Anonymization of Data Sets with NULL Values

Margareta Ciglic, Johann Eder(✉), and Christian Koncilia

Department of Informatics Systems, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria
{margareta.ciglic,johann.eder,christian.koncilia}@aau.at

Abstract. Releasing, publishing or transferring microdata is restricted by the necessity to protect the privacy of data owners. k-anonymity is one of the most widespread concepts for anonymizing microdata but it does not explicitly cover NULL values which are nevertheless frequently found in microdata. We study the problem of NULL values (missing values, non-applicable attributes, etc.) for anonymization in detail, present a set of new definitions for k-anonymity explicitly considering NULL values and analyze which definition protects from which attacks. We show that an adequate treatment of missing values in microdata can be easily achieved by an extension of generalization algorithms. In particular, we show how the proposed treatment of NULL values was incorporated in the anonymization tool ANON, which implements generalization and tuple suppression with an application specific definition of information loss. With a series of experiments we show that NULL aware generalization algorithms have less information loss than standard algorithms.

Keywords: Privacy · k-anonymity · NULL values · Missing values

1 Introduction

Detailed data collections are an important resource for research, for fact based governance, or for knowledge based decision making. In the field of statistical databases any collection of data with detailed information on entities, in particular persons and organizations, is called microdata.

A crucial requirement for the release of microdata is the preservation of the privacy of the data owners, which is protected by laws and regulations. Furthermore, for data collections requiring the willingness of data owners to share (donate) their data, studies [9] clearly indicate that the protection of privacy is one of the major concerns of data owners and decisive for a consent to donate data [13]. For protecting privacy from linkage attacks the concept of k-anonymity [25] received probably the widest attention. Its core idea is to preserve privacy

The work was supported by *Austrian Ministry of Science and Research* within the project BBMRI.AT (GZ 10.470/0016-II/3/2013) and *Technologie- und Methodenplattform für die vernetzte medizinische Forschung e.V.* (*TMF*) within the project ANON.

A. Hameurlain et al. (Eds.): TLDKS XXIV, LNCS 9510, pp. 193–220, 2016.
DOI: 10.1007/978-3-662-49214-7_7

by hiding each individual in a crowd of at least k members. Many anonymization algorithms implementing these concepts were developed.

Surprisingly, neither the original definition of k-anonymity nor any of the many anonymization algorithms deals explicitly with unknown, or missing values (NULL values in database terms) in microdata. We could not find a single source discussing the problem of NULL values in microdata for anonymization. Recent surveys [19] or textbooks [12] do not mention NULL values or missing values. However, all techniques and algorithms we found, explicitly or implicitly require that all records with at least one NULL value have to be removed from a table before it can be anonymized ([2,15–18,23,29,32], and many more). There is only some treatment of NULL values in form of suppressed values, i.e. NULL values resulting from removing ("suppressing") data in the course of anonymization procedures. Attacks on suppressed rows can be found in [22,30]. [1,7,21] discuss suppression of values in single cells. However, neither of these approaches discusses the problem of missing values in the original data or of non-existing values due to non-applicable attributes.

NULL values, nevertheless, are not exceptional in microdata, e.g. they appear frequently in data sets for medical research [8,10,28,33]: Some attributes might not be applicable for each patient. A patient might have refused to answer some questions in a questionnaire or could not be asked due to physical or mental conditions. In an emergency some test might not have been performed, etc.

Anonymization by generalization and suppression of data cause loss of information. The aim of reducing this information loss triggered many research efforts. The ignorance of NULL values in anonymization algorithms results in dropping rows from a table, causing a considerable loss of information. Furthermore, dropping rows with NULL values also could introduce some bias in the data set, which is not contained in the original table. This is of course unfortunate for further analysis of the data (for example in evidence based medicine) and might compromise the statistical validity of the results (e.g. dropping rows with a NULL value in the field *occupation* would skip all children from the data set and introduce an age bias, which was not present in the original data set).

This paper is an extension of [4]. We provide a thorough grounding for the treatment of NULL values in anonymization algorithms. We show that we can reduce the problem of NULL values in k-anonymity to different definitions of matching between rows of a data set based on extending the comparison of values and NULL values. We show that generalization algorithms, widely used for anonymization, can be easily extended to cover NULL values and we show that this extension reduces information loss during anonymization significantly. In particular, we show how the treatment of NULL values was incorporated into the anonymization tool ANON, which implements a generalization algorithm with tuple suppression, which optimizes application specific usability of the anonymized data by minimizing the information loss, which is defined by the user, specifically for the application and the intended use of data [3,27,28].

k-anonymity (which is defined on the quasi identifiers) has to be complemented with ℓ-diversity for sensitive attributes [18] to avoid that an adversary

might infer data of individual. In this paper, however, we focus on k-anonymity and only briefly treat ℓ-diversity, as it is implemented in ANON.

2 k-Anonymity Revisited

A detailed collection and representation of data on information subjects is called microdata - as opposed to data in less detail like statistical data. For this paper a microdata table is a multiset of rows [14]. We can classify the attributes in the schema of such table in four categories: (1) identifiers: all attributes which uniquely identify a row in the table, (2) quasi identifier: all attributes which an adversary might know, (3) sensitive attributes: attributes with values that should not be inferable by an adversary and (4) all other data. For this paper we assume that the identifiers have already been removed from a table and that the schema of a table includes a set of quasi identifiers Q, which we denote by $Q_1, \ldots, Q_n$.

The aim of anonymization is to assure that a table can be published without opening an adversary the possibilities to gain additional knowledge about the data subjects.

Table 1 shows our running example for such a table with the quasi identifiers `Gender`, `Height`, `Job`, and `ZIP` and the sensitive attribute `Condition`.

Table 1. Original table

	Gender	Height	Job	ZIP	Condition
A	f	165	NULL	9020	Cancer
B	m	187	Mayor	9020	Hepatitis
C	f	163	Clerk	9020	Flu
D	m	NULL	Technician	9020	Pneumonia
E	m	183	NULL	9020	Malaria
F	m	189	Pilot	9020	Gastritis

Samarati and Sweeney [25] proposed an approach to preserve the privacy of a data owner by hiding each data owner in a crowd of at least k individuals, such that an adversary might not get detailed information about an individual, but only information about a group of k individuals. The larger the k, the smaller the possible information gain of an adversary.

In [26] k-anonymity is defined as follows: 'Each release of data must be such that every combination of values of quasi identifiers can be indistinctly matched to at least k individuals'. The term *indistinct match* is not defined explicitly, nevertheless, it is clear from the context that two rows match, if they have identical values in the quasi identifiers. However, missing values are not mentioned. We basically follow this definition here, and analyze, how rows of a table match in case some values are NULL. Hence we formalize the notion of k-anonymity, dependent on some match operator.

Definition 1 ***(k-Anonymity).*** *Let T be a table and Q the set of quasi identifier attributes and let $\sim$ be a match predicate on T. T is k-anonymous with respect to $\sim$, iff $\forall t \in T : |\{t' | t \sim t'\}| \geq k$.*

k-Anonymity as well as ℓ-diversity can be achieved with two basic techniques, *generalization* and *suppression*. Both techniques decrease information content of the data to meet the required degree of privacy. They can be used separate or in combination. Generalization [26,30] replaces the values of quasi identifiers (QID) with more general values defined in the generalization hierarchies (taxonomy trees or intervals with step definitions) for all QIDs. The leaves of a generalization hierarchies are the original values of the domain of an attribute, the top level is a single value ALL, which has no information content. The generalization hierarchy and its corresponding domains of the QID *Job* of the running example (Table 1) are shown in Fig. 1.

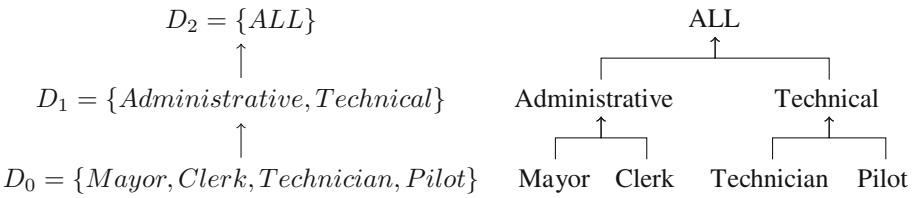

Fig. 1. Generalization hierarchy and its corresponding domains (generalization levels) of the QID *Job* of the running example

The anonymization procedure in general is as follows: When a row does not match at least $k-1$ other rows, then some attribute values are generalized, i.e. replaced with the parent of this value in the generalization hierarchy defined for each domain (resp. each attribute). In the case of local recoding this is done for individual rows, for global recoding or full-domain generalization scheme the generalization is performed for all the rows [12,31]. This is repeated until the table is k-anonymous or the highest level of generalization is reached in all attributes. A shortcoming of this method is that outlier tuples in the microdata can lead to a very coarse grain generalization. Outlier tuples are those, which hardly match any other tuples. If they remain in the table that is being anonymized, the overall information loss increases. To avoid information loss caused by such outliers, full domain generalization is mostly accompanied with row suppression, where given fraction of rows might be suppressed, i.e. these rows are removed from the table or all their values are replaced by ALL (resp. NULL). To avoid attacks at least k rows have to be suppressed [30].

It is easy to see that in general several tables qualify as result. The aim is now to compute the table with the lowest information loss. The problem is known to be NP complete [21]. However, for global recoding the complexity of

the method is exponential in the number of quasi identifiers and their generalization hierarchy height and not in the number of tuples. Many algorithms have been proposed, which apply heuristics to reduce complexity and which apply different measures for information loss to efficiently compute "good" anonymized tables.

In contrast to generalization, suppression does not transform the values to other, more general values, but simply deletes (eliminates) them. Suppression can be undertaken on single values (called cell suppression), on whole tuples (called tuple suppression) or on whole attributes (called attribute suppression). The impact of the attribute suppression is the same as the one of the generalization of an attribute to the top level. Approximation algorithms that use cell suppression are described in [1,21]. In combination with generalization [26,30], tuple suppression can be used to eliminate outlier tuples, while the remaining tuples get generalized.

3 k-Anonymity with NULL values

3.1 NULL Values

NULL values [20] are the standard way of representing missing information in database tables. We can distinguish three kinds of NULL values: (1) attribute not applicable: in this case there is no value for this attribute for this row in the world represented in the database. (2) missing value: there exists a value in the world, but it is not contained in the database. (3) no information: it is not known whether the value exists in the world or not. In SQL the semantics of NULL is "no information".

For the following considerations we follow the treatment of NULL values in SQL [14]. This means in particular, that a comparison of a NULL value with any other value never results in TRUE and there is a special unary predicate *is* NULL to test for NULL values.

3.2 Matching NULL Values

For matching of NULL values we have the following options:

- basic match: NULL values do not match with NULL values, nor with any other value.
- extended match: NULL values match with NULL values.
- maybe match: NULL values match with any value including NULL values.

In the original definition of k-anonymity and in the current anonymization algorithms basic match is used. It is in accordance with the definition in SQL, where 'A = B' is not true, if A or B are NULL values. Extended match treats a NULL value like any other value. Maybe match sees NULL values as wildcards for matching. It corresponds to Codd's maybe selection [6], where rows are returned, if the selection predicate is true for a substitution of the NULL values.

3.3 Basic Match

We call the match used in [26], where rows with NULL values are discarded, *basic match*, and formally define it as follows:

Definition 2 ***(Basic match).*** *Let T be a table and Q the set of quasi identifier attributes.*

$t_1 \sim_b t_2 :\Longleftrightarrow \forall q \in Q : t_1[q] = t_2[q]$

For illustrating the different match definitions and their consequences we use Table 1 and transform it to a 2-anonymous table using the different match definitions in turn. Table 2 shows the result of the anonymization of Table 1 to a 2-anonymous table. The table has only 3 rows, as all rows of the original table which contain NULL values (rows A, D, and E) had to be removed before the generalization - hence the table is also 3-anonymous. For the rest of this paper we always follow the full-domain generalization scheme [17,24,26] in our examples, however, the considerations are applicable to all algorithms for k-anonymization.

Table 2. 3-anonymity with basic match

	Gender	Height	Job	ZIP	Condition
B	All	All	All	9020	Hepatitis
C	All	All	All	9020	Flu
F	All	All	All	9020	Gastritis

3.4 Extended Match

In extended match NULL values are treated like any other value, in particular, a NULL value only matches with another NULL value but not with any values from the domains of the attributes.

Definition 3 ***(Extended match).*** *Let T be a table and Q the set of quasi identifier attributes of T. For two rows $t_1, t_2 \in T$ we define the extended match as*

$$t_1 \sim_e t_2 :\Longleftrightarrow \forall q \in Q : t_1[q] = t_2[q] \vee (t_1[q] is \text{NULL} \wedge t_2[q] is \text{NULL})$$

The extended matching definition can be used to extend existing anonymization algorithms. First we have to extend all generalization hierarchies with a branch with the value NULL on each level of the hierarchy below the root 'ALL'. Using these extended hierarchies we can apply the generalization method again and receive Table 3 which is 2-anonymous with respect to the extended match. Note that in contrast to the basic match no row has been lost.

The aim of k-anonymity is to prevent attacks on released data, in particular, record linking attacks [12], i.e. joining a table with some known information

Table 3. 2-anonymity with extended match

	Gender	Height	Job	ZIP	Condition
A	All	All	NULL	9020	Cancer
B	All	All	Admin	9020	Hepatitis
C	All	All	Admin	9020	Flu
D	All	All	Technical	9020	Pneumonia
E	All	All	NULL	9020	Malaria
F	All	All	Technical	9020	Gastritis

to associate values of sensitive attributes with some data owner. In particular, matching any record containing quasi identifiers with the released table should result in no or at least k hits. It is easy to see that this requirement is fulfilled, if any query posed on the released table in the form of "Select * From T where search_condition" yields 0 or at least k result rows, if the search condition only contains predicates on quasi identifiers.

Theorem 1 ***(link-safe).*** *Let T be a table and Q a set of quasi identifier attributes and let $\pi_Q T$ be the projection of T on Q. If T is k-anonymous with respect to $\sim_e$ then for all search conditions p the query "Select * From $\pi_Q T$ where p" returns 0 or at least k rows.*

Proof. The theorem follows from the observation that if a row $t \in \pi_Q T$ satisfies the search condition of the query then all rows matching t according to the extended match also satisfy the search condition. Because T is k-anonymous with respect to $\sim_e$ there are at least k such rows.

3.5 Maybe Match

For extending the domain of the match predicate to also consider NULL values we can build on the treatment of NULL values in Codd's *maybe* operations for the relational algebra [6]. The maybe selection operator does not only return those rows for which the selection predicate is satisfied, but also all those rows which satisfy the selection predicate if NULL values are replaced by suitable values.

Applying the concept of 'maybe' selects to the matching of rows, we define the maybe match as follows: NULL matches both with NULL and other values. NULL values in the rows can be used as wildcards in both directions. For an example: The tuples (a, b, c), (a, NULL, c), (NULL, NULL, NULL) all match can be grouped together.

Definition 4 ***(maybe match).*** *Let T be a table and Q the set of quasi identifier attributes of T. For two rows $t_1, t_2 \in T$ we define the maybe match as*

$$t_1 \sim_m t_2 :\Longleftrightarrow \forall q \in Q : t_1[q] = t_2[q] \vee (t_1[q] is \text{NULL} \vee t_2[q] is \text{NULL}).$$

The maybe match is not transitive and does not lead to an equivalence partitioning of the table (in contrast to basic and extended match). We call a row t and its matching rows the *match-group* of t. k-groups of different rows might overlap without being equal. For an example, the tuple (a, NULL, NULL) matches both with (a, b, c) and with (a, e, f) and so (a, NULL, NULL) is contained in several match-groups.

Applying maybe matching in the generalization method we compute the table shown in Table 4. In this table the match-groups are built from the following match relations $A \sim_m C$, $B \sim_m E$, $D \sim_m E$, $E \sim_m F$, such that each row matches with one other row.

Table 4. 2-anonymity with maybe match

	Gender	Height	Job	ZIP	Condition
A	f	161–180	NULL	9020	Cancer
B	m	181–200	Mayor	9020	Hepatitis
C	f	161–180	Clerk	9020	Flu
D	m	NULL	Technician	9020	Pneumonia
E	m	181–200	NULL	9020	Malaria
F	m	181–200	Pilot	9020	Gastritis

Let us now analyze whether this table is safe. First we try an attack on missing values.

Hampering Reconstruction [22] is an attack that shows how a value that was suppressed in the anonymization process can be reconstructed. We extend it here to cover also missing values in the original table. For example, if an adversary knows that Daniel's data are in the table and Daniel is 205 cm tall, then he can associate row D with Daniel.

Hampering reconstruction requires that a value for some attribute exists in the real world, but is not recorded in the database. It shows that tables, where NULL values have the semantics of missing values, may be compromisable.

Next we show that there are also attacks possible on NULL values, which have the semantics 'not applicable', i.e. for which no value exists in the real world.

We introduce the novel NULL-*identifier attack*, which uses knowledge whether an attribute is applicable for some row. For example, let us assume that an adversary knows that Alice, a female patient, is not employed, and therefore the value in the Job attribute has to be NULL. He can thus query Table 4 with the search condition 'Gender = "f" and Job is NULL' and retrieves row A.

The NULL-identifier attack leverages on the knowledge that a certain value does not exist. Therefore, the row in the table has to have the value NULL in the corresponding attribute. Anonymization based on maybe match is thus not safe against NULL identifier attacks.

With hampering reconstruction and NULL-identifier attack we show that tables which are k-anonymous with respect to maybe match are not safe from

linking attacks. Both attacks exploit situations where there are less than k NULL values in some attribute within a k-group.

In conclusion we observe that generalizations using maybe-match are vulnerable against extended hampering reconstruction (NULLs as 'missing values') and NULL-identifier attack (NULLs as 'attribute not applicable').

3.6 Right Maybe Match

We restrict the definition of the maybe match, such that within a match-group a NULL value in some attribute has to appear at least k times, but use the wildcard character of NULL for matching other values with NULL. For an example, (a, b, c) would match with (a, NULL, NULL) and (a, b, NULL) but not vice versa. The motivation for this is to enforce the existence of k NULL values in some attribute within a match-group to avoid hampering reconstruction and NULL-identifier attacks. In our example, (a, b, c) matches with the two other tuples, (a, b, NULL) with one other tuple. The expectation was that a single tuple like (a, b, c) demanding further generalization with extended match would match with tuples containing NULL values and therefore would not require further generalization.

Definition 5 ***(right maybe match).*** *Let T be a table and Q the set of quasi identifier attributes of T. For two rows $t_1, t_2 \in T$ we define the right maybe match as*

$$t_1 \sim_r t_2 :\Longleftrightarrow \forall q \in Q : t_1[q] = t_2[q] \vee t_2[q] is \text{NULL}.$$

The right maybe match relation is reflexive and transitive, but not symmetrical and does not define an equivalence partitioning on a table. The non-symmetry is viable, as the definition of k-anonymity requires that each tuple matches with k other tuples, however, it does not require that the match relation defines equivalence classes.

The result of anonymization with right maybe match is shown in the example of Table 5. Here the following match relations are found: A $\sim_r$ E, B $\sim_r$ A, B $\sim_r$ E, C $\sim_r$ A, C $\sim_r$ E, D $\sim_r$ A, D $\sim_r$ E, E $\sim_r$ A, F $\sim_r$ A, F $\sim_r$ E.

Table 5. 2-anonymity with right maybe match

	Gender	Height	Job	ZIP	Condition
A	All	All	NULL	9020	Cancer
B	All	All	Mayor	9020	Hepatitis
C	All	All	Clerk	9020	Flu
D	All	All	Technician	9020	Pneumonia
E	All	All	NULL	9020	Malaria
F	All	All	Pilot	9020	Gastritis

Now let us analyze, whether the right maybe match admits tables which are safe. We first observe that there are queries on the quasi identifier projection of

a table which yield less than k results. For example, the search condition 'Job = "technician"' would only return row D. However, this query might lead to a wrong result for an adversary, because it is possible, that the 'true' value in the Job attribute is also "technician" for the rows A and E but just missing. Therefore, a rational adversary would use a maybe query instead ('Job = "technician" or Job is NULL').

It is possible to show that right maybe match is safe for all maybe queries.

We introduce the *singularity attack* to show that straight (not maybe) queries make sense and possibly compromise the data. A singularity attack uses knowledge that some value is unique, at least for some combinations of other attribute values. For example, ZIP code and Job may be compromised, when the job = "Mayor" and there is never more than one mayor in a town (i.e. per ZIP code). In such cases an adversary will use straight queries rather than maybe queries and can compromise tables which are k-anonymous with respect to the right maybe match.

The singularity attack shows that tables which are k-anonymous with respect to right maybe match are not safe against attacks. The singularity attack does not depend on the type of NULL values (non-applicable, missing, no-information), such that we have to dismiss anonymization based on the right-maybe match.

3.7 An Extended Generalization Algorithm

A detailed analysis of NULL values and their matching operators allows to extend generalization algorithms (see Sect. 2) with minimal effort to cover also NULL values in the input table using the k-anonymity definition with extended match. There are only two extensions necessary: (1) the generalization hierarchies of each quasi identifier is extended with an additional branch below the root that contains a NULL value in each level of the hierarchy. (2) k-anonymity is tested with extended match. With these extensions, the anonymization algorithms can accept microdata with NULL values without any preprocessing. It is easy to see that the complexity of the algorithms is not changed by this extension.

Figure 2 shows, how the generalization hierarchies of our running example were extended, in order to apply extended matching. Note that for consistency reasons, there has to be a NULL value on each level of the hierarchy.

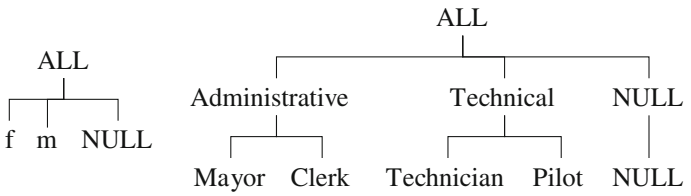

Fig. 2. Generalization hierarchies for NULL value handling in the running example

The extended generalization algorithm as well as the extended match are implemented in the tool ANON that is introduced in the next section. ANON was used to execute experiments on microdata with NULL values to analyze whether the explicit treatment of NULL values actually reduces information loss. The results are shown in Sect. 5.

4 The Anonymization System ANON

We implemented a flexible and customizable tool, called ANON [3], for computing k-anonymous and ℓ-diverse tables based on anonymization by generalization and tuple suppression [26,30] where the information loss can be defined by the users explicitly through penalties in the generalization hierarchies and through priorities for attribute generalizations [27,28]. The contribution of ANON is on one hand the computation of a k-anonymous and ℓ-diverse generalization of a given table with minimal information loss, where this information loss can be defined application specific in a fine grained way when defining the generalization hierarchies [3] and on the other hand the explicit treatment of NULL values. We implemented both anonymization with basic match and with extended match such that ANON offers two ways of handling NULL values: removing rows with NULL values before anonymization (basic match), or treating NULL values as any other value (extended match). We did not implement anonymization with maybe match or right maybe match due to their vulnerability as shown in Sect. 3.

4.1 User-Specific Requirements and Information Loss

ANON aims at adapting to the application specific data requirements by allowing the user to customize the anonymization procedure by defining application specific information loss calculation. The motivation for this is the observation that the requirements for the precision of attribute values vary enormously between different applications. For an example, in one application the age is needed in fine granularity while in a different application the body mass index is needed in detail and age is sufficient in 10-year intervals. So we argue that technical information loss definitions like those based on Shannon's definition of entropy cannot reflect the usability of a data set for a specific application. For steering the search for an optimal solution the users may specify priorities for the generalization of quasi identifiers, to specify an information loss for each generalization step and to set generalization limits. The calculation of information loss is used in a best-first search to determine the best anonymized table with generalization and tuple suppression.

Weighted information loss is calculated with the formula

$$\mathrm{WIL} = \sum_{i=1}^{n} prios[i] * \varphi_{\alpha_i}^{levels[i]},$$

where n is the number of quasi identifiers, $\varphi_{\alpha_i}^{levels[i]}$ the loss of information if the attribute α_i is generalized to the level $levels[i]$, and $prios[i]$ is the priority of the attribute α_i.

With this formula, the user can influence the information loss with the following user-specified information:

Attribute priorities. A User can assign a particular priority to each quasi identifier attribute. These priorities influence the computation of generalization loss. The intention is that attributes with a high priority are generalized to lower levels than attributes with lower priorities.

Generalization limits. If the values are useful only up to a particular generalization level, the user can limit the generalization of an attribute to this level.

Generalization penalties. Generalization penalties define the information loss for each generalization level. The top level of a generalization hierarchy has an information loss of 100 %.

This user-specified information and its impact in the search algorithm guarantee that the user will obtain such results that suit the user's requirements in the best possible way.

4.2 Architecture and Implementation

ANON is implemented in Java and is available in two distributions: as a Web Service and as an executable platform-independent java archive (JAR) with a simple graphical user interface. The anonymization is controlled by the ANON definition file, which is a combination of an anonymization settings file and a metadata file. As shown in Fig. 3, the ANON definition file is the main ANON input that determines the microdata source(s) and the outputs, as well as the anonymization process itself.

ANON definition is an XML file that consists of the following five fundamental sections:

Parameters define the anonymization settings (the value of k, maximal suppression threshold max_supp, type of the search algorithm, ANON report settings and missing value handling details).

Datasource definition defines the source(s) of data that should be anonymized (database(s), XML or CSV-file(s)).

Output definition defines the target where the anonymized data should be saved (database, XML or CSV-file).

Attributes definition defines which attributes should be read from the source data and how these should be handled. Each quasi identifier attribute (k-attribute) should have assigned its priority and maximal generalization limit. Each sensitive attribute (l-attribute) must be configured with the ℓ-diversity type that should be used for checking, and its desired parameter values (ℓ).

Generalization hierarchies define value generalization hierarchies that are used for anonymization. For every quasi identifier attribute (k-attribute) there should be one generalization hierarchy, upon which the values are

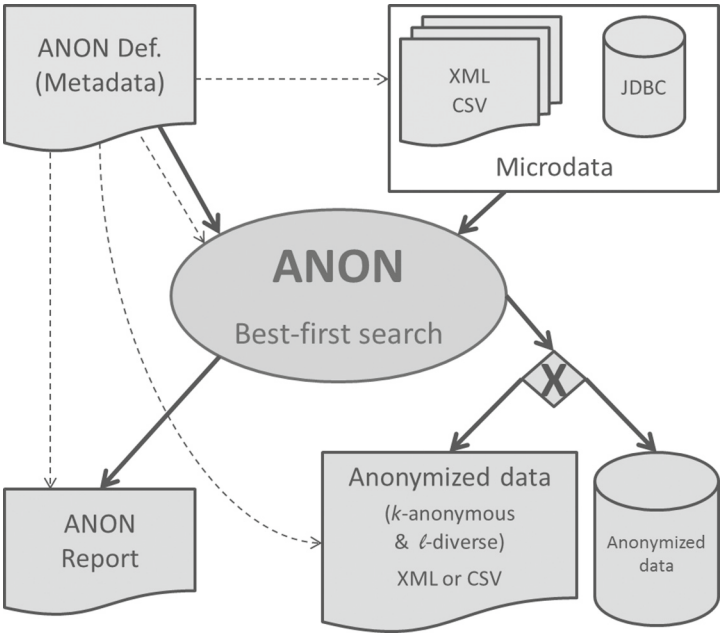

Fig. 3. ANON architecture

generalized. Each generalization hierarchy contains information about the hierarchy levels inclusive their information loss and a value generalization tree. This tree must contain all the values that the corresponding quasi identifier can hold in the microdata.

ANON is capable of anonymizing data from multiple sources that have one of the following formats: database connection (JDBC), XML-file or CSV-file. The result can be saved as one of these formats as well. Besides the anonymization outcome, user can decide to receive an ANON report, which informs about the anonymization process and eventual failures.

ANON offers the user the possibility to select the attributes that the user wants to handle in the anonymization process and mark them with one of the following anonymization types:

- k-attribute,
- l-attribute,
- dontcare,
- ignore.

The attributes that should be skipped from the result should be marked with "ignore". Alternatively they can be left out of the ANON definition file to raise the same effect. If an attribute does not play any role for the individuals privacy and should appear unchanged in the result, then it should be

marked as "dontcare". The remaining two types are those that are relevant for the anonymization process.

Attributes marked with "k-attribute" are quasi identifiers that must be transformed to a particular generalization level, such that k-anonymity for the whole table is satisfied. For this kind of attributes, the user should also specify the generalization limit and attribute priority. For l-attributes (sensitive attributes), the ℓ-diversity type and its parameter(s) should be defined.

ANON is designed to provide anonymized tables with multi-attribute ℓ-diversity. The ℓ-diversity can be defined on the attribute level. Furthermore, ANON allows to assign different ℓ-diversity types to different attributes.

4.3 Anonymization Algorithm

The anonymization Algorithm 1 features both basic match and extended match for computing partitions defined by the quasi identifiers. The variables used in ANON's Partitioning Algorithm 1 and their meaning are listed in Table 6. After a table has been partitioned, ANON checks whether each partition has at least k tuples. If this is not the case, ANON uses generalization and suppression to generate a more coarse grain table.

ANON's anonymization algorithm uses best-first search algorithm to find the optimal solution and weighted information loss described in Sect. 4.1 to evaluate the cost of a potential solution (generalized table).

The algorithm consists of 2 main parts: table search (function ANONYMIZE-TABLE - see Algorithm 2) and privacy test (function PRIVACY-TEST - see Algorithm 3). Required input parameters of both functions of Algorithms 2 and 3 are listed and described in Table 7. Furthermore, other variables and values are listed and described in Table 8, instance variables of a node in Table 9 and functions in Table 10.

ANON is customizable, so the implementation offers an abstract class of search algorithms that can easily be extended by new search algorithms. Similar interfaces are provided for information loss calculation, as well as for ℓ-diversity check. Details about the implementation are given in the next section.

Table 6. Variables used in ANON's Partitioning Algorithm 1

Variable	Description
null_handling	Denotes if NULL values are handled or not. If *null_handling* is TRUE then the partitioning algorithm will use the extended match, otherwise the basic match
partition	A partition is a set of tuples that match each other
partitionset	Set of partitions
table	A table with microdata
tuple	A table row
matched	Denotes if a tuple matches a partition

Algorithm 1. ANON's Partitioning Algorithm

Input: *table*, *null_handling*
Output: set of partitions *partitionset*, where each partition contains tuples with identical values of quasi identifiers or NULLs instead of a quasi identifier value. Partitions are disjoint.

```
 1: function PARTITION-TABLE(table, null_handling)
 2:     partitionset ← { }
 3:     for each tuple in table do
 4:         matched ← FALSE
 5:         for each partition in partitionset do
 6:             if tuple matches partition then
 7:                 add tuple into partition
 8:                 matched ← TRUE
 9:             end if
10:         end for
11:         if matched = FALSE then
12:             if tuple does not contain NULL or null_handling = TRUE then
13:                 partition ← {tuple}
14:                 add partition into partitionset
15:             end if
16:         end if
17:     end for
18:     return partitionset
19: end function
```

Table 7. Input parameters required by the function ANONYMIZE-TABLE

Parameter	Description
table	Original table that has to be anonymized
limits	Array with generalization level limits for all quasi identifiers
prios	Array with priorities for all quasi identifiers
k_param	k - the minimal partition size
l_params	Array with minimal required diversities for all sensitive attributes
max_supp	Number of tuples that are allowed to be suppressed
null_handling	Denotes if NULL values are handled or not. If *null_handling* is TRUE then the partitioning algorithm will use the extended match, otherwise the basic match

4.4 Performance Analysis

ANON's anonymization algorithm (a best-first search instantiation) is optimal and complete. If generalization limits are set to less than the number of generalization levels, it is possible that the algorithm will not find a solution (because there is none). If no limits are set, it always finds a solution, which is in worst case a completely generalized table.

Table 8. Variables and values used in ANON's Priority-Based Algorithms 2 and 3

Variable/value	Description
open	List of potential solutions (generalized tables)
visited	List of potential solutions that have already been added to *open*
best	Generalized table from *open* with the lowest information loss
levels	Generalization levels of a potential solution
child	Child node - a potential solution with a table generalized to the next higher level at one quasi identifier, while the other attributes' levels remain the same as the levels of the parent
supp_tuples	Number of tuples from all partitions that violate k-anonymity and/or ℓ-diversity
partition	Table partition - a set of records with the same quasi identifier values
diversities	Array with diversities of one partition for all sensitive attributes
NIL	Represents a NULL value
FAILURE	Denotes that an anonymized table, which satisfies all constraints, could not be found

Table 9. Instance variables of a node used in ANON's Priority-Based Algorithms 2 and 3

Variable	Description
node.PARENT	Parent node - potential solution from which *node* was deduced
node.LEVELS	Generalization levels of *node*.TABLE - represents action
node.WIL	Weighted information loss - represents total path cost
node.TABLE	Table with values generalized to *node*.LEVELS - represents state

As all optimizing generalization algorithms ANON has in the worst case exponential time complexity caused by the state space, which grows exponentially with the number of quasi identifier attributes and their limits. However, for a given set of quasi identifiers it scales nicely for increasing sizes of the data set. The set of experiments presented in this section were intended to analyze whether the anonymization with customizable calculation of information loss is feasible for large real-world datasets.

We did not compare ANON with other algorithms. The intention of the experiments was not to show that ANON reduces information loss in general, as other algorithms do not feature an application specific calculation of information loss so any such comparison would be pointless. We also do not claim that ANON is the fastest anonymization algorithm and the system could be accelerated e.g. by applying other heuristic search procedures.

The microdata used for the experiments was the *Adult Data Set* from UCI Machine Learning Repository [11] commonly used for performance experiments

Table 10. Functions used in ANON's Priority-Based Algorithms 2 and 3

Function	Description
ANONYMIZE-TABLE	Input: *table*, *limits*, *prios*, *k_param*, *max_supp* Output: Anonymized table, which satisfies all input constraints, or FAILURE if no solution could be found
PARTITION-TABLE	Input: *table*, *null_handling* Output: set of partitions *partitionset*, where each partition contains tuples with identical values of quasi identifiers or NULL instead of a quasi identifier value.
PRIVACY-TEST	Input: *node*, *k_param*, *max_supp* Output: TRUE if *node*.TABLE satisfies k-anonymity and ℓ-diversity, else FALSE
MAKE-NODE	Input: *table* Output: Node that represents *table* in the search space with LEVELS initialized with 1.
GENERALIZE	Input: *table*, *node*.LEVELS Output: Table with values generalized to node.LEVELS.
CHILD-NODE	Input: *parent*, *levels*, *prios* Output: Node with PARENT = *parent*, LEVELS = *levels*.
CALCULATE-WIL	Input: *levels*, *prios* Output: Weighted information loss of a generalized table, calculated as follows: $\sum_{i=0}^{length(levels)-1} prios[i] * \varphi_{\alpha_i}^{levels[i]} \; \varphi_{\alpha_i}^{levels[i]}$ is the loss of information if the attribute α_i gets generalized to the level $levels[i]$.
COUNT-TUPLES	Input: *partition* Output: Number of tuples in the *partition*.
length	Input: *array* Output: Length of the *array*.

in the microdata privacy literature. The Adult Data Set contains real data collected by the U.S. census bureau in the year 1994. This data is split in a training set and a test set. For our experiments we merged both sets together and tuples with unknown values were removed. After data cleaning, the set contained 45,222 tuples. To provide comparable results, the same data preparation was undertaken as described in [17,18]. From the 15 attributes in the data set, the identical nine were chosen as in [18]. As shown in Table 11, the attributes *age*, *gender*, *race*, *marital status*, *education*, *native country* and *workclass* were used as quasi identifiers and the attributes *salary class* and *occupation* were used as sensitive attributes. Generalization hierarchies for the used quasi identifiers were constructed in a semantically logical way.

There were two experiment runs, each with almost 800 anonymizations: one with the use of priorities and information loss and one without them, to imitate the optimal search algorithms without a cost function (e.g. MinGen).

Algorithm 2. ANON's Priority-based Algorithm - Part 1 (Search Algorithm)

Input: *table*, *limits*, *prios*, *k_param*, *l_params*, *max_supp*, *null_handling*
Output: anonymized table satisfying k-anonymity and ℓ-diversity or FAILURE if no solution could be found

```
 1: function ANONYMIZE-TABLE(table, limits, prios, k_param, l_params, max_supp)
 2:     open ← {MAKE-NODE(table)}
 3:     visited ← { }
 4:     while open is not empty do
 5:         best ← node n in open with the lowest n.WIL value
 6:         if best.TABLE = NIL then
 7:             best.TABLE ← GENERALIZE(table, best.LEVELS)
 8:         end if
 9:         if PRIVACY-TEST(best, k_param, l_params, max_supp, null_handling)
    then
10:             return best.TABLE
11:         else                                              ▷ expand best
12:             for i ← 0 to length(limits) − 1 do
13:                 levels ← best.LEVELS
14:                 if limits[i] > levels[i] then
15:                     levels[i] ← levels[i] + 1
16:                     child ← CHILD-NODE(best, levels, prios)
17:                     if child not in visited then
18:                         add child into open
19:                         add child into visited
20:                     end if
21:                 end if
22:             end for
23:             best.TABLE ← NIL
24:             remove best from open
25:         end if
26:     end while
27:     return FAILURE
28: end function

29: function CHILD-NODE(parent, levels, prios)
30:     return a node with
31:             PARENT ← parent,
32:             LEVELS ← levels,
33:             WIL ← CALCULATE-WIL(levels, prios)  ▷ Weighted Information Loss
34:             TABLE ← NIL
35: end function
```

Information loss values are listed within generalization hierarchies that come with ANON. The priority order of attributes (starting with a low priority) in the first run was {*age*, *native country*, *education*, *marital status*, *workclass*, *race*, *sex*}. In the second run (without the cost function), an implicit priority order was derived from the attributes order in the ANON definition file, which was

Algorithm 3. ANON's Priority-based Algorithm - Part 2 (Privacy Test)

```
36: function PRIVACY-TEST(node, k_param, l_params, max_supp, null_handling)
37:     supp_tuples ← 0
38:     for each partition in PARTITION-TABLE(node.TABLE, null_handling) do
39:         if COUNT-TUPLES(partition) ≥ k_param then        ▷ k-anonymity satisfied
40:             diversities ← CALCULATE-DIVERSITIES(partition)
41:             for i ← 0 to length(l_params) − 1 do
42:                 if diversities[i] < l_params[i] then      ▷ ℓ-diversity not satisfied
43:                     remove partition from node.TABLE
44:                     supp_tuples ← supp_tuples + COUNT-TUPLES(partition)
45:                     break
46:                 end if
47:             end for
48:         else                                              ▷ k-anonymity not satisfied
49:             remove partition from node.TABLE
50:             supp_tuples ← supp_tuples + COUNT-TUPLES(partition)
51:         end if
52:         if supp_tuples > max_supp then                    ▷ privacy not satisfied
53:             return FALSE
54:         end if
55:     end for
56:     return TRUE                          ▷ privacy satisfied (supp_tuples ≤ max_supp)
57: end function
```

{*age*, *sex*, *race*, *marital status*, *education*, *native country*, *workclass*}. Generalization limits were not set (they equaled to the no. of generalization levels) to avoid an anonymization without a solution.

The experiments were performed to estimate the "real case" complexity and the impact of different parameters on the number of visited nodes, resulting information loss and average partition size. These parameters are listed in Table 12.

Table 11. Adult Data Set description (adapted from [18])

	Attribute	Domain size	Generalization type	No. of gen. levels
1	Age	74	Ranges (5, 10, 20, 100)	4
2	Gender	2	Taxonomy tree	1
3	Race	5	Taxonomy tree	1
4	Marital Status	7	Taxonomy tree	2
5	Education	16	Taxonomy tree	3
6	Native Country	41	Taxonomy tree	2
7	Work Class	7	Taxonomy tree	2
8	Salary class	2	Sensitive att	
9	Occupation	14	Sensitive att	

Table 12. Experiments' parameters and their values

Parameter	Chart notation	Values
n	QID	1, 2, 3, 4, 5, 6, 7
k	k	2, 3, 5, 10, 14, 20, 100, 200, 1000
ℓ_{α_8}	l1	1, 2
ℓ_{α_9}	l2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
max_supp	supp	0 %, 1 %, 10 %
cost function (WIL)	With priorities	Used (first run),
	W/o priorities	not used (second run)

There were over 1,500 anonymizations produced, where the parameters were set to almost all possible value combinations.

The first two experiments deal with the time complexity. In these experiments, the impact of the quasi identifiers' increase on the number of visited nodes and anonymization time was analyzed. Both experiments were executed with four different anonymization settings groups: (1) $k = 2$ with 0 % tuple suppression (black line with white markers), (2) $k = 2$ with 1 % suppression limit (black line with black markers), (3) $k = 10$ with 0 % suppression and $\ell_{\alpha_8} = 2$, $\ell_{\alpha_9} = 10$ (gray line with white markers) and (4) $k = 10$ with 1 % suppression and $\ell_{\alpha_8} = 2$, $\ell_{\alpha_9} = 10$ (gray line with black markers). The dotted line denotes the maximal possible number of visited nodes ($\Pi_{i=1}^{n} limits[i]$) and the approximated maximal required anonymization time, respectively. As approximation, the anonymization settings with $k = 14$, 0 % suppression and $\ell_{\alpha_8} = 2$, $\ell_{\alpha_9} = 14$ were used. These settings were noticed to result in maximal possible values, because only the last queued node with completely generalized table satisfies these settings.

Time complexity of k-anonymity algorithms similar to ANON grows with the number of quasi identifiers [5,12,17]. In contrast to that, the number of tuples in a table does not have a big impact on time complexity. Table size is just a constant factor multiplied by the number of nodes, which does not affect the number of visited nodes itself and can therefore be neglected. Figure 4 confirms for ANON that time complexity grows with the increase in quasi identifiers.

Figure 5 represents the same experiment, where, instead of number of visited nodes, the time was measured. If we compare both figures, it is easy to see that time depends on the number of nodes and some other factors like generalization hierarchy height. If we observe the ℓ-diversity lines (gray lines) in Figs. 4 and 5, we can notice that these lines have a slightly higher slope in the chart with time on the y-axis than in the chart with nodes on the y-axis. The explanation for this phenomenon is hidden in diversity calculation effort. ANON does not need to calculate the size of a partition (relevant for k-anonymity checking) explicitly, because it is managed together with a partition. The diversity (relevant for ℓ-diversity checking), in opposition to partition size, has to be calculated extra for each partition, if the partition is k-anonymous.

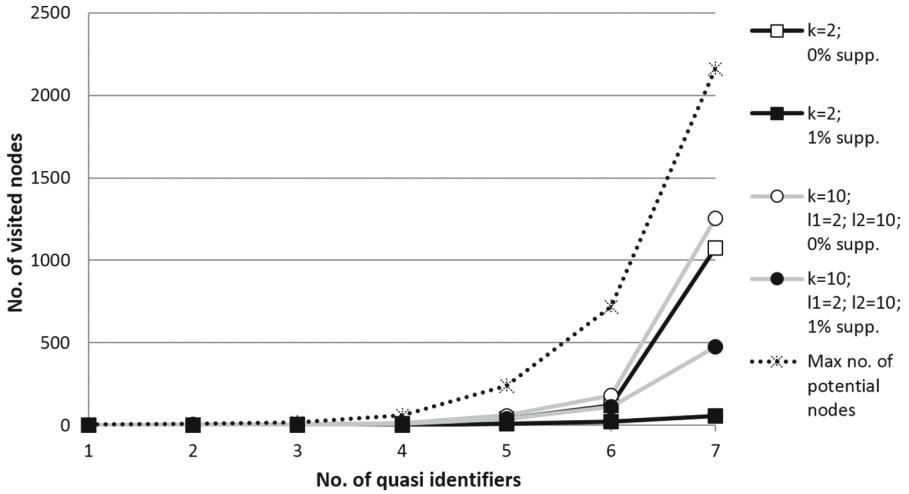

Fig. 4. Search algorithm complexity (number of nodes)

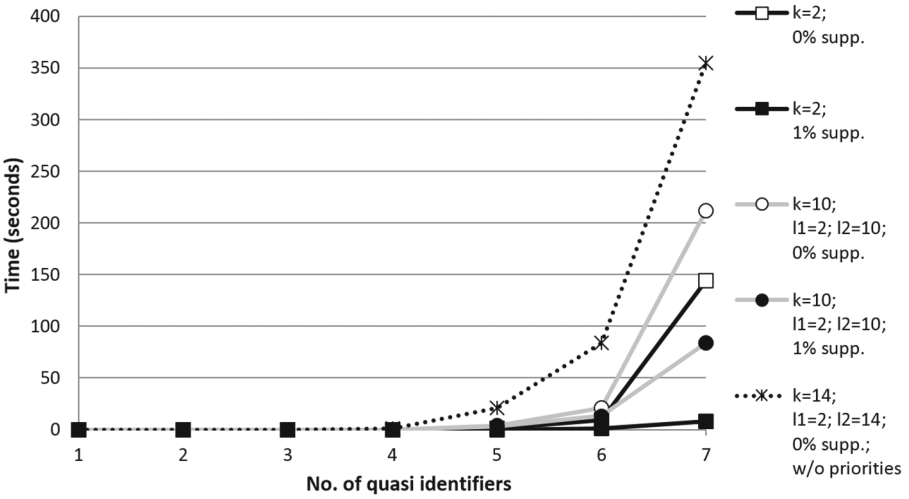

Fig. 5. Search algorithm complexity (time)

The charts Figs. 4 and 5 show the highly significant impact of tuple suppression. Both black lines have the same settings, except the maximal suppression limit (white markers 0 %, black markers 1 %). However, the difference in the results is huge. It took 1,075 nodes to anonymize 7 quasi identifiers to a 2-anonymous table with no suppression (black line with white markers) and with just 1 % suppression (max. 453 tuples may be eliminated), it took only 60 nodes to find the optimal solution. More experiment results and analysis of different parameters' impact can be found in [3].

5 NULL value experiments

In this section we report experiments which analyze the differences in information loss between the different treatments of NULL values: basic and extended match. In Tables 1, 2, 3, 4 and 5 we showed in a tiny example the differences resulting from applying different match definitions. Here we analyze the differences between anonymization with basic match and with extended match in more detail and more exhaustive. Since in case of basic match whole rows have to be removed (suppressed) from the table if they contain a NULL value, we expect an improvement in the information content of the results through the application of extended match. Our hypothesis, therefore, was that anonymization with extended match has less information loss than anonymization with basic match.

To test the hypothesis we conducted a series of 3168 experiments using the anonymization tool ANON applied to datasets derived from the Adult Database from the UCI Machine Learning Repository, with varying parameter settings and varying ratios of NULL values. We included 8 quasi identifiers in the following order: *age*, *sex*, *race*, *marital_status*, *workclass*, *education*, *native_country*, *occupation*. According to this order, tables with $n < 8$ quasi identifiers contain the first n quasi identifiers. Generalized values were provided with help of taxonomy trees for all quasi identifiers except the *age*, where we used ranges with steps {5, 10, 20, 100}.

The Adult Database itself contains several NULL values in the attributes *workclass*, *occupation* and *native_country*. To assure that every table has the right target amount of randomly placed NULL values used in the experiments (and not more than that), we first eliminated the rows with NULL values from the original table to obtain a common base for all test tables. From this base table with 45222 records we created 88 test tables with NULL values as a result of combination of the number of quasi identifiers (1 to 8) and the percentage of randomly inserted NULL values (0.1 %, 0.5 %, 1 %, 2.5 %, 5 %, 7.5 %, 10 %, 15 %, 20 %, 25 %, 30 %). We used random number generation in Java for determination of cells in the table where NULL values were inserted.

For the anonymization runs of the 88 tables we used following parameters:

- k-parameter: 2, 3, 4, 5, 10, 15, 20, 50, 100
- max. allowed suppression: 0 %, 1 %
- matching: basic match, extended match.

The max. allowed suppression specifies the quota for suppressing rows (to avoid adverse effects of outliers). For basic match this is in addition to the rows with NULL values, which are removed in a preprocessing step.

Information loss in the anonymization algorithm is caused by row suppression (fraction of rows being suppressed) and by generalization (generalization level of the attributes). For the following experiments the information loss was calculated with the following formula:

$$IL = \frac{s}{n} + \sum_{i=1}^{m} \frac{gl_{\alpha_i}}{ht_{\alpha_i}} \times \frac{1}{m} \times \frac{n-s}{n}$$

n Number of rows
s Number of removed rows
m Number of quasi identifier attributes
gl_{α_i} Generalization level of the attribute α_i in the generalized table
ht_{α_i} Height of the generalization hierarchy of the attribute α_i (Number of total generalization levels of α_i).

Information loss due to row suppression is represented in the first summand as the fraction of removed tuples. The information loss caused by generalization is calculated as the weighted sum of information losses of all attributes multiplied with the fraction of non-removed tuples. The information loss of an attribute is defined as fraction of the generalization level of this attribute by the number of levels, where the most general level (which carries no information at all)is h and the level in the original table is 0. So if an attribute is generalized to the level 3 of a 5 level hierarchy we define the information loss a 3/5. Note that the values of an attribute in all tuples are generalized to the same level as we apply the global recording strategy. Therefore we can calculate the information loss at the attribute level.

We show first representative comparisons of the information loss between basic match and extended match without and with row suppression. In the figures results of extended match is shown in light gray bar and those of basic match in black bars. Each bar represents the information loss of one anonymization run.

Figures 6 and 7 show anonymizations without row suppression (max. supp. = 0 %). Anonymizations with extended match (light gray bars) tend to have constant information loss, whereas the information loss of anonymizations with basic match (black bars) has a growing trend with increasing percentage of NULL values. This behavior can be explained with 2 influence factors: (1) number of

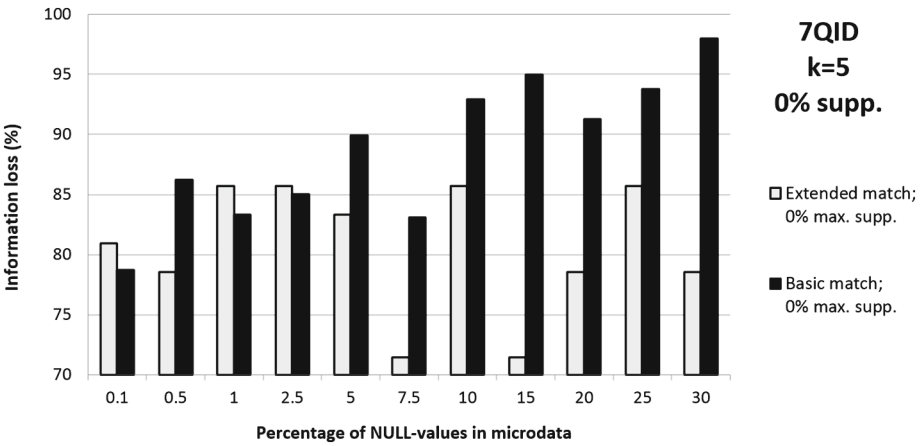

Fig. 6. Impact of the percentage of NULL values on the information loss (7 quasi identifiers, k-parameter = 5, 0 % max. row suppression).

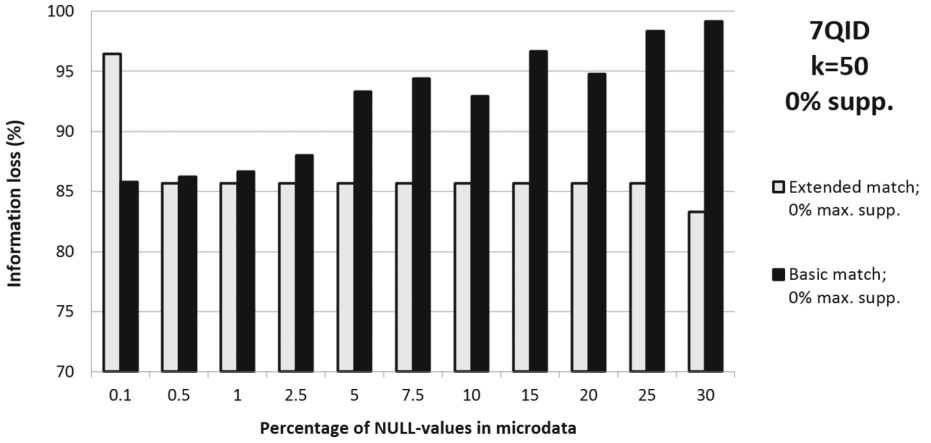

Fig. 7. Impact of the percentage of NULL values on the information loss (7 quasi identifiers, k-parameter = 50, 0 % max. row suppression).

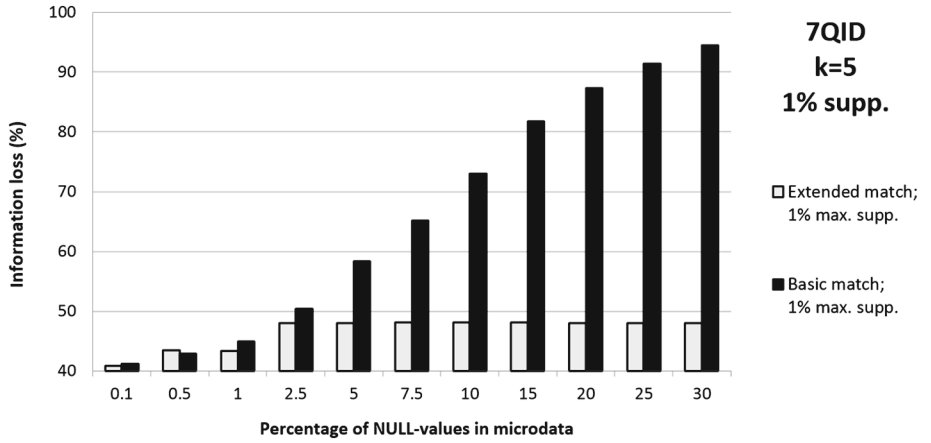

Fig. 8. Impact of the percentage of NULL values on the information loss (7 quasi identifiers, k-parameter = 5, 1 % max. tuple suppression).

removed rows (in case of basic match) and (2) outlier rows and the corresponding high generalization. If the percentage of NULL values is low, the rows with NULL values are outlier rows, causing information loss to grow if extended match is used. If basic match is used instead, those "NULL-outliers" are simply removed and thus do not cause massive generalizations. On the other end, where the ratio of NULL values is high, rows with NULL values are not outliers anymore. Therefore, they do not increase the information loss if extended match is used. For basic match, however, information loss is proportional to the ratio of rows with

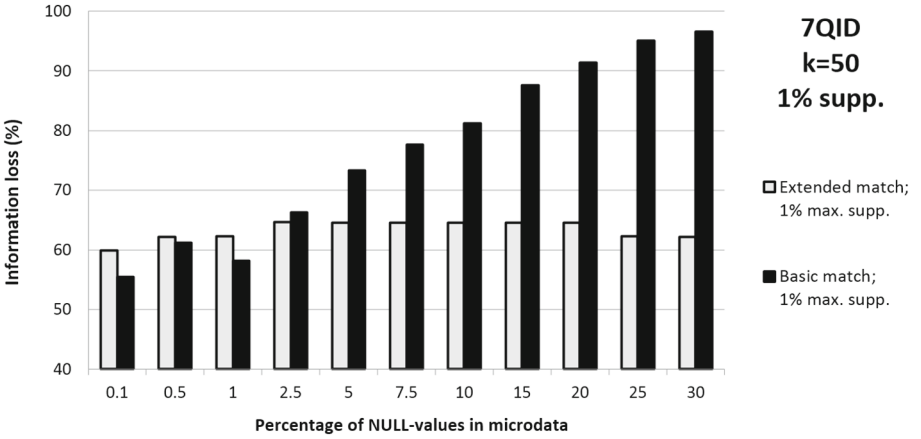

Fig. 9. Impact of the percentage of NULL values on the information loss (7 quasi identifiers, k-parameter = 50, 1 % max. tuple suppression).

NULL values leading to an increase in information loss with increasing ratios of NULL values.

Figures 8 and 9 show anonymizations with the same setup as those in Figs. 6 and 7, but with row suppression of up to 1 %. Here extended match is no longer so sensible on NULL outliers and results in an almost constant information loss over increasing ratio of NULL values (light gray bars), while information loss grows drastically for basic match (black bars). That for low ratios of NULL values (below 1 %) basic match is slightly better than extended match might be due that for basic match more rows are removed (number of rows with NULL plus 1 % of the rows without NULL).

Figure 10 shows the aggregated results of all 3168 anonymizations in our experiment. Each bar represents the average difference in information loss of anonymizations with basic match and anonymizations with extended match, calculated over all 8 quasi identifiers and all 9 different k-parameters. The light gray bars represent the setups without row suppression (max. supp. = 0 %) and the dark gray bars the setups with 1 % max. suppression.

To summarize the results: The experiments showed that the best method in general was extended match with 1 % row suppression. For very low ratios of NULL values basic match was slightly better. Extended match without row suppression performs worse for low ratios of NULL values, because it suffers from the generalizations caused by NULL outliers. Basic match was only favorable for very low numbers of NULL values and the quality of the results deteriorates with increasing ratios of NULL values, caused by the removal of all rows with NULL values. Furthermore, the information loss for extended match with row suppression did not seem to be influenced by the number of NULL values in the data set, as shown by the almost constant information loss over varying ratios of NULL values.

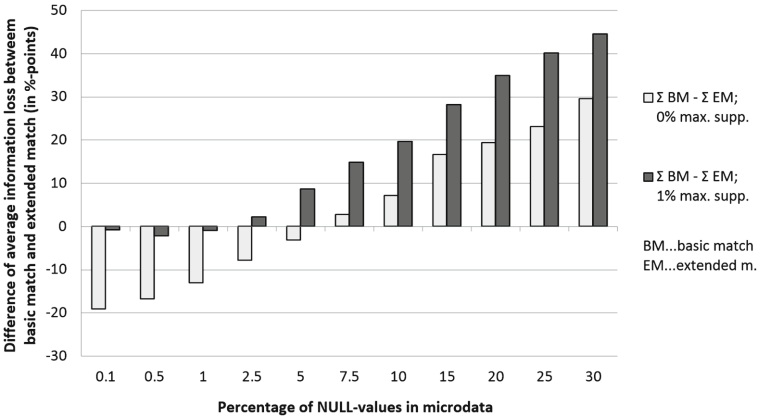

Fig. 10. Average advantage in %-points of information loss of the extended match over the basic match, depending on the percentage of NULL values in a table. In the positive y-area the extended match outperforms the basic match.

6 Conclusions

NULL values (missing values, not applicable attributes) appear frequently in microdata. Surprisingly, current anonymization algorithms require that all rows containing NULL values are removed from a table before it can be anonymized. We analyzed the effects of including NULL values in the definition of k-anonymity in detail and showed that the extended match where NULL values match (only) with other NULL values is a correct approach for extending k-anonymity to cover missing values. We introduced two new attacks that show that a further relaxation of the match operator which interprets NULL values as wildcards in the sense of Codd's maybe select leads to tables which can be attacked successfully. The extension of k-anonymity to tables with NULL values reduces the information loss induced by the removal of rows with NULL values by current anonymization algorithms and avoids the introduction of biases. Experiments showed that extended match reduces information loss for a generalization algorithm with row suppression considerably. The definition of k-anonymity we propose here can be used easily as basis for extending other anonymization algorithms to also cover tables with NULL values in an adequate and save way.

References

1. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Approximation algorithms for k-anonymity. In: Proceedings of the International Conference on Database Theory, ICDT 2005 (2005)
2. Bayardo, R.J., Agrawal, R.: Data privacy through optimal k-anonymization. In: Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, pp. 217–228 (2005)

3. Ciglic, M., Eder, J., Koncilia, C.: ANON - a flexible tool for achieving optimal k-anonymous and ℓ-diverse tables. Technical report, University of Klagenfurt (2014). http://isys.uni-klu.ac.at/PDF/2014-ANON-Techreport.pdf
4. Ciglic, M., Eder, J., Koncilia, C.: k-anonymity of microdata with NULL values. In: Decker, H., Lhotská, L., Link, S., Spies, M., Wagner, R.R. (eds.) DEXA 2014, Part I. LNCS, vol. 8644, pp. 328–342. Springer, Heidelberg (2014)
5. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Samarati, P.: k-anonymity. In: Yu, T., Jajodia, S. (eds.) SDMDS 2007. AISC, vol. 33, pp. 323–353. Springer, New York (2007)
6. Codd, E.F.: Extending the database relational model to capture more meaning. ACM Trans. Database Syst. **4**(4), 397–434 (1979)
7. Cox, L.H.: Suppression methodology and statistical disclosure control. J. Am. Stat. Assoc. **75**(370), 377–385 (1980)
8. Eder, J., Dabringer, C., Schicho, M., Stark, K.: Information systems for federated biobanks. In: Hameurlain, A., Küng, J., Wagner, R. (eds.) Transactions on Large-Scale Data- and Knowledge-Centered Systems I. LNCS, vol. 5740, pp. 156–190. Springer, Heidelberg (2009)
9. Eder, J., Gottweis, H., Zatloukal, K.: IT solutions for privacy protection in biobanking. Public Health Genomics **15**(5), 254–262 (2012)
10. Eder, J., Stark, K., Asslaber, M., Abuja, P.M., Gottweis, H., Trauner, M., Mischinger, H.J., Schippinger, W., Berghold, A., Denk, H., Zatloukal, K.: The genome austria tissue bank. Pathobiology **74**(4), 251–8 (2007)
11. Frank, A., Asuncion, A.: UCI machine learning repository (2010). http://archive.ics.uci.edu/ml
12. Fung, B.C.M., Wang, K., Fu, A.W.-C., Yu, P.S.: Introduction to Privacy-Preserving Data Publishing: Concepts and Techniques, 1st edn. Chapman & Hall/CRC, Boca Raton (2010)
13. Gaskell, G., Gottweis, H., Starkbaum, J., Gerber, M.M., Broerse, J., Gottweis, U., Hobbs, A., Ilpo, H., Paschou, M., Snell, K., Soulier, A.: Publics and biobanks: Pan-European diversity and the challenge of responsible innovation. Eur. J. Hum. Genet. **21**(1), 14–20 (2013)
14. ISO: ISO/IEC 9075–2:2011 Information technology – Database languages – SQL – Part 2: Foundation (SQL/Foundation), December 2011
15. Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002, pp. 279–288 (2002)
16. Kifer, D., Gehrk, J.: Injecting utility into anonymized datasets. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of data, SIGMOD 2006, pp. 217–228 (2006)
17. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: efficient full-domain k-anonymity. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of data, SIGMOD 2005, pp. 49–60 (2005)
18. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: L-diversity: privacy beyond k-anonymity. ACM Trans. Knowl. Disc. Data (TKDD) **1**(1), 3 (2007)
19. Matthews, G.J., Harel, O.: Data confidentiality: a review of methods for statistical disclosure limitation and methods for assessing privacy. Stat. Surv. **5**, 1–29 (2011)
20. Meyden, R.: Logical approaches to incomplete information: a survey. In: Chomicki, J., Saake, G. (eds.) Logics for Databases and Information Systems. The Springer International Series in Engineering and Computer Science, vol. 436, pp. 307–357. Springer, New York (1998). Chapter 10

21. Meyerson, A., Williams, R:. On the complexity of optimal k-anonymity. In: Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2004, pp. 223–228 (2004)
22. Ohrn, A., Ohno-Machado, L.: Using boolean reasoning to anonymize databases. Artif. Intell. Med. **15**(3), 235–254 (1999)
23. Park, H., Shim, K.: Approximate algorithms for k-anonymity. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of data, SIGMOD 2007, pp. 67–78 (2007)
24. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. (TKDE) **13**(6), 1010–1027 (2001)
25. Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information (abstract). In: Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 1998, p. 188 (1998)
26. Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report (1998)
27. Stark, K., Eder, J., Zatloukal, K.: Priority-based k-anonymity accomplished by weighted generalisation structures. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 394–404. Springer, Heidelberg (2006)
28. Stark, K., Eder, J., Zatloukal, K.: Achieving k-anonymity in datamarts used for gene expressions exploitation. J. Integr. Bioinform. **4**(1), 57 (2007)
29. Sun, X., Wang, H., