General purpose database summarization

Régis Saint-Paul, Guillaume Raschia, Noureddine Mouaddib

LINA - Polytech'Nantes
ATLAS-GRIM Group
2, rue de la Houssinire, BP 92208
44 322 Nantes cedex 03, FRANCE
{saint-paul, raschia, mouaddib}@lina.univ-nantes.fr

Abstract

In this paper, a message-oriented architecture for large database summarization is presented. The summarization system takes a database table as input and produces a reduced version of this table through both a rewriting and a generalization process. The resulting table provides tuples with less precision than the original but yet are very informative of the actual content of the database. This reduced form can be used as input for advanced data mining processes as well as some specific application presented in other works. We describe the incremental maintenance of the summarized table, the system capability to directly deal with XML database systems, and finally scalability which allows it to handle very large datasets of a million record.

1 Introduction

Because of the ever increasing amount of information stored each day into databases, users can no longer have an exploratory approach for visualizing, querying and analyzing their data without facing the problem often referred to as 'Information Overload'. Hence, as mentioned in [1], the data summarization paradigm has become "a ubiquitous requirement for a variety of application environments, including corporate data warehouses, network-traffic monitoring and large socio-economic or demographic surveys". In addition, downsizing massive datasets allow to address some critical issues such as individual data obfuscation, optimization of the usage of system resources like storage space and network bandwidth, as well as effective approximate answers to queries.

Depending on the application environment and the preferred goal of the approach, we distinguish three families of approaches concerned with database summarization. The first one focuses on aggregate computation, the second, so-called semantic compression (SC), deals with intentional characterization of groups of individuals. The last one is interested in metadata-based semantic compression (MDBSC) to overcome the low self-description ability of the usual SC methods. Those categories are obviously not strict and there exist a lot of hybrid approaches. It is worth mentioning here that we do not refer to traditional (or syntactic) compression methods in which the database is viewed as a large byte string and usual compression algorithms (such as Huffman or Lempel-Ziv coding) can be used.

Related work

In the early 80's, conceptual models of statistical databases (SDB) [16] have been studied to provide tools for macro-data management. The main idea is to build aggregates using statistical functions from micro-data especially taken from scientific and socio-economic application environments. In that context, two major issues are i) to solve the *statistical inference* problem [3], dealing with the privacy of individual data, and ii) to guarantee the *summarizability property* of micro-data [11], which ensures that raw data could be aggregated. In SDB, raw data are not preserved and only summarized views are maintained.

Closely related to SDB, On-Line Analytical Processes (OLAP) and multidimensional databases are arising great interest from the summarization task point of view, since they allow an end-user to query, visualize and access part of the database using cubes of aggregate values computed from raw data. In a data warehousing environment, data marts are designed as

Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

a subject-specific OLAP system and data cubes are then designed in response to a given intent.

At the frontier of aggregate computation and semantic compression methods, Quotient Cube [9, 10] also aims at summarizing the generalized *group by* operator of the cube in OLAP systems. This method tries to group together cells of the cube with similar aggregation values while preserving the semantic of the cube operators. Quotient Cube, also referred by the authors as Semantic OLAP, offers a way to reduce the size of a particular cube, but it does not reduce the size of the dataset itself. As such, it can be seen as a specialized subject-oriented summary rather than a general purpose summary of the dataset.

Using SC for dataset summarization, a lot of work has been done, especially in the data mining research area, to characterize groups of database observations and to provide higher-level models such as decision trees or association rules. The few methods [7, 8, 1] which explicitly refer to semantic compression of structured data point out the guaranteed-quality approximate answer by taking into consideration an error tolerance. Thus, lossy compression can be performed to enhance the compression ratio. Those models provide intentional descriptions of the data but, as is, they are usually complex and useless structures for the end-user and they need third-party applications as well as experts to become relevant.

Finally, MDBSC approaches use metadata such as semantic nets to guide the compression process in a way that summaries are highly understandable since their descriptions rely on user-defined vocabulary. The main difference between SC and MDBSC is that in metadata-based approaches, rather than trying to identify hidden patterns from data, provide a short description which precisely fit the user knowledge of the domain. One representative of that family is the attribute-oriented induction process (AOI). It provides small versions of database relations using is-a hierarchies a priori built over each attribute domain. First mentioned in [19], the core AOI process has been presented and implemented in the DBLEARN system [5]. It roughly consists in replacing attribute values of database records by more general terms taken from the is-a hierarchies, and merging identical generalized tuples until the size of the database has reached a given threshold. Even if this approach seems to be well-suited to compress the database content, it frequently leads to over-generalization of tuples, and the well fitting granularity, from the user point of view for a specific task, is barely obtained. Furthermore the overall process of AOI has to be performed off-line. Besides, roll-up and drill-down operations performed on cubes in OLAP systems are also MDBSC methods since they rely on conceptual hierarchies built on each dimension of the cube. Finally, fuzzy set-based methods [20] have been proposed to construct robust summaries from datasets, using linguistic variables [22].

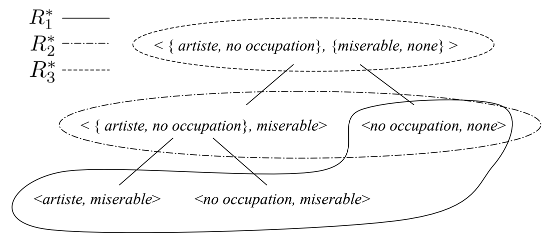

R_1^*
R_2^*
R_3^*
< { artiste, no occupation}, {miserable, none} >
< { artiste, no occupation}, miserable>
<no occupation, none>
<artiste, miserable>
<no occupation, miserable>

Figure 1: Examples of R^* of various size

Our contribution

We propose SAINTETIQ, an online linguistic summarization system of tables and/or views. Our approach considers a first normal form relation $R(A_1, \ldots, A_n)$ in the relational database model, and constructs a new relation $R^*(A_1, \ldots, A_n)$, in which tuples $z \in R^*$ are summaries and attribute values are linguistic labels describing a set of tuples R_z, sub-table of R. Thus, the SAINTETIQ system identifies statements of the form *"Q tuples of R are* (a_1^1 *or* a_1^2 *... or* $a_1^{m_1}$) *and ... and* (a_n^1 *... or* $a_n^{m_n}$)*"*.

