by end users, and, on the other hand, performance issues have been quite neglected in research on fDBMSs so far [14].

With our work, we fill this gap by addressing the problem of efficiently querying a fuzzy database. In fact, for the access of fuzzy data and the evaluation of fuzzy queries, the usual index structures are no longer useful due to the expanded internal representation and evaluation of fuzzy attribute values or query predicates [2]. This is also true for querying fuzzy constraints in DBMSs, where a fuzzy constraint is understood as the restriction of the domain of an attribute [7].

In all existing approaches to fDBMSs, a fuzzy attribute value is represented by a single possibility distribution. In practice, however, it is unrealisitic to assume that an attribute can be described by a single fuzzy set, it is rather described by a complex expression of linguistic terms containing conjunction, disjunction and negation mapped onto fuzzy sets. This linguistic description should not be lost, although the complex expression can be represented by a single fuzzy set by applying the corresponding fuzzy operators. We already consider this requirement from applications, e.g. design applications [1], on efficient index structures for fDBMSs in this article. Thus, the representation of an attribute by a single fuzzy set is seen as a special case of this more general case. For more details on the fuzzy object oriented data model underlying this work see [7, 8].

The index structures, we propose in this article, are based on the technique of superimposed coding, which was originally developed for mechanical retrieving systems in libraries [13]. Roberts made the first use of this concept for partial match retrieval in computer systems [17].

The remainder of this article is organized as follows. In section 2, we introduce the access principles for fuzzy data bases. Then, the basics of superimposed coding are introduced as far as necessary to describe the indexing principles. In section 4 we present two different index structures based on superimposed coding. Section 5 covers the cost models for those structures. The results of practical measurements and the analysis of the cost models are presented in section 6. The article concludes with a summary in section 7.

1.1 Related Work

Most previous approaches with respect to fuzzy queries have addressed the case that the level-cuts of the fuzzy attribute values as well as of the fuzzy query predicates are characterized by a closed interval, and that there is a linear order imposed on the attribute's domain. This is also true for the little existing work on index structures for fDBMSs, [2] and [3]. We, instead, address the case that the domain of the attributes is finite. Further assumptions with respect to the possibility distributions are not made. Our work has in common with the work presented in [2] that it makes use of existing indexing techniques in DBMSs, and that it uses the same basic indexing principles. These are presented in the next section. However, our work goes far beyond that in [2] by not only providing

a concept for index structures but by additionally providing detailed cost models and measurements to prove the correctness of the cost models and the efficiency of the index structures.

In most cases concerning the use of superimposed coding only superset predicates are discussed, since only partial match queries are considered. It has been shown recently, that the technique of superimposed coding is suitable for fast set comparisons including super- and subset predicates in ooDBMSs [12]. In the context of fuzzy retrieval we need subset as well as intersection predicates. Therefore, we supplement the technique of superimposed coding by adding the evaluation of intersection set predicates.

2 Queries and access principles in fuzzy databases

2.1 Fuzzy sets and fuzzy attribute values

As already mentioned, the underlying model is object oriented. Let us consider an attribute T of an object class C taking values in Ω. Let the objects o_i be instances of the class C, with $1 \leq i \leq n$. The value of a (fuzzy) attribute T of an object o_i, $T(o_i)$, corresponds to a linguistic description composed out of a set of given linguistic terms $F_1, ..., F_m$ containing conjunction, disjunction and negation. A linguistic term F_l, $1 \leq l \leq m$, corresponds to the fuzzy set F_l described by its characteristic function $= \mu_{F_l}$ defined on the domain of the attribute Ω: $\mu_{F_l} : \Omega \mapsto [0, 1]$. Let $\pi_{T(o_i)}(\omega)$ measure the possibility that $\omega \in \Omega$ is the correct value of $T(o_i)$. We refer to $\pi_{T(o_i)}$ as the *restriction* of the object. $\pi_{T(o_i)}(\omega)$ does not have to be normalized, i.e. a value of $\omega \in \Omega$ for which $\pi_{T(o_i)}(\omega) = 1$ may not exist. We use the terms 'fuzzy set' and 'possibility distribution' synonymously in this article.

We denote an α-level-cut of a fuzzy set F by $L_\alpha(\mu_F)$, with $0 \leq \alpha \leq 1$. Hence, the support and core of a fuzzy-set are represented by $L(\mu_F)$ and $L_1(\mu_F)$, respectively. If strict level-cuts are meant, we include a "greater than" symbol, like $L_{>\alpha}(\mu_F)$. Several different operators can be applied to fuzzy sets. We are using the following definitions: $\mu_{F_i \wedge F_j} = \min(\mu_{F_i}, \mu_{F_j})$ (conjunction), $\mu_{F_i \vee F_j} = \max(\mu_{F_i}, \mu_{F_j})$ (disjunction), and $\mu_{\neg F_i} = 1 \Leftrightarrow \mu_{F_i}$ (negation).

2.2 Queries and access principles

We use the same basic indexing scheme as [2]. It aims at providing selective access to the objects, which possibly (necessarily) satisfy a fuzzy query predicate Q. A query Q consists of a fuzzy set defined by μ_Q and a threshold α. The threshold indicates that objects o_i have to satisfy Q possibly (necessarily) to at least the degree α, i.e. $\Pi(Q/o_i) \geq \alpha$ ($N(Q/o_i) \geq \alpha$). $\Pi(Q/o_i)$ and $N(Q/o_i)$ denote the possibility measure and necessity measure, respectively, extended to fuzzy events as introduced by [16].

We encountered difficulties already experienced by [16] and [19]. Namely, $\Pi(Q/o_i)$ and $N(Q/o_i)$ are no fuzzy measures, when abandoning the normalization of the possibility distributions. Nevertheless, we use the term measure for $\Pi(Q/o_i)$ and $N(Q/o_i)$, due to the close similarity to fuzzy measures. We redefine $N(Q/o_i)$ in a way to preserve as much properties of fuzzy measures as possible. [1]

[1] If we assumed the fuzzy sets to be normalized, we would have $N(Q/o_i) \geq \alpha \Leftrightarrow L_{>1-\alpha}(\pi_{T(o_i)}) \subseteq$

$$\Pi(Q/o_i) \geq \alpha \quad \Leftrightarrow^{def} \quad L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset \tag{1}$$
$$N(Q/o_i) \geq \alpha \quad \Leftrightarrow^{def} \quad \emptyset \subset L_{>1-\alpha}(\pi_{T(o_i)}) \subseteq L_\alpha(\mu_Q) \wedge L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset \tag{2}$$

One important property that still holds is $\Pi(Q/o_i) \geq N(Q/o_i)$, i.e. given the same threshold value, the set of objects necessarily satisfying the predicate is always included in the set of objects possibly satisfying it.

Let us consider that the index entries are the supports (cores) of the attribute values. In terms of the entries, *weakened access principles* can be deduced:

$$\Pi(Q/o_i) \geq \alpha \quad \Rightarrow \quad L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset \tag{3}$$
$$N(Q/o_i) \geq \alpha \quad \Rightarrow \quad L_1(\pi_{T(o_i)}) \subseteq L_\alpha(\mu_Q) \tag{4}$$

Due to the weakened access principles there are objects that do not satisfy the query Q, but the cores or supports of which do. These objects are called *false level-cut drops*. These false drops have to be sorted out in a separate step.

2.3 False level-cut probability

In this section we present the probability that an object turns out to be a false level-cut drop. We distinguish between several subcases, depending on which fuzzy measure is used in the evaluation of the query. For the following calculations we assume the values of the fuzzy sets to be uniformly distributed among all values of Ω.

False core drops

The probability that the core of an object satisfies the query Q weakly via the necessity measure and the object itself does not is

$$\begin{aligned} & d_{core\subseteq} =^{def} \\ & Pr(L_{>1-\alpha}(\pi_{T(o_i)}) = \emptyset \vee L_{>1-\alpha}(\pi_{T(o_i)}) \not\subseteq L_\alpha(\mu_Q) \vee L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) = \emptyset | L_1(\pi_{T(o_i)}) \subseteq L_\alpha(\mu_Q)) \\ & = 1 - \frac{\begin{pmatrix} |L_\alpha(\mu_Q)| - |L_1(\pi_{T(o_i)})| \\ |L_{>1-\alpha}(\pi_{T(o_i)})| - |L_1(\pi_{T(o_i)})| \end{pmatrix}}{\begin{pmatrix} |\Omega| - |L_1(\pi_{T(o_i)})| \\ |L_{>1-\alpha}(\pi_{T(o_i)})| - |L_1(\pi_{T(o_i)})| \end{pmatrix}} \end{aligned} \tag{5}$$

for $0 < |L_{>1-\alpha}(\pi_{T(o_i)})| \leq |L_\alpha(\mu_Q)|$, otherwise $d_{core\subseteq} = 1$.

