A Semantic Approach for Semi-Automatic Detection of Sensitive Data

Jacky AKOKA[1], Isabelle COMYN-WATTIAU[2], Cédric DU MOUZA[3], Hammou FADILI[4], Nadira LAMMARI[3], Elisabeth METAIS[3], Samira SI-SAID CHERFI[3]

1 CEDRIC-CNAM & Institut Mines-Telecom TEM, Paris, France, akoka@cnam.fr
2 CEDRIC-CNAM & ESSEC Business School, Paris, France, wattiau@cnam.fr
3 CEDRIC-CNAM, Paris, France, {dumouza, lammari, metais, samira.cherfi}@cnam.fr
4 Maison des Sciences de l'Homme, fadili@msh-paris.fr

ABSTRACT

We propose an innovative approach and its implementation as an expert system to achieve the semi-automatic detection of candidate attributes for scrambling sensitive data. Our approach is based on semantic rules that determine which concepts have to be scrambled, and on a linguistic component that retrieves the attributes that semantically correspond to these concepts. Because attributes cannot be considered independently from each other, we also address the challenging problem of the propagation of the scrambling process through the entire database. One main contribution of our approach is to provide a semi-automatic process for the detection of sensitive data. The underlying knowledge is made available through production rules, operationalizing the detection of the sensitive data. A validation of our approach using four different databases is provided.

Keywords: Security, integrity, protection, database semantics, expert system.

INTRODUCTION

In an ever-changing competing environment, organizations are under increasing pressure to find ways of protecting the sensitivity of data regarding both individuals and organizations. Companies face the challenge of creating and/or updating non-production environments for testing purposes. In general, companies create a copy of the production system, which may include the data repository and the administrative settings. They provide a test environment for improving application delivery process. However, there are many risks associated with the test environments open to external consultants. In some cases, testing can become a liability. Thus, there is a need to provide realistic test data. This requires masking techniques for sensitive data. By masking the data, users see only a representation of the data without having access to the sensitive ones. Data related to customers, products, materials, and financial accounts may be sensitive and should be masked or anonymized. Sensitive information like address, telephone numbers, and contact information has to be de-identified. By scrambling the data, we substitute sensitive information on customers, orders, products, order profitability, etc., with fictitious, but still consistent data, preserving the overall structure and semantics of the test database. Data masking, also referred as data obfuscation, data de-identification, data depersonalization, data scrubbing, etc., represents a solution for data protection from both internal and external security threats. It enables the creation and/or the updating of data in non-production environments, without the risk of exposing sensitive information to unauthorized

users, such as external consultants in environments like ERP systems. Let us mention that, unlike encrypted data, masked data maintain their usability in testing environments. Data masking encompasses several techniques such as generalization, mutation algorithms, customization, etc. It can use shuffling techniques for names protecting. A related technique called linked shuffling can de-identify the address. Phone numbers can be scrambled using random number generators. Date transformer allows obfuscating dates. Finally, account generator performs data de-personalization of account numbers. Thus, data masking tasks provides several benefits, such as providing realistic data for off-site and offshore software testing. Even if techniques and tools are available, scrambling huge databases is a fastidious process. Surveys reveal that companies often neglect to scramble their datasets, generating business and financial risks. ERP systems rely on thousands of tables, each one composed of more than twenty columns. Deciding which column is sensitive and has to be scrambled is an enormous task.

Protecting data privacy using scrambling is composed of three steps. The first step deals with the choice of data to be hidden, notably anonymized, randomized, swapped or, more generally, obfuscated (Bakken, Parameswaran, Blough, Franz, & Palmer, 2004). The second step consists in choosing, for each sensitive part of the database, the adequate scrambling technique, particularly but not exhaustively among those mentioned above (Fung, Wang, Chen, & Yu, 2010). The third step is related to the application of data sanitization to the entire dataset preserving data integrity. To the best of our knowledge, the literature and industrial solutions are concentrated on this third step (Ravikumar et al., 2011; Askari, Safavi-Naini,& Barker, 2012). This paper contributes to the first step by proposing an innovative technique that automates the detection of the sensitive attributes. By semantically modeling the different data, we enable the semi-automatic detection of data sensitivity. This technique encompasses two functionalities: (1) automatic detection of the values to be scrambled; that have to be validated by a domain expert and, (2) automatic propagation to other semantically linked values.

Our contribution is original in the sense that it encapsulates general and domain knowledge into rules. We propose a rule-based approach implemented under an expert system architecture. Rules are devoted to the selection of sensitive data with regard to their semantics. Furthermore we present a deduction mechanism modeled by a semantic graph to ensure the propagation of the sensitivity on near values and the consistency with the other relations. Moreover we propose a prototype with a set of clever interfaces to capture the rules. Let us mention three important aspects of our approach:

i) It is well-known that which information acts as identifying depends on the adversary's knowledge. The main assumption underlying our approach is that the adversary model is embedded in the domain knowledge. To this end, our expert rules capture this domain knowledge.

ii) Another originality of our approach is that our scrambling process preserves data integrity and generalizes this notion to take into account a more comprehensive notion of data dependency.

iii) Finally, the automatic detection of sensitive data represents a strength of our approach since companies facing privacy-preserving problem cannot perform manually this detection especially with huge databases containing several thousands of tables and columns.

Motivating example

A typical example of a large database which contains sensitive data, and which is often outsourced when new managing software is developed, is a Human Resources Department (HRD) database. This database stores basically information about the employees like employee's id, name, city,

department, wage, etc. As an illustration for our proposal, let's consider the sample HR schema provided by Oracle, enriched with additional information on employees such as the evaluation by the hierarchy (manager_evaluation). We implemented this schema and populated the tables (see Figure 1).

Countries (**country_id,** country_name,#region_id)
Departments (**department_id**, department_name, #manager_id, #location_id)
Employees (**employee_id**, first_name, last_name,email, phone_number, hire_date, #job_id,salary, commission_pct, #manager_id, #department_id)
Jobs (**job_id**,job_title,min_salary,max_salary)
Jobs_history (**#employee_id, start_date**, **end_date**, #job_id, #department_id, manager_evaluation)
Locations (**location_id**, street_address, postal_code, city, state_province, #country_id)
Regions (**region_id**,region_name)

Figure 1. The motivating example database structure

This database contains sensitive information about the employee salary or evaluation, but also about its coordinates (phone, email, etc.). Some attributes are obviously identified as sensitive and should be scrambled. Others like manager_id that gave an evaluation or the (start_date, end_date) that allows isolating an employee, for instance, are not so easily identified, especially when the number of tables reaches dozens or hundreds.

The paper is organized as follows. Next section summarizes related work. The following sections are dedicated respectively to the definition of the main concepts involved in our approach and to the description of the different components of our approach, allowing us to set the rule base for detecting sensitive values. In the following section, we introduce the propagation mechanism of attributes to be scrambled. Next section presents the scrambling process. Finally, we validate our approach before concluding and describing future work.

RELATED WORK

Determining a sanitization strategy which guarantees that the data provided preserve sensitivity is a complex task. In Atallah, Elmagarmid, Ibrahim, Bertino, and Verykios (1999), the authors prove that finding the sanitization that minimizes the sensitivity of the values with respect to some sensitive rules is a NP-hard problem. A large number of heuristics have been proposed (Wang & Lee, 2008; Chakaravarthy, Gupta, Roy, & Mohania, 2008) to find a satisfying sanitization under precise hypotheses.

A first family of approaches is based on sensitive association rules. These approaches hide the frequent itemsets corresponding to these rules by modifying the sensitive transactions that contain those itemsets. In Oliveira and Zaïane (2003), for instance, the authors present a privacy preservation heuristic algorithm named sliding window algorithm (SWA) that hides, in one pass on a transactional database, association rules by decreasing their support. Amiri (2007) proposes three heuristics also based on rules that outperform SWA in terms of maximizing data utility of the sanitized databases but that require computational overhead.

Several approaches are semantics-free and rely on the number of occurrences inside each equivalence class (i.e. a set of records that could not be distinguished w.r.t. a given identifying attribute). The most famous ones are k-anonymity (Sweeney, 2002) that imposes for a class to contain at least k records, and l-diversity (Machanavajjhala et al., 2006) that improves the k-anonymity by forcing equivalence classes to contain at least l well represented values for each sensitive attribute. Li et al. (2007) go beyond both k-anonymity and l-diversity, and define t-closeness that requires that the distribution of an attribute in an equivalence class is close to the one of the real table. Since discovering frequent patterns in databases is largely used for commercial purposes, some approaches hide sensitive patterns like in (Wang & Lee, 2008). However all these approaches assume that sensitive attributes or patterns are known and do not consider links between attributes.