The SAINTETIQ system computes, and incrementally maintain, a hierarchically arranged set of summaries, from the root (the most generic summary) to the leaves (the most specific ones). Within this structure, any non-leaf summary generalizes the content of its children nodes. From this hierarchy, it is then possible to select a set of summaries to form the relation R^* such that they cover the complete original relation R (i.e. $\bigcup_{z \in R^*} R_z = R$). However, the size of R^* can be freely chosen: the most precise relation R^* that represents R is given by the set of all leaf summaries and, at the opposite, the maximally concise relation is made of a single summary, the root node. A relation R^* of any given intermediate size, composed of more or less precise summaries chosen to best represent the content of the database, can be calculated from the hierarchy. Figure 1 shows the intuitive idea of the various R^* that can be extracted from the summary hierarchy, where R_1^* is the most precise representation and R_3^* is the most concise. In this example, the intentional content of each summary is presented for attributes OCCUPATION and INCOME.

The relation R^* thus forms a downsized version of the initial relation. R^* size can be adjusted posteriorly to the summarization process due to the produced hierarchy structure. The semantic compression introduced by the SAINTETIQ system respects the original dataset schema and can directly be queried or used as an alternative dataset for other operations including querying, browsing, data mining process input or any other operation that requires a reduced view of the database.

A summarization process has to harmoniously run

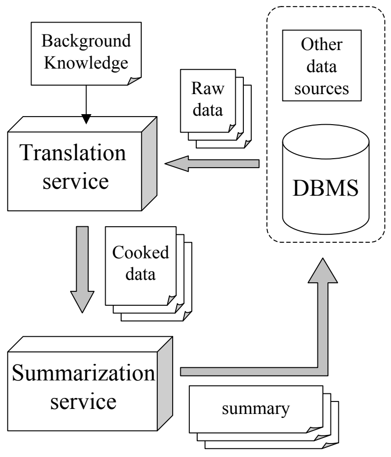


Figure 2: Message-Oriented Architecture of SaintEtiQ based on Web Services

on top of existing database management systems. As a result, the system requires a low memory consumption, a reliable serialization and a linear time complexity for atomic operations. This paper will focus on the SaintEtiQ system proposal to meet those requirements as well as on the DBMS coupling through various scenarios.

The rest of the paper is organized as follows. First, an overview of the SaintEtiQ process as well as its current message-oriented architecture are presented. The algorithmic time and space complexity will then be discussed and the benefits of the chosen architecture in terms of scalability will be exposed. In order to validate the results, section 4 is devoted to the process behavior in a real case situation. Section 5 will discuss applications of our approach. Finally, in section 6, we give conclusion and future direction of our work.

2 System architecture

A major requirement of any Semantic Compression process is to easily integrate into existing information systems. Among the various standardized means for system interoperability, web services [2], an open standard with cross-platform capabilities, presents a unique simplicity. The simplicity of the web services architecture, if compared with CORBA or COM+ for instance, comes in a large part from the loosely coupled way in which collaborative systems are intended to interact, with the assumption that web methods are independent, stateless and atomic.

Hence, to benefit the web services ease of use, the SaintEtiQ architecture has been designed around autonomous agents that interact through one-way messages. Figure 2 shows the overall organization of the SaintEtiQ system into two separate web services. The translation service corresponds to a pre-processing step that prepares the data for summarization while the summarization services actually produces the summaries.

2.1 The translation service

A unique feature of the SaintEtiQ system is its extensive use of Background Knowledge (BK). Databases store precise and application-oriented information to describe individuals (people, transaction, event, object, ...) with raw attributes values which are far from the user vocabulary on the same domain. For example, a database record in a business application gives the annual income of a given customer. This information is mandatory when application directly deals with the individual. A marketing manager however will probably refer to those individuals who have a *comfortable* or a *modest* income. The accuracy of such a vocabulary is much less than the available precision of the underlying database raw value. Nevertheless, it will be preferred by the marketing department since it allows an immediate understanding of the typology of customers.

To best reflect the way users manipulate knowledge about their data, the SaintEtiQ system relies on Zadeh's fuzzy set theory [22] and, more specifically on linguistic variables [21] and fuzzy partitions [12] to grasp the inherent imprecision and vagueness of natural language. Descriptors used for summary intentional content representation are defined as linguistic labels on the attribute domain. For example, figure 3 shows a user defined vocabulary on the attribute INCOME where descriptor *reasonable* is defined as being plainly satisfactory to describe values between 42000 and 59000 and less satisfactory as the income is out of this range.

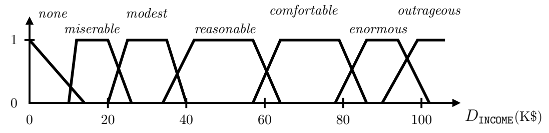


Figure 3: Fuzzy Linguistic Partition defined on attribute INCOME

The *translation service* supports the process of finding the best representation of a database tuple according to background knowledge provided by the user. It takes as input *Raw Data*, ie data directly extracted from the database in a common XML representation.

