Privacy-Preserving Schema Reuse

Author

Nguyen, Quoc Viet Hung, Do, Son Thanh, Nguyen, Thanh Tam, Aberer, Karl

Published

2014

Journal Title

Lecture Notes in Computer Science

Version

Accepted Manuscript (AM)

DOI

10.1007/978-3-319-05813-9_16

Privacy-Preserving Schema Reuse

Nguyen Quoc Viet Hung, Do Son Thanh, Nguyen Thanh Tam, and Karl Aberer

École Polytechnique Fédérale de Lausanne
{quocviethung.nguyen,sonthanh.do,tam.nguyenthanh,karl.aberer}@epfl.ch

Abstract. As the number of schema repositories grows rapidly and several web-based platforms exist to support publishing schemas, *schema reuse* becomes a new trend. Schema reuse is a methodology that allows users to create new schemas by copying and adapting existing ones. This methodology supports to reduce not only the effort of designing new schemas but also the heterogeneity between them. One of the biggest barriers of schema reuse is about privacy concerns that discourage schema owners from contributing their schemas. Addressing this problem, we develop a framework that enables privacy-preserving schema reuse. Our framework supports the contributors to define their own protection policies in the form of *privacy constraints*. Instead of showing original schemas, the framework returns an *anonymized schema* with maximal *utility* while satisfying these privacy constraints. To validate our approach, we empirically show the efficiency of different heuristics, the correctness of the proposed utility function, the computation time, as well as the trade-off between utility and privacy.

1 Introduction

Schema reuse is a new trend in creating schemas by allowing users to copy and adapt existing ones. The key driving forces behind schema reuse are the slight differences between schemas in the same domain; thus making reuse more realistic. Reusing existing schemas supports to reduce not only the effort of creating a new schema but also the heterogeneity between schemas. Moreover, as the number of publicly available schema repositories (e.g. schema.org[1], Factual[2]) grows rapidly and several web-based platforms (e.g. Freebase [3], Google Fusion Tables [4]) exists to support publishing schemas, reusing them becomes a great interest in both academic and industrial worlds.

One of the biggest barriers of reuse is about privacy concerns that discourage contributors from contributing their schemas [5]. In traditional approaches, all original schemas and their own attributes are presented to users [6]. However, in practical scenarios, the linking of attributes to their containing schemas, namely *attribute provenance*, is dangerous because of two reasons. First, providing the whole schemas (and all of their attributes) leads to privacy risks and potential attacks on the owner database. Second, since some attributes are the source of revenue and business strategy, the schema contributors want to protect their sensitive information to maintain the competitiveness. As a result, there is a need of developing new techniques to cope with these requirements.

In this paper, we develop a privacy-preserving schema reuse framework that protects the attribute provenance from being disclosed. To this end, our framework enables schema owners to define their own protection policies in terms of privacy constraints. Unlike previous works [6–8], we do not focus on finding and ranking schemas relevant to a user search query. Instead, our framework takes as input these relevant schemas and visualizes them in a unified view, namely *anonymized schema*, which satisfies pre-defined

privacy constraints. Constructing such an anonymized schema is challenging because of three reasons. First, defining the representation for an anonymized schema is non-trivial. The anonymized schema should be concise enough to avoid overwhelming but also generic enough to provide comprehensive understanding. Second, for the purpose of comparing different anonymized schemas, we need to define a utility function to measure the amount of information they carry. The utility value must reflect the conciseness and the completeness of an anonymized schema. Third, finding an anonymized schema that maximizes the utility function and satisfies privacy constraints is NP-complete.

The main goal of this paper is to construct an anonymized schema with maximal utility while preserving the privacy constraints, which are defined to prevent an adversary from linking the shown attributes back to the original schemas (and to the schema owners). Our key contributions are summarized as follows.

- We model the setting of schema reuse with privacy constraints by introducing the concept of *affinity matrix* (represents a group of relevant schemas) and *presence constraint* (is a privacy constraint that translates human-understandable policies into mathematical standards).
- We develop a quantitative metric for assessing the utility of an anonymized schema by capturing two important aspects: (i) *attribute importance*—which reflects the popularity of an attribute—and (ii) *completeness*—which reflects the diversity of attributes in the anonymized schema.
- We show the intractability result for the problem of finding an anonymized schema with maximal utility, given a set of privacy constraints. We propose a heuristic-based algorithm for this problem. Through experiments on real and synthetic data, we show the effectiveness of our algorithm.

The paper is organized as follows. Section 2 gives an overview of our approach. Section 3 formally introduces the notion of schema group, anonymized schema and privacy constraint. Section 4 demonstrates the intractability result and the heuristic-based algorithm to the *maximizing anonymized schema* problem. Section 5 empirically shows the efficiency of our framework. Section 6 and 7 present related work and conclusions.

2 Overview

System overview. Fig. 1 illustrates a schema reuse system, in which there is a repository of schemas that are willingly contributed by many participants in the same domain. We focus on the scenario in which end-users want to design a new database and reuse the existing schemas as hints, by exploring the repository through search queries [6–8] to find schemas of relevance to their applications. In traditional approaches, all the relevant schemas and their whole attributes are shown to users. Nevertheless, it is important to support the contributors to preserve the privacy of their schemas [9] because of several privacy issues. One possible issue is the threats of being attacked and unprivileged accesses to the owner database systems. For example, knowing the schema information (e.g. schema name, attribute names), an adversary can use SQL injection [10] to extract the data without sufficient privileges (details are described in the report [11]). Another possible issue is the policies of schema owners that require hiding a part of schemas under some circumstances. For instance, a hospital needs to hide the personal information of its patient records due to legal concerns [12], or an enterprise makes an agreement with its clients about the privacy of business practices [13]. Since schema reuse only

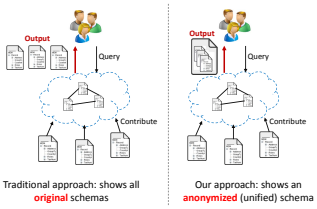

Fig. 1: System Overview.

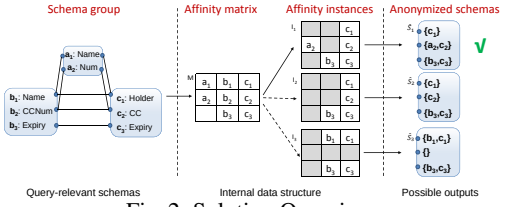

Fig. 2: Solution Overview.

shares schema information, the scope of this paper is to preserve the privacy of schema only.