Other proposals have been devoted to the sanitization of free-text, mainly in the medical domain (Neamatullah et al., 2008). However the problem is different in free-text and consists basically in identifying sensitive words based on specialized domain semantics. They do not consider any links between terms except potentially synonymy and usually do not aim at guaranteeing any data utility after sanitization. One interesting exception for health information is (Gardner & Xiong, 2008) that presents a prototype for extracting information and identifying entities. They applied an anonymization process for both structured and unstructured data. Here again authors rely on k-anonymity and l-diversity to determine sensitive attributes. Chakaravarthy et al. (2008) present the Erasme framework for sanitization of unstructured documents based on term scoring functions. However no link between attributes is considered. (To, Dang, & Küng, 2011b) address the problem of obfuscation of spatiotemporal data, such as users' location. Conventional querying process consists of two phases, first querying the database to retrieve the accurate positions of users and then modifying them to decrease the quality of location information, which is time-consuming. The authors propose a structure called OST-tree that embeds the user's privacy policy in its node and obfuscates the spatiotemporal data. (To, Dang, & Küng, 2011a) is a B+− tree variant that contains geographic aware information on its nodes to perform obfuscation of location information while processing database-level queries. Shou, Shang, Chen, Chen, & Zhang (2011) study the anonymization of time series data while supporting complex queries, such as range and pattern matching queries, on the published data. They propose an anonymization model called (k,P)-anonymity for pattern-rich time series.

Anonymization may be required for various applications. For instance, for identity obfuscation in graphs, Bonchi, Gionis, & Tassa (2011) consider three types of methods: the first category provides k-anonymity in graphs by adding or deleting edges. The second category consists in adding noise to the data in the form of random additions, deletions or switching of edges. The last category does not alter the graph data; instead, they group together vertices into super-vertices of size at least k, which is the threshold of anonymity. The graph obfuscation is a hot topic in social networks. In another context, Saini, Atrey, Mehrotra, & Kankanhalli (2011) deal with the problem of anonymity dedicated to the images in the context of video surveillance. Hiding the faces is not sufficient when the context on the image is sufficient to de-anonymize the picture, given the adversary knowledge.

Several softwares are proposed to de-identify databases (*Datamasker*, n.d.; *Camouflage*, n.d.; *Solix*, n.d.; *Datavantage Globa*, n.d.; *Pse Data Security*, n.d.; *JumbleDB Orbium Software*, n.d.; *HCM Data Scrambler*, n.d.; *HCM Data Scramble Tool*, n.d.). They basically offer the same functionalities, i.e., to select sensitive attributes, to choose a scrambling technique among a set (shuffling, replacing with synthetic data, masking, deleting, encrypting, etc.) to apply for each

attribute. (*Datamasker*, n.d.) proposes enhanced functionalities like using templates for replacing data with adapted synthetic data or respecting integrity constraints (within tuples, between tuples or between tables). (*JumbleDB Orbium Software*, n.d.) capitalizes on domain knowledge by allowing database administrators to store simple rules such as column names containing SAL string (for salary). However this detection is limited to a comparison between columns in tables and words in a dictionary which is supposed to contain the names of sensitive columns. (*HCM Data Scrambler*, n.d.; *HCM Data Scramble Tool*, n.d.) are dedicated to human resources databases. Vinogradov and Pastsyak (2012) present an evaluation of different anonymization tools. Nonetheless, no tool provides any help for detecting sensitive attributes that can lead to important security flaws.

SENSITIVE INFORMATION

A database in production may contain sensitive information that must not be visible (or at least exploitable) when the database is used during development or test phases. We distinguish *identity information* that allows identifying a person or an entity stored in the database from *confidential information* whose content may be harmful if revealed. We are convinced that both kinds of information must be considered when sanitizing a database. Thus we consider the following definitions.

Let D be a database and S be the set of all attributes in D. Let k be a parameter that depends on the application and that represents the minimal number of occurrences required for assuming anonymity (see the *k*-anonymity approach).

Definition 1 Confidential attribute. *The confidential attributes set, denoted $S_c \subseteq S$ is the set of attributes whose content is confidential, whatever their number of occurrences.*

Definition 2 Identifying attribute. *The k-identity attributes set, denoted $S_i \subseteq S$ is the set of attributes such that for any $x \in S_i$ it exists a subset $s_i \subseteq S_i$ within a single table T of D and with $x \in s_i$, such that (i) each instance of s_i occurs less than k times in the records from T and (ii) there is an attribute $y \in S_c$ in T. We call in the following, an element from S_i, a k-identifying attribute (identifying attribute for short).*

In other words, each instance of an (or a group of) identifying attribute has less than *k* occurrences, and is considered selective enough to identify a small number of persons. If there is a confidential attribute in the same table, it means that the individual privacy is endangered. Note that we assume *k* set for the application, but we can easily extend our definition to capture applications where a different value for *k* is set for each table. Our notion of *identifying attribute* is similar to the notion of quasi-identifier in (Sweeney, 2002) except it cannot be considered independently from the confidential attributes. Observe that $Si \cap Sc$ may be not empty. Finally we define a sensitive attribute as follows:

Definition 3 Sensitive attribute. *The sensitive attributes set, denoted S_s, is the set of identifying and confidential attributes for the table D, i.e., $S_s = S_i \cup S_c$.*

The rationale for considering both confidential and identifying attributes in the scrambling process is based on the following observations. The scrambling of identity attributes preserves anonymity. However confidential attributes keep their initial distribution, which is clearly not sufficient when the presence of some instances of attributes must remain itself sensitive or at least difficult to exploit. Conversely, the scrambling of confidential attributes aims at protecting

individual privacy by modifying their value while information that identifies persons remains unchanged. But in this case, local (*e.g.*, value range, precision, etc.) and global (*e.g.*, average, min, max, etc.) properties of the concerned attributes are changed. This may invalidate a test phase. Consequently, both types of attributes must be simultaneously considered as sensitive and thus candidates for scrambling.

Example 1. *In our HRD database example, employee_id or (first_name, last_name) permit to identify an employee. So Si = {employee_id, first_name, last_name}. Information on address, department and wage properties are apparently less sensitive. Nonetheless, one may avoid revealing the highest salary or the minimal salary of a given job. Such properties must then be considered as confidential (Sc = {min_salary, max_salary}). Moreover in smaller companies one can argue that the couple (start_date, end_date) for a job is sufficient to identify a small subset of employees and consequently must be added to the k-identity set also, while for larger companies this information is not identifying enough. So finally for our large company we have to scramble Ss = {employee_id, first_name, last_name, min_salary, max_salary, start_date, end_date}.*

The main concepts of our approach are represented in a meta-model described at Figure 2. Information in a database is structured within tables. Tables are composed of columns on which constraints are defined. There is a variety of constraints such are primary key, referential integrity, domain constraints, etc. In our approach we are interested in what we call sensitive information. The latter refers to a single column or to a group of columns and could also include constraints that allow us, under certain conditions, to infer sensitive information upon non sensitive information. Information to be scrambled appears in grey. Sensitive information includes identifying data and confidential data. Notice that identifying data comprise not only keys but also quasi-identifiers (e.g. birth date, sex, and postal codes are considered as quasi-identifiers, since some surveys have shown that very few persons share the same birth date, sex and postal codes). In some cases, only some instances are sensitive. For example, all salaries may be considered as sensitive: in this case Salary as a type is sensitive information. We can also consider that only turnovers of important customers are sensitive. In this case, only some instances of turnovers are sensitive.