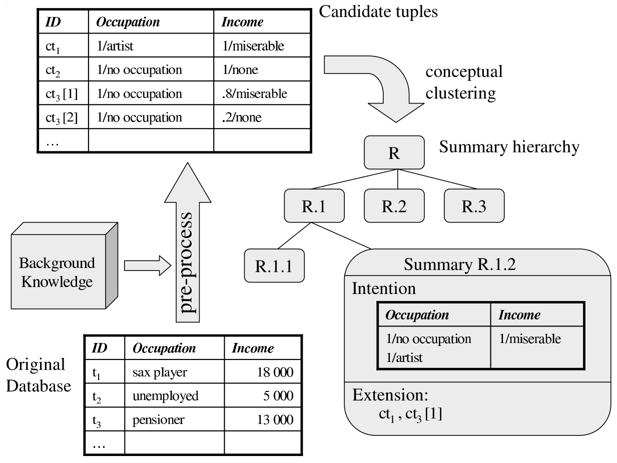
Almost any modern Relational DBMS provide mechanisms to directly handle XML data [15]. For instance, Microsoft SQL-Server uses the `FOR XML` directive of the Transact SQL language while Oracle 9*i* uses the `DBMS_XMLGEN` function. IBM's DB2 through Xperanto also provides similar features. Users of such DBMS can either define their own Relational/XML Schema mapping or use the default built-in output format. The following example shows a possible convention (the 'element' normal form) for such a representation of a collection of tuples:

```
<MarketingData>
  <Client>
    <ClientID>0005</ClientID>
    <Occupation>sax player</Occupation>
    <Income>18,000</Income>
  </Client>
  <Client>
    <ClientID>0013</ClientID>
    <Occupation>unemployed</Occupation>
    <Income>5,000</Income>
  </Client>
  <Client>
    <ClientID>0015</ClientID>
    <Occupation>pensioner</Occupation>
    <Income>13,000</Income>
  </Client>
</MarketingData>
```

Tuple values can be represented either by elements or by attribute values in the XML document. Thus, any usual representations of documents can actually be handled by SAINTETIQ without any modification. The system will try to guess the attribute names as well as each of the "tuple" root nodes and create its own mapping into an internal attribute/value representation.

The translation service transforms the *Raw data* document into *Cooked data*. Basically, the operation replaces the original value of all attributes by the set of descriptors defined in the BK that have a non-zero matching value. Figure 4 gives an example of preprocessing operation. In this figure, the separation of data into Raw Data document has not been shown because this separation is up to the user. As shown above, a Raw Data document may contain a single tuple as well as several tuples. However, we consider one tuple at a time, and tuple t_3 of figure 4 is rewritten into two distinct *candidate tuples* denoted by $ct_3[1]$ and $ct_3[2]$ in figure 4. Flexibility in the vocabulary definition of BK permits to express any single value with more than one descriptors. In this case, the income value of t_3 could be described as both *miserable* and *none* with corresponding satisfaction degrees.

Cooked data are documents that contain a single or a collection of candidate tuples and are the input of the summarization service. They provide a common representation of the data in the form of a set of descriptors for each attribute. Each descriptor is associated with a satisfaction degree. This makes the summarization service independent of the underlying data type. We have seen that the task of rewriting the attribute values into descriptors is trivial for nominal or numerical attributes. Separating the summarization service from the translation service allows for direct input of data prepared with user-defined translation service. Automatic extraction of image features such as color, size or texture, for example, can be used to describe an image attribute into a set of linguistic descriptors. An application of the summarization process to image databases can be found in [13].



ID	Occupation	Income
ct_1	1/artist	1/miserable
ct_2	1/no occupation	1/none
ct_3 [1]	1/no occupation	.8/miserable
ct_3 [2]	1/no occupation	.2/none
…		

Occupation	Income
1/no occupation 1/artist	1/miserable

ID	Occupation	Income
t_1	sax player	18 000
t_2	unemployed	5 000
t_3	pensioner	13 000
…		

Figure 4: The Overall Process of DB Summarization

2.2 The summarization service

The *Summarization service* performs a dual learning and classification tasks on the data. It takes cooked data as input, and outputs a collection of summaries, hierarchically arranged according to their precision. This hierarchical structure can later be used to produce reduced and size adjustable version R^* of the original relation R.

New data, prepared in the form of candidate tuples, are first incorporated in the root node of the hierarchy one at a time. Then, in a top-down conceptual clustering proposed by D.H. Fisher Cobweb [4], data are processed from the root to the leaves. At each node, a measure evaluates the quality of a few hypothetical summary arrangement resulting from applying a set of learning operators which locally modify the underlying partition P_z, i.e. the set of child nodes.

At a node z, the algorithm considers *incorporating* the current candidate tuple ct into each child node of z as well as *creating* a new child node accommodating ct. Furthermore, the system evaluates the preference of *merging* the two best children nodes of z and *splitting* the best child node. Then SAINTETIQ uses a heuristic objective function, the *partition quality*, based on contrast and typicality of summary descriptions to determine the best operator to apply at each level of the hierarchy.

The splitting and merging operation make local bidirectional modification of the hierarchy. They are used to weaken sensitivity of the object ordering, simulating the effect of backtracking in the space of summary hierarchies, without storing previous hypotheses on the resulting structure. Thus, the system does not adopt a purely agglomerative or divisive approach, but rather uses both kind of operators for the construction of the tree. The so-called *hill-climbing* search method locally optimize the summary hierarchy such that the tree is an estimated structure built from past observations and refined every time a new tuple is inserted. Deletion of tuples is performed symmetrically, from the leaves to the root, updating each concerned summary to take into account tuple deletion.

Once a candidate tuple is incorporated (or deleted) from a summary z, the intentional description of z is updated with respect to this new content and measures. Basically, this operation consists in merging the candidate tuple descriptors with those of the summary intention. Each descriptor is associated with two additional measures :

- satisfaction degree: this value expresses the accuracy of the descriptor regarding the actual content of the summary.

- support: it measures the number of tuples within the summary that are actually described by this descriptor.

This process repeats until a leaf is reached. Then, the system considers the creation of a new level. This decision is based on a user-defined criteria that set the desired maximal precision of the hierarchy. The maximal precision is achieved when all leave summary intentions are described due to a single descriptor per attribute. When a new level is created, two children nodes are incorporated to the current summary, the first one containing the leaf in its previous state and the second one containing the single candidate tuple ct.

The quality measure derives from the category utility as defined by D. Fisher in the COBWEB system [4], itself based on the Contrast Model introduced by A. Tversky [17] for the judgment of similarity. This measure tries to maximize both the contrast between classes and the internal cohesion. Hence, the quality measure is an aggregate of two measures: the average *contrast* between summaries and the average *typicality* of the summaries.