Solution overview. To encourage schema reuse while preserving privacy, we propose a novel approach that shows a unified view of original schemas, namely *anonymized schema*, instead of revealing all of the schemas. In our approach, schema owners are given the rights to protect sensitive attributes by defining privacy constraints. The resulting anonymized schema has to be representative to cover original ones but opaque enough to prevent leaking the provenance of sensitive attributes (i.e. the linking to containing schemas). Towards our approach, we propose a framework that enables privacy-preserving schema reuse. The input of our framework is a group of schemas (which is relevant to user query as returned by a search engine) and a set of privacy constraints. Respecting these constraints, the framework returns an anonymized schema with maximal utility, which reflects the amount of information it carries. The problem of constructing such an anonymized schema is challenging and its details will be described in Section 4. Fig. 2 depicts the simplified process of our solution for this problem, starting with a schema group and generated correspondences (solid lines) that indicates the semantic similarity between attributes. First, we represent a schema group by an *affinity matrix*. In that, attributes in the same column belongs to the same schema, while attributes in the same row have similar meanings. Next, we derive various *affinity instances* by eliminating some attributes from the affinity matrix. For each affinity instance, we construct an anonymized schema with many *abstract attributes*. Each abstract attribute is a set of original attributes in the same row. Among constructed anonymized schemas, we select the one that maximizes the utility function and then present this "best" anonymized schema to user as final output. Table 1 summarizes important notations, which will be described in the next section.

3 Model

In this section, we describe three important elements of our schema reuse approach: (i) schema group – a set of schemas relevant to a user search query, (ii) anonymized schema – a unified view of all relevant schemas in the group, and (iii) privacy constraint – a mean to represent human-understandable policies by mathematical standards. All these elements are fundamental primitives for potential applications built on top of our framework. Right after defining the anonymized schema, we also propose a single comprehensive notion to quantify its utility in Section 3.3.

3.1 Schema Group

We model a schema as a finite set of attributes $A_s = \{a_1, ..., a_n\}$, which is generic enough for various types of schemas [14]. Let $\mathcal{S} = \{s_1, ..., s_n\}$ be a schema group that is a set

of schemas relevant to a specific search query. Let $A_{\mathcal{S}}$ denote the set of attributes in $\mathcal{S}$, i.e. $A_{\mathcal{S}} = \bigcup_i A_{s_i}$. Two schemas can share a same-name attribute; i.e. $A_{s_i} \cap A_{s_j} \neq \emptyset$ for some i, j. Given a pair of schemas $s_1, s_2 \in \mathcal{S}$, an attribute correspondence is an attribute pair $(a \in A_{s_1}, b \in A_{s_2})$ that reflects the similarity of the two attributes' semantic meanings [15]. The union of attribute correspondences for all pairs of schemas is denoted as C, each of which can be generated by state-of-the-art schema matching techniques [16, 17]. Note that the quality of generated correspondences is not our concern, since there is a large body of research on reconciling and improving schema matching results [18–20]. In the context of schema reuse, the generated correspondences are useful for users to see possible variations of attributes among schemas.

To model a schema group as well as generated correspondences between attributes of its schemas, we define an affinity matrix $M_{m\times n}$. Each of n columns represents an original schema and its attributes. Each of m rows represents the set of equivalent attributes, each pair of which have correspondences between them. An entry being null means that the schema-column of that entry does not have an equivalent attribute against the attributes of other schema-columns in the same row.

Table 1: Summary of Notations

Symbol	Description
$\mathcal{S} = \{s_i\}$	a group of schemas
A_s	a set of attributes of the schema s
C	a set of attribute correspondences
M, I	affinity matrix, affinity instance
$\hat{S}, \hat{A}$	anonymized schema, abstract attribute
$u(\hat{S})$	utility of an anonymized schema
$\Gamma = \{\gamma_i\}$	a set of privacy constraints

Definition 1 ***Affinity Matrix.*** *Given a schema group $\mathcal{S}$ and a set of correspondences C, an affinity matrix is a matrix $M_{m\times n}$ such that: (i) $\forall i,j : M_{ij} \in A \cup \{null\}$, (ii) $\forall i,j : M_{ij} \in A_{s_j}$, (iii) $\forall i, j_1 \neq j_2, M_{ij_1} \neq null, M_{ij_2} \neq null : (M_{ij_1}, M_{ij_2}) \in C$, and (iv) $\nexists i_1 \neq i_2, j_1, j_2 : (M_{i_1 j_1}, M_{i_2 j_2}) \in C$.*

In order to construct a unique affinity matrix, we define an order relation on $A_{\mathcal{S}}$. We consider each attribute $a \in A_{\mathcal{S}}$ as a character. The order relation between any two attributes is the alphabetical order between two characters. Consequently, a row $M_{i.}$ of the affinity matrix M is a string of n characters. An affinity matrix is *valid* if the string of row r_1 is less than the string of row r_2 ($r_1 < r_2$) w.r.t. the lexicographical order. Since the alphabetical and lexicographical order are unique, a valid affinity matrix is unique.

Example 1. Consider a group of 3 schemas $\mathcal{S} = \{s_1, s_2, s_3\}$ and their attributes $A_{s_1} = \{a_1, a_2\}$, $A_{s_2} = \{b_1, b_2, b_3\}$, $A_{s_3} = \{c_1, c_2, c_3\}$. We have 8 correspondences $C = \{(a_1, b_1), (a_1, c_1), (b_1, c_1), (a_2, b_2), (a_2, c_2), (b_2, c_2), (b_3, c_3)\}$. Assume that the lexicographical order of all attributes is $a_1 < a_2 < b_1 < b_2 < b_3 < c_1 < c_2 < c_3$. Then we can construct a unique affinity matrix M as in Fig. 2. The string of the first, second, third row of M is "$a_1b_1c_1$", "$a_2b_2c_2$", "b_3c_3" respectively. M is valid because "$a_1b_1c_1$" < "$a_2b_2c_2$" < "b_3c_3" w.r.t. the assumed lexicographical order.

3.2 Anonymized Schema

An anonymized schema should meet the following desirable properties: (i) representation, and (ii) anonymization. First, the anonymized schema has to be representative to cover the attributes of a schema group. It should show to end-user not only all possible attributes but also the variations of each attribute. Second, the anonymized schema should be opaque enough to prevent adversaries from reconstructing a part or the whole of an original schema.