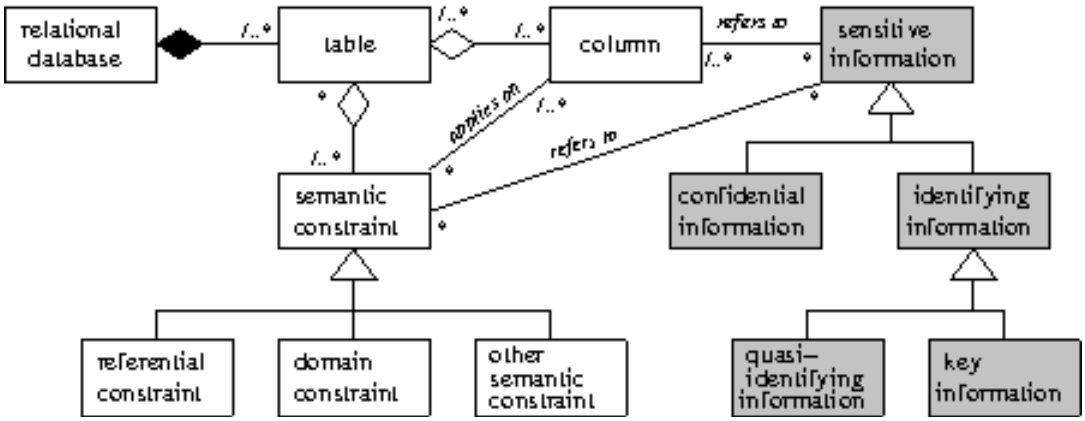

Figure 2. Meta-model of sensitive information

One main interest of our approach is that it scrambles data while maintaining existing semantic links expressed as semantic constraints. These semantic constraints are not always completely defined in existing databases. Therefore, our approach encompasses semantic rules enabling to detect hidden semantic links such as functional dependencies resulting from database de-normalization. Our process scrambles data by processing all these semantic links. Therefore test teams are faced with fictitious but realistic data.

The scrambling problem is complicated by the fact that attackers may have access to other information sources, which permit to de-anonymize data by joining these sources with the scrambled database. This external knowledge is often called adversarial knowledge (Chen, LeFevre, & Ramakrishnan, 2007). In our approach, we aim to scramble test databases. We suppose that we don't have any knowledge about attackers. However we propose to model potential adversarial knowledge using domain rules. As an example, in the chemistry industry, potential attackers have fine-grain knowledge on their competitors. A domain rule will edict that competitors turnovers are sensitive information in a given activity sector. Thus, the adversarial model is embedded in domain rules.

DETECTING SENSITIVE DATA

While most existing tools need as an input the attributes to be scrambled, our tool aims at helping in the detection of sensitive attributes. We automate the detection of sensitive attributes with a combination of techniques based on deduction rules, statistics and natural language processing (NLP). Deduction rules are mainly used to build S_c, statistics to compute S_i, and NLP to expand S_s with semantically close attributes. The whole process may be enriched by a human expert validation consisting in either adding new rules or modifying sensitivity scores.

The Rule based approach

Our approach, automating the identification of sensitive attributes, relies mainly on rules that represent the knowledge of experts on the sensitivity of the data in a given context. The rule based approach is divided into two steps: (*i*) the acquisition step that implies the human expertness, and (*ii*) the rules application step that can be fully automated. We distinguish the two following kinds of rules:

- *intentional rules* with conditions on the database schema (mainly attribute names);
- *extensional rules* with conditions on the attribute's instances.

Example 2 *Rules like "Salary is a highly sensitive attribute" or "attributes with type* `autoincrement` *must be scrambled" (they generally denote identifiers) are examples of the first type of rules. Conversely, rules based on the fact that a column with some instances that contain words like euros or street may refer respectively to private data on salary or on address of the employees, belong to the second kind of rules.*

Let Δ be the set of all possible domains of application, Φ the set of all possible attribute names and Ψ the set of all possible attribute values. An attribute instance is an instance $(\delta, \varphi, \psi) \in \Delta \times \Phi \times \Psi$

of the triple (domainName, attributeName, attributeValue). While theoretically rules may be complex, we adopt the simple following rule definition.

Definition 4 Rule condition. *A rule condition* $\chi = \chi_1 \theta \chi_2$ *is a condition with* $\chi_1 \in \{domainName, attributeName, attributeValue\}$, $\chi_2 \in \Delta \cup \Phi \cup \Psi$, *and* θ *is an operator in* $\{=, !=, <, >, \leq, \geq, contains, !contains\}$.

Observe that an attribute value could be set during the rule definition if it holds for all databases of the related domain (such as a pathology name) or set by the expert during the scrambling process if the attribute value is dependent of specific applications such as the maximal salary value.

Definition 5 Rule. *A rule is composed of a couple* (α, σ) *where* $\alpha \in \Omega$ *and* σ *is composed of disjunctions and conjunctions of rule conditions along a rule sensitivity score* $\sigma \in [0,1]$, *where* σ *permits to evaluate how sensitive is an attribute that satisfies the rule.*

Similarly to the attribute values, the sensitivity score value could either be set during the rule definition or during the scrambling process by an expert.

The sensitivity score is set by the expert to express how sensitive are data that match a given rule, the higher the score is, the more sensitive the data are. This sensitivity score allows us to order the different attributes according to their sensitivity. Thus the user can decide the security level she wants for her application by deciding the sensitivity threshold. All attributes with a sensitivity score above this threshold must be scrambled. A rule example follows.

Example 3. *Assume we consider that a column whose name contains "SALAR", if the domain is HRD and there are values greater than 15,000 or lower than 5,000 then this column is highly sensitive (score=0.9). The corresponding rule is defined by the following expression: ((domainName='HRD')* ∧ *(attributeName contains 'SALAR')* ∧ *(attributeValue>15000* ∨ *attributeValue<5000)) , 0.9).*

Finally, if an attribute α has one or several instances or metadata that satisfy at least one rule, this attribute is candidate for scrambling. The sensitivity score of α for a given set of rules is defined as follows:

Definition 6 Attribute's sensitivity score. *Let I be the set of instances and metadata for* α *and R be the set of rules such that* $\forall \rho \in R, \exists \iota \in I, \iota \text{ ‘ } \rho$. *The sensitivity score of the attribute* α *is defined as:*

$$score(\alpha) = \begin{cases} 0 & if\ R = \varnothing \\ max_{\rho \in R}(\sigma_\rho) & otherwise \end{cases}$$

where σ_ρ *denotes the score of the rule* ρ. *In other words, we consider that either the attribute doesn't satisfy any rule and its sensitivity score is null, or several rules are satisfied for this attribute and consequently its sensitivity score is the highest of all the rule sensitivity scores. We have chosen this*

way of computation among other candidate formulae (min, average, Bonczek-Eagin, hybrid mixture, etc. See (Blanning, 1988)) since we give priority to the highest security.

The existence or not of the *domainName* in a rule allows us to classify the rules in two families. On the one hand there are context-free rules (when no *domainName* is set) that are applied whatever the domain. On the other hand, we have noticed domain-dependent rules: they may be valid in a given domain and false in other domains. A practical way to define some rules is based on expert knowledge. Simple rules concerning one single attribute may be acquired from the experts by the mean of a matrix such as the one depicted Figure 3. The given marks allow them to set the sensitivity scores of the attributes. A mark is given for an attribute in a given domain starting from A which corresponds to "highly sensitive" to E "not at all sensitive" (public). Of course many other techniques may be applied to populate our knowledge database, such as Delphi questionnaire methodologies querying non-experts impacted by the disclosure, limiting expert's bias.

Domain / Attribute	Medical	Banking	Insurance	Mail Order Sailing
Birth Day	B	D	D	D
Name	A	A	A	D
Salary	A	A	A	A
Transaction Date	A	A	A	B
Gender	E	E	E	E
Town	E	E	E	E
Room	E	E	E	E

Figure 3: Matrix for acquisition by voting

The statistical computation

First we suppose that the *primary key* and the *unique* integrity constraints are always stored in the metabase. They directly give indications on candidates for S_i, that is the set of attributes for which a restriction query will return only one tuple. Some candidates for the set S_i of identity attributes can be computed via the statistics of the database. In most DBMS the selectivity of each attribute is stored in the metabase for query optimization purpose. Thus the system may know if an apparently mild attribute such as car brand is in fact an identifying attribute for some databases with few tuples and some unusual cars.

Unfortunately the statistics stored in the metabase are generally not sufficient to supply all the required information on the distribution and the selectivity since they consider only single attributes. Generally, no information about the selectivity of a subset of attributes is stored in metadata tables. SQL queries sent to the database can be performed to fulfill this requirement. However with large databases this method is costly: if we assume n attributes for a table, 2^n queries must be performed. When considering a large application like an ERP, with thousands of tables with dozens of attributes, this solution is not conceivable. Actually it has been shown that determining the optimal k-anonymity in a database is *NP*-hard (Meyerson & Williams, 2004).

However several heuristics have been proposed to provide fast *k*-anonymization algorithms (Park & Shim, 2007). We do not aim in this paper at presenting a new heuristic for detecting composed identity attributes so we rely on existing ones. Finally, to determine S_i we consider in turn the different candidates found and we check if the table they belong to also presents sensitive attributes.