The *typicality* measure is based on the computation of an intra-similarity degree $\sigma(z_i)$ for all the summaries z_i in P_z. This value is updated in an incremental manner, each time a new cooked data is integrated into a summary. $\sigma(z_i)$ is maximal when the intentional description of z_i contains only a single descriptor per attribute, it is minimal (i.e. null) when the summary is described by all the available descriptors as defined in BK.

The *contrast* measure represents an average degree of the dissimilarity $d(z, z')$ between all summary pairs (z, z') of the partition. The dissimilarity $d(z, z')$ is null when all descriptors are common between z and z'. The process prevents such a situation in order to avoid redundancy in the summaries. Dissimilarity, on the other hand, is maximal when there are no common descriptors between the intentional descriptions of z and z'. The computation of the contrast of a partition is the most time-consuming since a summary with n child nodes must perform $n(n-1)/2$ operations for the computation of the dissimilarity vector needed by the contrast measure. Fortunately, most of the values of the dissimilarity vector remain the same for the n tests of incorporating a new tuple into each of the child nodes. Actually, $n-1$ calculations are required for each partition hypothesis $H_k(P_z)$ since for a given updated summary $z_1 \in H_k(P_z)$, only dissimilarity values $d(z_1, z_i)$, $i \in \{2..n\}$, have to be computed. Furthermore, each of the contrast evaluation is made very efficiently through the use of a matrix of pairwise dissimilarity. Incremental operators are used to update this matrix to reflect the changes that occur once a new tuple is inserted.

Those two measures are complementary: going from the root to the leaves of the summary hierarchy, the internal cohesion increases while the contrast between summaries decreases. The trade-off values are found in the middle of the hierarchy and correspond to summaries that are both generalized while still informative. Therefor, in order to represent the relation R in a concise R^* form, summaries of intermediate level are selected. Depending on the precision required in R^*, summaries will be selected at a varying level within the summary hierarchy structure. This last step is performed through an iterative process that reduces R^* at each step until the desired size has been reached. At each step, the less informative summaries are removed and replaced by a more general summary (a parent in the hierarchy).

A distinctive feature of SAINTETIQ is that changes in the database are reflected through the incremental maintenance of the complete summary hierarchy. This means that at any time, the process is able to provide a semantic compression of R at any chosen precision level. The summary selection step is efficient because generalized summaries have already been calculated.

3 Scalability Issues

In this section, we discuss about the robustness and efficiency of the proposed system, putting forward some relevant features of the SAINTETIQ process, especially related to the summarization service. The translation service will not be further discussed as it is a trivial rewriting process.

Memory consumption and time complexity are the two main factors that need to be taken care off in order to guaranty the capacity of a semantic compression system to handle massive datasets. Those two points will be discussed in the following subsections. Also, the parallelization of the system will be presented as this feature is a key to ensure a smooth scalability.

Figure 5 shows the component-based architecture of the summarization service. Messages between autonomous components are self-contained XML documents representing cooked data. They are treated as a stream and each document is relayed from the root of the hierarchy to different summary nodes along a branch until it reaches a leaf.

3.1 Time complexity of summarization service

The algorithm depicted in section 2.2 shows that constructing the hierarchy requires finding the best learning operator at each level of the hierarchy. Thus, in order to properly estimate the time complexity of the process, we first need to make reasonable assumption on the size and balance of the hierarchy.

In the worst case, i.e. when the hierarchy leaves have the maximal precision allowed by the given BK, the number of leaves L is a direct consequence of the number of combinations of descriptors that are present in the dataset. Of course, the exact number will greatly depend on the considered dataset, and more specifically, on the existing correlation between attribute values. For example, in a dataset with attributes PRODUCT and PRICE, it is likely that we will not find the combination of *Ferrari* and *Cheap*.

The total number of leaves L is not an issue because it can be adjusted by the user according to the desired precision:

- A detailed BK will lead to a greater precision in the summary description, with the natural consequence of a larger summary. For nominal attribute, the worst case is achieved when the translation step is skipped and the data are summarized directly with the values as they appear in the dataset.
- The hierarchy is constructed in a top-down approach and it is possible to set the process so that the leaves have any desired precision (see section 2.2).

The average hierarchy depth will be denoted d, an estimation of d is given by $d = \log_B L$ where B is the average number of children nodes of a summary.

At each step, the process evaluates the quality of the partition P_z resulting from the incorporation of a candidate tuple ct into each of the children nodes as well as k (actually 3: splitting, merging and creating a new node) additional operations. This process repeats along the hierarchy until a leaf node is reached. The time cost C for the incorporation of ct can thus be expressed as :

$$C(B, L) = \log_B L \cdot [(B + k) \cdot c(B, N)]$$

where c is the time taken for the evaluation of a partition quality. Thanks to the contrast matrix described in section 2.2, this cost is a linear function of the number of attributes N and of the number of children nodes B.

The cost for the incorporation of a candidate tuple is then a fixed time $C(B, L)$ depending only on the size of the hierarchy. The SAINTETIQ process time complexity is thus in $O(n)$ with n the number of candidate tuples. The number of candidate tuples that will be produced by the translation service is dependent only on the fuzziness of the BK definition. A crisp BK will produce exactly as many candidate tuples as there are original tuples.

We can see that the balance of the hierarchy is the parameter that has the most influence on the computation cost. In fact, this estimation corresponds to a worst case scenario because many of the $(B + k)$ evaluations do not have to be done in order to decide the correct learning operator. Here are some of the heuristics that helps in avoiding the complete computation:

- If the intentional content of a child node already subsume the description of the candidate tuple, the process can stop evaluating the other child nodes: none will be a better candidate. During the learning process, more descriptor combination have already been learned by the hierarchy. Hence, this situation will occur more frequently. In such a situation, the summary hierarchy performs a simple classification task rather than a learning one. On very large datasets, it is likely that at some point, almost all the existing combination are already known.
- In order to guaranty a deterministic classification, the process avoids the creation of a partition such as two summaries subsume the same set of descriptors. If the incorporation of a candidate tuple into one of the children node leads to such a partition, then the incorporation of the candidate within the conflicting children node can be skipped.