Our approach is inspired by the generalization technique [21], with some modifications, to protect the owners of original schemas. The core idea is that by not showing all the attributes but only a subset of them (e.g. remove one or many cells of the affinity matrix), we preserve the privacy of important attributes. However, hiding too many attributes would lead to poor information presented to end-users. Thus, the problem of selecting which attributes and how many of them is challenging and will be formulated in Section 4. Here we provide the basic notion of an anonymized schema by representing it as an *affinity instance*. An affinity instance $I_{m\times n}$ is constructed from an affinity matrix $M_{m\times n}$ by removing one or many cells; i.e. $\forall i,j : I_{ij} = M_{ij} \vee I_{ij} = null$. Formally, an anonymized schema $\hat{S}$ is transformed from an affinity instance $I_{m\times n}$ as follows.

$$\hat{S} = \{\hat{A}_1, \ldots, \hat{A}_m \mid \hat{A}_i = \bigcup_{1\le j\le n} I_{ij}\} \tag{1}$$

where $\hat{A}$ is an abstract attribute (a set of attributes at the same row). In other words, an abstract attribute of an anonymized schema is a set of equivalent attributes (i.e. any pair of attributes has a correspondence). Following the running example in Fig. 2, a possible affinity instance is I_1 and the corresponding anonymized schema is $\hat{S_1} = \{\hat{A}_1, \hat{A}_2, \hat{A}_3\}$ with $\hat{A}_1 = \{c_1\}$, $\hat{A}_2 = \{a_2, c_2\}$, and $\hat{A}_3 = \{b_3, c_3\}$.

With this definition, the anonymized schema meets two aforementioned properties—representation and anonymization. First, the representation property is satisfied because by showing abstract attributes, users can observe all possible variations to design their own schemas. Second, the anonymization property is satisfied because it is non-trivial to recover the original affinity matrix from a given anonymized schema. By knowing only the anonymized schema (as a set of abstract attributes), brute-force might be the only way for an adversary to disclose the ownerships of contributors, their schemas, and schema attributes. In Section 3.4, we will illustrate this property in terms of probability theory.

3.3 Anonymized Schema Utility

The utility of an anonymized schema $\hat{S}$ is measured by a function $u : \overline{\mathcal{S}} \to \mathbb{R}$, where $\overline{\mathcal{S}}$ is the set of all possible anonymized schemas. The utility value reflects the amount of information $\hat{S}$ carries. To define this utility function, we first introduce two main properties of an anonymized schema: *importance* and *completeness*.

Importance. The importance of an anonymized schema is defined as a function $\sigma : \bar{S} \to \mathbb{R}$. We propose to measure $\sigma(\hat{S})$ by summing over the popularity of attributes it contains; i.e. $\sigma(\hat{S}) = \sum_{a\in\hat{S}} t(a)$, where $t(a)$ is the popularity of attribute a. The higher popularity of these attributes, the higher is the importance of the anonymized schema. Intuitively, an attribute is more popular than another if it appears in more original schemas. At the same time, an original schema is more significant if it contains more popular attributes. To capture this relationship between attributes and their containing schemas, we attempt to develop a model that propagates the popularity along attribute-schema links. We notice some similarities to the world-wide-web setting, in which the significance of a web page is determined by both the goodness of the match of search terms on the page itself and the links connecting to it from other important pages. Adapting this idea, we apply the hub-authority model [22] to quantify the popularity of an attribute and the significance of related schemas, whose details are described in our report [11] for the sake of brevity.

Completeness. The completeness reflects how well an anonymized schema collectively covers the attributes of original schemas, where each abstract attribute is responsible for original attributes it represents. This coverage is necessary to penalize trivial anonymized schemas that contain only one attribute appearing in all original schemas. To motivate the use of this property, we illustrate some observations in Fig. 2. In this running example, $\hat{S}_2$ has more abstract attributes than $\hat{S}_3$ ($\hat{S}_2$ has three while $\hat{S}_3$ has two). As $\hat{S}_3$ lacks information about *CCNum* or *CC* attributes, users would prefer $\hat{S}_2$ in order to have a better overview of necessary attributes for their schema design. Intuitively, an anonymized schema is more preferred if it covers more information of original schemas. In this paper, we measure the completeness of an anonymized schema by the number of abstract attributes it contains. The more abstract attributes, the higher is the completeness. Technically, the completeness is defined as a function $\lambda : \bar{S} \rightarrow \mathbb{R}$ and $\lambda(\hat{S}) = |\hat{S}|$.

Put It All Together. As mentioned above, there are two properties (importance and completeness) that should be considered for the purpose of comparing different anonymized schemas. We attempt to combine these properties into a single comprehensive notion of *utility* of an anonymized schema. Formally, we have:

$$u(\hat{S}) = w \cdot \sigma(\hat{S}) + (1 - w) \cdot \lambda(\hat{S}) \tag{2}$$

where $w \in [0, 1]$ is a regularization parameter that defines the trade-off between completeness and importance. In terms of comparison, an anonymized schema is more preferred than another anonymized schema if its utility is higher. Back to the running example in Fig. 2, consider three anonymized schemas $\hat{S}_1$, $\hat{S}_2$ and $\hat{S}_3$. By definitions, we have $\sigma(\hat{S}_1) = 5$, $\sigma(\hat{S}_2) = \sigma(\hat{S}_3) = 4$ and $\lambda(\hat{S}_1) = \lambda(\hat{S}_2) = 3$, $\lambda(\hat{S}_3) = 2$. Assuming $w = 0.5$, we have $u(\hat{S}_1) > u(\hat{S}_2) > u(\hat{S}_3)$. $\hat{S}_1$ is more preferred than $\hat{S}_2$ since $\hat{S}_1$ contains more original attributes (importance property). While, $\hat{S}_2$ is more preferred than $\hat{S}_3$ since $\hat{S}_2$ contains more abstract attributes (completeness property).

3.4 Privacy Constraint

Privacy constraint is a mean to represent human-understandable policies by mathematical standards. In order to define a privacy constraint, we have to identify two elements: *sensitive information* and *privacy requirement*. In our setting, the sensitive information is attributes of a given schema. Each attribute has different levels of sensitivity (e.g. credit card number is more sensitive than phone number) and some of them are the source of revenue and business strategy and can only be shared under specific conditions. Whereas, the privacy requirement is to prevent adversaries from linking sensitive attributes back to original schemas. To capture these elements, we employ the concept of *presence constraint* that defines an upper bound of the probability $Pr(D \in s|\hat{S})$ that an adversary can disclose the presence of a set of attributes D in a particular schema s, if he sees an anonymized schema $\hat{S}$. Formally, we have:

Definition 2 ***Presence Constraint.*** *A presence constraint γ is a triple $\langle s, D, \theta \rangle$, where s is a schema, D is a set of attributes, and θ is a pre-specified threshold. An anonymized schema $\hat{S}$ satisfies the presence constraint γ if $Pr(D \in s|\hat{S}) \leq \theta$.*

Intuitively, the presence constraint protects a given schema from being known what attributes it has. The more chances of knowing the exact attribute(s) a schema contains, the riskier an adversary can attack and exploit the owner database of that schema. Hence, the presence threshold θ is given to limit the possibilities of linking the attributes back

to original schemas by hiding the attributes whose provenances (i.e. attribute-schema links) can be inferred with probabilities higher than θ. Choosing an appropriate value for θ depends on the domain applications and beyond the scope of this work.