Natural Language Processing

Rules are stated upon concepts. However in a given application the attributes may not have been named with exactly the same term that the one used in the rules. Thus the matching between the term used in the rule and the attributes name involves NLP techniques. Since the nineties numerous works have been proposed using ontologies like WordNet (*WordNet: An Electronic Lexical Database*, n.d.) enabling measuring the similarity between two terms (Lin & Sandkuhl, 2008). In our WordNet based solution the matching between names in the rules and names in the relations requires a function $\mathrm{SIMILAR}(att\_name, att\_desc, att\_name\_in\_rule)_{ssim}$. The inputs of this function are *att_name* the attribute name in the relation, *att_desc* the explanatory text on the attribute that can be found in the metabase and *att_name_in_rule* the name of the attribute as specified in the rule. The explanatory text is essential in case of particularly inexpressive attribute's names. For example early versions of the SAP ERP allowed only 5 characters terms for naming attributes (*e.g.* PERNR, KUNNR, SPTXT, etc.). Those original names are often still in use; fortunately an attribute called "short description" is systematically filled with a description like "total amount of premium".

In a first step *att_name* is the object of a cleaning process aiming at avoiding a lack or an overuse of stop words or delimiters (space, underscore, etc.), as for example in *customername*, *employee_id*, *num_of_customers* and at homogenizing notations. Basically, for these text pre-processing techniques the following operations may be performed:

- removal of word-separators like "_" or "-";
- word-completion when an abbreviation is found, like "number" for "#" or "num", "identifier" for "id", etc;
- stop-words removal, while generally rarely present in table or column names;
- stemming of the words, since two words with the same root have a similar meaning regarding a given rule. For instance we keep "salar" for words "salary", "salaries", etc.

Several tools exist for performing this preprocessing, like (*The Apache Lucene Project*, n.d.) that presents modules for stopwords removal or stemming. As an example, a morphological analyzer is able to recognize conceptual relationships (mainly hyponymy) between parts of the expression (Morin & Jacquemin, 2004). In a first implementation we haven't yet explored these relationships and we simply state that there is matching in the following cases: *att_name* and *att_name_in_rule* are members of the same set of synonyms, or are naming the same concept in different languages, or are hyponym/hyperonym.

A score $s_{sim} \in]0, 1[$ corresponds to any other proximity distance regarding the semantics of the two terms. The membership of *att_name* and *att_name_in_rule* to the set of hyponyms of the same

hyperonym is considered as an expansion of the rule and is treated in the following section. In case of failing in the matching between *att_name* and *att_name_in_rule* an attempt of matching is triggered using *att_desc* and *att_name_in_rule*. Terminological variations have been mainly studied between two terms and fewer works include comparisons between multi-terms expressions based on the analysis of the relationships between parts of the expression (Morin & Jacquemin, 2004). In a first implementation we haven't yet explored these relationships and we simply state that there is matching in the following cases: *att_name* includes *att_name_in_rule*; or *att_desc* shares more than 80% with the description of *att_name_in_rule* in Wordnet.

PROPAGATING SENSITIVITY SCORES

Applying the previous techniques to a database results in a set of attributes S_{init_s} identified for scrambling. However up to now we have considered each table separately. Halting the process at that step would probably lead to an incomplete result since there exist links between attributes from different tables and any sensitivity score for an attribute must be propagated to another. First we present shortly the propagation algorithm proposed in (Mouza et al., 2010) that exploits both referential and semantic links between attributes.

The propagation graph model

We consider two kinds of links between attributes: links explicitly defined in the database schema as integrity referential constraints, and implicit links based on semantics.

Referential integrity links

Since a foreign key attribute references a primary or secondary key attribute, any modification of the former must impact the latter. However the foreign keys are generally not detected neither as identity attribute since their selectivity is low (a primary key value is referenced by the foreign key of many tuples) nor as sensitive data since they are not explicitly targeted by rules.

Example 4 *Back to our HRD database example, since many employees share the same manager, the techniques presented above do not detect the attribute manager_id as sensitive. However we decided to scramble the id of the employee (primary key), then we have to propagate and to scramble also the attribute manager_id (foreign key).*

Since referential integrity constraints are explicitly stored in the database we can extract them to propagate sensitivity scores. Assume the set PK of primary or secondary keys, we use the following notation to refer to the referential integrity constraints: $\gamma_r : 2^{|S|} \rightarrow 2^{|S|}$ ($2^{|S|}$ denotes the power set of S) {we use the traditional $2^{|S|}$ notation to denote the power set of S } defined as $\forall x \in 2^{|S|}, \gamma_r(x) = \begin{cases} \{y \mid y \in 2^{|S|}, y \text{ foreign key referring to } x\} & \text{if } x \in PK \\ \varnothing & \text{otherwise} \end{cases}$

Finally we denote for any set $P \subseteq S$, the result set $\Gamma_r(P)$ defined as $\Gamma_r(P) = \bigcup_{x \in 2^{|P|}} \gamma_r(x)$

Semantic links

Referential integrity constraints are not the only links that exist between attributes. For instance an attribute in a table may have the same semantics than another one in another table. The NLP

approach for the rules allows firing rules on attributes based on the semantics, whatever the attribute's name is. So if a rule is applied to a given attribute, this same rule will also be applied to any other attribute sharing the same meaning. However the expert may also decide that an attribute has to be scrambled independently of what our system proposes. Such a decision must consequently propagate to all the "semantically linked" attributes.

Example 5 *Assume there does not exist any rule concerning the sensitivity of the salary of employees and that the expert decides that this information must not be revealed. When she sets the attribute salary in one table as sensitive, she intends that all the other attributes in any table that refer to the same kind of information, like* `salary` *in another table, but also wages, bonus, income, etc, have to be set in the same way (*e.g. `max_salary` *and* `min_salary`*). Later she realizes that the address attribute must be scrambled too. Starting from this selected attribute, the sensitivity must be propagated to the couple (*`street,city`*) in another table for instance.*

These semantic links may be either stored in the rules base or extracted from some general (*e.g.* WordNet or domain-based ontologies.

We use the notation $\gamma_s : 2^{|S|} \to 2^{|S|}$ to refer to the semantic constraints defined as:

$$\forall x \in 2^{|S|}, \gamma_s(x) = \quad \{y \mid y \in 2^{|S|}, x \text{ is semantically linked to } y\}$$

Finally we denote for any set $P \subseteq S$, the result set $\Gamma_s(P)$ defined as $\Gamma_s(P) = \bigcup_{x \in 2^{|P|}} \gamma_s(x)$

Propagation algorithm

We use the referential and semantic links between attributes to extend the set of attributes S_s^{init} identified for scrambling and validated by the expert using the techniques presented above. We proceed to the following iterative algorithm to determine the final set S_s of attributes to scramble:

(i) $\quad S_s^{(0)} = S_s^{init}$

(ii) $\quad S_s^{(k+1)} = S_s^{(k)} \cup \Gamma_r(S_s^{(k)}) \cup \Gamma_s(S_s^{(k)})$

Lemma 1 Convergence. *The algorithm converges to with at most |S| iterations.*

Proof*: The proof is straightforward:* $S_s^{(k)}, k \in \mathbb{N}$ *, is monotonic increasing and is bounded by* S *, therefore it converges. Moreover note that we have* $S_s^{(k+1)} = S_s^{(k)}$ *when we reach the convergence and the algorithm stops since it means that no link permits to extend* $S_s^{(k)}$ *and the result is stabilized. While convergence is not reached, the result set extends at each step by at least one attribute. Consequently the algorithm converges in at most* $|S|$ *steps.*

If the propagation process leads to a conflict set of different sensitivity values for the same attribute, the maximum level is preferred as presented above. Finally when a candidate attribute has been selected for scrambling one must determine the adequate algorithm to apply. This is however out of the scope of the paper and remains as future work.

SCRAMBLING PROCESS

The whole scrambling process is decomposed into three main steps: labeling, detection and validation (Figure 4). Labeling and detection are performed by the mean of rules that can be either general rules or application-dependent rules. Validation involves a panel of experts.