The derivation of the formula is found in appendix B.

False support drops

The probability that the support of an object satisfies the query Q weakly via the possibility measure and the object itself does not is for $|L(\pi_{T(o_i)})| \Leftrightarrow |L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| \geq |L_\alpha(\pi_{T(o_i)})|$ (for details see appendix B):

$L_\alpha(\mu_Q)$ [2].

$$
\begin{aligned}
d_{support\cap} &=^{def} Pr(L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) = \emptyset | L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset) \\
&= \frac{\left(\begin{array}{c} |L(\pi_{T(o_i)})| \Leftrightarrow |L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| \\ |L_\alpha(\pi_{T(o_i)})| \end{array} \right)}{\left(\begin{array}{c} |L(\pi_{T(o_i)})| \\ |L_\alpha(\pi_{T(o_i)})| \end{array} \right)}
\end{aligned} \tag{6}
$$

Otherwise $d_{support\cap} = 0$.

Since all objects satisfying a query predicate necessarily also do so possibly, the probability that an object satisfies the predicate weakly via the possibility measure, but does not satisfy the query Q necessarily is also of interest (for details see appendix B), for $0 < |L_{>1-\alpha}(\pi_{T(o_i)})| \leq |L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)|$ and $0 < |L_\alpha(\pi_{T(o_i)})|$:

$$
\begin{aligned}
&d_{support\subseteq} =^{def} \\
&Pr(L_{>1-\alpha}(\pi_{T(o_i)}) = \emptyset \vee L_{>1-\alpha}(\pi_{T(o_i)}) \not\subseteq L_\alpha(\mu_Q) \vee L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) = \emptyset | L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset) \\
&= 1 - \frac{\left(\begin{array}{c} |L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| \\ |L_{>1-\alpha}(\pi_{T(o_i)})| \end{array} \right)}{\left(\begin{array}{c} |L(\pi_{T(o_i)})| \\ |L_{>1-\alpha}(\pi_{T(o_i)})| \end{array} \right)}
\end{aligned} \tag{7}
$$

Otherwise $d_{support\subseteq} = 1$.

3 Superimposed Coding

In this section the basic concepts of superimposed coding are illustrated. As can be seen in (3), we need to support intersection predicates besides the support of subset predicates.

3.1 Encoding sets

Let K_i be the *set of key values* for each object o_i. K_i is composed of the elements in the support (core) of the possibility distribution representing the value of the attribute T. The formation of a superimposed code word *(scw)*, S_i, for an object o_i involves the transformation of the values contained in K_i into binary code words (*bcw*'s), which are bit-strings of length b consisting of exactly k binary 1's and b-k binary 0's. The scw of the attribute value is obtained by superimposing (inclusive ORing) the bcw's generated for o_i [17, 18].

A superimposed code word, S_Q, for a query Q is obtained in a similar fashion. We are looking for all objects, whose value on the attribute T satisfies the fuzzy query predicate Q with the threshold α. For the set of key values of a query the α-level-cut of μ_Q is chosen: $K_Q = L_\alpha(\mu_Q)$.

We illustrate the technique of superimposed coding by giving an example. Let *B(S)* denote the positions of bits set to '1' in the signature S, $b = 5$, and $k = 2$, and $L_1(\pi) = \{3, 7\}$. Say, '3' is mapped into the bcw 11000, and '7' is mapped into the bcw 10010, then, by superimposing, we gain 11010 as the signature for {3,7}, *B(11010)* = {*1,2,4*}.

3.2 False signature drop probability

All objects, the signatures of which satisfy the query Q, are called *drops*. The objects that do not satisfy the query predicate, but the signatures of which do, are called *false signature drops*. False signature drops exist, because by hashing the data elements and superimposing them, it is possible that different sets are mapped onto the same signatures. After extracting all drops from the database, all false drops must be eliminated in a second step. For the sets of key values mentioned above, we can deduce:

$$K_i \subseteq K_Q \quad \Rightarrow \quad B(S_i) \cap B(S_Q) = B(S_i) \tag{8}$$

$$K_i \cap K_Q \neq \emptyset \quad \Rightarrow \quad |B(S_i) \cap B(S_Q)| \geq k \tag{9}$$

The application of superimposed coding is only sensible, if the number of false drops can be held low and more importantly, the comparison of the signatures is more efficient than the direct comparison of the objects with the query set. The latter condition is fulfilled in our case: executing bit-operations is clearly faster than comparing sets element by element. We proceed by discussing the false signature drop probability, that is the probability that an object becomes a false signature drop.

Let the set describing the query K_Q include r_q elements, the sets describing the data objects K_i include on the average $\overline{r_i}$ elements, which are encoded using bitstrings of length b in which exactly k bits are set, $\overline{w}_q$ denotes the average number of bits set in the query signature.

Then, the probability $d_{f\subseteq}$ that an object turns out to be a false drop with respect to a subset predicate is approximately [17]:

$$d_{f\subseteq}(b, k, r_q, \overline{r_i}) \approx (1 \Leftrightarrow e^{-\frac{k}{b} r_q})^{k \cdot \overline{r_i}} \tag{10}$$

Since this probability was already cited and derived in many other articles easy to obtain, we do not give the derivation of it, but focus on queries with intersection predicates. We approximate the corresponding false drop probabilitiy $d_{f\cap}$ by

$$d_{f\cap}(b, k, r_q, \overline{r_i}) \approx 1 \Leftrightarrow \sum_{j=0}^{k-1} \left(\begin{array}{c} \overline{w}_q \\ j \end{array} \right) \cdot (1 \Leftrightarrow (1 \Leftrightarrow \frac{k}{b})^{\overline{r_i}})^j \cdot (1 \Leftrightarrow \frac{k}{b})^{\overline{r_i} \cdot (\overline{w}_q - j)} \tag{11}$$

Its derivation is found in appendix A.

Discussion on optimization of the parameters k and b to yield a small false drop probability can be found, e.g., in [12], [17] for subset predicates, and, for intersection predicates, in [11].

3.3 Superimposed coding and access principles

Superimposed coding can be used to create an efficient index structure supporting access principles in fuzzy databases. Details on how this is done can be found in the next section, we just outline the main idea in the remainder of this section.

Assume, that the signatures S_i and S_Q of the object o_i and the query Q are given, and that the set of key values of o_i refers to the *support* of the fuzzy attribute value. Then we have for an access via the possibility measure (see formulas (3) and (9)):

$$\Pi(Q/o_i) \geq \alpha \quad \Rightarrow \quad |B(S_i) \cap B(S_Q)| \geq k \tag{12}$$

Now assume, the set of key values refers to the *core* of the fuzzy attribute value. Then we have for an access via the necessity measure (see formulas (4) and (8)):

$$N(Q/o_i) \geq \alpha \quad \Rightarrow \quad B(S_i) \cap B(S_Q) = B(S_i) \tag{13}$$

4 Index structures

Instead of using the supports and cores themselves as index entries we use the signatures of the supports and cores. Two variants of index structures based on superimposed coding are presented, the sequential signature file (short SSF) and the compressed signature file (short CSF). We have also analyzed bit-slice signature files and compared them to SSF and CSF index structures. We do not include the results (see [11]) in our paper, because we can only confirm the comparisons of (non-fuzzy) sequential signature files and bit-slice signature files already done by [12] and [17], for example, and no novel aspects were discovered.

