Querying the SaintEtiQ summaries – a first attempt

W. A. Voglozin, G. Raschia, L. Ughetto, and N. Mouaddib

Laboratoire d'Informatique de Nantes Atlantique, Université de Nantes
2 rue de la Houssinière, BP 92208, 44322 Nantes Cedex 3, France
{voglozin,raschia,ughetto,mouaddib}@lina.univ-nantes.fr

Abstract. For some years, data summarization techniques have been developed to handle the growth of databases. However these techniques are usually not provided with tools for end-users to efficiently use the produced summaries. This paper presents a first attempt to develop a querying tool for the SaintEtiQ summarization model. The proposed search algorithm takes advantage of the hierarchical structure of the SaintEtiQ summaries to efficiently answer questions such as "how are, on some attributes, the tuples which have specific characteristics?". Moreover, this algorithm can be seen both as a boolean querying mechanism over a hierarchy of summaries, and as a flexible querying mechanism over the underlying relational tuples.

1 Introduction

In order to handle the growth in size of databases, many approaches have been developed to extract knowledge from huge databases. One of these approaches consists in summarizing data (see for example [2, 4, 6, 9, 11]). However, summarization techniques are usually not provided with tools for end-users to efficiently use the summaries. As a consequence, users have to directly interpret the summaries, which is conceivable with a few summaries only. In other cases, tools are necessary.

In this paper, the structured data summarization model SaintEtiQ developed in our research team [11] is considered. SaintEtiQ provides a compact representation of a database, rewriting the tuples by means of linguistic variables [15] defined on each attribute, and classifying them in a hierarchy of summaries. The set of summaries, produced by the process, describes the data in a comprehensible form. Thus, each summary, expressed with fuzzy linguistic labels, symbolizes a concept that exists within the data.

This paper proposes a querying mechanism for users to efficiently exploit the hierarchical summaries produced by SaintEtiQ. The first idea is to query the summaries using the vocabulary of the linguistic variables defined in the summarization process. Although linguistic terms are used in the expression of queries, the querying process is clearly boolean. Since the querying vocabulary is the one used within the summaries, the linguistic terms have become the attribute values in the summaries and query answers contain linguistic terms only.

Then, basic queries such as "how are, on attribute(s) A_k, the tuples which are $d_{i,j}$ on A_i", can be answered very efficiently as the querying process relies on boolean operations. Moreover, the algorithm takes advantage of the hierarchical structure of the summaries, and answers are obtained rapidly. The gain is particularly important in case of a null answer, as only a small part of the summaries hierarchy has to be explored, instead of the entire relation.

Querying the summaries as explained above is interesting in order to rapidly get a rough idea of the properties of tuples in a relation. In case of null answers, it clearly saves time. In other cases, a rough answer is often not enough. Thus, the second idea is to query the database through the summaries, which would be done by retrieving tuples from the summaries obtained as an answer in the previous kind of queries. This process is then related to the "flexible querying of relational databases" trend of research. Indeed, in this case, linguistic terms are used in the expression of queries, and the answer would be composed of tuples from the summarized relation, ranked according to a degree of satisfaction to the query.

The next section describes flexible queries of databases and their features compared to classical queries. It exposes some earlier works done in this field by other researchers. Section 3 presents an overview of the SaintEtiQ model, briefly depicting the representations of summaries and the different steps of the summary building process. It also highlights the distinctive aspects of our approach. Section 4 thoroughly explains how advantage can be taken from the use of the SaintEtiQ summaries hierarchies in a flexible querying process. Expression of queries, selection of summaries and formation of results are then reviewed.

2 Regular databases flexible querying

A flexible querying process operating on relational databases probes the tuples for adequacy to a query using a standard language, namely SQL. According to Larsen [8], the querying process can be divided in three steps: extension of criteria, selection of results and ordering.