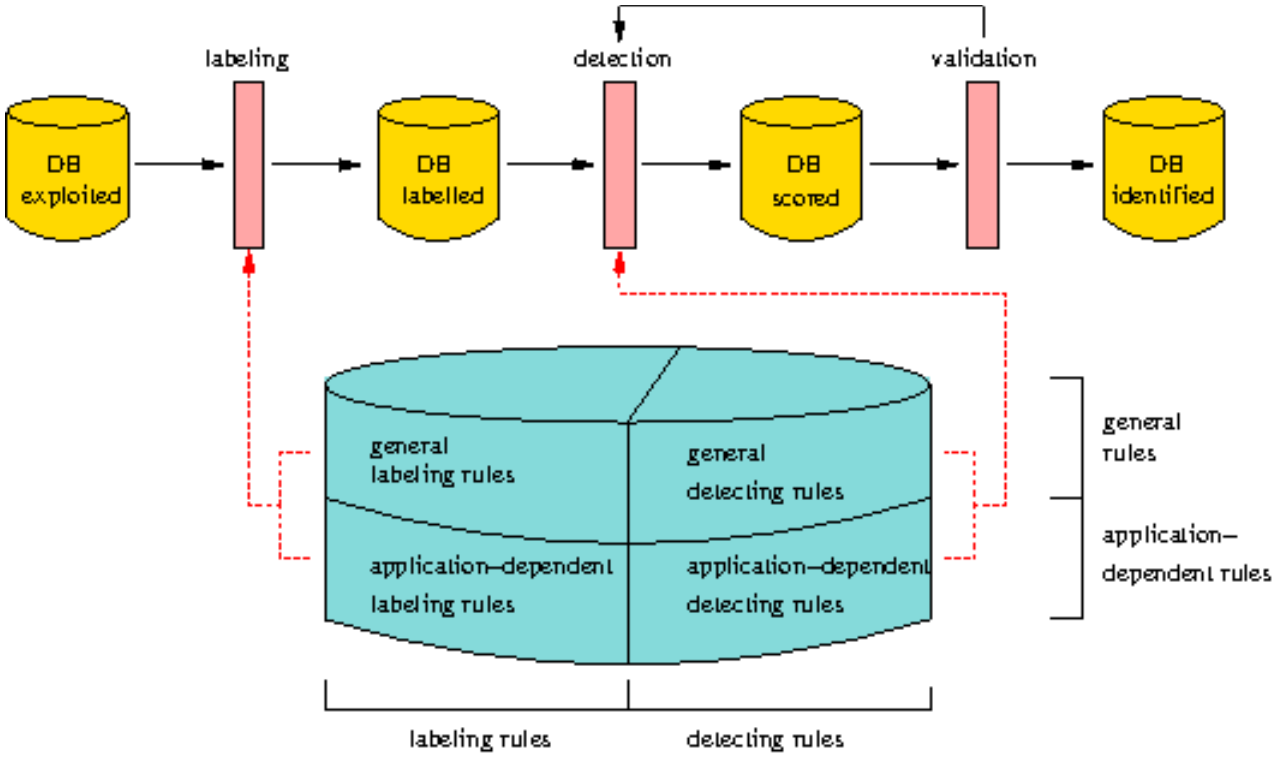

Figure 4: Scrambling process

Labeling the database

Detection rules are based on semantics and they are not necessarily connected to the exact attribute's names of the database to be scrambled. For example a detection rule can specify that "salaries have to be scrambled" while in a given application the column of salaries is named wage. Thus to facilitate rules detection and scores propagation the first step in the scrambling process consists in applying *labeling rules* that annotate the database with labels giving the semantics of the attributes, in the same terms as those used in the rules. The result of this step is a labeled database with meta-information about the semantics of its attributes. For example the attribute wage has a tag salary because the latter is the word used in the rules. As mentioned earlier, when the name of the attribute is not explicit enough, *e.g.* STX23, this process may imply the analysis of the documentary text stored in the metabase on each attribute. Among labeling rules, some are *general rules*, that means that they are application independent and are supplied in the rule base; and others are *application-*

dependent rules that can be supplied by the user of the expert system. This latter class of rules addresses a particular application and gives labeling directives for some of its attributes. An example of application-dependent labeling rule is “In my application the attribute Sal23 is set for salary”.

Detection of sensitive attributes

Detection rules are applied on the labeled database to detect which attributes have to be scrambled. For example, thanks to the tag salary on the attribute wage, the rule “salary is sensitive” will be applied to state that the attribute wage of the table Employee is sensitive and has to be scrambled.

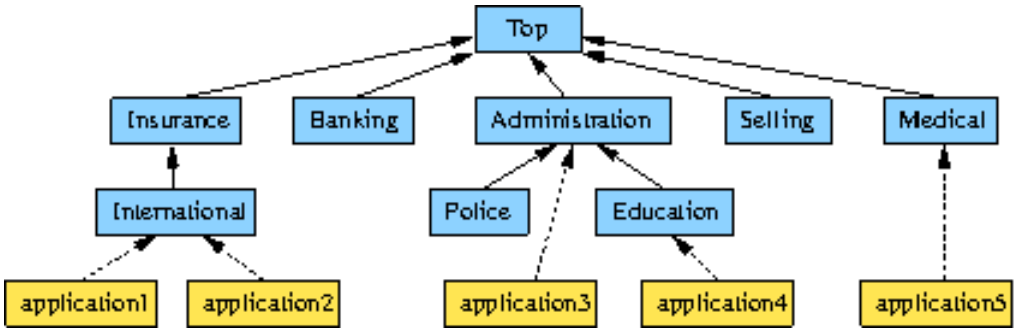

Figure 5: Taxonomy of domains for rule inheritance

As previous labeling rules, detecting rules may be either general rules or application-dependent rules. Application-dependent rules always have priority on general rules. A taxonomy of domains enables an automatic inheritance of rules (Figure 5). Context-free rules (e.g. rules concerning attribute’s types) are placed on the first levels. Application-dependent rules are on the leaves.

After this step the propagation algorithm performs the propagation of sensitivity through referential constraints and other semantic constraints.

Expert Validation

Both automatic detection and automatic propagation steps' results have to be proposed for validation to a panel of experts. The same validation process can be proposed to the users for a new application in order to customize the rule base. In the second case the user can either accept all default choices or change some that are not relevant for her particular firm’s requirements. It is very important in this step to propose realistic schemes, leading to an easy and quick process. For this purpose we propose to the expert:

- a direct access to the rules base filtered according to her domain thanks to the hierarchy in Figure 5;
- a clear vision of data samples (instances) across several tables, reporting on the attribute’s deduced sensitivity and the propagation of this sensitivity. This allows the expert to directly point to the attribute she disagrees with and to correct its level of sensitivity.

After an expert validation, if she performed any changes on the proposal, the identification and propagation processes are re-run. This step is iterated until the expert validates the whole proposal.

The process then terminates and provides as output an “identified DB”, *i.e.* a database with the different sensitive attributes identified and scored according to their degree of sensitivity.

EVALUATION

In order to validate our approach we developed a first prototype that includes most of the ideas introduced in this paper. Then we used it to sanitize a sample of databases and made the results analyzed by a panel of experts.

Prototype presentation

The tool presented in this paper for an automatic detection of sensitive data to be scrambled is part of a more global prototype gathering two modules:

- the identifying module presented in this paper for the detection of which attributes are sensitive and have to be scrambled;
- a scrambling module that chooses the most appropriate scrambling algorithms and applies them on the data to produce a scrambled database.

Figure 6 represents the architecture of the identifying module presented in this paper. This tool has been implemented in Java mainly for its portability using an Expert System approach. We have chosen JESS (*JESS, the Rule Engine for the Java Platform*, n.d.), a rule engine and scripting environment dedicated to Java applications, as an expert system. JESS stores the rules in files with *clp* extension which allows us to easily import/export rules files. Theses files can also be completed by the expert and/or user (depending on the genericity of the rule) through the tool interface.