4.1 Sequential signature file (SSF)

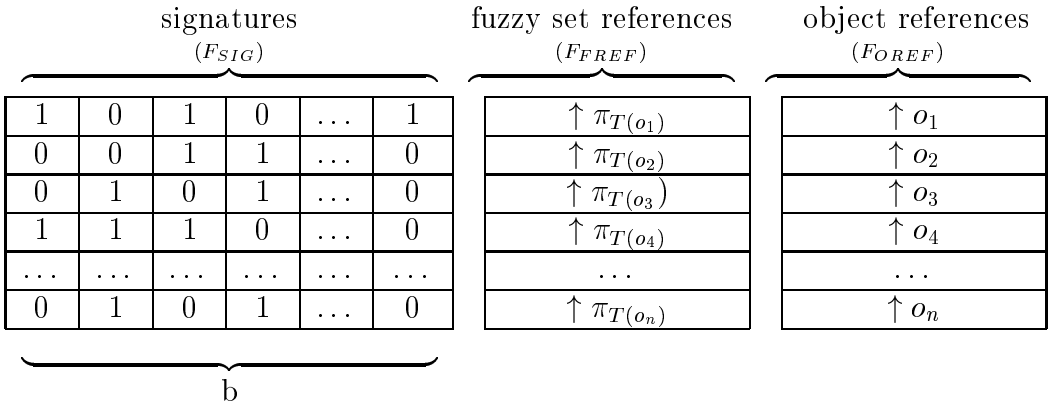

Figure 1: Sequential signature file index structure (SSF)

We explain the construction of a sequential signature index structure on the basis of figure 1. On the left hand side of the figure we have a file composed of signatures. These signatures are derived by encoding the supports (cores) of the restrictions of all objects, when access via the possibility measure (necessity measure) shall be supported. There are two more files, one containing the references to the restrictions of the objects and another one composed of references to the objects. The references correspond to unique object identifiers (OID), as usual in ooDBMSs. All files are organized sequentially. A signature

and its associated fuzzy set reference and object reference are stored at the same position within the files, allowing fast random access.

Given a query Q asking for all objects that possibly (necessarily) satisfy the query fuzzy predicate to at least the degree α, the matching objects are determined in four steps as follows.

1. Construct the signature S_Q of the query predicate Q (see section 3.1).

2. Determine all matching signatures, via possibility measure:

$$T_\Pi = \{j | S_j \in F_{SIG} \wedge |B(S_j) \cap B(S_Q)| \geq k\} \tag{14}$$

   via necessity measure:

$$T_N = \{j | S_j \in F_{SIG} \wedge B(S_j) \cap B(S_Q) = B(S_j)\} \tag{15}$$

3. Eliminate all restrictions which do not truly satisfy the predicate Q, via possibility measure:

$$M_\Pi = \{j \in T_\Pi | L_\alpha(\pi_{T(o_j)}) \cap L_\alpha(\mu_Q) \neq \emptyset\} \tag{16}$$

   via necessity measure:

$$M_N = \{j \in T_N | L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset \wedge \emptyset \subset L_{>1-\alpha}(\pi_{T(o_i)}) \subseteq L_\alpha(\mu_Q)\} \tag{17}$$

4. Collect all objects that are described by fuzzy sets which were not eliminated in previous steps. Via possibility measure:

$$R_\Pi = \{o_j | j \in M_\Pi\} \tag{18}$$

   via necessity measure:

$$R_N = \{o_j | j \in M_N\} \tag{19}$$

The weakened access principles and superimposed coding are also called *filters*. A filter is a mechanism, that does a rough preselection on objects, ensuring that no valuable information (i.e. objects, that satisfy the query) is lost. Instead of applying the filters sequentially, we eliminate all false drops (the ones caused by the superimposed coding, the others caused by the weakened access principles) in a single step. This adds to the efficiency of the index.

Encoding the cores to support queries via the necessity measure as described above can only be used efficiently, when most of the cores of the possibility distributions contain at least one element. We recommend to preselect objects in step 2. by an access via the possibility measure regardless of the measure demanded by the query. No relevant objects are lost in this way, because all objects satisfying a query predicate Q necessarily also do so possibly. When eliminating the false drops, the measure corresponding to the query is applied. A drawback is the higher false drop probability $d_{support\subseteq}$ now applied instead of $d_{core\subseteq}$ for an access via the necessity measure (see section 2.3).

4.2 Compressed signature file (CSF)

In the SSF index there exists an entry in the index files for each and every object, regardless of objects sharing the same fuzzy set. The compressed signature file (CSF) index structure we present in this section considers this redundancy explicitly. We assume that the number m of fuzzy sets defined for an application is much smaller than the number n of objects in the database. Negated fuzzy sets are treated as fuzzy sets of their own. Then, there is only one entry for each of the m different fuzzy sets in the signature and fuzzy set reference files. Instead of a single object reference for each fuzzy set, we have a set of OIDs now, because several objects may share a fuzzy set. The object reference file is replaced by a file containing references to OID sets (see figure 2).

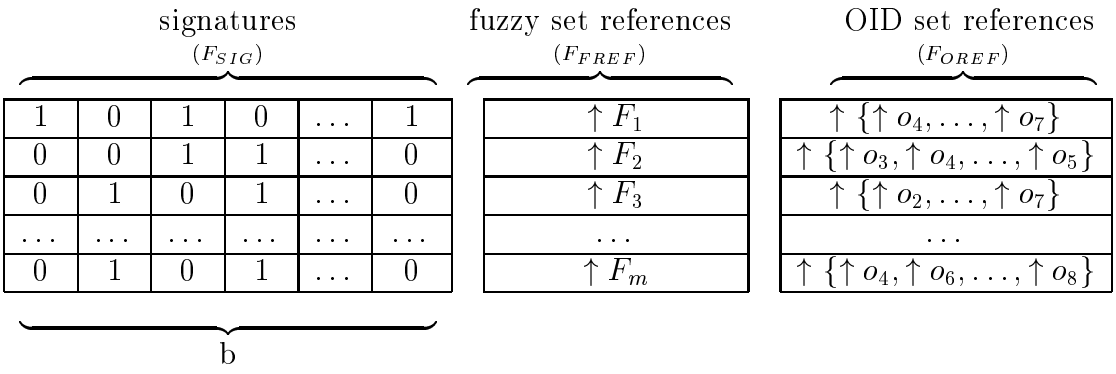

Figure 2: Compressed signature file index structure (CSF)

As already mentioned an object is described by a restriction composed of several fuzzy sets. In the SSF index the objects are indexed directly by their corresponding restriction, i.e. a query predicate can only address the restriction as is. The CSF index supports the evaluation of more complex queries. This is achieved by breaking up the restrictions into their individual fuzzy sets, thereby allowing access to the individual fuzzy sets. This allows for new types of queries. However, their discussion is out of the scope of this article, for details see [7]. An object identifier may now appear in different object references sets. Object o_4, e.g., is described by at least the fuzzy sets F_1, F_2, and F_m.

In the CSF index the query predicate Q is not directly compared to the restriction of an object, but to the individual fuzzy sets of the restriction. Formula (4) does not hold anymore, because $L_1(\pi_{T(o_i)}) \subseteq L_\alpha(\mu_Q)$ does not imply $L_1(\mu_{F_j}) \subseteq L_\alpha(\mu_Q)$ for each fuzzy set F_j that $\pi_{T(o_i)}$ is composed of. Therefore the preselection of the objects has to be done by an access via the possibility measure in the CSF index regardless of the measure demanded by the query, whereas in the SSF index this was just a recommendation.

Given a query Q the matching objects are determined in five steps. The first four steps remain the same as for queries via the possiblity measure for the SSF-index, a fifth step is added [6]:

1. Construct the signature S_Q of the query predicate Q.

2. Determine all matching signatures,

$$T = \{j | S_j \in F_{SIG} \wedge |B(S_j) \cap B(S_Q)| \geq k\} \tag{20}$$

3. Eliminate all fuzzy sets F_j which do not truly satisfy the predicate Q,

$$M = \{j \in T | L_\alpha(\mu_{F_j}) \cap L_\alpha(\mu_Q) \neq \emptyset\} \tag{21}$$

4. Collect all objects that are described by fuzzy sets which were not eliminated in previous steps. The objects are now accessed through sets consisting of object identifiers. Let $fRef(F_j)$ be the set of OID referenced by the fuzzy set F_j, i.e., the set of OIDs references in line j of the OID set reference file F_{OREF}.

$$O = \{o_i | j \in M \wedge \uparrow o_i \in fRef(F_j)\} \tag{22}$$

5. An object o_i satisfies the predicate Q possibly (necessarily) to at least the degree α, iff its restriction satisfies Q, i.e., via possibility measure:

$$R_\Pi = \{o_i \in O | L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset\} \tag{23}$$

via necessity measure:

$$R_N = \{o_i \in O | L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset \wedge \emptyset \subset L_{>1-\alpha}(\pi_{T(o_i)}) \subseteq L_\alpha(\mu_Q)\} \tag{24}$$

5 Cost models

In this section we describe the cost models for storage, retrieval, and update for the SSF and CSF index structures. We assume that the performance of the access facilities mainly depends on the I/O costs, i.e., the number of page accesses. Each object and fuzzy set can be directly accessed via its OID. Buffering is not taken into account, so in order to reach an object it is assumed that at least one page access is necessary.

