The K-Anonymity Approach in Preserving the Privacy of E-Services that Implement Data Mining

Ion LUNGU[1], Alexandru PIRJAN[2]
[1]Economic Informatics Department, Academy of Economic Studies, Bucharest, Romania
[2]Romanian-American University, Bucharest, Romania
ion.lungu@ie.ase.ro, alex@pirjan.com

In this paper, we first described the concept of k-anonymity and different approaches of its implementation, by formalizing the main theoretical notions. Afterwards, we have analyzed, based on a practical example, how the k-anonymity approach applies to the data-mining process in order to protect the identity and privacy of clients to whom the data refers. We have presented the most important techniques and algorithms used in order to enforce k-anonymity. We have studied possible approaches to ensure that k-anonymity preserves the privacy of the data mining process in order to assure an efficient, safe and effective data mining delivery as an e-service. We have depicted several software implementations that employ k-anonymity. We have developed, using the Compute Unified Device Architecture, an algorithm that analyzes k-anonymity and is suitable for extracting tuples with the same quasi-identifying values from a large database structured as a private table.

Keywords: *K-anonymity, Data mining, Compute Unified Device Architecture, Privacy, Confidentiality*

1 Introduction

Due to the tremendous evolution of the information technology and the increasing necessity for storage, access and analysis, the field of knowledge discovery and data mining emerged in the recent past. In parallel, the Internet spreading provided a platform where organizations conduct commercial transactions, leading to the development of e-commerce. After that, the research and development in the field of e-commerce led to the next evolutionary phase, namely, e-service. As it is generally defined, an e-service contains three main components: the service provider, the service receiver and the channels of service delivery that are based on the information technology [1]. Usually, the main channel used for delivering an e-service is the Internet, but channels such as a call center, a telephone, a TV channel, can also be taken into account when delivering e-services to customers. Because e-services involve a lot of personal data exchange, privacy is one of the most important challenges when implementing e-services, because people who use such services want to be sure that their personal information is safe and will remain secure and confidential. On one hand, these privacy concerns might make the service provider hesitant to offer online services and on the other hand the service receiver might refuse the usage of such services without having the guarantee that his own data privacy is secured.

Data mining in conjunction with business intelligence applications interferes with the e-services domain and becomes suitable for being delivered as an e-service, mainly because of the fact that several small to medium range businesses are constrained by the high costs required for setting up and maintaining the infrastructure of support technologies and business intelligence software [2]. In order to assure an efficient, safe and effective data mining delivery as an e-service using the Internet, Web service technologies are introduced to provide a layer of abstraction above existing software systems. The entire data mining e-service model requires interaction and communication between e-service agents and clients, and between e-service providers and e-service agents. These interactions need to be implemented in a reliable, stable, scalable and secure way and this is the reason why we study in this paper a modern and novel approach, the k-anonymity concept, used to preserve the privacy in the data mining process, including those e-services that employ data mining.

In recent years, the information society, the private and public organizations are facing an exponential increase in the amount and variety of daily collected data. In this context, data mining techniques are becoming increasingly important in assisting the decision making process and in extracting knowledge from large volumes of data in the form of models, patterns and trends. Although most of the time the results of the data mining process do not explicitly contain the original data stored in databases, these results could still be used to deduce other pieces of information contained in the original data that were not intended for release because it could affect the privacy of people referred in the data. Data owners or holders face difficult challenges to manage, store, protect and release information without compromising the privacy, confidentiality or even national interests.

The data mining technology raised a serious challenge for the researchers in the field: combining the legitimate usage and sharing of mined information while preserving the privacy of individuals. Therefore, the guaranteeing of proper protection of data is of paramount importance taking into account the risk of violating the privacy of individuals to whom the stored information refers. The collection of data that includes personal information and its analysis requires a high degree of privacy protection.

The techniques used for extracting knowledge from databases must be designed as to provide guarantees on the privacy of individuals referred to the data in question. In this context, the concept of privacy preserving data mining has emerged, as a response to these concerns [3]. The privacy preserving data mining seeks to ensure a compromise between the exchange of information for data mining analysis and keeping that information safely, in order to protect the privacy of parties involved. In order to preserve confidentiality, researchers have proposed several approaches whose techniques for protecting data are based on modifying (masking or erasing) the original sensitive data that should not be revealed [3].

The basic concepts of these approaches are "loss of privacy" and "loss of information". The loss of privacy concept measures the capacity of estimating the original data from the modified data while the loss of information concept measures the loss of accuracy in the data [4]. If the level of confidentiality regarding respondents to which the data refers to is higher, the results from the process of data mining are less accurate and vice versa. Therefore, the main purpose of these approaches is to provide equilibrium between privacy and accuracy.

There are also other approaches to privacy preserving data mining, like those implementing cryptographic techniques to prevent leaks of information [5] [6] [7]. The main disadvantage of cryptography-based techniques is that they usually require a lot of processing power and consume a significant amount of time. After establishing a clear definition of privacy and after identifying the information that is sensitive in the original data and must be protected against disclosure, one can choose privacy preserving data mining techniques. In the following we will address the notion of k-anonymity [8] [9] [10] a recent approach related to the privacy used for modeling the protection of released data against possible retrieval of private data from respondents to which the data refers to. Recently introduced, the notion of k-anonymous data mining is an approach linked to ensuring the preservation of privacy when data mining results are being released. According to the basic principle of k-anonymity, each released data must respect the condition that every combination of values, externally available, corresponding to the released attributes, can be associated to a minimum number of k respondents.

2 The k-anonymity

As mentioned before, the notion of k-anonymity [8] [9] [10] highlights the protection of released data, as a consequence of the data mining process, against possible retrieval of private data from respondents to which the data refers to. We first briefly depict the main theoretical notions that we will use later when we illustrate the k-anonymity approach by a series of examples. A detailed discussion of these notions and concepts is presented in [6].