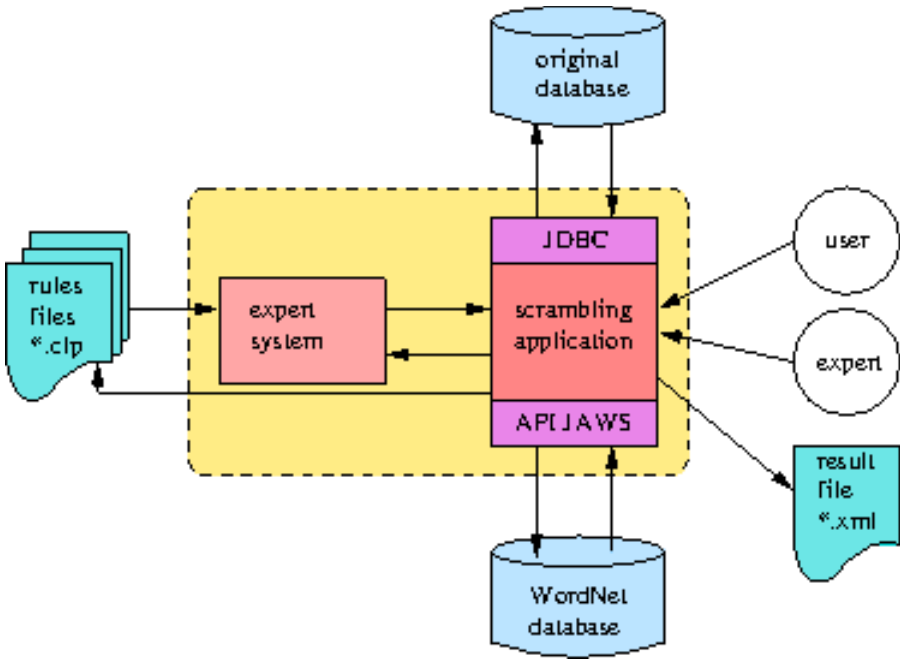

Figure 6: Prototype architecture

The NLP treatments are supported by the Wordnet ontology that provides, among other links, synonymy and proximity links between words. Currently our prototype takes into consideration only synonymy links to detect if a rule written for a given attribute’s name applies to an attribute in the

database to be scrambled while its name is different but synonym. We intend to consider other links using existing similarity measures based on path length between concepts like *Ich* (Leacock & Chodorow, 1998) or *wup* (Wu & Palmer, 1994) for instance, or based on information content like *res* (Resnik, 1995) or *lin* (Lin, 1998). The WordNet::Simarity package (Pedersen, Patwardhan, & Michelizzi, 2004) is another interesting solution. Our implementation relies on the JAWS API as an interface between our application and WordNet, and JDBC to connect the application to the database.

Validation has been performed on ORACLE. A next prototype will focus on SAP applications. The tool finally provides as a result an XML file with the set of attributes for each table along their sensitive score. This XML file is then processed by the second module (not presented here) in charge of determining adequate scrambling strategies for each sensitive attribute. Using an XML file as an output also allows the expert or the advanced user to directly edit the XML files for adding or modifying some rules.

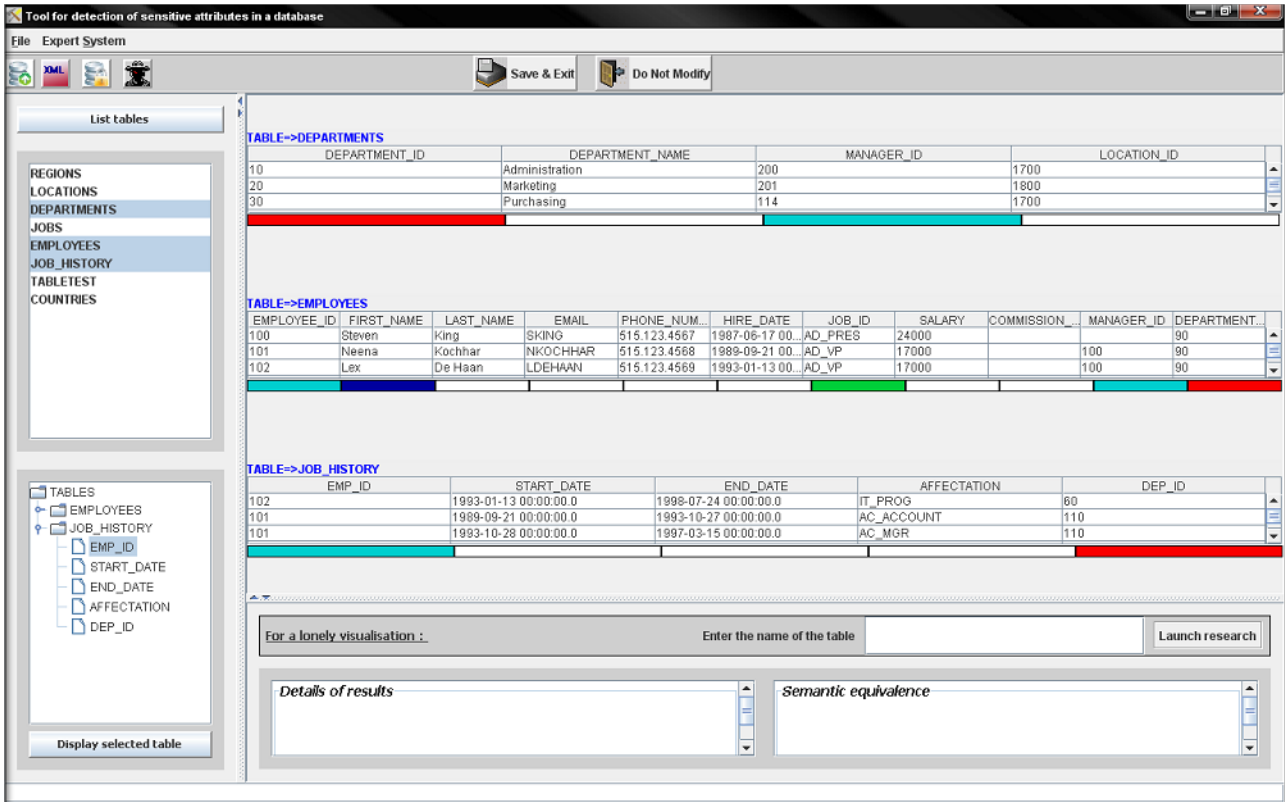

Figure 7: Prototype's interface

Our experiment has convinced us that, unlike computer scientists, domain experts and users are more familiar with data than with attribute names. Thus we provide them with some examples of data in order to help experts making their decisions. Figure 7 shows such a proposal based on simple `select` queries on the tables. A sample of a query's result is proposed with a different color for each attribute, corresponding to the level of sensitivity based on the acquired rules.

The expert can change a color each time she doesn't agree. This change is propagated in cascade to other attributes connected either by a referential constraint link or a semantic one (see propagation mechanism above). Here for instance we decide to increase the sensitivity score for the attribute

department_id from green (score of 0.2) to red (score of 0.6) in table Department. This impacted also the sensitivity score for attributes department_id and dep_id in respectively tables Employee and Job_history that get in turn a red label. A visual alert warns the user when tables not currently displayed have an attribute whose sensitivity has changed when cascading. Moreover the attribute first_name satisfied a rule on family_name attribute and got a very high sensitivity score symbolized by the dark blue color.

Our tool also contains an interface to edit, add or delete rules on attributes (see Figure 8) or instances (see Figure 9).