Table 1 summarizes the parameters that are used throughout the article.

5.1 Cost model for the SSF index

In this section we present the cost model for the SSF index. The storage costs S_{SSF} are defined by:

$$S_{SSF} = S_{SIG} + S_{FREF} + S_{OREF} \tag{25}$$

S_{SIG} denotes the size of the signature file and can be calculated as follows:

$$S_{SIG} = \left\lceil \frac{n \cdot b}{P \cdot y} \right\rceil \text{ pages} \tag{26}$$

P is the number of bytes per page, y the number of bits per byte, and *oid* the size of an object identifier in byte. The value of b, the number of bits in a signature, depends on the desired false signature drop probability (see 3.2) and whether the supports or cores are encoded. This depends on whether queries via the possiblity or necessity measure shall

n	number of objects in the database
m	number of different fuzzy sets
g	number of fuzzy sets describing an object (CSF)
P	number of bytes per page
y	number of bits per byte
oid	size of an object identifier in byte
c	number of right signature drops ($c = f + f_f$)
$\overline{\gamma}$	expected number of right signature drops per page
c_f	number of false signature drops
$\overline{\gamma}_f$	expected number of false signature drops per page
f	number of right level-cut drops ($f = c - f_f$)
$\overline{\phi}$	expected number of right level-cut drops per page
f_f	number of false level-cut drops ($f_f = c \cdot d_{[support \cap \mid support \subseteq]}$)
a	number of objects satisfying the query
$\overline{\alpha}$	expected number of objects satisfying the query per page
O_f	number of page accesses to get a fuzzy set
O_a	number of page accesses to get an object

Table 1: Parameters used in cost models

be supported. S_{FREF} and S_{OREF} represent the size of the files containing the references to the fuzzy sets and the references to the objects.

$$S_{FREF} = \left\lceil \frac{n \cdot oid}{P} \right\rceil \text{ pages} \tag{27}$$

$$S_{OREF} = \left\lceil \frac{n \cdot oid}{P} \right\rceil \text{ pages} \tag{28}$$

The retrieval costs $C_{FR[\Pi|N]_{SSF}}$ for an access via the [possibility|necessity] measure are described by

$$C_{FR[\Pi|N]_{SSF}} = S_{SIG} + L_{FREF} + O_f \cdot (c + c_f) + L_{OREF} + O_a \cdot a \tag{29}$$

and can be explained by considering the steps taken during the evaluation of a query (see section 4.1). First of all, all matching signatures have to be found. For that reason the signature file has to be traversed completely leading to the cost S_{SIG}.

In the next step all false drops have to be eliminated. For that reason we have to access the fuzzy sets, but unlike the signature file not all pages of the fuzzy set reference file have to be referenced. We will only fetch the fuzzy sets, whose signatures satisfy the query. Therefore, the number of I/O operations on the fuzzy set reference file is limited to

$$L_{FREF} = \left\lceil S_{FREF} \cdot \min\{\overline{\gamma} + \overline{\gamma}_f, 1\} \right\rceil \tag{30}$$

with $\overline{\gamma}$ being the expected number of right drops per page given by

$$\overline{\gamma} = \frac{c}{S_{FREF}} \tag{31}$$

(where c is the number of right signature drops) and with $\overline{\gamma}_f$ being the expected number of false drops per page given by

$$\overline{\gamma}_f = \frac{c_f}{S_{FREF}} \tag{32}$$

(where c_f is the number of false signature drops).

We assume that the number of right and false drops is uniformly distributed among the pages of the fuzzy set reference file.

The number of false signature drops c_f can be calculated by using the false drop probabilities given in section 3.2.

$$c_f = (n \Leftrightarrow c) \cdot d_{f[\cap|\subseteq]} \tag{33}$$

Then we have $L_{FREF} + O_f \cdot (c + c_f)$ page accesses for fetching all fuzzy sets belonging to the matching signatures, with O_f being the cost for accessing a fuzzy set.

Finally, $L_{OREF} + O_a \cdot a$ is the cost for accessing the objects, with O_a being the number of page accesses necessary to reach an object. The page accesses for the object identifier file can be calculated analogous to (30).

$$L_{OREF} = \left\lceil S_{OREF} \cdot \min\{\overline{\phi}, 1\} \right\rceil \tag{34}$$

With f being the number of right level-cut drops, which can be calculated by using the false level-cut drop probability (see section 2.3): $f = c \cdot (1 \Leftrightarrow d_{[support \cap | core \subseteq]})$, the expected number of right level-cut drops per page is $\overline{\phi} = \frac{f}{S_{FREF}}$. The number of right level-cut drops is identical to the number of objects matching the query, i.e. $f = a$, because in the last step of the evaluation (when accessing the objects) there are no more false drops. Thus, we also have $\overline{\phi} = \overline{\alpha}$ with $\overline{\alpha}$ being the expected number of right object drops per page.

When inserting a new object, an item is added to all three files (signature, fuzzy set reference, object reference), resulting in three page accesses.

$$C_{FI[\Pi|N]_{SSF}} = 3 \tag{35}$$

When deleting an object, the reference to the object is marked as deleted. The deletion of the object and the adjustment of the index take place during times of low user activity. This leads to a slightly higher false drop probability, however. The average costs for deleting an object are

$$\overline{C}_{FD[\Pi|N]_{SSF}} = \frac{S_{OREF}}{2} \tag{36}$$

5.2 Cost model for the CSF index

In this section the cost model for the CSF index is presented. It bears resemblance to the SSF index, but there are some changes. The storage costs S_{CSF} are calculated as follows:

$$S_{CSF} = S_{SIG} + S_{FREF} + S_{OREF} + S_{OSET} \tag{37}$$

The size of the signature file S_{SIG} and the fuzzy set reference file S_{FREF} is smaller than S_{SIG} and S_{FREF} for the SSF index, because there is only one entry for each fuzzy set in the index. The formulas for S_{SIG}, S_{FREF} and S_{OREF} remain the same, except that n must be substituted by m, the number of entries in the signature and fuzzy reference file. Hereby, it is assumed that a reference to a set of object identifiers has the same size as a reference to a single object. We assume that every object is described by the same number of fuzzy sets g, i.e., a fuzzy set F_j describes on the average $\frac{n \cdot g}{m}$ objects.

Additionally, we have costs S_{OSET} for storing the sets of OIDs themselves:

$$S_{OSET} = \left\lceil \frac{n \cdot g \cdot (oid + Set_{elem}) + m \cdot Set_{info}}{P} \right\rceil \text{ pages} \tag{38}$$

with S_{elem} being the administrative cost for a single set element, and S_{info} being the administrative cost for the whole set.

The retrieval costs $C_{FR[\Pi|N]_{CSF}}$ for an access via the [possibility|necessity] measure in a CSF index are described by

$$\begin{aligned} C_{FR[\Pi|N]_{CSF}} &= S_{SIG} + L_{FREF} + O_f \cdot (c + c_f) \\ &\quad + L_{OREF} + f \cdot \left(O_s + \frac{n \cdot g}{m} \cdot (O_a + O_f) \right) \end{aligned} \tag{39}$$

and can be explained by considering the evaluation steps described in section 4.2. The first three steps resemble those for the SSF index, so the costs can be calculated analogous to the SSF index (see formula (29)). We concentrate on the differences here. Again, n has to be substituted by m in formula (33) to gain the correct results. Since we first check whether an object possibly satisfies the query in the CSF index, we have $c_f = (m \Leftrightarrow c) \cdot d_{f\cap}$ independently of whether we have a query via the possibility measure or via the necessity measure.

For the CSF index the number of right level-cut drops is not equal to the number of objects satisfying the query ($f \neq a$), because an object may be described by several fuzzy sets. Similar to the SSF index $f = c \cdot (1 \Leftrightarrow d_{[support\cap|support\subseteq]})$, only with the difference that $\pi_{T(o_i)}$ is substituted in (6) and (7) by the corresponding possibility distribution μ_{F_j} of the fuzzy set being checked.

After accessing and comparing the fuzzy sets the OID sets corresponding to the right level-cut drops have to be traversed. The costs O_s for this task are (at least) the number of page accesses equal to the number of pages occupied by the sets: $\left\lceil \frac{\frac{n \cdot g}{m}(oid + Set_{elem}) + Set_{info}}{P} \right\rceil$ per OID set. Getting the objects themselves costs us O_a page accesses. Checking, whether they truly satisfy the query predicate is done by obtaining the restrictions. The reference

to the restriction is assumed to be stored with each object and this leads to additional access costs of O_f.