Definition 1. For a relational table $B(A_1, \dots, A_n)$ with a finite number of tuples, the finite set of *attributes* of B is $\{A_1, \dots, A_n\}$. For a table $B(A_1, \dots, A_n)$ if we consider a subset of the set of attributes, $\{A_i, \dots, A_j\} \subseteq \{A_1, \dots, A_n\}$ and a tuple $t \in B$, we will denote the sequence of values $v_i, \dots, v_j$ of $A_i, \dots, A_j$ in t by $t[A_i, \dots, A_j]$ and the projection of attributes $A_i, \dots, A_j$ in B (maintaining duplicate tuples), by$B[A_i, \dots, A_j]$. We will assume that each tuple is specific to one person and every person appears only in one tuple. In order to protect the data and limit the ability to link released information to other external data, the data holder must identify all the attributes that contain private information and could be used to link with external information about this person. These attributes include private data and explicit identifiers such as name, address and others, or they may include attributes whose combination could help to certainly identify a person, such as birth date and gender. Dalenius [11] named these attributes quasi-identifiers. Person specific data has to be released such that the quasi-identifier does not link to other information that could affect the privacy of the persons in question.

Definition 2. If U is a population of entities, $T(A_1, \dots, A_n)$ an entity specific table, $f_c: U \rightarrow T$and $f_g: T \rightarrow U'$, where $U \subseteq U'$, a *quasi-identifier* of T, $Q_T$is a set of attributes $\{A_i, \dots, A_j\} \subseteq \{A_1, \dots, A_n\}$ ie $\exists \alpha_i \in U$ such that $f_g(f_c(\alpha_i)[Q_T]) = \alpha_i$.

Attributes that appear both in private and public data are suitable candidates for linking and constitute the quasi-identifier, an important reason to control their disclosure. Usually, the data holder can easily identify these attributes. But what happens if the data holder does not correctly identify the sensitive attributes for linking? In this case, it is possible for the individuals to be easily identified as the released data becomes less anonymous. Many approaches in the literature try to solve the conflict between disclosure risk and information loss [9], [6]. A protected dataset satisfies the k-anonymity requirements if for every combination of quasi-identifiers there are at least k records that share the same combination in the dataset. When k-anonymity reaches the desired level of protection, minimizing information loss becomes the next concern.

Definition 3. If $T(A_1, \dots, A_n)$ is a table and QI_T its associated quasi-identifier, T is said to satisfy the *k-anonymity* if and only if each sequence of values in $T[QI_T]$ appears at least k times in $T[QI_T]$.

Lemma. If $T(A_1, \dots, A_n)$ is a table and $QI_T = (A_i, \dots, A_j)$ its associated quasi-identifier, $\{A_i, \dots, A_j\} \subseteq \{A_1, \dots, A_n\}$, and T satisfies the *k-anonymity*, then each sequence of values in $T[A_x]$ appears at least k times in $T[QI_T]$, for $x = i, \dots, j$.

In the following, we will illustrate the k-anonymity by a series of examples. We will first consider a private table PT, where explicit identifiers of persons (such as the name, the ID, the address) to which the stored data refers to, were removed. On the other hand, a number of values of other attributes (such as birth date, zip code, gender, marital status, etc.) can appear in some external tables jointly with the respondents' identities. If certain combinations of values of these attributes are unique or rare, then it is possible that people who have access to such data, can identify the respondent to which the data refers to or can select a small group of persons having a common set of characteristics, group in which the person in question belongs to.

Based on the k-anonymity requirements, each tuple in the private table may be indistinguishably linked to at least k respondents. In order to protect the identity and anonymity of individuals whose data are involved and to make the identification of

these people more difficult by a potential attacker, k-anonymity requires that if a combination of values of a set of attributes (called quasi-identifying attributes) appears in the table, then this combination must appear at least k times [4].
The private table chosen to illustrate the k-anonymity contains a series of attributes and among them, the education level of the person, his/her sex, the age of the person and also an information related to the fact that the person wears or not glasses. The quasi-identifier is constituted from attributes „Education_level", „Sex" and „Age". In Table1, one can observe a simplified representation of the private table, highlighted through the quasi-identifier.

Table 1. A simplified representation of a private table

Education_level	Sex	Age	Tuples	Glasses
Bachelor	M	32	20	10Y;10N
MA	F	40	7	2Y;5N
PhD	M	42	2	0Y;2N
Baccalaureate	F	36	8	3Y;5N
PhD	F	45	2	2Y;0N
Gymnasium	M	31	15	6Y;9N

In this representation, tuples with the same quasi-identifying values were represented in the table as a single tuple. On the right side of Table 1, on each row, is placed information regarding the number of tuples with common characteristics and the number of those people who wear or not glasses. Possible values of this attribute are denoted by Y (yes), corresponding to cases in which a person wears glasses and N (no), if that person does not wear glasses. In the table, there are only two occurrences of males being 42 years old and a PhD degree. As a consequence of these two occurrences, the k-anonymity degree for preserving the privacy in the private table PT in Table 1 is very poor, being k=2 or less. If there is another linked external table that contains other supplementary attributes, referring to respondents from the PT table, the identity of respondents can be reduced to one of the two people having these common features.
One can observe that if a tuple has k occurrences, then any of its sub-tuples must have at least k occurrences, which leads to the following conclusion: a condition for having k occurrences of a tuple is that there are k occurrences of any sub-tuple. This condition is necessary but not sufficient. In our example, if we consider the k-anonymity over the quasi-identifier {Education_level, Sex and Age}, it requires that each value of individual attributes appears at least k times and the same condition is available for any sub-tuple corresponding to a combination of them. Referring to our example, one can observe that there are only two tuples referring to males having a PhD degree and as a consequence we can certainly say that the table will not satisfy the k-anonymity for k>2 because after adding the attribute "Age", these occurrences will remain two or less. In order to enforce the k-anonymity on such a private table, two main techniques with the property of preserving the truthfulness of the data have been proposed: the generalization and the suppression techniques, which we will depict in the following.
The first technique, the generalization, is based on replacing each attribute with a more general version of it, taking into account a domain generalization hierarchy and the associated generalization hierarchy of values, over the values in the domains. In Figure 1 we represent an example of the domain and value generalization hierarchies, based on the quasi-identifying attributes used in our example. The generalization can be applied at two levels: at a single cell level, if it is substituted the cell value with a generalized version of it or at the attribute level, if all the cells in the corresponding column are

generalized. All the values, which differ in the private table, can be easily generalized to the same value and for this value, the number of occurrences is the sum of all occurrences of the values from which the generalization comes. In this way, the generalization enforces the k-anonymity.