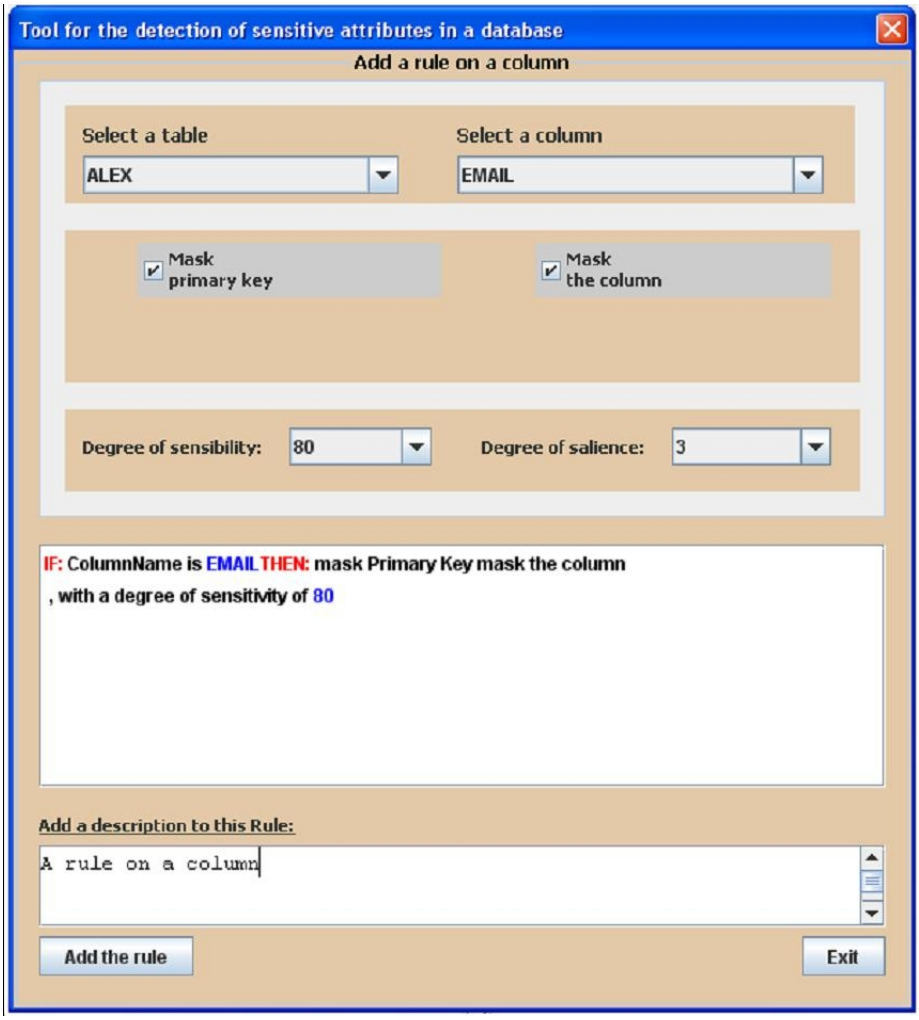

Figure 8: Adding a rule on a column

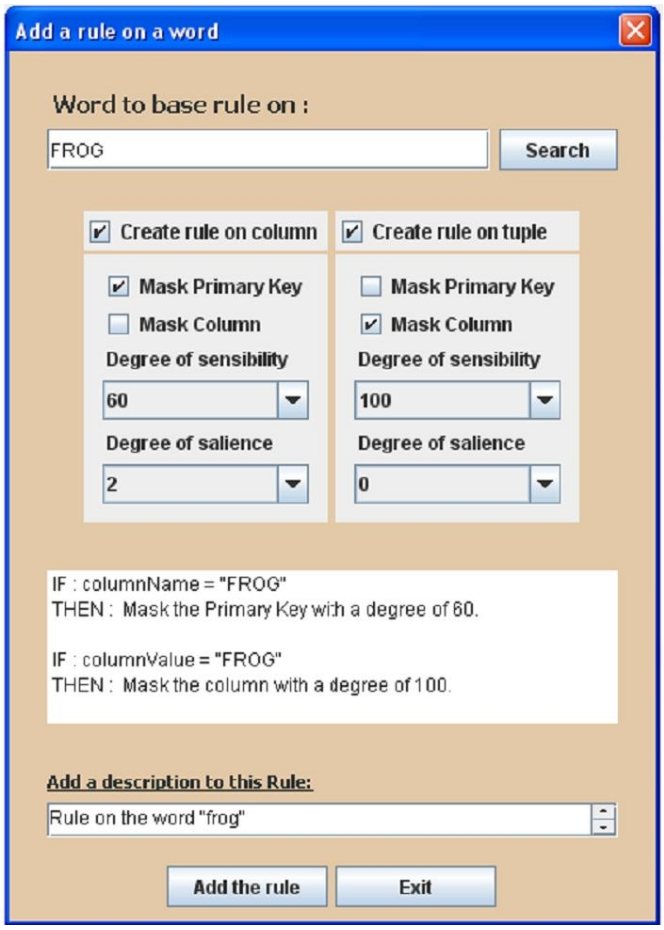

Figure 9: Adding a rule on an instance

Validation

We evaluate our proposal and prototype with four different databases: samples based on *Dellstore*, *IMDB* and *MediaWiki* (used in *Wikipedia*) databases and a classical sales application that we refer as "Order". Table 1 describes the databases.

Database	# of tables	# of attributes	# of rows
Dell store	8	52	172,716
IMDB	47	151	1,834,483
Media Wiki	45	289	756
Order	8	59	3459

Table 1: Description of the databases for the experiments

Our expert system relies on a set of 30 rules collected from a panel of 8 experts. Table 2 presents an extract from this rules set. They are general rules or rules about the sales domain.

1. *A column whose name contains the word "address" (or one of its synonyms or one of it's abbreviations) is sensitive*
2. *A column whose name contains the word "income" (or one of its synonyms or one of it's abbreviations) is sensitive*
3. *A column whose name contains the word "discount" (or one of its synonyms or one of it's abbreviations) and for which the standard deviation is high then is sensitive.*
4. *A column of type date for which some values occur less than 10 times is sensitive.*

Table 2. Examples of sensitivity rules for the Order and Dell store databases

They concern either attributes' names (80% of the rules) or both attributes' names and instances. In addition implicit rules on the primary keys, unique attributes and attributes with high selectivity (here we assume that attributes whose single value identifies less than 2% of the data is sensitive) are also considered. Implicit rules on identifiers or quasi-identifiers (here we assume that attributes with instances that identify less than 2% of the data are sensitive) are also considered. For each rule a score that represents the sensitivity of an attribute that follows this rule was *a priori* set.

Database	total time (in s)	sensitive attributes (# and %)		Experts evaluation			
		mod. ($\sigma \leq 50$)	high ($\sigma > 50$)	missed	false/positive	inadequate	approved
Dell store	24.9	25 (48%)	13 (25%)	2	5	2	32
IMDB	566.8	44 (29%)	15 (10%)	3	9	8	42
Media Wiki	1.6	5 (11%)	2 (4%)	2	1	0	6
Order	1.9	25 (42%)	21 (36%)	3	4	4	38

Table 3: Execution time and result of the identification process

We report the results obtained with our prototype for the four databases in Table 2. First we observe that the execution time greatly depends on the size of the database. For instance `Dell Store` and `Order` databases have approximately the same number of attributes but the former consists of 172,716 records and the latter consists only of 3,459 records, what results respectively in 24.9*s* and 1.9*s* as execution time. This is due to the rules checking process of JESS that performs linearly in the size of the data that consists of both attribute's names and attribute's values. The execution time remains acceptable since around 9*mn* are needed for processing a database with 1.8 million of records. The number of sensitive attributes identified by the prototype varies with applications. As expected, sales applications with much information about customers present more

sensitive attributes than public applications. To simplify analysis we group in this table attributes with a moderate (resp. high) sensitivity, *i.e.* with a sensitivity score $\sigma \leq 50$ (resp. $\sigma > 50$).

We ask to our panel of experts to assign to each column of the four tables a tag among `not sensitive`, `moderately sensitive`, `highly sensitive`. Comparisons of their evaluation with the results produced by our prototype are reported in Table 2. We notice that our prototype has identified around 80% of the sensitive attributes and very few ones were missed (around 5%). The rate of false positives and the rate of attributes with an inadequate sensitive score (tagged as highly sensitive instead of moderately or conversely) are respectively around 10%. As expected, best results are achieved for sales databases since experts provided mainly generic rules and rules for sales applications. Our *recall* of 80%, combined with a precision of 90% of correctly identified sensitive attributes, is not sufficient but is promising. We intend to improve these results by adding other rules and also by relying on domain ontologies to achieve a more adequate matching between rules and information. However remember that even if the current recall and precision scores do not avoid an expert validation, the expert may benefit from our approach since our system can detect possible sensitive attributes difficult to identify, and also "hidden" semantic links between attributes that the expert could have missed.

CONCLUSION

Scrambling test databases is a crucial need for an increasing number of companies. However nothing has been proposed to automatically determine which part of the database needs scrambling. In this paper we have proposed an approach to detect sensitive attributes and its implementation based on an expert system architecture. Our approach relies on a meta-model describing the main concepts used in this scrambling process. We have proposed a rule based approach for determining the attribute sensitivity level. These rules may be general or specific to one application. General rules are provided in the rule base with the tool; they are categorized in an inheritance hierarchy of domains. Application dependent-rules can be added by the user. Primary keys, indexes and statistics on the database stored in the DBMS for optimization purpose are used to detect attributes that are nearly identifying for the tuples. Referential integrity constraints and other semantic links are exploited for the propagation of the sensitivity among attributes. Labeling rules using the WordNet ontology are provided to match the attribute's names used in the rules with the exact names of the attributes in a given application. In addition, referential integrity constraints are preserved as well as other semantic dependencies.

Our future work will focus on the evaluation of the resulting scrambled database. In particular it is difficult to be certain that the scrambled database doesn't contain any inconsistency due to a bad propagation of the scrambling among all the tables. Experimentation will be performed on an SAP application, where data are strongly connected together, sometimes through complex deduction and management rules. Another step will deal with the generalization of rules allowing us not only to assign a sensitivity score to each attribute but also to map the relevant scrambling technique to be applied to this attribute.