When updating a CSF index, several cases have to be distinguished, depending on which assumptions are made. Presenting them would go beyond the scope of this paper, because each case is in turn split into several subcases. For details see [7].

6 Benchmarking and cost analysis

In this article we confine ourselves to the description of the implementation and analysis of the CSF index, because the SSF index can be seen as a special case of the CSF index (with the number of fuzzy sets describing an object $g = 1$ and the total number of fuzzy sets m equal to n, the total number of objects). We present the results of the various benchmark runs done with experimental databases in this section.

6.1 Implementation

The index structures were implemented in OBST, a freeware ooDBMS developed at the Forschungszentrum Informatik (FZI) in Karlsruhe [9, 15]. We tried to keep the implementation as simple as possible, i.e., we did not attach great importance to optimizing. Figure 3 shows the implementation of a CSF index structure in OBST.

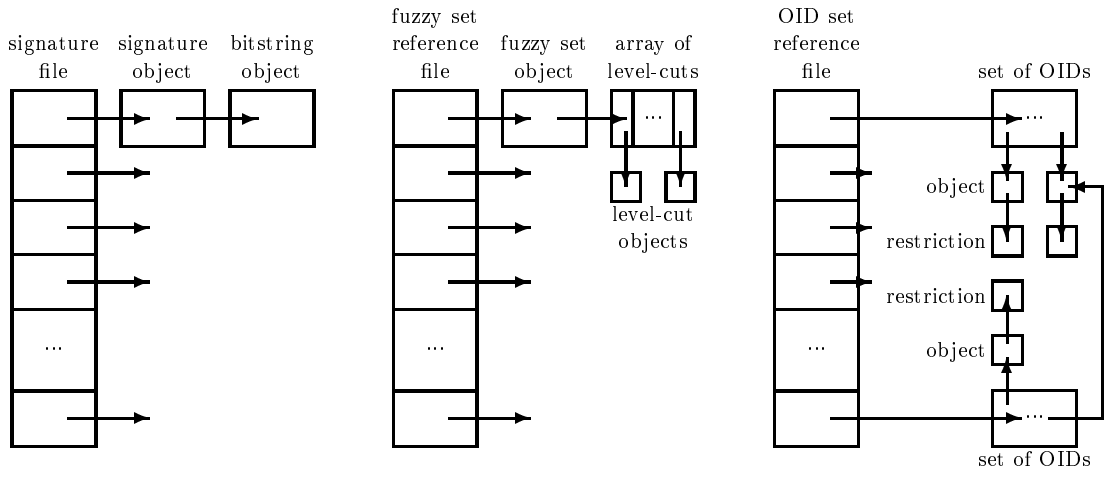

Figure 3: Implementation of a CSF index structure

There are several parameters we used in the benchmark runs worth mentioning (see table 2):

- For the domain Ω of the fuzzy sets and query predicates we choose the closed interval $[0, 1000]$. The elements are of the type *sos_Int* (i.e. integer) and have a length of 32 bits.
- The implementation of the fuzzy sets is done by explicitly storing a finite number of level-cuts of the fuzzy sets. We have avoided redundancy by only storing $L_{j_i} \setminus L_{j_{i+1}}$ for each level-cut of the fuzzy set with $0 < j_i < j_{i+1} \leq 1$.

- In our benchmark runs a fuzzy set is represented by eight level-cuts $(\frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \ldots, 1)$. The core contains three elements and in each subsequent level-cut two elements are added. A fuzzy set needs about 480 bytes storage (S_{fuzzy}).

- For the benchmark runs, r_q, the number of elements in the query fuzzy set, is equal to 5.

- Each of the n objects is associated with g fuzzy sets selected randomly from the m fuzzy sets. For the benchmark runs g is equal to 2.

- Each object references its restriction. The restriction of an object is the fuzzy set resulting from the conjunction (using the min operator) of all g fuzzy sets describing the object[2]. A restriction needs about 270 bytes storage (S_{rst}).

- The length b of a signature is 512 bits, the number of bits set in a single codeword k is equal to 2. These values guarantee an optimal false drop probability $d_{f\cap}$ under the benchmark conditions.

- The signature file is larger than the sum of all signatures, because each signature is stored as an object (the representation of the signatures was not optimized). Therefore we decided to consider the costs for comparing two signatures. 1.6 msec is the average time measured for the comparison of two signatures E_s.

- Table 3 shows the average time E_f and $E_{f'}$ needed to check if two fuzzy sets intersect. The reasons for considering CPU time are given in section 6.3.

- Each object is stored with its object identifier of 16 bytes. Altogether an object needs about 80 bytes storage (S_{obj}). Each (fuzzy) set is stored as an object, so $Set_{info} = 16$ bytes. Within each set short pointers (4 bytes) are used as references, so $Set_{elem} = 4$ bytes.

The benchmarks were run on a SPARCstation 10 with 32 Mbyte main memory and 218 Mbyte virtual memory. The size of a page was 8 Kbyte with an average access time ($\overline{SZ}$) of 10 msec. The experimental databases, which were constructed with 1000, 2000, 3000, 4000, and 5000 objects and 500 fuzzy sets (high compression rate) and 2000 fuzzy sets (low compression rate), were stored on a local disk. The thresholds 0.25, 0.5, 0.75, and 1.0 and the access via the possibility measure were used in the queries. The benchmarks were run at night, because the machine was not a stand-alone machine.

6.2 The influence of thresholds

First of all we want to discuss the influence of the query threshold on the query evaluation. Figure 4 shows an example for a database with 4000 objects. The costs were calculated using (39).

It can be clearly seen that the costs for evaluating a query are high when choosing low thresholds and low when choosing high thresholds regardless of whether high or low compression rates are used. The reason for this is that comparing low level-cuts (containing

[2]That is, we did not consider disjunctions.

	no units
n	1000,2000,3000
	4000,5000
m	500,2000
g	2
$\|\Omega\|$	1001
Ω	{0,...,1000}
b	512
k	2
r_i	17
r_q	5
$d_{f\cap}$	0.13
$d_{support\cap}$	0.34

	byte
P	8192
y	32
oid	16
S_{sig}	82
S_{fuzzy}	480
S_{obj}	80
S_{rst}	270
Set_{info}	16
Set_{elem}	4

	msec
E_s	1.6
E_f,$E_{f'}$	see separate
	table
$\overline{SZ}$	10

Table 2: Parameters chosen, determined, estimated or measured for benchmarks

level-cut	average time for testing, if query predicate Q intersects with	
	fuzzy set descr. object E_f	restriction of object $E_{f'}$
0.25	47 msec	25 msec
0.5	28 msec	22 msec
0.75	18 msec	17 msec
1.0	8 msec	10 msec

Table 3: Average costs for checking intersections

many elements) of two fuzzy sets with each other is more expensive than comparing high level-cuts (containing few elements), when using the possibility measure as seen in table 3. The larger false-drop probability of high level-cuts does not compensate for this effect. Another reason is the fact, that the probability that two sets intersect is the larger the larger the two sets are (assuming a finite domain and a uniform distribution).