Checking presence constraint. Given an anonymized schema $\hat{S}$ constructed from the affinity instance I and a presence constraint $\gamma = \langle s, D, \theta \rangle$, we can check whether $\hat{S}$ satisfies γ by computing the probability $Pr(D \in s \mid \hat{S})$ as follows:

$$Pr(D \in s \mid \hat{S}) = \begin{cases} \frac{1}{\prod_{i=1}^{k} |\hat{A}_i|}, \text{ if } \hat{A}_1, \ldots, \hat{A}_k \neq \emptyset \\ 0, \qquad \textit{otherwise} \end{cases} \tag{3}$$

where $D = \{a_1, \ldots, a_k\}$ is a subset of attributes of s and $\hat{A}_i \in \hat{S}$ is the abstract attribute (a group of attributes at the same row of I) that contains a_i. In case there is no abstract attribute which contains a_i, we consider $\hat{A}_i = \emptyset$. Eq. 3 comes from the assumption that to check an attribute a_i belongs to s, an adversary has to exhaustively try every attribute of $\hat{A}_i$, thus the probability of a successful disclosure is $Pr(a_i \in s|\hat{S}) = 1/|\hat{A}_i|$. Consequently, the probability of checking a subset of attributes is simply a product of elemental probabilities, since the anonymized schema generated by our model does not imply any dependence between abstract attributes.

Example 2. Continuing our running example in Fig. 2, we consider an anonymized schema $\hat{S} = \{\hat{A}_1, \hat{A}_2\}$, where $\hat{A}_1 = \{a_1, c_1\}$ and $\hat{A}_2 = \{a_2, b_2\}$. An adversary wants to check if two attributes a_1 and a_2 belongs to s_1 but he can only see $\hat{S}$. Based on Eq. 3, we have $Pr(\{a_1, a_2\} \in s_1 \mid \hat{S}) = \frac{1}{|\hat{A}_1| \cdot |\hat{A}_2|} = 0.25$. If we consider a presence constraint $\gamma = \langle s_1, \{a_1, a_2\}, 0.6 \rangle$, the anonymized schema $\hat{S}$ can be accepted as system output.

Privacy vs. utility trade-off. It is worth noting that there is a trade-off between privacy and utility [23]. In general, anonymization techniques reduce the utility of querying results when trying to provide privacy protection [24]. On one hand, the utility of an anonymized schema is maximal when there is no presence constraint (i.e. $\theta = 1$). On the other hand, when a contributor set the threshold θ of a presence constraint $\langle s, D, \theta \rangle$ very small (closed to 0), D will not appear in the anonymized schema. Consequently, the utility of this anonymized schema is low since we loss the information of these attributes. As a result, privacy should be seen as a spectrum on which contributors can choose their place. From now on, we use two terms—*privacy constraint* and *presence constraint*—interchangeably to represent the privacy spectrum defined by contributors.

4 Maximizing Anonymized Schema

Maximizing anonymized schema is the selection of an optimal (w.r.t. utility function) anonymized schema among potential ones that satisfy pre-defined privacy constraints. Formally, the maximization problem of our schema reuse setting is defined as follows.

Problem 1 ***(Maximizing Anonymized Schema)*** *Given a schema group $\mathcal{S}$ and a set of privacy constraints Γ, construct an anonymized schema $\hat{S}$ such that $\hat{S}$ satisfies all constraints Γ, i.e. $\hat{S} \models \Gamma$, and the utility value $u(\hat{S})$ is maximized.*

Theorem 1 *Let $\mathcal{S}$ be a schema group and Γ be a set of privacy constraints. Then, for a constant U, the problem of determining if there exists an anonymized schema $\hat{S} \models \Gamma$, whose utility value $u(\hat{S})$ is at most U, is NP-complete.*

The key of the proof is to transform the problem to *Maximum Independent Set* problem, which is known to be NP-complete [25]. For brevity sake, further details are referred to our report [11]. In this section, we consider the heuristic-based approach to relax optimization constraints and find the approximate solution in polynomial time.

4.1 Algorithm

In light of Theorem 1, we necessarily consider heuristic approaches to the maximizing anonymized schema problem. Our heuristic-based algorithm takes two parameters as input: a schema group modeled by an affinity matrix M and a set of presence constraints Γ. It will efficiently return an approximate solution—an anonymized schema $\hat{S}$ which satisfies Γ—with the trade-off that the utility value $u(\hat{S})$ is not necessarily maximal. The key difficulty is that during the optimization process, repairing the approximation solution to satisfy one constraint can break another. At a high level, our algorithm makes use of greedy and local-search heuristics to search the optimal anonymized schema in the space of conflict-free candidates (i.e. anonymized schemas without violations). The details are illustrated in Algorithm 1 and described in what follows.

Technically, we begin by constructing a valid affinity instance from the original affinity matrix (line 2 to line 7). The core idea is we generate an initial affinity instance $I = M$ and then continuously remove the attributes of I until all constraints are satisfied ($\Gamma_s = \Gamma$). The challenge then becomes how to choose which attributes for deletion such that the resulting instance has maximal utility. To overcome this challenge, we use a repair routine (line 3 to line 7) which greedily removes the attributes out of I to eliminate all violations. The greedy choice is that at each stage, we remove the attribute $\hat{a}$ with lowest utility reduction (line 5); i.e. choose an attribute a such that the utility of I after removing a is maximized. However, this greedy strategy could make unnecessary deletions since some constraints might share common attributes, thus making the utility not maximal. For example in Fig. 2, consider the anonymized schema $\hat{S} = \{\{a_1, c_1\}, \{a_2, b_2\}\}$ with two predefined constraints $\gamma_1 = Pr(a_1 \in s_1 \mid \hat{S}) \leq 0.4$ and $\gamma_2 = Pr(\{a_1, a_2\} \in s_1 \mid \hat{S}) \leq 0.1$. The best way to satisfy these constraints is removing only attribute a_2, but the algorithm will delete a_1 first and then a_2. Hence, we need to insert a_2 back into $\hat{S}$ to increase the utility. For this purpose, A_{del} stores all deleted attributes (line 6) for the following step.