The first step uses similarity between values to extend the criteria (sometimes using non-binary operators) and to find potentially interesting results. The second step, namely the selection of results, determines which data will participate in the answer to the query. These data are afterwards referred to by the term "results": the set of all results constitute the answer to a query. The last step (ordering) follows from the extension of criteria. It discriminates among the results on the basis of their relative satisfaction to the query.

The fuzzy set theory is often used in flexible querying (see [3]) because it provides a formal framework to handle the vagueness inherent to natural language. The following works, which are representative of the research on flexibility in database querying, exemplify the use of fuzzy sets. They are essentially characterized by a tuple-oriented processing, the possibility to define new terms and especially, the use of satisfaction degrees, which we have not accomplished yet.

2.1 SQLf

The querying language SQLf, proposed by Bosc and Pivert [1], is an extension of SQL aiming at "introducing fuzzy predicates into SQL wherever possible". An augmentation of both the syntax and semantics of SQL is performed in order to express elements of a query in a fuzzy form. These elements include operators, aggregation functions, modifiers (very, really, more or less), quantifiers (most, a dozen) as well as general description terms such as young or well-paid.

Evaluation of the query may be based on a particular interpretation of the query, for instance fuzzy sets crisp cardinality or Yager's ordered weighted averaging operators [14]. It occurs for each record and yields a grade of membership (of the record to the relation symbolizing the query) which is used to rank the results. An example of query in SQLf is "**select** 10 `dpt` **from** EMPLOYEE **group by** `dpt` **having** *most-of*(`age` = *young*) **are** *well-paid*" where standard SQL keywords are in bold face, and `dpt` and `age` are attributes from a relation named EMPLOYEE. The query selects the 10 departments which have the best satisfaction of the condition "most of the young employees are well-paid".

2.2 FQUERY

FQUERY [7] is an integration of flexible querying into an existing database management system, namely Microsoft Access. The system allows queries with vague predicates expressed through fuzzy sets. Queries may contain linguistic quantifiers and attach different levels of importance to attributes. In a such way of doing, the authors try to apply the *computing with words* paradigm and so, deal with linguistic values, quantifiers, modifiers and relations.

FQUERY uses fuzzy sets for the imprecision aspect and performs a syntax and semantics extension of SQL. Linguistic values and quantifiers are represented as fuzzy sets. The query is assimilated to a fuzzy set resulting from the combination of these sets. Accordingly, each record selected by a classical SQL query, has a matching degree used to rank that record since it indicates how well it corresponds to the query.

2.3 SummarySQL

Developed by Rasmussen and Yager [12], SummarySQL is a fuzzy query language intended to integrate summaries into a fuzzy query. The language can not only evaluate the truth degree of a summary guessed by the user but also use a summary as a predicate in a fuzzy query.

A summary expresses knowledge about the database in a statement under the form "**Q** objects in **DB** are **S**" or "**Q R** objects in **DB** are **S**". **DB** stands for the database, **Q** is a linguistic quantifier and **R** and **S** are summarizers (linguistic terms). One can obtain statements like "**most** people in **DB** are **tall**" or "**most tall** people in **DB** are **heavy**".

Predicates (summaries) and linguistic terms and are fuzzy sets in the expression that represents the selection condition. The expression is evaluated for each

tuple and the associated truth values are later used to obtain a truth value for the summary. SummarySQL is used to determine whether, or to what extent, a statement is true. It can also be used to search for fuzzy rules.

3 Querying the SaintEtiQ summaries

To concentrate flexible queries on database records may lead to prohibitive response times when a large number of records is involved, or when subqueries are expressed. Waiting for an answer for a long time is frustrating, particularly when the query fails.

Database summaries offer a means of significantly reducing the volume of input for processes that require access to the database. The response time benefits from the downsizing. Furthermore, for this querying process, performance does not depend on specific combinations of attributes, i.e., whether the attributes are indexed or not, since these summaries are general indexes for the underlying data [10]. This eliminates possible restrictions due to predefined queries tailored for efficiency.