6.3 Validating the cost model

Calling methods in ooDBs can be very time consuming in regard to CPU time. After comparing the benchmark results with our cost models, we decided that in our experimental databases the CPU time should not be neglected. So, we extended the formula for the retrieval cost (39) by the costs for level-cut and signature comparisons. The costs for method calling is given in msec, so all parameters given in number of page accesses have to be multiplied with $\overline{SZ}$, the average page access time. Each page storing fuzzy sets and objects only had to be read once, because afterwards the page was held in the system buffer (this was due to the fact that the experimental databases were quite small). Our earlier cost model on the other hand assumes that for every right drop a page access is necessary to get the associated entry in the fuzzy set reference and OID reference file, which is the case for large databases. The modified formula is

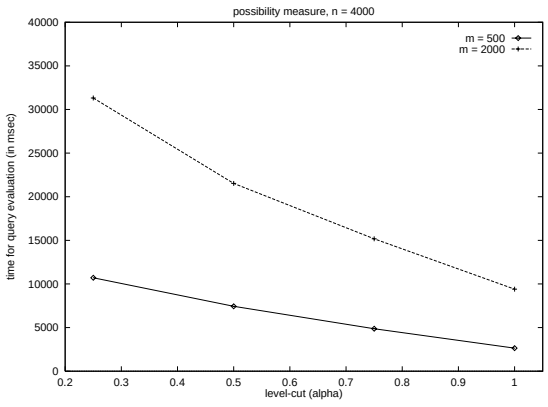

Figure 4: Evaluation costs according to cost model for CSF

$$
\begin{aligned}
C_{FR\Pi_{CSF}} \quad = \quad & (S_{SIG} + L_{FREF} + L_{OREF} + \\
& O_f \cdot \min\{(c + c_f), m \cdot S_{fuzzy}\} + \\
& O_s \cdot \min\{f, m \cdot S_{OSET}\} + \\
& O_a \cdot \min\{f \cdot \frac{n \cdot g}{m}, n \cdot S_{obj}\} \\
& O_f \cdot \min\{f \cdot \frac{n \cdot g}{m}, n \cdot S_{rst}\}) \cdot \overline{SZ} + \\
& E_s \cdot m + E_f \cdot (c + c_f) + E'_f \cdot f \cdot \frac{n \cdot g}{m}
\end{aligned} \tag{40}
$$

In figure 5 the comparison of the calculations using the formula (40) with the actual values of the benchmark runs are given. All parameters presented in 6.1 were considered in the calculations.

As can be seen in figure 5 formula (40) gives good approximations of the benchmark runs. The best approximation is for high compression rate and high α (only 1% deviation), the worst for low compression rate and very low α (up to 20% deviation). Further investigation may lead to the introduction of additional parameters thereby improving the cost model even further.

6.4 Examining the efficiency

In order to prove the efficiency of the index structures proposed, we also ran benchmarks on the database using the index structures without the signature files (the results are shown in figure 6). The following analysis is made exemplary for a database with 500 fuzzy sets and for $\alpha = 0.25$ and $\alpha = 0.75$ (see figure 7). It can be clearly seen that an access via an index structure with signature files is much faster than an access via an index structure without signature files. In the case of a level-cut of 0.25 the access is up to four times faster, in the case of a level-cut of 0.75 it is still two times faster. As seen in table 3, the comparison of the 0.25-level-cuts of two fuzzy sets is more expensive

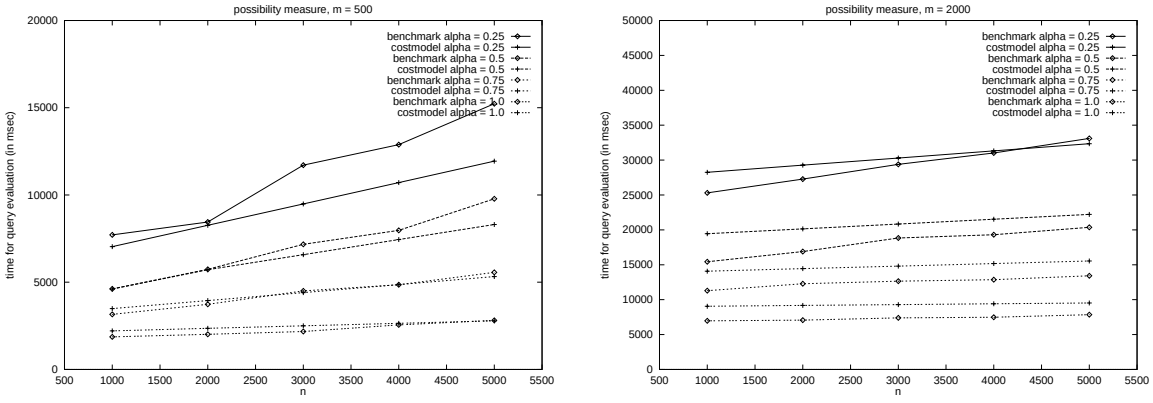

Figure 5: Comparison of cost model with benchmark runs

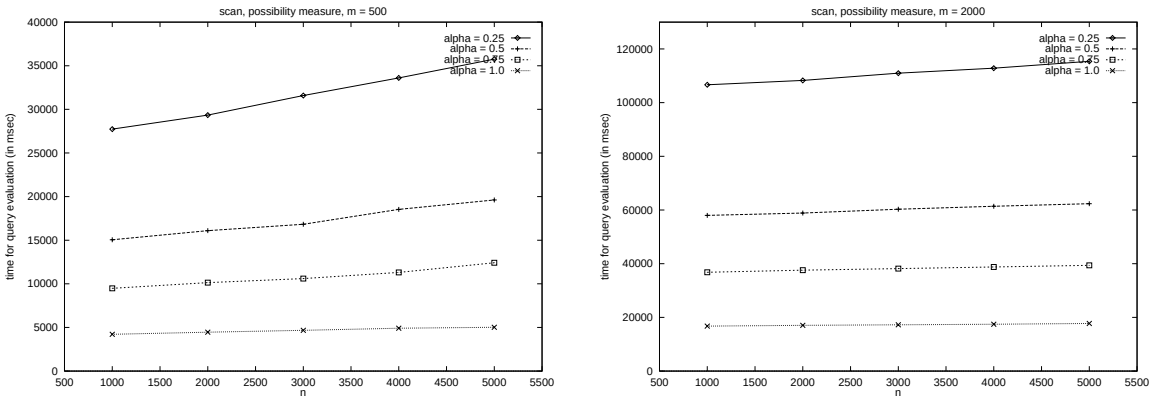

Figure 6: Benchmark runs without signature files

than the comparison of the 0.75-level-cuts. An index structure with signature files lowers the number of fuzzy set comparisons, therefore high comparison costs are significantly improved by signature files. The compression rate of an index also influences the number of needed comparisons. A signature file is not as effective on an index structure with a high compression rate (500 fuzzy sets) as on an index structure with a low compression rate (2000 fuzzy sets), because the compression of the index structure already lowers the number of comparisons even without a signature file.

6.5 Comparison of CSF index with SSF index

We already showed in [11] that the cost model for the SSF index approximates the evaluation costs very accurately, the largest deviations were 5%. We compare the CSF index with the SSF index by using the results of the benchmark runs for the CSF index

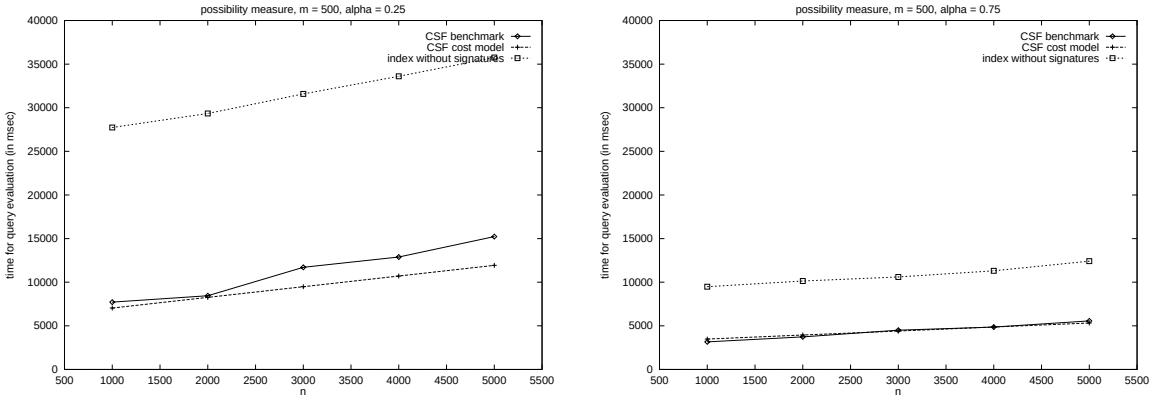

Figure 7: Comparison of benchmark runs with and without signature files

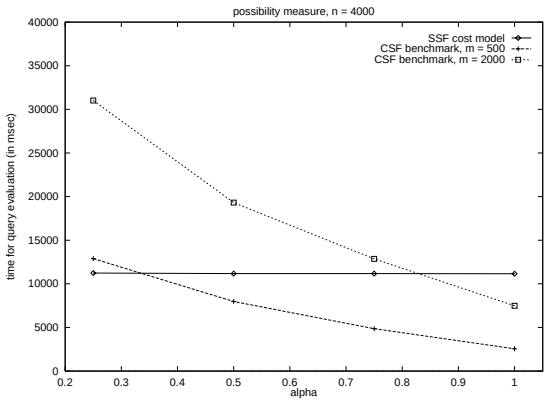

Figure 8: Comparison of CSF index with SSF index

and the cost model for the SSF index. Figure 8 shows an example for 4000 objects and 500 describing fuzzy sets. For the SSF index all fuzzy sets describing an object are combined to a restriction, which in turn is used as entry in the signature and fuzzy set files. The restrictions contain very few elements in this example (the core was usually empty and the support included one element), so the query evaluation costs for the SSF index are almost constant. When a high compression rate can be achieved ($m = 500$ in our example), the CSF index is clearly the better choice. For $\alpha = 1$ a query can be evaluated more than four times faster than in the SSF index. When the compression rate is low ($m = 2000$ in our example), the CSF index is only better for high level-cuts.