The above observation motivates the next step in which we will explore neighbor instances using local search and keep track of the instance with highest utility (line 9 to line 20). Starting with the lower-bound instance from the previous step ($I_{op} = I$), the local search is repeated until the termination condition Δ_{stop} is satisfied (e.g. k iterations or time-out). In each iteration, we incrementally increase the utility of the current instance I by reverting unnecessary deletions from the previous step. The raising issue is how to avoid local optima if we just add an attribute with highest utility gain; i.e. choose an attribute a such that the utility of I is maximized after adding a. To tackle this issue, we perform two routines. (1) *Insertion:* Each attribute $a \in A_{del}$ is a possible candidate to insert into I (line 11). We use Roulette-wheel selection [26] as a non-deterministic strategy, where an attribute with greater utility has higher probability to add first (line 12). (2) *Repair:* When a particular attribute is inserted, it might make some constraints unsatisfied ($\Gamma_s \neq \Gamma$). Thus, the repair routine is invoked again to eliminate new violations by removing the potential attributes out of I (line 15 to line 19).

In an iteration of local-search, an attribute could be inserted into an affinity instance and then removed immediately by the repair routine, making this instance unchanged

Algorithm 1: Heuristics-based Algorithm

input : $\langle M, \Gamma \rangle$ // An affinity matrix M and a set of predefined constraints Γ
output : $\mathcal{I}_{op}$ // Maximal Affinity Instance
denote : $u(I^{-a})$: utility of I after removing attribute a out of I
$u(I^{+a})$: utility of I after adding attribute a into I

1 $A_{del} = \emptyset$ // Set of deleted attributes; $\Gamma_s = \emptyset$ // Set of satisfied constraints
2 $I = M$
3 **while** $\Gamma_s \neq \Gamma$ **do**
4 **/* Violation Repair: greedy deletion until satisfying constraints */**
5 $\hat{a} = \text{argmax}_{a \in \{I_{ij}\}} u(I^{-a})$
6 $I.delete(\hat{a})$; $A_{del} = A_{del} \cup \{\hat{a}\}$
7 $\Gamma_s = I.checkConstraints()$
8 $I_{op} = I$; $T = Queue[k]$ // a queue with fixed size k
9 **while** Δ_{stop} **do**
10 **/* Local search by non-deterministic insertion */**
11 $\Omega = \{\langle a, u(I^{+a}) \rangle : a \in A_{del}\}$
12 $\hat{a} = \Omega.rouletteSelection()$
13 $I.add(\hat{a})$; $T.add(\hat{a})$
14 $\Gamma_s = I.checkConstraints()$
15 **while** $\Gamma_s \neq \Gamma$ **do**
16 **/* Violation Repair: greedy deletion until satisfying constraints */**
17 $\hat{a} = \text{argmax}_{a \in \{I_{ij}\} \setminus T} u(I^{-a})$
18 $I.delete(\hat{a})$
19 $\Gamma_s = I.checkConstraints()$
20 **if** $u(I) > u(I_{op})$ **then** $I_{op} = I$
21 **return** I_{op}

and leading to being trapped in local optima. For this reason, we employ the Tabu search method [27] that uses a fixed-size "tabu" (forbidden) list T of attributes so that the algorithm does not consider these attributes repeatedly. In the end, the maximal affinity instance I_{op} is returned by selecting the one with the highest utility among explored instances (line 20).

4.2 Algorithm Analysis

Our algorithm is terminated and correct. Termination follows from the fact that the stopping condition Δ_{stop} can be defined to execute at most k iterations (k is a tuning parameter). The correctness follows from the following points. (1) When a new attribute a added into I (line 13) violates some constraints, I is repaired immediately. This repair routine takes at most $O(|M|)$, where $|M|$ is the number of attributes of the schema group represented by M. (2) The newly added attributes are not removed in the repair routine since they are kept in the "tabu" list T. The generation of Ω takes at most $O(|M|)$. (3) I_{op} always maintains the instance with maximal utility (line 20). Sum it up, the worse-case running time is $O(k \times |M|^2)$—which is reasonably tractable for real datasets.

Looking ahead for repair cost. We have a heuristic to improve the repair technique. The idea of this heuristic is to avoid bad repairs (that cause many deletions) by adding some degree of lookahead to the repair cost (i.e. number of deletions). In Algorithm 1, the utility value $u(I^{-a})$ is used as an indicator to select an attribute for deletion. Now we modify this utility value to include a look-ahead approximation of the cost of deleting one further attribute. More precisely, when an attribute a is considered for deletion (first step), we will consider a next attribute a' given that a is already deleted (second step). Our look-ahead function will be the utility of the resulting instance after these two steps. In other words, we prevent a bad repair by computing the utility of the resulting instances one step further. Due to space limit, further heuristics for improve the algorithm performance are omitted and can be found in the report version of this work [11].

5 Experiments

To verify the effectiveness of the proposed framework in designing and constructing the anonymized schema, following experiments are performed: (i) effects of different heuristics for the maximizing anonymized schema problem, (ii) the correctness of utility function, (iii) the computation time of our heuristic-based algorithm in various settings, and (iv) the tradeoff between utility and privacy. We proceed to report the results on real datasets and synthetic data.

5.1 Experimental Settings

Datasets. Our experiments are conducted on two types of data: real data and synthetic data. While the real data provides a pragmatic view on real-world scenarios, the synthetic data helps to evaluate the performance with flexible control of parameters.

- *Real Data.* We use a repository of 117 schemas that is available at [28]. To generate attribute correspondences, we used the well-known schema matcher COMA++ [16].
- *Synthetic Data.* We want to simulate practical cases in which one attribute is used repeatedly in different schemas (e.g. 'username'). To generate a set of n synthetic schemas, two steps are performed. In the first step, we generate k core schemas, each of which has m attributes, such that all attributes within single schema as well as across multiple schemas are completely different. Without loss of generality, for any two schemas a and b, we create all 1-1 correspondences between their attributes; i.e. $(a_i, b_i)_{1 \le i \le m}$. In the second step, $n - k$ remaining schemas are permuted from k previous ones. For each schema, we generate m attributes, each of which is copied randomly from one of k attributes which have all correspondences between them; i.e. attributes at the same row of the affinity matrix.