When querying the summaries, the response time gain is made clearly at the expense of a loss of precision in the answer. This is of no importance when only a rough answer is required. But when more details about the tuples are required, querying the summaries is only a first step: the entire set of relevant tuples can be easily retrieved from the answer summaries. The querying mechanism remains efficient, and there is no loss of precision in the answer. However, the loss is in the querying language expressiveness, since for now only the linguistic variables used to build the summaries hierarchy can be used in the expression of the queries.

3.1 Summaries in SaintEtiQ

The SaintEtiQ model aims at apprehending the information from a database in a synthetic manner. This is done through linguistic summaries structured in a hierarchy. The model offers different granularities, i.e., levels of abstraction, over the data. The steps necessary to build a summary hierarchy are described below.

First, the fuzzy set theory is used to translate records in accordance with a *background knowledge* provided by the user. For each attribute, linguistic variables (which are part of the background knowledge) offer a mapping of the attribute's value to a linguistic label describing that value. For instance, with a linguistic variable for attribute INCOME (figure 1), a tuple value $t.\texttt{income}$ =50,000 is expressed as $t.\texttt{income} = \{1.0/\textit{reasonable}\}$ where 1.0 tells how well the label *reasonable* describes the value '$50,000'. Applying this mapping to each attribute of a relation corresponds to a translation of the initial tuple into another expression called a *candidate tuple*.

Because one initial attribute value may be described by more than one fuzzy label (for instance, $37,000 is described by both modest and reasonable), one

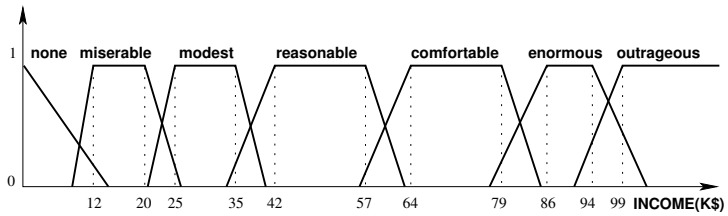

Fig. 1. *Linguistic variable defined on* `INCOME`

initial tuple may yield many translated representations, i.e., many candidate tuples. For a tuple $t = \langle v_1, v_2, \ldots, v_n \rangle$, a candidate tuple ct is under the form $ct = \langle \alpha_1/d_1, \alpha_2/d_2, \ldots, \alpha_n/d_n, \rangle$ where d_i is a fuzzy label and α_i is a satisfaction degree which tells how well v_i is described by d_i.

In a second place, comes a generalization step, allowing to represent fuzzy linguistic labels using more general labels.

Concept formation is the last step of the summary hierarchy building process. Each candidate tuple from each database record is incorporated into a tree and reaches a leaf node. This can be seen as a classification of the candidate tuple. It is important to notice that the tree is modified throughout candidate tuples incorporation: it progressively becomes a complete representation of the data. The way the tree evolves is partially controlled by learning operators. These operators discover new concepts (summaries) when necessary so that the current set of concepts reflects better the data.

For more clarity, let us mention that the tree represents the summary hierarchy and that a node represents a summary. In the hierarchy structure, the level can be associated with the relative proportion of data that is described by a summary: the deeper the summary in the tree (or the lower its level in the hierarchy), the finer the granularity. It follows that the lowest levels contain the most precise and specific summaries. The intensional expression of such summaries is similar to candidate tuples: $z = \langle \alpha_1/d_1, \alpha_2/d_2, \ldots, \alpha_n/d_n, \rangle$. There is only one label per attribute.

Inversely, the root of the tree is the most general summary view one can have over the data. The intensional expression of more general summaries may have one or more multi-labeled attributes. This depends on the labels that are present in the candidate tuples captured by the summaries. An example of such expression is $z = \langle \alpha_{11}/d_{11} + \alpha_{12}/d_{12}, \alpha_2/d_2, \ldots, \alpha_n/d_n, \rangle$.