In the SAINTETIQ algorithm, raw data have to be parsed only once. Furthermore it is performed in a minimal time cost, due to both the incremental learning, the hill-climbing search method and the predefined vocabulary of summary descriptions.

3.2 Space complexity of the summarization service

From the memory consumption point of view, only the current state of the summary hierarchy as well as a single cooked data are simultaneously required to perform the SAINTETIQ algorithm. We denote by S the average size in kilo-bytes of a summary or cooked data. In the average-case assumption, there are $\sum_{k=0}^{d} B^k = (B^{d+1} - 1)/(B - 1)$ nodes in a B-arity tree with d, the average depth of the hierarchy. Thus, the average space requirement is given by:

$$c_m = S \cdot \frac{B^{d+1} - 1}{B - 1}$$

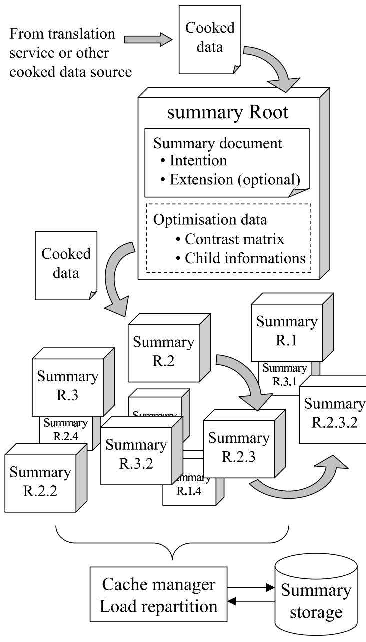


Figure 5: Sub-Systems and their Interactions in the Summarization Service

Each summary node contains not only the intentional description, but also an optional list of the incorporated tuples (i.e. the extension) and some optimization data: the contrast matrix and other incremental measures. Based on some real test, $S = 3\text{kB}$ gives a rough estimation of the space required for each summary.

In practice, the number of summary nodes is derived from the variety of attribute values in the database. In the worst case where each leaf of the hierarchy represents only one descriptor combination, the number of leaves L is equal to the number of distinct cooked data descriptions. L is expected to be much lower than the number n of tuples since the translation vocabulary is at most as precise as the initial raw values. For instance, if we consider a 1 million tuples database, a reasonable assumption is $L \approx n/4$, that is, $L = 250000$. The number of leaves of an average B-arity tree is also given by $L = B^d$, where d is the average depth. Therefor, $B = 6$ and $d = 7$ are candidate values to approximate the number of leaves in a summary tree of a 1 million tuples database. Hence, the total number of summaries in the hierarchy is given by $6 \cdot 335923 \cdot 3\text{kB} \approx 1\text{GB}$.

Figure 5 gives an overview of the internal implementation of the summarization service. Each summary, while being part of a global hierarchy is designed to be an autonomous agent and can be serialized as a small binary stream.

A supervisor, the *cache manager*, is in charge of summary caching in memory and it can be bounded to a given memory requirement. Usually, less than a hundred of summaries is needed in the cache since this number covers the two or three top levels of even a wide hierarchy. Least recently used summaries are discarded when a required summary is not found in the cache. The cache manager requests the resurrection of a summary, as stored in the database. Once resurrected, the parent summary component invokes the incorporation method in the current summary with the new tuple as argument. Then, the *cooked data* document is transmitted to the appropriate child summary and this operation is repeated until the message reaches a leaf node. The summary stores itself in the database only when it is unloaded by the cache manager.

The swap system supported by cache management drastically reduces memory consumption to a user or system-defined threshold. The natural counterpart is some more frequent disk accesses, even if they are controlled by the cache policy. In a conventional page cache system, objects in memory would be loaded according to their relative position in the stack. This default behavior however can be enhanced by the SaintEtiQ cache policy where instances are disposed according to their use frequency. With such a method, complete branches of the hierarchy are resident in memory while rarely accessed summaries (which represent outliers data) are stored on the disk. A general disk access cost model has not been defined since performance mainly depends on data insertion order as well as attribute values distributions, and as such, it is very application-dependent.

3.3 Distributed Processing

As mentioned in the previous section, the implementation of the SaintEtiQ system is based on the Message-Oriented Programming paradigm. Each subsystem is autonomous and collaborates with the others through disconnected asynchronous method invocations. It is among the least demanding approaches in terms of availability and centralization.

The autonomy of summary components allows for a distributed computing of the process. Once a component completes the treatment and evaluates the best operator for the hierarchy modification, if needed, a similar method is successively called on children nodes. The cache manager is able to handle several lists of summaries residing on different computers. The manager is also responsible for the load balancing of the newly created process.

Moreover, in order to minimize bandwidth consumption, summaries are grouped by branches of the hierarchy. Obviously, the most frequently used nodes are situated at the top of the hierarchy. Those nodes however are not very prone to hierarchy rearrangements when the number of summarized tuples significantly grows. In that case, their main task consists in classifying the new tuples, that is finding the next node which will further process the tuple. Once it is oriented to a branch, all other sibling nodes become available for parallel computing. The load balancing is directly achieved by analysing the number of processed tuples for each node: when data are treated in no particular order, the relative content size of a summary is a direct function of its frequency of use.

As shown by Figure 5, each basic step, separated by a wide arrow, can take place on distinct parallelized nodes:

- The source DBMS do not require any change other than the publication or the request for raw data. Usually, the DBMS is already in place and isolated from other tasks like the summarization process.

- The translation process is simultaneously performed on any single data or batch of data over different computers. It produces a stream of cooked data (see Section 2.1) that can reach the summarization service in any order.