Privacy Constraints. To generate a set of privacy constraints Γ for simulation, we need to consider two aspects:

- *The number of attributes per constraint*. To mix the constraints with different size, we set a parameter α_i as the number of constraints with i attributes. Since the chance for an adversary to find the correct provenance of a group of four or more attributes is often small with large abstract attributes (Eq. 3), we only consider constraints with less than four attributes (i.e. $\alpha_{i \ge 4} = 0$). For all experiments, we set $\alpha_1 = 50\%$, $\alpha_2 = 30\%$, and $\alpha_3 = 20\%$.
- *Privacy threshold*. We generate the privacy threshold θ of all constraints according to uniform distribution in the range $[a, b]$, $0 \le a, b \le 1$. In all experiments, we set $a = (1/n)^i$ and $b = (2/n)^i$, where n is the number of schemas and i is the number of attributes in the associated constraint.

In general, the number of constraints is proportional to the percentage of the total number of attributes; i.e. $|\Gamma| \propto m \cdot n$). In each experiment, we will vary the percentage value differently.

Evaluation Metrics. Beside measuring the computation time to evaluate proposed heuristics, we also consider two important metrics:

- *Privacy Loss:* measures the amount of disagreement between the two probability distributions: actual privacy and expected privacy. Technically, given an anonymized schema $\hat{S}$ and a set of presence constraints $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ where $\gamma_i = \langle s_i, D_i, \theta_i \rangle$,

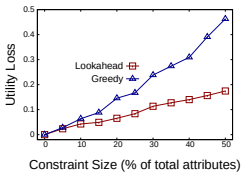

Fig. 3: Utility Loss (the lower, the better)

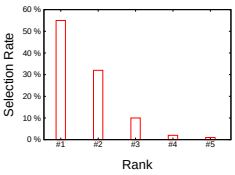

Fig. 4: Correctness of Utility Function

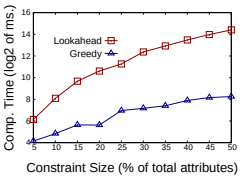

Fig. 5: Computation Time vs. #Constraints

we denote $p_i = Pr(D_i \in s_i|\hat{S})$. Privacy loss is computed by the Kullback-Leiber divergence between two distributions $P = \{p_i\}$ (actual) and $\Theta = \{\theta_i\}$ (expected):

$$\Delta p = KL(P \parallel \Theta) = \sum_i p_i \log \frac{p_i}{\theta_i} \tag{4}$$

– *Utility Loss:* measures the amount of utility reduction with regard to the existence of privacy constraints. Technically, denote $u_\emptyset$ is the utility of an anonymized schema without privacy constraints and u_Γ is the utility of an anonymized schema with privacy constraints Γ, utility loss is calculated as:

$$\Delta u = \frac{u_\emptyset - u_\Gamma}{u_\emptyset} \tag{5}$$

We implement our schema reuse framework in Java. All the experiments ran on an Intel Core i7 processor 2.8 GHz system with 4 GB of RAM.

5.2 Effects of Heuristics

This experiment will compare the utility loss of the anonymized schema returned by different heuristics. In this experiment, we study the proposed algorithm with two heuristics: without lookahead (Greedy) and with lookahead (Lookahead). We use a synthetic data with a random number of schemas $n \in [50, 100]$ and each schema has a random number of attributes $m \in [50, 100]$. In that, the copy percentage is varied from 10% to 20%; i.e. $k/n \in [0.1, 0.2]$. The number of constraints is varied from 5% to 50% of total number of attributes. The results for each configuration are averaged over 100 runs. Regarding utility function, we set $w = 0.5$ which means the completeness is equally preferred as the importance.

Fig. 3 depicts the result of two described heuristics. The X-axis shows the number of constraints, while the Y-axis shows the utility loss (the lower, the better). A noticeable observation is that the Lookahead heuristic performs better than the Greedy heuristic. Moreover, when there are more constraints, this difference is more significant (up to 30% when the number of constraints is 50%). This is because the Lookahead heuristic enables our algorithm to foresee and avoid bad repairs, which cause new violations when eliminating the old ones, thus improves the quality of the result.

5.3 Correctness of Utility Function

In this experiment, we would like to validate the correctness of our utility function by comparing the results of the proposed algorithm with the choices of users. Technically, we expect that the ranking of anonymized schemas by utility function is equivalent to their ranking by users. Regarding the setting, we construct 20 different schema groups from the real data by extracting a random number of schemas and attributes. For each

group, we present five of constructed anonymized schemas (ranked #1, #2, #3, #4, #5 by the decreasing order of utility) to 10 users (users do not know the order). These presented anonymized schemas are not top-5, but taken from the highest such that the utility difference between two consecutive ranks is at least 10% to avoid insignificant differences. Each user is asked to choose the best anonymized schema in his opinion (200 votes in total). We fix the number of constraints at 30% of total attributes. Since the Lookahead strategy is better than the Greedy strategy as presented in the previous experiment (Section 5.2), we use it to construct anonymized schemas.

The results are depicted in Fig. 4. X-axis shows top-5 anonymized schemas returned by our algorithm. Y-axis presents the selection rate (percentage of users voting for an anonymized schema), averaged over all schema groups. A key finding is that the order of selection rate corresponds to the ranking of solutions by the utility function. That is, the first-rank anonymized schema (#1) has highest selection rate (55%) and the last-rank anonymized schema (#5) has lowest selection rate (1%). Moreover, most of users opt for the first-rank solution (55%) and second-rank solution (33%), indicating that our algorithm is able to produce anonymized schemas similar to what users want in real-world datasets.

5.4 Computation Time

In this experiment, we design various settings to study the computation time with two above-mentioned strategies (Greedy vs. Lookahead) using synthetic data. In each setting, we measure the average computation time over 100 runs.

Effects of Input Size. In this setting, the computation time is recorded by varying the input size of the affinity matrix; i.e. the number of schemas and the number of attributes per schema. A significant observation in Table 2 is that the Greedy strategy outperforms the Lookahead strategy. This is because the cost of looking-ahead utility change is expensive with the trade-off that the utility loss is smaller as presented in Section 5.2.

Table 2: Computation Time (log2 of msec.)

Size of M *	Greedy	LookAhead
10×20	2.58	5.04
10×50	5.13	10.29
20×50	7.30	13.36
20×100	9.95	17.60
50×100	12.62	21.58

* $m \times n$: m attributes (per schema) and n schemas

Effects of Constraints Size. In this setting, the computation time is recorded by varying #constraints from 5% to 50% of total number of attributes. Based on results of the previous setting, we fix the input size at 20×50 for a low starting point. Fig. 5 illustrates the output, in which the X-axis is the constraint size and the Y-axis is the computation time (ms.) in logarithmic scale of base 2. A noticeable observation is that the Lookahead strategy is much slower than the Greedy strategy. In fact, adding degrees of look-ahead into utility function actually degrades the performance. The running time of those extra computations increases due to the large number of combinations of privacy constraints.