As far as precision is concerned, the compact expression of a summary induces only an additional processing: for more details about tuples, one has to probe all of the summary's leaf nodes. No precision is lost since a candidate tuple is an exact representation of an initial tuple using the vocabulary given.

3.2 Distinctive features

Our approach to flexible queries uses the SaintEtiQ summaries to answer questions such as "how are objects that have such and such characteristics?" (a few examples of questions can be found in Section 4.1). As the data searched by the querying process is made of summaries, it seems obvious to make results from summaries. Moving a step further, it is plain to retrieve other kinds of results such as candidate tuples or initial relational tuples.

Since one summary can be linked to candidate tuples by simply probing all its children leaves, the sort of results can be changed from summaries to candidate tuples. And one further step links candidate tuples to database records. Because the retrieval of candidate tuples or database records is immediate, in the rest of this document, results are assimilated to summaries only.

Another distinction exists in the classification step of the process: no preference is expressed as there is no evaluation of result quality with respect to the query. All results are shown to the user in a non-discriminated form and it is up to them to determine what results are better than others. As far as summaries are concerned, no ranking is performed.

4 Description of the process

As stated in Section 2, the first step of a database flexible querying process consists in extending criteria. Thanks to linguistic variables defined for each attribute, criteria extension is already performed in SaintEtiQ. From then, we can use binary operators to identify the data to be considered as results. This section deals with all aspects of selection from the expression and meaning of a query to its matching against summaries.

To illustrate the querying process, we use a toy example based on a relation R =(thickness, hardness, temperature). R describes some imaginary steel industry materials over arbitrary features. For each attribute of R, the following descriptors are used:

- thickness: small, thin, medium, thick, huge;
- hardness: malleable, flexible, soft, medium, hard, compact, impenetrable;
- temperature: cold, low, moderated, normal, high, extreme.

4.1 Expression of a query

This approach to flexible querying intends to answer questions such as "how are materials which are thin?" or "how are materials which are high-temperature and medium-hardness?". In the prototype developed for querying, the questions are expressed using a user-friendly interface that composes the corresponding query in an SQL-like language. For the previous questions, the queries formed are respectively:

Q_1: DESCRIBE ON temperature, hardness
WITH thickness IN (thin)
Q_2: DESCRIBE ON thickness
WITH temperature IN (high)
AND hardness IN (medium)

Because an answer, for example "thin materials have a soft hardness and a low temperature", is a description of basic data (summaries, candidate tuples, database records), we consider description as an elementary operation. Embedding the DESCRIBE operator (and other operators from summary-based querying) in an extension of SQL is a future project. For a more formal expression, let:

- S be a set of attributes;
- $\mathsf{R}(\mathsf{S})$ be the relation whose tuples are summarized;
- Q be a query, for instance Q_1 or Q_2;
- A be an attribute appearing in the query ($A \in \mathsf{S}$);
- d be a label (or descriptor) also appearing in the query.

A question explicitly defines some values (thin, high or medium) called *required characters*. In a query, descriptors embody required characters and serve as a basis for determining what data partake in the answer.

A question also defines, sometimes implicitly, the attributes for which required characters exist. The set of these input attributes for a query is denoted by X. The expected answer is a description over the other attributes, whose set is denoted by Y. Without further precision, Y is the complement of X relatively to S: $X \cup Y = \mathsf{S}$ and $X \cap Y = \emptyset$.

Hence a query defines not only a set X of input attributes A_i but also, for each attribute A_i, the set C_{A_i} of its required characters. The set of sets C_{A_i} is denoted by C, as shown in the following example.

Example 1. Let Q_1 and Q_2 be the queries stated above.

For Q_1, X = {thickness}, Y ={hardness, temperature}, $C_{\texttt{thickness}}$ ={thin} and $C = \{C_{\texttt{thickness}}\}$.