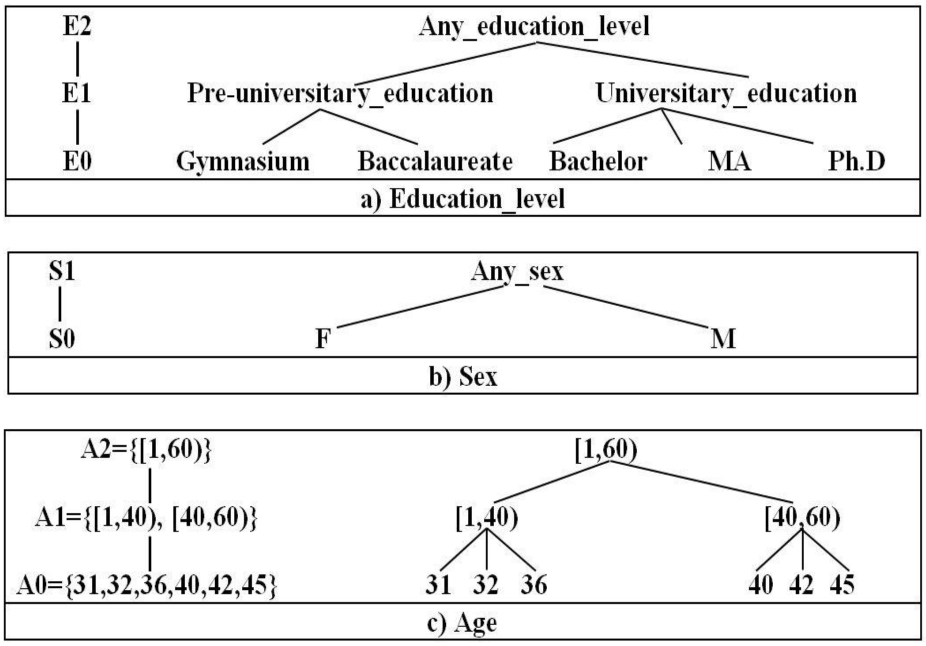

Fig. 1. An example of the domain and value generalization hierarchies

The second technique used for enforcing the k-anonymity on a private table is the suppression technique, which is based on removing sensitive information in order to protect it and is applicable for a single cell, for an entire tuple or for an entire column. The suppression method consists in removing from the table information that forces a large amount of generalization to satisfy a k-anonymity constraint. Therefore, the k-anonymity is satisfied but with less generalization and with a reduced loss of information.

3 Algorithms for enforcing k-anonymity

If the generalization and suppression techniques are applied over a private table, the obtained tables will provide an improved protection of the respondents' identity but the information will be less precise due to the generalization and less complete because some values are suppressed. Therefore, an information loss occurs (both in precision and in completeness) and it is very important to maintain the information loss under control and minimize it.

In the literature, there are many proposed definitions of minimality and studies about the process of finding minimal k-anonymous tables using the attribute generalization and tuple suppression techniques, highlighting that these problems are hard to compute [4] [12] [13]. In order to select the optimal solution, different criteria can be applied to all the tables that ensure minimal information loss and satisfy the definition of minimality. This aspect is of paramount importance in data mining where the usefulness of the data has a great impact over the whole process. In the following, we will present the most important existing algorithms used for obtaining k-anonymous tables.

3.1 The Samarati's algorithm

The Samarati's algorithm is based on the generalization over quasi-identifier attributes combined with the tuple suppression and its main goal is to suppress less tuples by finding a k-minimal generalization [9]. As mentioned before, the generalization technique is based on the domain and value generalization hierarchies and in this case, the definition of the domain generalization hierarchy is extended at tuples of domains, based on the fact that the algorithm operates on a set of attributes. In order to obtain the domain generalization hierarchy of a domain tuple, the involved attributes corresponding to the domain tuple must be generalized and some of the tuples must be suppressed for satisfying the k-anonymity constraint.

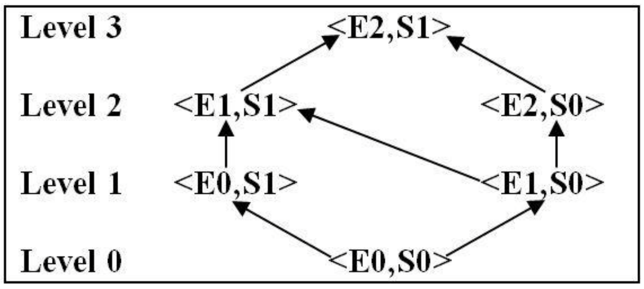

Fig. 2. An example of a domain generalization hierarchy

In Figure 2 it is presented an example of a domain generalization hierarchy obtained by using two quasi-identifying attributes, “Education_level” and “Sex” which have been depicted in Figure 1. The domain tuple is, in this case, <E0, S0>. By generalizing the original private table in accordance with a generalization strategy one can obtain different paths of the hierarchy located at different levels. The algorithm has to compute a generalization that satisfies k-anonymity within the MAX constraint, a threshold that must be specified, representing the maximum number of suppressible tuples. The algorithm evaluates all the solutions at a specific level of the hierarchy starting with the Level 0, and if there isn’t any k-anonymous table that satisfies the MAX threshold, it passes to the next superior level and repeats the procedure until it finds the lowest level where there is a solution that satisfies the k-anonymity constraint.

In our example, the quasi-identifying attributes as shown in Table 1 are “Education_level” and “Sex”, the domain and value generalization hierarchy are depicted in Figure 1, and the generalization hierarchy is represented in Figure 2. If we choose k=4 and the threshold MAX=1, the algorithm starts the verification of solutions at the Level 0. The solution <E0, S0> corresponds to the original table; it is not 4-anonymous and does not satisfy the MAX constraint since the 4-anonymity requires the suppression of the two tuples Ph.D, M. The algorithm verifies the solutions at Level 1, <E0, S1>and <E1, S0> and one can observe that only the solution <E0,S1> is 4-anonymous within the MAX constraint. Therefore, this is the lowest level that has a solution satisfying the k-anonymity constraint and this solution <E0, S1>is considered as minimal.

3.2 The k-Optimizealgorithm

Another algorithm for generalization over quasi-identifier attributes and tuple suppression is called k-Optimize, proposed by Bayardo and Agrawal [14]. For a private table PT and an ordered set of quasi-identifying attributes $QI = \{A_1, \dots, A_n\}$, the k-Optimize algorithm assumes that each attribute $A_i \in QI$ is defined over a totally ordered domain denoted by D_i. If A is an attribute, its generalization on domain D is obtained by partitioning D into a set of ordered intervals $\{I_1, \dots, I_m\}$ with $D = \bigcup_{i=1}^{m} I_i$ and for every elements $v_i \in I_i$and $v_j \in I_j$, if $i < j$ then $v_i < v_j$. For each interval in any domain of the quasi-identifying attributes, the approach associates an integer called index whose assignment reflects on one hand the total order relationship over intervals in the domains and on the other hand the total order relationship among the quasi-identifying attributes [14].