5.5 Trade-off between Privacy and Utility

In this experiment, we validate the trade-off between privacy and utility. Technically, we relax each presence constraint $\gamma \in \Gamma$ by increasing the threshold θ to $(1 + r)\theta$, where r is called relaxing ratio and varied according to normal distribution $\mathcal{N}(0.3, 0.2)$. Based on previous experiments, we use the Lookahead strategy on the synthetic data with

different input size and fix the number of constraints at 30% of total attributes. For a resulting anonymized schema, privacy is quantified by privacy loss (Eq. 4), while utility is quantified by utility loss (Eq. 5). For comparison purposes, both utility loss and privacy loss values are normalized to [0, 1] by the thresholding technique: $\Delta u = \frac{\Delta u - min_{\Delta u}}{max_{\Delta u} - min_{\Delta u}}$ and $\Delta p = \frac{\Delta p - min_{\Delta p}}{max_{\Delta p} - min_{\Delta p}}$, where $max_{\Delta u}$, $min_{\Delta u}$, $max_{\Delta p}$, $min_{\Delta p}$ are the maximum and minimum values of utility loss and privacy loss in this experiment.

Fig. 6 shows the distribution of all the points whose x and y values are privacy loss and utility loss, respectively. Each point represents the resulting anonymized schema in a specific configuration of the relaxing ratio and the input size. One can easily observe a trade-off between privacy and utility: the utility loss decreases when the privacy loss increases and vice-versa. For example, the maximal privacy (privacy loss = 0) is achieved when the utility is minimal (utility loss = 1). This is because the decrease of presence threshold θ will reduce the number of presented attributes, which contributes to the utility of resulting anonymized schemas.

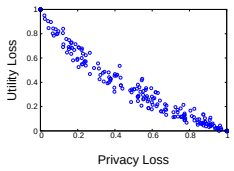

Fig. 6: Trade-off between privacy and utility

6 Related Work

We now review salient work in schema reuse, privacy models, anonymization techniques, and utility measures that is related to our research.

Schema reuse. There is a large body of research on schema-reuse. Some of the works focus on building large-scale schema repositories [4, 6, 29, 30], designing management models to maintain them [31, 32], and establishing semantic interoperability between collected schemas in those repositories [15, 16, 33]. While, some of the others support to find relevant schemas according to a user query [7, 8, 34], techniques to speed-up the querying process [35], and visualization aspects [36, 37]. Unlike these works, we study the problem of preserving privacy of schema information when presenting the schema attributes as hints for end-users to design new schemas.

Privacy Models. Privacy model is a mean to prevent information disclosure by representing human-defined policies under mathematical standards. Numerous types of information disclosure have been studied in the literature [38], such as attribute disclosure [39] and identity disclosure [40], and membership disclosure [41]. In the context of schema reuse, the linking of attribute to its containing schema can be considered as an information disclosure. There is a wide body of work have proposed privacy models at data level, including differential entropy [42], k-anonymity [43], l-diversity [44], and t-closeness [45]. Different from these works, the model proposed in this paper employs the concept of presence constraint defined at schema-level.

Anonymization Techniques To implement privacy models, there are a wide range of techniques have been proposed in literature such as generalization, bucketization, randomization, suppression, and differential privacy. In terms of schema reuse, these techniques can be interpreted as follows. The generalization technique [21] replaces the sensitive attribute with a common attribute, which still preserves the utility of attributes shown to users (no noises). The bucketization technique [46] extends the generalization

technique for multi-dimensional data through partitioning. The randomization technique [47, 48] adds a "noisy" attribute (e.g. 'aaa'). The suppression technique [49] replaces a sensitive attribute by a special value (e.g. '*'). The differential privacy technique [40] enhances both generalization and suppression techniques by balancing between preserving the utility of attributes and adding noises to protect them. Similar to most of other works, we adopt the generalization technique (which does not add noises) to preserve the utility of attributes shown to users. This utility is important in schema design since adding noisy attributes could lead to data inconsistencies.

Quality Measures. To quantify the quality of a schema unification solution, researchers have proposed different quantitative metrics, the most important ones among which are completeness and minimality [50, 51]. While the former represents the percentage of original attributes covered by the unified schema, the latter ensures that no redundant attribute appears in the unified schema. Regarding the quality of a schema attribute, [52] also proposed a metric to determine whether it should appear in schema unification. Combining these works and applying their insights in the schema-reuse context, this paper proposed a comprehensive notion of *utility* that measures the quality of an anonymized schema by capturing how many important attributes it contains and how well it covers a wide range of attribute variations.

7 Conclusions

In this work, we addressed the challenge of preserving privacy when integrating and reusing schemas in schema-reuse systems. To overcome this challenge, we introduced a framework for enabling schema reuse with privacy constraints. Starting with a generic model, we introduce the concepts of schema group, anonymized schema and privacy constraint. Based on this model, we described utility function to measure the amount of information of an anonymized schema. After that, we formulated privacy-preserving schema reuse as a maximization problem and proposed a heuristic-based approach to find the maximal anonymized schema efficiently. In that, we investigated various heuristics (greedy repair, look-ahead, constraint decomposition) to improve the computation time of the proposed algorithm as well as the utility of the resulting anonymized schema. Finally, we presented an empirical study that corroborates the efficiency of our framework with main results: effects of heuristics, the correctness of utility function, the guideline of computation time, and the trade-off between privacy and utility.

Our work opens up several future research directions. One pragmatic direction is to support a "pay-as-you-use" fashion in schema-reuse systems. In that, reuse should be defined at a level finer than complete schemas, as schemas are often composed of meaningful building blocks. Another promising direction is the scalability issues of browsing, searching, and visualizing the anonymized schema when the number of original schemas becomes very large. Moreover, our techniques can be applied to other domains such as business process and component-based design.

Acknowledgment

The research has received funding from the EU-FP7 EINS project (grant number 288021) and the EU-FP7 PlanetData project (grant number 257641).