The presented approach could also be generalized to contexts different from the one tests environments. In this case, an extension of the actual work will integrate the users' profiles to elaborate a data access strategy.

ACKNOWLEDGEMENTS. The authors want to thank the reviewers for their helpful comments.

REFERENCES

Amiri, A. (2007). Dare to Share: Protecting Sensitive Knowledge with Data Sanitization. *Decision Support Systems (DSS), 43*(1), 181191. *The Apache Lucene Project.* (n.d.). (http://lucene.apache.org/)

Askari M., Safavi-Naini R.,& Barker K. (2012): An Information Theoretic Privacy and Utility Measure for Data Sanitization Mechanisms. In *Proc. Intl. ACM Conf. on Data and Application Security and Privacy (CODASPY)* (p 283-294).

Atallah, M., Elmagarmid, A., Ibrahim, M., Bertino, E., & Verykios, V. (1999). Disclosure Limitation of Sensitive Rules. In *Proc.Intl. Workshop on Knowledge and Data Engineering Exchange (KDEX)* (p. 45-52).

Bakken, D. E., Parameswaran, R., Blough, D. M., Franz, A. A., & Palmer, T. J. (2004). Data Obfuscation: Anonymity and Desensitization of Usable Data Sets. *IEEE Security & Privacy*, *2*(6), 34-41.

Blanning, R. (1988). Sensitivity Analysis in Hierarchical Fuzzy Logic Models. In *Proc. Intl Conf. on Decision Support and Knowledge Based Systems Track* (pp. 471–476).

Bonchi, F., Gionis, A., & Tassa, T. (2011). Identity Obfuscation in Graphs Through the Information Theoretic Lens. In *Proc. Intl. Conf. on Data Engineering (ICDE)* (p. 924-935). *Camouflage.* (n.d.). (http://www.datamasking.com)

Chakaravarthy, V. T., Gupta, H., Roy, P., & Mohania, M. K. (2008). Efficient Techniques for Document Sanitization. In *Proc. Intl. Conf. on Information and Knowledge Management (CIKM)* (p. 843-852).

Chen, B.-C., LeFevre, K., & Ramakrishnan, R. (2007). Privacy Skyline: Privacy with Multidimensional Adversarial Knowledge. In *Proc. intl. conf. on very large data bases (vldb)* (pp. 770–781).

Datamasker. (n.d.). (http://www.datamasker.com)

Datavantage Globa. (n.d.). (http://www.datavantage.com)

Fung, B. C., Wang, K., Chen, R., & Yu, P. S. (2010). Privacy-Preserving Data Publishing: A Survey on Recent Developments. *ACM Computing Surveys*, *42*(4).

Gardner, J., & Xiong, L. (2008). HIDE: An Integrated System for Health Information DE-identification. In *Proc. Intl. Symp. on Computer-Based Medical Systems (CBMS)* (p. 254-259).

HCM Data Scrambler. (n.d.).

Meyerson, A., & Williams, R. (2004). On the *HCM Data Scramble Tool*. (n.d.). (http://mudiaminc.com/Hcm-Data-scrambel-tool.html)

JESS, the Rule Engine for the Java Platform.(n.d.). (http://www.jessrules.com)

JumbleDB - Orbium Software. (n.d.). (http://www.orbiumsoftware.com/)

Leacock, C., & Chodorow, M. (1998). Combining Local Context and WordNet Similarity for Word Sense Identification. *MIT Press*, 265-283.

Li, N., Li, T., & Venkatasubramanian, S. (2007). t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *Proc. Intl. Conf. on Data Engineering (ICDE)* (p. 106-115).

Lin, D. (1998). An Information-Theoretic Definition of Similarity. In *Proc. Intl. Conf. on Machine Learning (ICML)* (p. 296-304).

Lin, F., & Sandkuhl, K. (2008). A Survey of Exploiting WordNet in Ontology Matching. In *Proc. Intl. Conf. on Intelligent Information Processing, Artificial Intelligence in Theory and Practice (IFIP AI)* (p. 341-350).

Machanavajjhala, A., Gehrke, J., Kifer, D., & Venkitasubramaniam, M. (2006). lDiversity: Privacy Beyond k-Anonymity. In *Proc. Intl. Conf. on Data Engineering (ICDE)* (p. 24). Complexity of Optimal K-Anonymity. In *Proc. Intl. Symp. on Principles of Database Systems (PODS)* (p. 223-228).

Morin, E., & Jacquemin, C. (2004). Automatic Acquisition and Expansion of Hypernym Links. *Computer and the Humanities*, *38*(4).

Mouza, C. du, Métais, E., Lammari, N., Akoka, J., Aubonnet, T., Comyn-Wattiau, I., et al. (2010). A Semantic Approach to Improve Automatically Data Security During Test of Information Systems. In *Proc. Intl. Conf. on Advances in Databases, Knowledge, and Data Applications (DBKDA)* (p. 247-252).

Neamatullah, I., Douglass, M. M., H Lehman, L. wei, Reisner, A., Villarroel, M., Long, W. J., et al. (2008). Automated DeIdentification of Free-Text Medical Records. *BMC Medical Informatics and Decision Making*, *8*(32).

NooJ, a free linguistic development environment. (n.d.). (http://www.nooj4nlp.net)

Oliveira, S. R. M., & Zaïane, O. R. (2003). Protecting Sensitive Knowledge By Data Sanitization. In *Proc. Intl. Conf. on Data Mining (ICDM)* (p. 613-616).

Park, H., & Shim, K. (2007). Approximate Algorithms for K-Anonymity. In *Proc. intl. conf. on management of data (sigmod)* (p. 67-78).

Pedersen, T., Patwardhan, S., & Michelizzi, J. (2004). WordNet: : Similarity Measuring the Relatedness of Concepts. In *Proc. Nat. Conf. on Artificial Intelligence (AAAI)* (p. 1024-1025).

Pse Data Security. (n.d.). (http://www.psedatasecurity.com)

Ravikumar, G. K., Manjunath, T. N., Ravindra, S. H., & Umesh, I. M. (2011). A survey on recent trends, process and development in data masking for testing. *International Journal of Computer Science*, (8), 2.

Resnik, P. (1995). Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proc. Intl. Conf. on Artificial Intelligence (IJCAI)* (p. 448-453).

Saini, M. K., Atrey, P. K., Mehrotra, S., & Kankanhalli, M. S. (2011). Anonymous Surveillance. In *Proc. Intl. Conf. on Multimedia and Expo (ICME)* (p. 1-6).

Shou, L., Shang, X., Chen, K., Chen, G., & Zhang, C. (2011). Supporting Pattern-Preserving Anonymization For Time-Series Data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, *99*(PrePrints).

Solix. (n.d.). (http://www.solix.com)

Sweeney, L. (2002). Achieving k-Anonymity Privacy Protection Using Generalization and Suppression. *Intl. Jour. of Uncertainty, Fuzziness and Knowledge-Based Systems*, *10*(5), 571-588.

To, Q. C., Dang, T. K., & Küng, J. (2011a). B^{ob}-Tree: An Efficient B^{+}-Tree Based Index Structure for Geographic-Aware Obfuscation. In *Proc. Intl. Conf. on Intelligent Information and Database Systems (ACIIDS)* (p. 109-118).

To, Q. C., Dang, T. K., & Küng, J. (2011b). OST-Tree: An Access Method for Obfuscating Spatio-Temporal Data in Location Based Services. In *Proc. Intl. Conf. New Technologies, Mobility and Security (NTMS)* (p. 1-5).

Vinogradov, S., & Pastsyak, A. (2012). Evaluation of Data Anonymization Tools. In *Proc. Intl. Conf. on Advances in Databases, Knowledge, and Data Applications (DBKDA)* (p. 163-168).

Wang, E. T., & Lee, G. (2008). An Efficient Sanitization Algorithm for Balancing Information Privacy and Knowledge Discovery in Association Patterns Mining. *Data Knowl. Eng.(DKE)*, *65*(3), 463-484.

WordNet: An Electronic Lexical Database. (n.d.). (http://wordnet.princeton.edu)

Wu, Z., & Palmer, M. S. (1994). Verb Semantics and Lexical Selection. In *Proc. of the Association for Computational Linguistics (ACL)* (p. 133-138).