For Q_2, X ={hardness, temperature}, Y ={thickness}, $C_{\texttt{hardness}}$ ={medium}, $C_{\texttt{temperature}}$ ={high} and $C = \{C_{\texttt{hardness}}, C_{\texttt{temperature}}\}$.

When users formulate a question, they expect data with some characteristics to be put forward. The meaning of that question becomes arguable when many characteristics are expressed for one attribute or when conditions exist for more than one attribute.

The first case is illustrated by the question "how are materials which are flexible or soft?". Because the database records are one-valued tuples, the question is interpreted as "how are materials which hardness is one of {flexible, soft}?" and not as "how are materials which hardness is both flexible and soft?". The equivalent query for the correct interpretation is Q_3: DESCRIBE ON thickness, temperature WITH hardness IN (flexible, soft), interpreted as the condition *hardness = flexible OR hardness = soft.*

The second case is illustrated by the question "how are thick compact materials?". The querying process should put forward only data that comply with the characterization on both thickness and hardness. This precludes, for instance, thick soft materials and thin compact materials from being selected. The equivalent query for this second question is Q_4: DESCRIBE ON temperature WITH thickness IN (thick) AND hardness IN (compact). The condition of Q_4 is interpreted as *thickness = thick AND hardness = compact*.

4.2 Evaluation of a query

This section deals with matching one particular summary against a query to decide whether it corresponds to that query and can then be considered as a result. The query is transformed into a logical proposition P used to qualify the link between the summary and the query. P is under a conjunctive form in which all descriptors appear as literals. In consequence, each set of descriptors yields one corresponding clause.

Example 2. For question q_5 "how are the materials which are thin or medium-thickness and normal or high-temperature?", the corresponding query is Q_5: DESCRIBE ON hardness WITH thickness IN (thin, medium) AND temperature IN (normal, high).

In this query, $X = \{thickness, temperature\}$, $C_{\texttt{thickness}} = \{thin, medium\}$ and $C_{\texttt{temperature}} = \{normal, high\}$. It follows that $P_5 = (thin \vee medium) \wedge (normal \vee high)$.

Let v be a valuation function. It is obvious that the valuation of P depends on the summary z: a literal d in P is positively valuated ($v(d) = \texttt{TRUE}$) if and only if d appears in z. So we denote by $v(P(z))$ the valuation of proposition P in the context of z.

Let $\mathcal{L}_{A_i}(z)$ be the set of descriptors that appear in z. An interpretation of P relatively to query Q leads to discarding summaries that do not satisfy P. But, as shown in the following example, some summaries might satisfy P and yet not match the intended semantics of the query.

Example 3. Table 1 shows the characteristics of materials covered by a summary z_0 along with z_0 itself. If z_0 is tested for conformance with Q_5 (see example 2), we can see that $v(P_5(z_0)) = \texttt{TRUE}$, but nowhere can one find a material responding to question q_5.

While confronting a summary z with a query Q, three cases might occur:

- **Case 1**: no correspondence. $v(P(z)) = \texttt{FALSE}$. For one attribute or more, z has no required character, i.e., it shows none of the descriptors mentioned in query Q.
- **Case 2**: exact correspondence. The summary being confronted with query Q matches its semantics. It is considered as a result. The following expression holds: $v(P(z)) = \texttt{TRUE}$ and $\forall i,\ \mathcal{L}_{A_i}(z) \subseteq C_i$.