7 Summary and outlook

Current research on fuzzy databases concentrates on abstract concepts like data models and query languages. Publications about the implementation aspects of fuzzy databases such as query optimization and index structures are few and far between. High performance of database management systems, however, is of great importance to end users, therefore we began to fill this gap by presenting index structures suitable for fuzzy databases. The index structures described in this paper are based on the well-known technique of superimposed coding. The calculations with our cost models and the conducted benchmarks have led us to the conclusion that this technique is a remarkable improvement for efficiently querying fuzzy databases.

We have tested several variants of index structures based on superimposed coding, thereby analyzing the strengths and weaknesses of each one. In this paper the most important variants, the SSF and CSF index, were illustrated. The index structures have not been tested in real-life applications, yet, but with the help of our cost models the task of deciding which variant to use for a given application has been made much easier.

Acknowledgments

We would like to thank Guido Moerkotte for his helpful hints on an earlier version of this paper.

A False signature drop probabilities

In this appendix we present the derivation of the false signature drop probability for intersection predicates. The false drop probability can be expressed by

$$d_{f\cap}(b,k,r_q,\overline{r_i}) = Pr(|B(S_i)\cap B(S_Q)| \geq k | L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) = \emptyset) \tag{41}$$

This is the probability that at least k bits of the w_q bits set in S_Q are also set in S_i under the assumption that $L(\pi_{T(o_i)})$ and $L_\alpha(\mu_Q)$ do not intersect. Let us now look closer at the probabilities that a certain bit in a signature is set to 0 or set to 1. When superimposing $\overline{r_i}$ codewords, the probability that a certain bit is set to 0 is [17]

$$(1-\frac{k}{b})^{\overline{r_i}} \tag{42}$$

The probability that a certain bit is set to 1 is therefore

$$1-(1-\frac{k}{b})^{\overline{r_i}} \tag{43}$$

When inspecting more than one bit in a signature, for instance j bits, then the probability that all j bits are set to 0 can be approximated by (assuming that j and k are sufficently small [17])

$$(1-\frac{k}{b})^{\overline{r_i}\cdot j} \tag{44}$$

The probability that all j bits are set to 1 can be approximated by [17]

$$(1-(1-\frac{k}{b})^{\overline{r_i}})^j \tag{45}$$

The probability that at least k bits of the w_q bits are set in the signature S_i can be approximated by a binomial distribution [11]:

$$\begin{aligned} d_{f\cap}(b,k,r_q,\overline{r_i}) &= \sum_{j=k}^{w_q} Pr(|B(S_i)\cap B(S_Q)|=j|L(\pi_{T(o_i)})\cap L_\alpha(\mu_Q)=\emptyset) \\ &\approx \sum_{j=k}^{\overline{w_q}} \left(\begin{array}{c} w_q \\ j \end{array}\right)\cdot(1-(1-\frac{k}{b})^{\overline{r_i}})^j\cdot(1-\frac{k}{b})^{\overline{r_i}\cdot(w_q-j)} \\ &\approx 1-\sum_{j=0}^{k-1}\left(\begin{array}{c} \overline{w}_q \\ j \end{array}\right)\cdot(1-(1-\frac{k}{b})^{\overline{r_i}})^j\cdot(1-\frac{k}{b})^{\overline{r_i}\cdot(\overline{w}_q-j)} \end{aligned} \tag{46}$$

The average number $\overline{w}_q$ of bits set in a query signature S_Q is equal to [17]

$$\overline{w}_q = b\cdot(1-(1-\frac{k}{b})^{r_q}) \tag{47}$$

B False level-cut drop probabilities

The false level-cut drop probabilities were presented in section 2.3. In this section we show how these results were obtained. We assume, that the elements in the level-cuts of the fuzzy-sets are uniformly distributed. Furthermore, we assume that for $x, y \in \mathbf{N}_0$ special cases of the binomial coefficient are defined as:

$$\left(\begin{array}{c} x \\ y \end{array}\right) = \left\{\begin{array}{ll} 1 & \text{for } y=0 \\ 0 & \text{for } y>x \end{array}\right. \tag{48}$$

B.1 False core drops

The probability that the core of an object satisfies the query predicate Q necessarily and weakly, but the object itself does not satisfy the query necessarily, is:

$$\begin{aligned} &d_{core\subseteq} =^{def} \\ &Pr(L_{>1-\alpha}(\pi_{T(o_i)})=\emptyset \vee L_{>1-\alpha}(\pi_{T(o_i)})\not\subseteq L_\alpha(\mu_Q) \vee L_\alpha(\pi_{T(o_i)})\cap L_\alpha(\mu_Q)=\emptyset | L_1(\pi_{T(o_i)})\subseteq L_\alpha(\mu_Q)) \\ &= 1-Pr(\emptyset\subset L_{>1-\alpha}(\pi_{T(o_i)})\subseteq L_\alpha(\mu_Q)\wedge L_\alpha(\pi_{T(o_i)})\cap L_\alpha(\mu_Q)\neq\emptyset | L_1(\pi_{T(o_i)})\subseteq L_\alpha(\mu_Q)) \end{aligned} \tag{49}$$

The level-cut $L_{>1-\alpha}(\pi_{T(o_i)})$ contains $|L_{>1-\alpha}(\pi_{T(o_i)})| - |L_1(\pi_{T(o_i)})|$ more elements than the core $L_1(\pi_{T(o_i)})$. $L_{>1-\alpha}(\pi_{T(o_i)})$ is a subset of $L_\alpha(\mu_Q)$, iff these additional elements are also members of the query set. There are $\left(\begin{array}{c} |L_\alpha(\mu_Q)|-|L_1(\pi_{T(o_i)})| \\ |L_{>1-\alpha}(\pi_{T(o_i)})|-|L_1(\pi_{T(o_i)})| \end{array}\right)$ different possible ways,

out of $\begin{pmatrix} |\Omega| - |L_1(\pi_{T(o_i)})| \\ |L_{>1-\alpha}(\pi_{T(o_i)})| - |L_1(\pi_{T(o_i)})| \end{pmatrix}$, to choose the additional elements from Ω so that $L_{>1-\alpha}(\pi_{T(o_i)}) \subseteq L_\alpha(\mu_Q)$. We are now able to describe the false drop probability by using a hypergeometric distribution. For $|L_{>1-\alpha}(\pi_{T(o_i)})| \neq 0$ and $|L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| \neq 0$

$$d_{core\subseteq} = 1 - \frac{\begin{pmatrix} |L_\alpha(\mu_Q)| - |L_1(\pi_{T(o_i)})| \\ |L_{>1-\alpha}(\pi_{T(o_i)})| - |L_1(\pi_{T(o_i)})| \end{pmatrix}}{\begin{pmatrix} |\Omega| - |L_1(\pi_{T(o_i)})| \\ |L_{>1-\alpha}(\pi_{T(o_i)})| - |L_1(\pi_{T(o_i)})| \end{pmatrix}} \tag{50}$$

otherwise, if $|L_{>1-\alpha}(\pi_{T(o_i)})| = 0$ or $|L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| = 0$, then $d_{core\subseteq} = 1$.