- The summarization service requires a front-end as an entry point to the hierarchy. Nodes are processed from the top of the hierarchy to the bottom. As detailed in Section 2.2, the partition quality measure is the most demanding one in terms of processing time. Each of the summary node have the information about its underlying hierarchy partition in order to independently perform those evaluations. As such, summary components can run on different computers and serialize summary documents in a centralized or distributed DBMS (which can be entirely different from the one providing raw data).

Between each of these steps, a single message containing the cooked data is transmitted as shown by the wide arrows of Figure 2. The serialized form of the cooked data document requires about 1kB for header, to identify the target hierarchy structure, and the size corresponding to the serialization of each attribute descriptor (about 20 bytes for each attribute).

4 Experiments

4.1 UCI dataset

The summarization performance was evaluated against the 1990 US Census Data from the UCI KDD Archive [6] which is the largest dataset of this repository. The dataset is a collection of 2,458,285 tuples defined on 67 discrete attributes. We used the prepared version of the dataset, as provided by C. Meek, B. Thiesson and D. Heckerman from Microsoft. They dropped the less useful attributes of the original dataset and discretized the few continuous attributes to a reasonable number of discrete values, since we do not have expert skills required to build

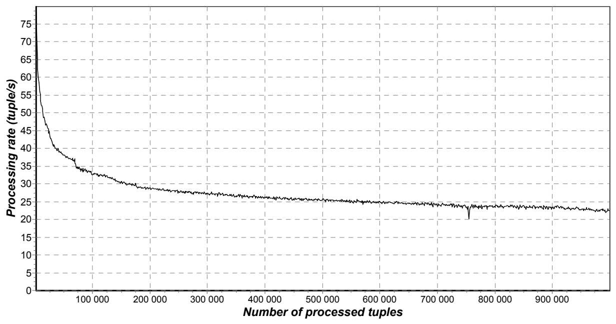


Figure 6: Process performance

realistic domain knowledge. Therefor, the data we used did not need any translation and were directly processed by the summarization service available at `http://www.simulation.fr/seq`.

In this experiment, the main goal is to check the validity of our processing cost function. From the 67 attributes, we selected 11 of them, with the greatest number of discretized values (many attributes were binary one). Thus, all the selected attributes have between 5 and 14 distinct values such that the cartesian product is $9.828e^8$.

The test was performed over the first million of tuples of the dataset. Figure 6 shows the processing rate, evaluated on a Intel Pentium IV 2.6GHz processor. Predictably, as the hierarchy size increases, the process slows down. The size of the hierarchy depends essentially on two factors which directly determine the total number of leaves in the hierarchy:

- maximum granulation of the summaries;

- size of the cartesian product of attribute domains.

The maximum granulation of summaries is fixed by the condition under which the hierarchy depth grows. For our experiments, the precision was set to the maximum, i.e. the hierarchy leaves contain exactly one element of the cartesian product (a single combination of attribute values). This situation produces a hierarchy where leaves have a maximal intra-similarity as only one descriptor is used for each attribute. Of course, it is not required to be as precise as this and the condition for the hierarchy growth can be set to any value of intra-similarity. The effect would be like cropping the summary tree by removing its too precise nodes.

Distinct tuples, i.e. elements of the relation defined over the cartesian product of discretized attribute domains, only depends on the translation step and the selected attributes. In our case, the translation step was already done and it appeared that a lot of combinations of attribute values were represented in the first million of tuples. Figure 7 shows the number of leaves according to the number of processed tuples. We can see that the curve is pretty regular with an average of one tenth of the database presenting different attribute values (over the selected subset of attributes).

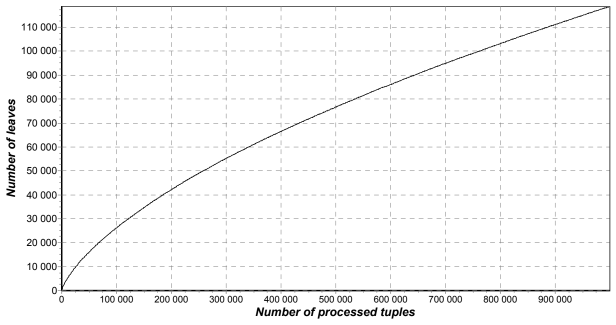


Figure 7: Number of leaf summaries

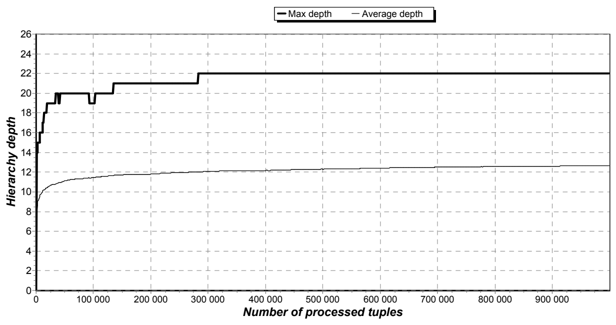


Figure 9: Evolution of the depth of the hierarchy

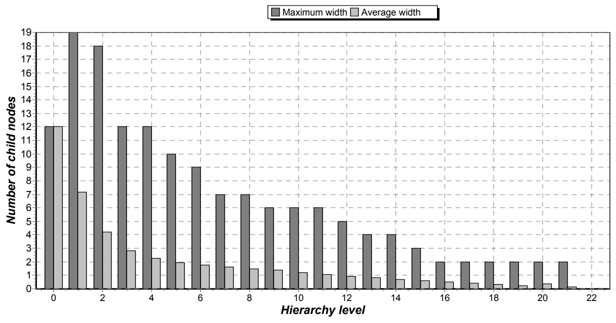


Figure 8: Wideness of the final hierarchy

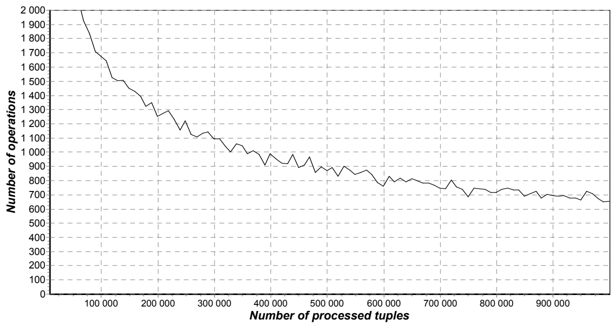


Figure 10: Learning operators needed for 10,000 tuples

The total number of leaves at the end of the process is 118707. Hence, a relation R^* can be produced of any size between this number and 1.