Using the same example as in Table 1, in this private table the quasi-identifying attribute is “Education_level” and the order among values inside this attribute is “Gymnasium”, “Baccalaureate”, “Bachelor”, “MA”, “PhD”. In Figure 3 is depicted the index assignment obtained without the applying of any generalization and considering that each attribute value is represented by an interval.

Education_level

<[Gymnasium]	**[Baccalaureate]**	**[Bachelor]**	**[MA]**	**[Ph.D]>**
1	**2**	**3**	**4**	**5**

Fig. 3. An example of Index assignment to attributes „Education_level”

Taking into account all possible subsets of the set I of index values, without duplications, the k-Optimize algorithm constructs a set enumeration tree over the set I. For a node n, children are obtained through all the sets that contain the elements in n and another single element of the set I appended to n, in accord with the previously defined total order. In Figure 4 is illustrated an example of a set enumeration tree over the set of indexes $I = \{1,2,3,4,5\}$.

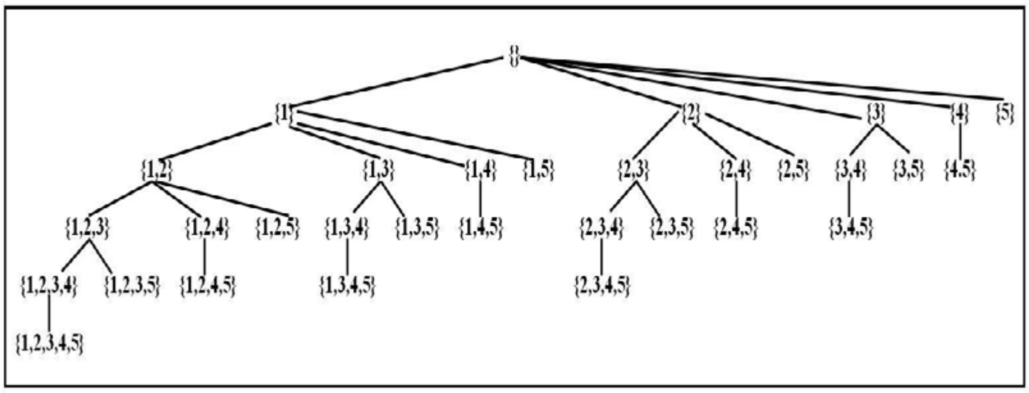

Fig. 4. An example of set enumeration tree over the set $I = \{1, 2, 3,4,5\}$of indexes

Each node in the tree depicts possible options of generalization for the original table PT and therefore the set enumeration tree helps to evaluate and select solutions for the k-anonymity problem. The null node represents the root of the node and by appending one item, which is lexicographically larger than the other items at that particular node of the tree, a new level of the tree is constructed. Therefore, the data’s dimension influences significantly the number of possible nodes in the tree, the increasing being exponentially and even for small values of n it is difficult to build the entire tree.

In order to eliminate this inconvenient, the k-Optimize algorithm uses a pruning strategy based on which a node of the tree is eliminated when no descendent of it can satisfy the k-anonymity requirements. When the maximum computational time has been reached, the algorithm can be stopped and the solution from that point can be used. This technique usually provides satisfactory results but it is possible for it not to be optimal.

4 Related works: software packages that implement certain levels of k-anonymity

In the following, we will depict several software implementations that employ the k-anonymity concept.

One of the most useful implementations is related to a globally optimal k-anonymity method for the de-identification of health data [15], through a globally optimal de-identification algorithm (Optimal Lattice Anonymization) that satisfies the k-anonymity criterion and is suitable for health datasets. It is also presented a comparison between this algorithm and another three existing k-anonymity algorithms (Datafly, Samarati and Incognito) on six public health registry datasets for different values of k and suppression limits. In order to compare these algorithms three information loss metrics have been used: precision, discernibility metric and non-uniform entropy. Each algorithm's performance speed was also evaluated. The conclusion of this study is that for the de-identification of health datasets, Optimal Lattice Anonymization is an improvement on existing k-anonymity algorithms, offering less information loss and faster performance compared to current de-identification algorithms. The Optimal Lattice Anonymization algorithm assumes that the dataset has more than k records and the objective of the algorithm is to find the optimal node in the generalization tree. A node is considered to be optimal if it is k-anonymous and has minimal information loss. The main three steps of the Optimal Lattice Anonymization algorithm are:

- A binary search is used for each generalization strategy to find all the k-anonymous nodes.
- The algorithm retains the k-anonymous node with the lowest height within the strategy for each generalization strategy.
- After the algorithm has selected the k-minimal nodes, it chooses the node with the smallest information loss as the globally optimal solution. Based on a monotonicity property, the k-minimal node with the smallest information loss must also have the smallest information loss among all k-anonymous nodes in the tree.

Another software implementation of the k-anonymity concept is the Datafly algorithm [8]. In order to find a k-anonymous dataset a heuristic method is used. From all the possible quasi-identifiers, the algorithm selects the one with the most distinct values and generalizes it. The algorithm stops if the output generalized dataset is k-anonymous.

In the following, we will depict the Incognito algorithm [8]. First, it starts by considering all possible subsets of the quasi-identifiers, using two optimization techniques:

- When evaluating nodes in the generalization tree, the algorithm tags as k-anonymous the nodes that are above k-anonymous ones in the same generalization strategies.
- If a node is not k-anonymous in a smaller quasi-identifier subset, then, by definition, it will not be k-anonymous in a larger subset of the quasi-identifiers and consequently the lattices for larger subsets of quasi-identifiers can be pruned.

After that, the algorithm evaluates the nodes by starting with a bottom up strategy and by tagging the generalizations of k-anonymous nodes that are found. The main advantage of this approach is a significant reduction in the number of nodes that need to be evaluated. The node with the lowest information loss value is selected as being the optimal solution. There are multiple versions of Incognito.