Table 1. *Example of descriptor combination*

	thickness	temperature
ct_1	thin	extreme
ct_2	medium	extreme
ct_3	thick	high
z_0	{thin, medium, thick}	{extreme, high}

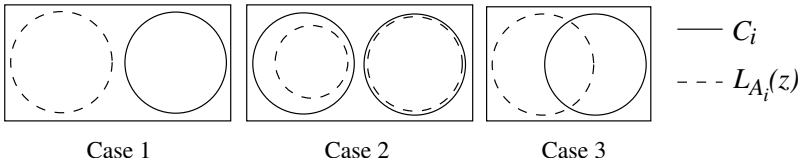

Fig. 2. *Comparison of descriptor sets* $\mathcal{L}_{A_i}(z)$ *and* C_i

– **Case 3**: no decision can be made. There is one attribute A_i for which summary z exhibits one or many descriptors besides those strictly required (i.e., those in C_i): $\exists i, \mathcal{L}_{A_i}(z) - C_i \neq \emptyset$.
Presence of required characters in each attribute of z suggests, but does not guarantee, that results may be found in the subtree starting from z. Exploration of the subtree is necessary to retrieve possible results: for each branch, it will end up in situations categorized by case 1 or case 2. Thus, at worst at leaf level, an exploration leads to accepting or rejecting summaries; the problem of indecision is always solved.

The situations stated above reflect a global view of the confrontation of a summary with a query. They can also be interpreted, from a crisp set point of view, as a combination of comparisons, still involving $\mathcal{L}_{A_i}(z)$ and C_i, concerning one required attribute A_i. Figure 2 shows all comparisons using a set representation with $\mathcal{L}_{A_i}(z)$ symbolized by a dashed circle and C_i by a solid circle.

4.3 Selection algorithm

This section applies the matching procedure from the previous section over the whole set of summaries organized in a hierarchy.

Since the selection should take into account all summaries that correspond to the query, exploration of the hierarchy is complete. The selection (algorithm 1) is based on a depth-first search and relies on a property of the hierarchy: the generalization step in the SaintEtiQ model guarantees that any descriptor that exists in a node of the tree also exists in each parent node. Inversely, a descriptor is absent from a summary's intension if and only if it is absent from all subnodes of this summary. This property of the hierarchy permits branch cutting as soon as it is known that no result will be found. Depending on the query, only a part

of the hierarchy is explored. In any case, all relevant results, and only relevant results, are captured.

Algorithm 1 describes the exploration and selection function with the following assumptions:

- the function returns a list of summaries;
- function Corr symbolizes the matching test reported in Section 4.2;
- operator '+' performs a list concatenation of its arguments;
- function Add is the classical constructor for lists, it adds an element to a list of the suitable type;
- L_{res} is a local variable.

Algorithm 1 Function Explore-Select(z, Q)

```
L_res ← ⟨⟩ {the list for this subtree is empty}
if Corr(z, Q) = indecisive then
  for all child node z_child of z do
    L_res ← L_res+ Explore-Select(z_child, Q)
  end for
else
  if Corr(z, Q) = exact then
    Add(z, L_res)
  end if
end if
return L_res
```

4.4 Classification

The classification step is an aggregation of selected summaries according to their interpretation with respect to proposition P: summaries that have the same required characters on all attributes of the input attributes set X form a class that is denoted by B.

Example 4. Once again query Q_5 from question q_5 is considered. The proposition P_5 induced by Q_5 (see example 2) admits 9 classes that match the following interpretations where only positively valuated literals are shown: {thin, normal}, {thin, high}, {medium, normal}, {medium, high}, {thin, medium, normal}, {thin, medium, high}, {thin, normal, high}, {medium, normal, high}, {thin, medium, normal, high}.

Let z_1 and z_2 be two summaries selected for Q_5. Assume that they are described by "thin" for the thickness and "normal" for the temperature. Then, they belong to the same class {thin, normal}. Had they been described by "medium" for the thickness, "normal" and "high" for the temperature, they would have belonged to the class {medium, normal, high}.