References

[1] http://schema.org/.
[2] http://www.factual.com/.
[3] K. Bollacker et al. “Freebase: a collaboratively created graph database for structuring human knowledge”. In: *SIGMOD*. 2008, pp. 1247–1250.
[4] H. Gonzalez et al. “Google fusion tables: web-centered data management and collaboration”. In: *SIGMOD*. 2010, pp. 1061–1066.
[5] M. Bentounsi et al. “Anonyfrag: an anonymization-based approach for privacy-preserving BPaaS”. In: *Cloud-I*. 2012, 9:1–9:8.
[6] K. Chen et al. “Exploring schema repositories with schemr”. In: *SIGMOD Rec.* (2011), pp. 11–16.
[7] A. Das Sarma et al. “Finding related tables”. In: *SIGMOD*. 2012, pp. 817–828.
[8] G. Limaye et al. “Annotating and searching web tables using entities, types and relationships”. In: *VLDB*. 2010, pp. 1338–1347.
[9] C. Clifton et al. “Privacy-preserving data integration and sharing”. In: *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. ACM. 2004, pp. 19–26.
[10] W. Halfond et al. “A classification of SQL-injection attacks and countermeasures”. In: *IEEE*. 2006, pp. 65–81.
[11] Q. V. H. Nguyen et al. “Towards Enabling Schema Reuse with Privacy Constraints”. In: *EPFL-REPORT-189971*. 2013.
[12] F.-C. Tsui et al. “Technical description of RODS: a real-time public health surveillance system”. In: *Journal of the American Medical Informatics Association* 10.5 (2003), pp. 399–408.
[13] A. I. Antón et al. “A roadmap for comprehensive online privacy policy management”. In: *Communications of the ACM* 50.7 (2007), pp. 109–116.
[14] Q. V. H. Nguyen et al. “Pay-as-you-go Reconciliation in Schema Matching Networks”. In: *ICDE*. 2014.
[15] P. Bernstein et al. “Generic Schema Matching, Ten Years Later”. In: *VLDB*. 2011, pp. 695–701.
[16] D. Aumueller et al. “Schema and ontology matching with COMA++”. In: *SIGMOD*. 2005, pp. 906–908.
[17] E. Peukert et al. “AMC - A framework for modelling and comparing matching systems as matching processes”. In: *ICDE*. 2011, pp. 1304–1307.
[18] N. Hung et al. “On Leveraging Crowdsourcing Techniques for Schema Matching Networks”. In: *DASFAA*. 2013, pp. 139–154.
[19] Q. V. H. Nguyen et al. “Minimizing Human Effort in Reconciling Match Networks”. In: *ER*. 2013.
[20] Q. V. H. Nguyen et al. “Collaborative Schema Matching Reconciliation”. In: *CoopIS*. 2013.
[21] R. J. Bayardo et al. “Data privacy through optimal k-anonymization”. In: *ICDE*. 2005, pp. 217–228.
[22] J. M. Kleinberg. “Authoritative sources in a hyperlinked environment”. In: *JACM* (1999), pp. 604–632.
[23] T. Li et al. “On the tradeoff between privacy and utility in data publishing”. In: *SIGKDD*. 2009, pp. 517–526.
[24] J. Brickell et al. “The cost of privacy: destruction of data-mining utility in anonymized data publishing”. In: *KDD*. 2008, pp. 70–78.
[25] R. M. Karp. “Reducibility Among Combinatorial Problems”. In: *CCC*. 1972, pp. 85–103.
[26] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
[27] F. Glover et al. “The general employee scheduling problem: an integration of MS and AI”. In: *COR* (1986), pp. 563–573.
[28] http://lsirwww.epfl.ch/schema_matching/.
[29] J. Madhavan et al. “Corpus-based schema matching”. In: *ICDE* (2005), pp. 57–68.
[30] K. P. Smith et al. “Unity: Speeding the creation of community vocabularies for information integration and reuse”. In: *IRI*. 2011, pp. 129–135.
[31] M. Franklin et al. “From databases to dataspaces: a new abstraction for information management”. In: *SIGMOD Rec.* (2005), pp. 27–33.
[32] P. A. Bernstein et al. “Model management 2.0: manipulating richer mappings”. In: *SIGMOD*. 2007, pp. 1–12.
[33] A. Das Sarma et al. “Bootstrapping Pay-As-You-Go Data Integration Systems”. In: *SIGMOD*. 2008, pp. 861–874.
[34] M. J. Cafarella et al. “WebTables: exploring the power of tables on the web”. In: *VLDB*. 2008, pp. 538–549.
[35] H. A. Mahmoud et al. “Schema clustering and retrieval for multi-domain pay-as-you-go data integration systems”. In: *SIGMOD*. 2010, pp. 411–422.
[36] B. Yost et al. “Visualizing Schema Clusters for Agile Information Sharing”. In: *InfoVis*. 2009, pp. 5–6.
[37] K. Smith et al. “Exploring schema similarity at multiple resolutions”. In: *SIGMOD*. 2010, pp. 1179–1182.
[38] D. Lambert. “Measures of disclosure risk and harm”. In: *JOS* (1993), pp. 313–313.
[39] G. T. Duncan et al. “Disclosure-limited data dissemination”. In: *JASA* (1986), pp. 10–18.
[40] C. Dwork. “Differential privacy”. In: *ICALP*. 2006, pp. 1–12.
[41] M. E. Nergiz et al. “Hiding the presence of individuals from shared databases”. In: *SIGMOD*. 2007, pp. 665–676.
[42] D. Agrawal et al. “On the Design and Quantification of Privacy Preserving Data Mining Algorithms”. In: *PODS*. 2001, pp. 247–255.
[43] L. Sweeney. “k-anonymity: a model for protecting privacy”. In: *IJUFKS* (2002), pp. 557–570.
[44] A. Machanavajjhala et al. “L-diversity: Privacy beyond k-anonymity”. In: *TKDD* (2007), pp. 24–24.
[45] N. Li et al. “t-closeness: Privacy beyond k-anonymity and l-diversity”. In: *ICDE*. 2007, pp. 106–115.
[46] X. Xiao et al. “Anatomy: Simple and effective privacy preservation”. In: *VLDB*. 2006, pp. 139–150.
[47] N. R. Adam et al. “Security-control methods for statistical databases: a comparative study”. In: *CSUR* (1989), pp. 515–556.
[48] R. Agrawal et al. “Privacy-preserving data mining”. In: *SIGMOD Rec.* (2000), pp. 439–450.
[49] V. S. Iyengar. “Transforming data to satisfy privacy constraints”. In: *SIGKDD*. 2002, pp. 279–288.
[50] M. Batista et al. “Information Quality Measurement in Data Integration Schemas”. In: *QDB*. 2007, pp. 61–72.
[51] F. Duchateau et al. “Measuring the quality of an integrated schema”. In: *ER*. 2010, pp. 261–273.
[52] C. Yu et al. “Schema summarization”. In: *VLDB*. 2006, pp. 319–330.