An interesting application of the k-anonymity concept is the k-anonymous message transmission [16], meaning simple and efficient protocols that are k-anonymous for both the sender and the receiver. In order for a communication protocol to be considered sender k-anonymous, it must assure that an adversary who tries to determine the identity of a particular message's sender can only narrow its search to a certain set of k suspects. A similar guarantee must be assured in what concerns the receiver in order to be considered receiver k-anonymous. In [16] there are presented a series of protocols that are k-anonymous for both the sender and the receiver in a described model where a polynomial time adversary is able to see all the traffic within a network and control a constant fraction of the participants. The protocol does not require the existence of trusted third parties and adds robustness against adversaries who try to disrupt the protocol through perpetual transmission or selective non-participation.

5 The Compute Unified Device Architecture approach for the k-anonymity concept

Graphics Processing Units have been used for a long time solely to accelerate graphics rendering on computers [17]. In order to satisfy the increasing need for improved three-dimensional rendering at a high resolution and a large number of frames per second, the GPU has evolved from a one-purpose specialized architecture to multiple purposes complex architectures, able to do much more than just provide video rendering. The acceleration of a broad class of applications became possible once with the introduction of the NVIDIA Compute Unified Device Architecture. The architecture and the main characteristics of the NVIDIA GPUs are summarized in Figure 5.

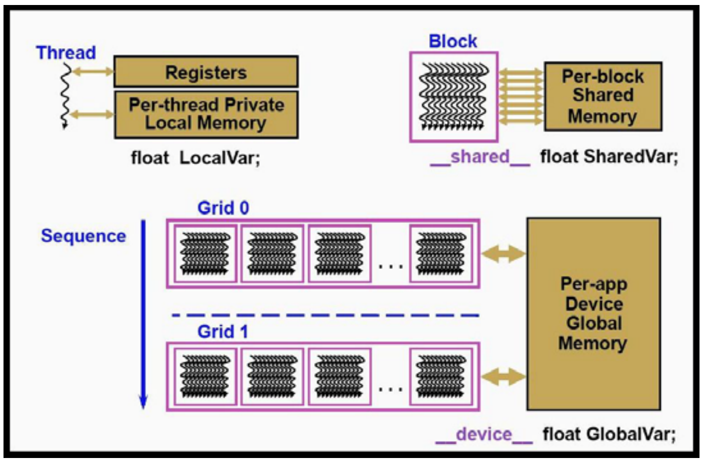
Thread
Registers
Per-thread Private Local Memory
float LocalVar;
Block
Per-block Shared Memory
__shared__ float SharedVar;
Sequence
Grid 0
Grid 1
Per-app Device Global Memory
__device__ float GlobalVar;

Fig. 5. NVIDIA Compute Unified Device Architecture (CUDA)[17]

CUDA is a software and hardware architecture that enables the NVIDIA graphics processor to execute programs written in C, C++, FORTRAN, OpenCL, Direct Compute and other languages. A CUDA program invokes more parallel program kernels. The kernel processes in parallel each set of parallel threads. The programmer or compiler manages these threads by grouping them into thread blocks (consisting of more threads) and grids of thread blocks (consisting of more thread blocks).

The GPU processor instantiates a kernel program on a grid containing parallel thread blocks. Each thread from the block executes an instance of the kernel and has a unique ID associated to registers, to thread's private memory within the thread block [17].

The Compute Unified Device Architecture hierarchy of threads is mapped to the hierarchy of the graphics processing units' hardware processor; a GPU executes one or more kernel grids; a streaming multiprocessor (SM) executes one or more thread blocks; the CUDA cores contained in the streaming multiprocessor SM run the threads within blocks. A streaming multiprocessor SM can process up to 32 groups of threads called warps. Regarding memory hierarchy, each multiprocessor contains a set of 32-bit registry with a zone of shared memory, which is easily accessible for each core of the multiprocessor but hidden from other multi-processors. Depending on the generation of a GPU, the number of registry and the size of shared memory vary. Besides shared memory, a multiprocessor contains two read - only memory caches, one for texture and another one for constants.

In order to improve software performance when programming in CUDA, developers have to optimize the number of concomitant active threads and balance each thread's resources: number of registers and threads per multiprocessor, global memory bandwidth and the amount of on-chip memory assigned per thread. Performance increases have been obtained by reordering accesses to off-chip memory in order to manage requests referring to the same memory locations (or contiguous memory locations). By applying these techniques, many applications improved their execution time up to 457X in kernel codes and 431X at

a general level [17].
In the NVIDIA CUDA programming model [17]a system is comprised of a traditional CPU (representing the host) and one or more massively data-parallel coprocessors (representing the devices). The CUDA runtime has library functions for managing both the device memory and transfers from the host to the compute devices.
All concurrent threads are based on the same code even if they may follow different paths of execution because each CUDA device processor supports the Single-Program Multiple Data (SPMD) model [17] and each thread resides in the same global address space. Data parallel functions, called kernels and data structures, corresponding to the compute devices, comply with standard ANSI C extended with keywords. A kernel is usually invoked on thousands of threads and describes the work of a single one. Inside thread blocks, through built-in primitives, threads synchronize their actions and share their data. The CUDA programming model enables a program's components, which are suited for data parallelism, to be separated and executed on a specialized massive data parallelism coprocessor. A detailed overview on the CUDA programming model is depicted in [17].This architecture offers a high degree of flexibility when it comes about allocating local resources like registers or local memory in threads. The programmer divides local resources among threads and every CUDA core can process a variable number of threads. Although this flexibility offers a high degree of control over an application performance, it also has a great impact on optimizing the performance of applications. Another important aspect is related to the way GeForce GTX480 can execute applications and what are the elements that improve or limit its performance. Numerous software applications were ported and evaluated on the CUDA platform as a result of its huge data processing power [17].