The hierarchy contains a set of intermediate levels involving any new tuple incorporation either in an existing node (classification task), or to a new one (learning task). Therefor, the average number of operations needed to incorporate a new tuple is closely related to the width and the depth of the hierarchy. In that experiment, the produced hierarchy is composed of a total of 234255 individual summaries, distributed among branches of an average depth of 12.67, with the maximum depth being 22. The maximum width (19) is located at level 2, whereas the average value of this level is 7 siblings. Figure 8 shows the maximal and average width at each level of the tree. Maximal and average wideness at each level of the final hierarchy had been reported in figure 8.

It is worth to note that while the number of new leaves increase monotonically, the depth or wideness of the hierarchy follows, as predictable, a logarithmic progression. Figure 9 shows the evolution of the maximum and average depth of the hierarchy regarding the number of incorporated tuples. Naturally, the performance of the process follows roughly the same progression.

The total time for the processing of one million tuples was 636 minutes. It should be noted that the algorithm was optimized in terms of number of operations (time complexity), but, as a research prototype, no low-level optimization has been done regarding the individual operation cost. For example, descriptors are compared through string comparisons while a bit map would dramatically reduce time cost for this operation. Those kind of low-level optimizations however would only affect the number of processed tuples per second (constant part of time cost), not the overall aspect of the curve shown on Figure 6.

Besides, as shown on Figure 10, the number of learning operators tends to be smaller as more tuples are processed. This is due to the decreasing proportion of new distinct tuples w.r.t. the ones already inserted into the hierarchy. As more tuples are processed, fewer changes are required on the hierarchy and, hopefully, once all existing combinations of attribute values have been processed, incorporating new tuples consists only in sorting it in a tree. In our experiments, the rate of new combinations is almost constant throughout the all process (see Figure 7). Then, the system permanently stays in *learning* mode. Hence, performance should increase once the curve of Figure 7 has reached its asymptotic behavior.

4.2 Real life dataset

Through an agreement, the CIC Banking Group provided us with an extract of statistical data used for behavioral studies of their customers. The database consists in a single table in which each record represents a customer, and fields (attributes) describe the customer in terms of age, income, or occupation, as well as banking products this customer is used to hold (accounts, credit cards, loan, . . .). And finally, several

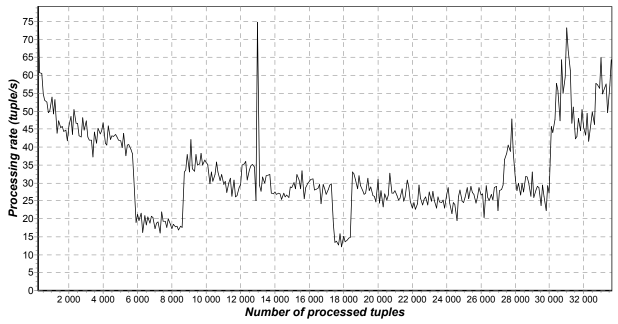


Figure 11: Process Performance

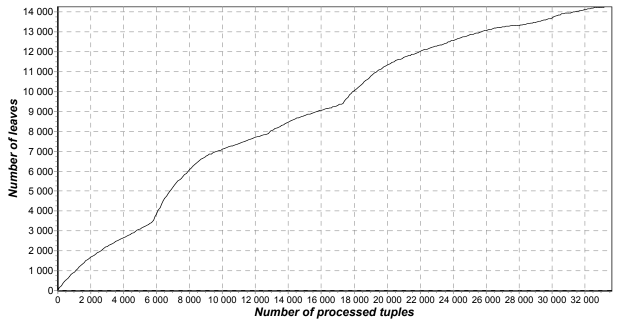


Figure 12: Number of leave summaries

attributes are dedicated to statistics over the operations the customer do on a monthly period (number of operations, total cash withdrawal, ...). The database represents a set of 33700 customers and 70 attributes. It is to be noted that some of the database values are absent and some are incoherent.

This dataset differs from the UCI dataset in several ways: it has not been prepared, customers were sorted by offices (each customer is attached to a particular office, there are about 120 offices spread in the western region of France concerned by the dataset), and BK knowledge has been designed by the expert of the marketing department. Figure 11 shows the evolution of the performance of the process. It is to be noted that the performance increases toward the end of the processing. As opposite to the previous dataset, the bank customers exhibit a greater correlation between attributes. Hence, less different combinations are found once the process progresses. We can see that the performance decreases around the 6000th and the 18000th processed tuple. Those two discontinuities also appear on the curve showing the number of leaves (see Figure 12). This shows that new value combinations were met at a higher rate during those stages than during the rest of the process, requiring more learning operations with a corresponding impact on the process performance. After a sequence of learning activity to take into account the new modalities found in the data set, the hierarchy becomes stable again and less operations are required for the incorporation of incoming tuples.

This situation is explained by the dataset organization: new category of customers are found when data from different offices are processed. We later learned that the bank offices are not only geographical divided, but are also specialized by customer types, some specializing, for example, in firms only.

5 Applications

Beside the classical usage of semantic compression techniques, the SAINTETIQ summarization scheme was successfully applied in a variety of applications, presented in separate publications [18, 13, 14]:

1. Selection of a subset of summaries of a given size for incorporation into a Galois Lattice;

2. Identification of the typology of a group (in a Decision Making paradigm);

3. Image database query by example;

4. Flexible query/answering support;

5. User profiling for an intelligent Personal Video Recorder;

The first application is a straightforward usage of the compressed version of a database. Galois Lattices are structures that offer some great features for image database browsing. However, their usage is limited due to the high time complexity ($O(n^2)$) of the lattice construction. Hence, rather than directly constructing the lattice on the images of the database, a summarized version is used. The lattice is constructed on a fixed number of generalized summaries rather than on the initial data. In this case, each summary describes a subset of the images thanks to descriptor automatically extracted from the low level features of the image.