Table 2. *Example of selected summaries*

Summary	`hardness`
z_1	flexible
z_2	soft
z_3	hard
z_4	compact

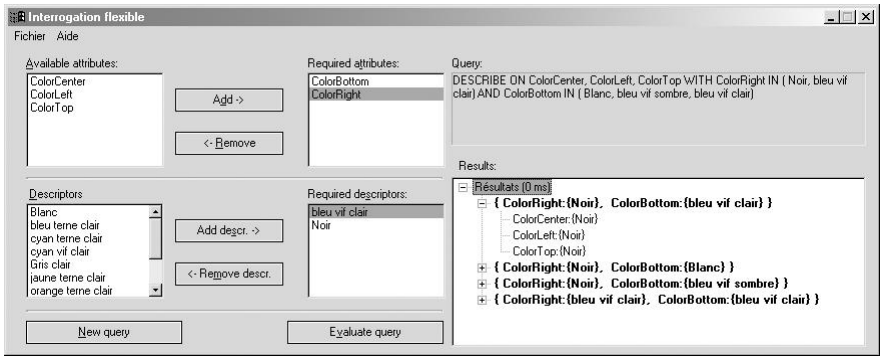

Fig. 3. *Screenshot of the current implementation*

Aggregation of summaries inside a class B is a union of descriptors (see example 5): for each attribute A_i of output set Y, the querying process supplies a set of descriptors. This set of descriptors characterizes summaries that respond to the query through the same logical interpretation (i.e., summaries that show the same labels for input attributes).

Example 5. Table 2 lists summaries selected for class {thin, normal} along with their descriptors for attributes of $Y = \{\text{hardness}\}$. For that class, we obtain an output set $C_{\texttt{hardness}} = \{\text{flexible, soft, hard, compact}\}$.

The use of classes stems from the desire to provide the user with an intensional expression of results. As a response to a query, the process returns a list of classes along with a characterization of the class for each output attribute. The list is interpreted as follows: *while searching for thin or medium-thickness and normal or high-temperature materials, it turned out that:*

- *thin normal-temperature materials are either flexible or soft;*
- *thin high-temperature materials are hard;*
- ...

Figure 3 shows a screenshot of the current implementation of the querying process. The data set used for this specific screenshot describes the images of an image repository based on their color characteristics. For the expression of

a query, we followed Larsen's idea that the query language should be "on the human user's condition" (see [8]). This lead to a Microsoft Access-like procedure where the user tells which attributes are required and for each of these, which fuzzy linguistic labels are wanted. The relevant query is formed by the system and displayed. The list of classes discussed in the previous paragraph can be seen on the right side of figure 3 with classes in bold face and output characterizations in normal style font.

4.5 From summaries to tuples in an answer

In the summarization process, the rewriting of data into fuzzy labels produces satisfaction degrees that tell how well a label describes the data. The satisfaction degrees, which are part of a summary's intension, are not used in this paper. But the qualitative information carried by these degrees is interesting, particularly to make a better distinction between results.

A fully flexible querying process with a ranking of results can build on our approach. In a first approximation, summaries are a general index over the data used to quickly reduce the search space. Then, a second step retrieves tuples from the output summaries extension. Finally, the degrees attached to each tuple can be taken into account to compute a total satisfaction degree of the query. This second step is the subject of a future paper.

An additional enhancement would deal with a quantitative aspect. Besides satisfaction degrees, the SaintEtiQ model provides frequency and proportion data attached to descriptors and candidate tuples [10]. By using that data, we will be able to express more information in a response, for instance the response in example 5 might be "thin normal-temperature materials are either flexible or soft but only a little part of them are soft".

5 Conclusion and future research

In this paper, a querying tool for the summarization model SaintEtiQ has been proposed. It allows end-users to efficiently retrieve summaries, and exploits the hierarchical structure of the summaries produced by SaintEtiQ.

From a technical point of view, it performs a boolean matching between the summaries and the query on the basis of linguistic labels from a user-defined vocabulary. It is therefore a classical boolean querying tool whose novelty lies in the use of summaries and in the efficient use of a hierarchy. The querying machinery, as well as a user-friendly interface have been developed and tested on toy examples in order to validate the method.