5.1 The algorithm's description

Using the Compute Unified Device Architecture, we have developed an algorithm that analyzes the k-anonymity approach and is suitable for extracting tuples with the same quasi-identifying values from a large database structured as a private table. The number of tuples with the same quasi-identifying values in a private table is important when evaluating the k-anonymity level and selecting possible options for suppression or generalization in order to obtain a desired level of k-anonymity. We took into consideration the main aspects of improving a CUDA application performance and GPU memory management through a sequence of progressively optimized kernels. The algorithm has been developed in two versions.
In the first version, denoted KNV1, threads within each thread block access data by using the texture memory, starting at different positions within the database while threads with the same ID from different blocks are starting from the same position.
The second version of the algorithm, denoted KNV2, uses block-level parallelism with shared memory database buffering. Instead of using the texture memory, this version loads a block of data from the database into a buffer of shared memory, processes data from the buffer, then loads another block of data in the buffer and the process is repeated until the entire database has been processed. The starting point for each thread of KNV2 depends on buffer size and not on the size of the database (as in KNV1). A thread will always access the same shared memory area during all searches, but data from the shared memory will change when buffer updates.
In the following, we will define four important requirements that were taken into account when we have designed the algorithm that analyses the k-anonymity approach and is suitable for extracting tuples with the same quasi-identifying values from a large database structured as a private table. These requirements represent a minimal necessary set and if they are not met then the de-identified data (obtained in order to k-anonymize the database) might become useless. The requirements are based on the

available documentation [15] in this domain and were of paramount importance in developing the algorithm.

- The values in the quasi-identifiers are generalized by reducing their precision and, therefore, the quasi-identifiers are represented as hierarchies and a de-identification algorithm needs to deal with this hierarchical nature of the variables.
- For all values of a given quasi-identifier a total order relationship is presumed and a quasi-identifier can be recorded to any partition of the values that preserves the order. The users of the data need to specify the interval sizes that are appropriate for the analysis that they will perform. Otherwise, if this partitioning is performed automatically, it may produce intervals of unequal sizes that are a disadvantage because it makes the analysis of such data quite complex and significantly reduces its utility.
- A very practical approach is to use global recording where all the records have the same recording within each variable, i.e. to use global recording instead of local recording. If the k-anonymity algorithm used the local recording, the generalizations performed on the quasi-identifiers are not consistent across all of the records and such inconsistency in constructing response categories makes the data very difficult to analyze in practice using standard data analysis techniques.
- When designing an algorithm for obtaining a de-identification solution, one must take into account that a globally optimal algorithm satisfies k-anonymity but at the same time minimizes information loss. A globally optimal solution prevents excessive information loss that would have led to inaccurate analysis results and inefficient use of data.

In order to improve the performance of extracting tuples having the same quasi-identifying values the following technical issues must be taken into consideration [17]:

- To assure a reduced bandwidth usage and to minimize the redundant execution, a programmer must optimize the use of the on-chip memory. This memory is called shared memory, is software managed and along with a register file it represents the working memory within a group of cores. The shared memory has low latency and is partitioned among all the thread blocks that belong to the same streaming multiprocessor during the runtime. The inter-thread data can be reused because all data in the shared memory is shared among threads from the same thread block. Even if there is a small increase in the registers or shared memory usage per thread, the number of simultaneous executed threads diminishes greatly.
- Using synchronization each thread can communicate only with other threads within the same thread block and there is no communication within threads from other blocks. Therefore, hardware resources do not have to be virtualized and so the hardware becomes highly scalable. The same program written in CUDA can be executed successfully on different generations of GPUs (for example one can use a GTX480 as well as a GTX280) but a single kernel call has a limited parallelism that can be applied.
- Every GPU thread has its own private per thread memory, private registers, program counter and thread execution state. Each thread can execute an independent code path. The GPU processor executes and manages at hardware level hundreds of concurrent threads avoiding scheduling overhead and hiding memory latency. The Fermi architecture offers 512 execution cores; a GTX480 has 480 execution cores available for use. Hundreds of threads are needed for all these cores to be completely occupied. The high latency of global memory is also an important technical issue that must be taken into consideration when a programmer defines the threads in order to improve the software performance in CUDA. While CPU designs use large caches to hide memory latencies, CUDA generates and uses thousands of active threads. In contrast to

traditional multicore systems, programmers may have to define threads at a finer granularity in order to assure that there are a sufficient number of threads and that there is a high compute-to-memory-access ratio in order to avoid saturation of memory channels.

5.2 Experimental Results

The data used for performance testing was exported from the AdventureWorks Sample Databases included in SQL Server 2008R2. The following configuration has been used: Intel i7-2600K at 3.4 GHz with 8 GB (2x4GB) of 1333 MHz DDR3, dual channel. The GPU used was GeForce GTX480 (from FERMI architecture). Programming and access to the GPUs used the CUDA toolkit 3.2.16-64 bits with NVIDIA driver version 266.58. In addition, all processes related to graphical user interface have been disabled to reduce the external traffic to the GPU. The benchmark took into consideration two stages. In the first one, the KNV1 and KNV2 algorithms computed the k-anonymity coefficient, denoted by $k01$, respectively by $k02$, corresponding to the data table and it is also computed and recorded the necessary amount of time for obtaining the result, using different number of treads per block. In this stage, we represented the effect of algorithm selection on execution time (measured in milliseconds) at different sizes of thread blocks (Figure 6).

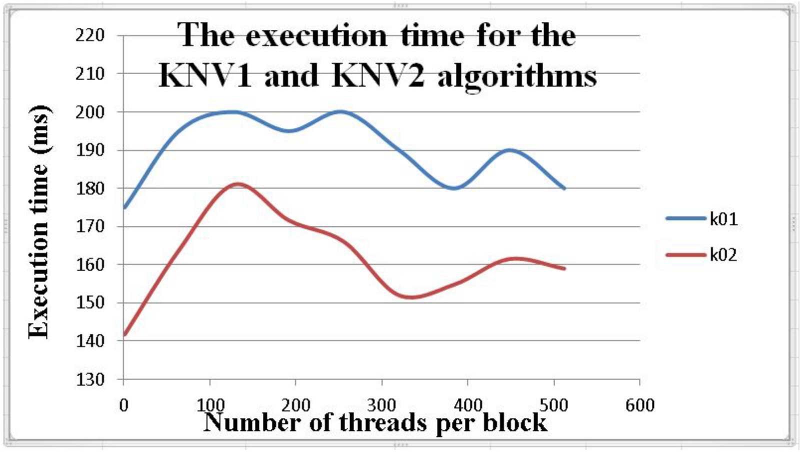

Fig. 6. The effect of algorithm selection on execution time at different sizes of thread blocks

Analyzing Figure 6 one can observe that the execution time for obtaining the k-anonymity coefficient is less in the case of the KNV2 algorithm than in the case of the KNV1 algorithm. Even if the buffering causes an increasing of the execution time in parallel thread processing, this increase is amortized. Algorithm KNV2 uses a buffer zone to combine the memory bandwidth of all threads in a memory block to reduce the texture load. This implies a long execution time because only one block can be resident on a multiprocessor at a time during the loading phase and other processing cannot be done. As more threads are added to a block, the execution time for Algorithm KNV2 decreases. This feature shows that Algorithm KNV2 is able to use the processing power of a large number of threads. As the number of threads increases, more results will be quickly calculated since all threads can access the shared memory block without additional resource consumption (until the moment when planning a large number of processes on the multiprocessor exceeds the total calculation time).