The second situation also uses the reduced version of a collection of data but, in this case, the summary structure is used to build reduced version of subset of the database tuples. For example, if the user wants to know the typology of customers that buy a particular product, (s)he select the corresponding subset of the database and uses the summary structure to find relevant generalization of those customers. This way, the intentional content of the summaries provide an immediate description of the customers in the user vocabulary.

Applications 3 to 5 benefit from the hierarchical arrangement of summaries, used as a general index over the data. Using the classification property of the summarization process, it is possible efficiently locate summaries that can answer a particular query. Hence, the summary structure can be used to give approximate answer to queries.

6 Conclusion

In this paper, we presented the integration of a database summarization process into existing corporate DBMS systems. This operation is facilitated by the intensive use of wide spread standards such as XML and Web services. The summarization process combines advantages such as scalability, controlled memory consumption, conservation of database schema and the capacity to handle various data types as well as user-defined ones. The produced summary hierarchy provides views on the data at different levels of granularity through perfectly understandable (because user-defined) high-level descriptors. Those descriptors rely on fuzzy set theory to avoid common drawbacks of crisp methods such as mis-categorization. The scalability and linear time complexity of the summarization process was validated against both a massive database and a real case application.

Future work could be to provide a wider range of data translation services for various data types and to develop a graphical user interface that can help the user to take the most benefits from a summarized version of the database. We will consider using the on-line version of summary integration to help mining dynamical behavioral tendencies.

Further developments include the use of database summaries to help querying e-communities catalog data. In this application, many peers collaborate to a community providing large databases (for example, the community of travel agencies). To answer a user query, each community member has to be separately queried and the results have to be gathered before being sent back to the user. The approximate query answer given by a summarized version of peer databases can be used to avoid unnecessary queries to peers that obviously don't have any valuable answer and, thus, help in sparing both bandwidth and response waits.

References

[1] S. Babu, G. Minos, and R. Rajeev. SPARTAN: A model-based semantic compression system for massive data tables. In *Proc. of the 2001 ACM Intl. Conf. on Management of Data (SIGMOD 2001)*, pages 283–295, Santa Barbara, CA, May 2001.

[2] W. W. W. Consortium. Web services activity, [http://www.w3c.org/2002/ws], 2002.

[3] D. E. Denning. Secure statistical databases with random sample queries. *ACM Trans. Database Syst.*, 5(3):291–315, 1980.

[4] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[5] J. Han, Y. Fu, Y. Huang, Y. Cai, and N. Cercone. DBLearn: a system prototype for knowledge discovery in relational databases. In R. T. Snodgrass and M. Winslett, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, page 516, New York, NY, USA, May 1994. ACM Press.

[6] S. Hettich and S. Bay. The UCI KDD archive [http://kdd.ics.uci.edu], 1999.

[7] H. V. Jagadish, J. Madar, and R. T. Ng. Semantic compression and pattern extraction with fascicles. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDB99)*, pages 186–198, 1999.

[8] H. V. Jagadish, R. T. Ng, B. C. Ooi, and A. K. H. Tung. ItCompress: An iterative semantic compression algorithm. In *20th International Conference on Data Engineering*, page 646, 2004.

[9] L. V. S. Lakshmanan, J. Pei, and J. Han. Quotient cube: How to summarize the semantics of a data cube. In *Proceedings of the 28 th International Conference on Very Large Data Bases*, pages 778–789, Hong Kong, China, August 2002.

[10] L. V. S. Lakshmanan, J. Pei, and Y. Zhao. Socqet: semantic olap with compressed cube and summarization. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 658–658. ACM Press, 2003.

[11] H.-J. Lenz and A. Shoshani. Summarizability in OLAP and statistical data bases. In *SSDBM '97: Proceedings of the Ninth International Conference on Scientific and Statistical Database Management*, pages 132–143. IEEE Computer Society, 1997.

[12] E. H. Ruspini. A new approach to clustering. *Information and Control*, 15(1):22–32, July 1969.

[13] R. Saint-Paul, G. Raschia, and N. Mouaddib. Image database summarization with the saintetiq system. In *Proc. of the 9th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'2002)*, pages 1179–1186, Annecy, France, July 2002. ESIA - Universit de Savoie.

[14] R. Saint-Paul, G. Raschia, and N. Mouaddib. Mining a commercial banking data set: The SaintEtiQ approach. In *Proc. of the IEEE Int. Conf. on Systems, Man & Cybernetics (SMC'2002)*, Hammamet, Tunisia, October 6-9 2002.

[15] J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pirahesh, and B. Reinwald. Efficiently publishing relational data as XML documents. *VLDB Journal: Very Large Data Bases*, 10(2–3):133–154, 2001.

[16] A. Shoshani. Statistical databases: Characteristics, problems, and some solutions. In *Eigth International Conference on Very Large Data Bases, September 8-10, 1982, Mexico City, Mexico, Proceedings*, pages 208–222. Morgan Kaufmann, 1982.

[17] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

[18] W. Voglozin, G. Raschia, L. Ughetto, and N. Mouaddib. Querying the SaintEtiQ summaries - a first attempt. In *6th International Conference on Flexible Query Answering Systems*, Lyon, France, June 24-26 2004.

[19] A. Walker. On retrieval from a small version of a large data base. In *Sixth International Conference on Very Large Data Bases, October 1-3, 1980, Montreal, Quebec, Canada, Proceedings*, pages 47–54. IEEE Computer Society, 1980.

[20] R. R. Yager. A new approach to the summarization of data. *Information Sciences*, 28(1):69–86, Oct. 1982.

[21] L. Zadeh. Concept of a linguistic variable and its application to approximate reasoning-I. *Information Systems*, 8:199–249, 1975.

[22] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.