Then, as it is easy to obtain tuples from summaries, and to rank the tuples according to their membership degree to the summaries, it has been shown that the method can also be considered as a flexible querying tool on a relational database. The flexibility entirely relies in the summarization process, and this is one of the reasons why the process is efficient.

Besides, this work is a first attempt at querying the summaries, and the richness of the framework is far from being entirely exploited. Several future developments, such as the possibility to use another vocabulary, are under consideration.

Above all, expressiveness is the main point future work will focus on as it will eventually allow imprecision in not just the representation of information but also in user queries. It might also cover preferences or priorities in queries as mentioned by Rocacher in [13].

An important concern resides in making use of the different levels of granularity that a summary hierarchy offers. For now, the selection procedure returns summaries that are at different levels in the hierarchy. As interpretation is not straightforward, we will investigate making the selection descend as far as possible, even when a summary that matches exactly the query is found on the way to leaf nodes.

From the logical or set-based expression of queries, determining the reasons of a search failure is simple (see 4.2). From then, an interaction with the user will permit us to implement one of the cooperative behaviors (corrective answers) surveyed by Gaasterland in [5]. We may display the reasons of a failure, that is the fuzzy labels in the query that cause the failure, so that the user could ask a new query based on the previous one.

References

1. Patrick Bosc and Olivier Pivert. Fuzzy queries and relational databases. In *Proc. of the ACM Symposium on Applied Computing*, pages 170–174, Phoenix, AZ, USA, March 1994.
2. Juan C. Cubero, Juan M. Medina, Olga Pons, and María Amparo Vila Miranda. Data summarization in relational databases through fuzzy dependencies. *Information Sciences*, **121 (3-4)**:233–270, 1999.
3. Didier Dubois and Henri Prade. Using fuzzy sets in database systems: Why and how ? In *Flexible Query Answering Systems*, pages 45–59. Kluwer Academic Publishers, Boston, September 1997.
4. Didier Dubois and Henri Prade. Fuzzy sets in data summaries — outline of a new approach. In *Proc. 8th Int. Conf. on Information Processing and Managament of Uncertainty in Knowledge-based Systems (IPMU'2000)*, volume 2, pages 1035–1040, July 2000.
5. Terry Gaasterland, Parke Godfrey, and Jack Minker. An overview of cooperative answering. *Journal of Intelligent Systems*, **1**(2):123–157, 1992.
6. Janusz Kacprzyk. Fuzzy logic for linguistic summarization of databases. In *Proceedings of the 8th International Conference on Fuzzy Systems (FUZZ-IEEE'99)*, volume 1, pages 813–818, August 1999.
7. Janusz Kacprzyk and Slawomir Zadrożny. Computing with words in intelligent database querying: standalone and Internet-based applications. *Information Sciences*, **134**:71–109, May 2001.
8. Henrik Legind Larsen. An approach to flexible information access systems using soft computing. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*, volume 6, January 1999.

9. Do Heon Lee and Myoung Ho Kim. Database summarization using fuzzy isa hierarchies. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics*, **27**:68–78, February 1997.
10. Guillaume Raschia. SAINTETIQ*: une approche floue pour la génération de résumés à partir de bases de données relationnelles.* PhD thesis, Université de Nantes, December 2001.
11. Guillaume Raschia and Noureddine Mouaddib. SAINTETIQ: a fuzzy set-based approach to database summarization. *Fuzzy Sets And Systems*, **129**:137–162, 2002.
12. Dan Rasmussen and Ronald R. Yager. SummarySQL - a fuzzy tool for data mining. *Intelligent Data Analysis*, **1**:49–58, 1997.
13. Daniel Rocacher. On fuzzy bags and their application to flexible querying. *Fuzzy Sets And Systems*, **140**:93–110, November 2003.
14. Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Sytems, man and Cybernetics*, **18**:183–190, 1988.
15. Lotfi A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, **8**:199–249 & 301–357, 1975. Part III in volume **9**, pages 43-80.