B.2 False support drops

Next we inspect the probability that the support of an object satisfies the query predicate Q possibly and weakly, but the object itself does not satisfy the query possibly.

$$d_{support\cap} =^{def} Pr(L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) = \emptyset | L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset) \tag{51}$$

The level-cut $L_\alpha(\pi_{T(o_i)})$ includes $|L_\alpha(\pi_{T(o_i)})|$ elements, that can also be found in the support $L(\pi_{T(o_i)})$. To be a false support drop the fuzzy set of an object o_i has to have a level-cut $L_\alpha(\pi_{T(o_i)})$ in which no element can be found that is also in the query set $L_\alpha(\mu_Q)$. There are $\begin{pmatrix} |L(\pi_{T(o_i)})| \\ |L_\alpha(\pi_{T(o_i)})| \end{pmatrix}$ different ways to choose $|L_\alpha(\pi_{T(o_i)})|$ elements from the support to construct $L_\alpha(\pi_{T(o_i)})$. To be a false drop, none may be chosen that are also in the query set. Assuming a hypergeometric distribution, we get

$$\begin{aligned} d_{support\cap} &= \frac{\begin{pmatrix} |L(\pi_{T(o_i)})| - |L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| \\ |L_\alpha(\pi_{T(o_i)})| \end{pmatrix} \begin{pmatrix} |L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| \\ 0 \end{pmatrix}}{\begin{pmatrix} |L(\pi_{T(o_i)})| \\ |L_\alpha(\pi_{T(o_i)})| \end{pmatrix}} \\ &= \frac{\begin{pmatrix} |L(\pi_{T(o_i)})| - |L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| \\ |L_\alpha(\pi_{T(o_i)})| \end{pmatrix}}{\begin{pmatrix} |L(\pi_{T(o_i)})| \\ |L_\alpha(\pi_{T(o_i)})| \end{pmatrix}} \qquad (52) \\ &\leq \frac{\begin{pmatrix} |L(\pi_{T(o_i)})| - 1 \\ |L_\alpha(\pi_{T(o_i)})| \end{pmatrix}}{\begin{pmatrix} |L(\pi_{T(o_i)})| \\ |L_\alpha(\pi_{T(o_i)})| \end{pmatrix}} \qquad (53) \end{aligned}$$

We look at one more case, the probability that the support of an object satisfies the query predicate possibly and weakly, but the object does not satisfy the query necessarily.

$$\begin{aligned} &d_{support\subseteq} =^{def} \\ &Pr(L_{>1-\alpha}(\pi_{T(o_i)}) = \emptyset \vee L_{>1-\alpha}(\pi_{T(o_i)}) \not\subseteq L_\alpha(\mu_Q) \vee L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) = \emptyset | L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset) \\ &= 1 - Pr(\emptyset \subset L_{>1-\alpha}(\pi_{T(o_i)}) \subseteq L_\alpha(\mu_Q) \wedge L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset | L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q) \neq \emptyset) \qquad (54) \end{aligned}$$

The level-cut $L_{>1-\alpha}(\pi_{T(o_i)})$ is a subset of the query set $L_\alpha(\mu_Q)$, iff all elements chosen from the support $L(\pi_{T(o_i)})$ to construct $L_{>1-\alpha}(\pi_{T(o_i)})$ are also in $L_\alpha(\mu_Q)$. No element may be chosen that cannot be found in $L_\alpha(\mu_Q)$. With the hypergeometric distribution we get for $|L_{>1-\alpha}(\pi_{T(o_i)})| \neq 0$ and $|L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| \neq 0$

$$
\begin{aligned}
d_{support\subseteq} &= 1 - \frac{\left(\begin{array}{c} |L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| \\ |L_{>1-\alpha}(\pi_{T(o_i)})| \end{array} \right) \left(\begin{array}{c} |L(\pi_{T(o_i)})| - |L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| \\ 0 \end{array} \right)}{\left(\begin{array}{c} |L(\pi_{T(o_i)})| \\ |L_{>1-\alpha}(\pi_{T(o_i)})| \end{array} \right)} \\
&= 1 - \frac{\left(\begin{array}{c} |L(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| \\ |L_{>1-\alpha}(\pi_{T(o_i)})| \end{array} \right)}{\left(\begin{array}{c} |L(\pi_{T(o_i)})| \\ |L_{>1-\alpha}(\pi_{T(o_i)})| \end{array} \right)} \qquad (55)
\end{aligned}
$$

otherwise, if $|L_{>1-\alpha}(\pi_{T(o_i)})| = 0$ or $|L_\alpha(\pi_{T(o_i)}) \cap L_\alpha(\mu_Q)| = 0$, then $d_{support\subseteq} = 1$.

References

[1] K. Abramowicz, B. Boss, V. Hovestadt, J. Mülle, R. Sturm, and P. C. Lockemann. Konsistenzüberwachung in objektorientierten Datenbanksystemen – Eine Anforderungsanalyse anhand der Entwurfsbereiche Architektur und Schiffbau. In G. Lausen, editor, *Datenbanksysteme in Büro, Technik und Wissenschaft*, Informatik Aktuell, pages 302–321, Dresden, March 1995. GI, Springer Verlag. In german.

[2] P. Bosc and M. Galibourg. Indexing Principles for a Fuzzy Data Base. *Information Systems*, 14(6):493–499, 1989.

[3] P. Bosc, M. Galibourg, and G. Hamon. Fuzzy Querying with SQL: Extensions and Implementation Aspects. *Fuzzy Sets and Systems*, 28:333–349, 1988.

[4] P. Bosc and J. Kacprzyk, editors. *Fuzzy Sets and Possibility Theory in Database Management Systems.* Physica-Verlag, A Springer-Verlag Company, Heidelberg, 1995.

[5] P. Bosc and H. Prade. An Introduction to Fuzzy Set and Possibility Theory-based Approaches to the Treatment of Uncertainty and Imprecision in Data Base Management Systems. In A. Motro and P. Smets, editors, *Proc. of Uncertainty Management in Information Systems: from Needs to Solutions Workshop (UMIS'94).* Catalina, California, April 1994.

[6] B. Boss. An Index Based on Superimposed Coding for a Fuzzy Object Oriented Database. In L. Hall, H. Ying, R. Langari, and J. Yen, editors, *Proc. Int. Joint Conf. of The North American Fuzzy Information Processing Society Biannual Conference, The Industrial Fuzzy Control and Intelligent Systems Conference, The NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic NAFIPS/IFIS/NASA'94*, pages 289–290, San Antonio, Texas, December 1994.

[7] B. Boss. *Fuzzy-Techniken in objektorientierten Datenbanksystemen zur Unterstützung von Entwurfsprozessen.* PhD thesis, Universität Karlsruhe, Fakultät für Informatik, Karlsruhe, December 1995. In german.

[8] B. Boss. Handling Vague Design Constraints in a Design Environment. In Luciano Faria, editor, *Proc. of the Int. Workshop on Concurrent/Simultaneous Engineering Frameworks and Applications*, pages 321–332, Lisboa, Portugal, April 1995.

[9] E. Casais, M. Ranft, B. Schiefer, D. Theobald, and W. Zimmer. OBST – An Overview. FZI Report 9/92, Forschungszentrum Informatik (FZI), Karlsruhe, June 1992. STONE-Bericht FZI.039.1.

[10] V. Cross. Fuzzy Information Retrieval. *Journal of Intelligent Systems*, 3:29–56, 1994.

[11] S. Helmer. Entwicklung von Kostenmodellen für eine auf Superimposed Coding basierende Zugriffsstruktur für Fuzzy-Datenbanken. Masters thesis, Forschungszentrum Informatik (FZI), Karlsruhe, February 1995. In german.

[12] Y. Ishikawa, H. Kitagawa, and N. Ohbo. Evaluation of Signature Files as Set Access Facilities in OODBs. In *Proc. of ACM SIGMOD International Conf. on Management of Data*, pages 247–256. Washington, DC, USA, May 1993.

[13] H.P. Luhn. Superimposed coding with the aid of randomizing squares for use in mechanical information searching systems. In R.S. Casey, J.W. Perry, M.M. Berry, and A. Kent, editors, *Punched Cards*, pages 492–509. Reinhold, New York, 1958.

[14] A. Motro. Management of Uncertainty in Database Systems. In Won Kim, editor, *Modern Database Systems: The Object Model, Interoperability, and beyond*, chapter 22, pages 457–476. ACM Press, Addison-Wesley, 1995.

[15] The OBject system of STONE – OBST. http://www.fzi.de/dbs/projects/-OBST.html, 1995.

[16] H. Prade and C. Testemale. Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries. *Information Sciences*, 34:115–143, 1984.

[17] C. S. Roberts. Partial-Match-Retrieval via the Method of Superimposed Codes. *Proc. of the IEEE*, 67(12):1624–1642, December 1979.

[18] R. Sacks-Davis. Performance of a Multi-Key Access Method based on Descriptors and Superimposed Coding Techniques. *Information Systems*, 10(4):391–403, 1985.

[19] M. Zemankova and A. Kandel. Implementing Imprecision in Information Systems. *Information Sciences*, 37:107–141, 1985.