In the second stage, the KNV1 and KNV2 algorithms compute the necessary time for obtaining certain desired k-anonymity coefficients, denoted by $ki1$ and $ki2, i = 1,2,3\ldots$ (using generalization and suppression techniques). In this stage, we represented the effect of k-anonymity coefficient’s selection on execution time (measured in milliseconds) at different sizes of thread blocks for each of the algorithms. If the desired k-anonymity coefficient cannot be reached, the necessary time for reaching this conclusion is registered and represented (Figure 7, Figure 8).

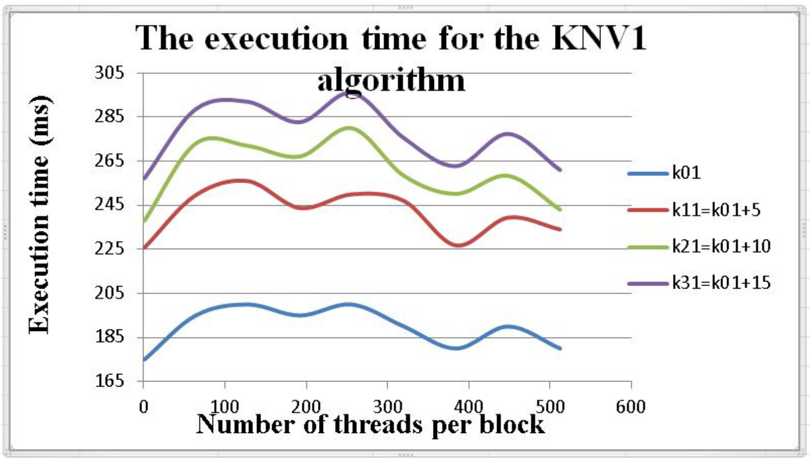

Fig. 7. The effect of k-anonymity coefficient's selection on execution time at different sizes of thread blocks for KNV1 algorithm

In Figure 7 there were represented four cases: the execution time for obtaining the k-anonymity coefficient of the original table $k01$ and execution times for obtaining another three increased levels of k-anonymity, $k01+5$, $k01+10$, $k01+15$ in the case of the KNV1 algorithm. Analyzing the obtained results one can observe that the execution time for obtaining the k-anonymity coefficient increases with the desired k-anonymity coefficient and the most significant increase is from $k01$to $k11$. The graphic shape is mostly the same in all four studied cases.

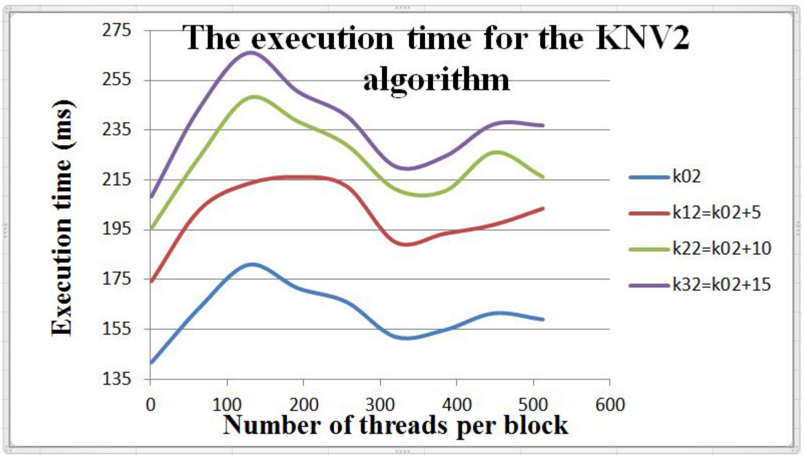

Fig. 8. The effect of k-anonymity coefficient's selection on execution time at different sizes of thread blocks for KNV2 algorithm

In Figure 8 there were represented four cases: the execution time for obtaining the k-anonymity coefficient of the original table $k02$ and execution times for obtaining another three increased levels of k-anonymity, $k02+5$, $k02+10$, $k02+15$ in the case of the KNV2 algorithm. Analyzing the obtained results one can observe that the execution time for obtaining the k-anonymity coefficient increases with the desired k-anonymity coefficient and the most significant increase is from $k02$to $k12$. The graphic shape is mostly the same in all four studied cases. Comparing Figure 7 and Figure 8, one can observe that the execution time for obtaining the k-anonymity coefficient (and the increased levels of k-anonymity) is less in the case of the KNV2 algorithm than in the case of the KNV1 algorithm. The increasing of the execution time in parallel thread processing caused by the buffering is amortized.

6 Possible attacks against k-anonymity

The k-anonymity has the potential to protect the identity and privacy of individuals referred to the data in question, but solutions that use the k-anonymity are still vulnerable to attacks even if the quasi-identifiers are very carefully chosen. We depict below three of the possible attacks and several methods to counter them [6].

1. The unsorted matching attack against the k-anonymity is based on the order of appearance of tuples in the released table, which can leak sensitive information if it is a related one. The solution against this problem is the randomly sort of tuples in the solution tables.
2. The complementary release attack against k-anonymity is based on the fact that usually the quasi-identifiers are a subset of the released attributes and as a consequence when a table T is released, even if it respects the k-anonymity, one must take into account that it could contain other external information. Therefore, subsequent releases of the same privately held information must consider all the attributes that have been previously released in order to prohibit linking on T [6].
3. The temporal attack against the k-anonymity is based on the dynamic nature of a data collection and on the fact that the tuples are frequently added, changed or removed. In this situation, releases of generalized data over time can be exposed to a temporal inference attack. We consider at a given moment of time an original private table T1 which leads to a solution based on the k-anonymity and a new table is released. We also consider another moment of time, when at the original table are added additional tuples, we denote this table by T2 and then a k-anonymity solution based on this new table is released. Linking the released tables may reveal sensitive information and thereby compromise k-anonymity protection. In order to prevent this problem, all of the attributes of the first released table would be used for subsequent releases.

7 Conclusions

In this paper, we have discussed the k-anonymity concept and different approaches of its implementation. We formalized the main theoretical notions regarding the k-anonymity and then we highlighted the application of the k-anonymity to the data mining process by developing a practical and intuitive example. We have developed, using the Compute Unified Device Architecture, an algorithm that analyzes k-anonymity and is suitable for extracting tuples with the same quasi-identifying values from a large database structured as a private table. The k-anonymity has the potential to protect the identity and privacy of clients who use e-services that employ data mining techniques.

We have tried this novel approach because numerous e-services use data mining techniques and to our best knowledge, at this moment the scientific literature lacks in the aspect of ensuring the privacy of the disclosed data. Because the k-anonymous data mining is a recent research area, many research issues are still open and worth being investigated. A first example is the possibility of combining k-anonymity with other scientific fields that employ the data mining process. Another example is the development of new optimized algorithms used for obtaining k-anonymous tables, designed to process huge amounts of data by using the increased computational power of novel parallel processing architectures.

We believe that developing and implementing a powerful solution for preserving the privacy of e-services that implement the data mining process is of a paramount importance, worth to be further developed.

References

[1] A. Buchmann, F. Casati, L. Fiege, M.C. Hsu, M.C. Shan, (Eds.), "Technologies for E-Services", in *Proc. Third International Workshop, TES 2002*, Vol. 2444, Hong Kong, China, 2002, pp. 175-233.

[2] S. Krishnaswamy, A. Zaslavsky, S. W.Loke, "Internet Delivery of Distributed Data Mining Services: Architectures, Issues and Prospects", Chapter 7 in the book *Architectural Issues of Web-enabled Electronic Business*, Murthy, V.K. and Shi, N. (eds.), Idea Group Publishing, 2003, pp. 113 - 127.

[3] R. Agrawal, R. Srikant, "Privacy-preserving data mining", in *Proc. of the ACM SIGMOD Conference on Management of Data*, Dallas, Texas, May 2000, pp.439-450.

[4] C.C. Aggarwal, P. S. Yu, *Privacy-Preserving Data Mining, Models and Algorithms,* Series: Advances in Database Systems, Springer, Vol. 34, pp. 105-134, 2008.

[4] Y. Lindell, B. Pinkas, "Privacy preserving data mining", *Journal of Cryptology*, vol.15, no.3, pp. 177–206, June 2002.

[5] L. Sweeney, K-anonymity: A model for protecting privacy, *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 557-570, 2002.

[6] Z. Yang, S. Zhong, R. N. Wright, "Privacy preserving classification of customer data without loss of accuracy", in *Proc. of the 5th SIAM International Conference on Data Mining*, Newport Beach, California, April 2005, pp.560-567.

[7] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, P. Samarati, "K-anonymity", in T. Yu and S. Jajodia, editors, *Security in Decentralized Data Management*, Springer, Berlin Heidelberg, 2007, pp. 323-353.

[8] P. Samarati, "Protecting respondents' identities in microdata release", *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 8, pp:1010–1027, November 2001.

[9] P. Samarati, L. Sweeney, "Generalizing data to provide anonymity when disclosing information", in *Proc.The 17th ACM-SIGMOD-SIGACT-SIGART Symposium on the Principles of Database Systems*, Seattle, WA, 1998, pp. 188-198.

[10] T. Dalenius, "Finding a needle in a haystack – or identifying anonymous census record", *Journal of Official Statistics*, vol. 2, nr.3, pp. 329-336, 1986.

[11] A. Meyerson, R. Williams, "On the complexity of optimal k-anonymity", in *Proc. of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Paris, France, June 2004, pp. 223-228.

[12] A. Evfimievski, R. Srikant, R. Agrawal, J. Gehrke, "Privacy preserving mining of association rules", in *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 2002, pp. 75-82.

[13] R. J. Bayardo, R. Agrawal, "Data privacy through optimal *k*-anonymization", In *Proc. of the International Conference on Data Engineering (ICDE'05)*, Tokyo, Japan, April 2005, pp. 217–228.

[14] K. Emam, F. Dankar, R. Issa, etal, "A Globally Optimal k-Anonymity Method for the De-Identification of Health Data", J*ournal of the American Medical Informatics Association*, vol. 16, no. 5, pp. 670-682 September-October 2009.

[15] L. Ahn, A. Bortz, N. Hopper, "K-anonymous message transmission", in *Proc. The 10th ACM conference on Computer and communications security*, New York, USA, 2003, pp. 122-130.

[16] A. Pîrjan, "Improving software performance in the Compute Unified Device Architecture", *Revista*

Informatica Economica, vol. 14, no. 4, pp. 30-47, 2010.

Ion LUNGU is a Professor at the Economic Informatics Department at the Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies of Bucharest. He has graduated the Faculty of Economic Cybernetics in 1974, holds a PhD diploma in Economics from 1983 and, starting with 1999 is a PhD coordinator in the field of Economic Informatics. He is the author of 41 books in the domain of economic informatics, 57 published articles (among which 20 articles ISI indexed or included in international databases) and 39 scientific papers published in conferences proceedings (among which 5 papers ISI indexed and 15 included in international databases). He participated (as director or as team member) in more than 20 research projects that have been financed from national research programs. He is a CNCSIS expert evaluator and member of the scientific board for the ISI indexed journal Economic Computation and Economic Cybernetics Studies and Research. He is also a member of INFOREC professional association and honorific member of Economic Independence academic association. In 2005 he founded the master program Databases for Business Support, who's manager he is and in 2010 he founded Databases Journal. His fields of interest include Databases, Design of Economic Information Systems, Database Management Systems, Decision Support Systems, Executive Information Systems, Business Intelligence.

Alexandru PÎRJAN has graduated the Faculty of Computer Science for Business Management in 2005. He holds a MA Degree in Computer Science for Business from 2007. He joined the staff of the Romanian-American University as a stagier teaching assistant in 2005 and a Lecturer Assistant in 2008. He is a PhD candidate since 2009 at the Doctoral School from the Bucharest Academy of Economic Studies. He is currently a member of the Department of Informatics, Statistics and Mathematics from the Romanian-American University. He is the author of more than 20 journal articles, and a member in 4 national scientific research projects. His work focuses on database applications, artificial intelligence and quality of software applications.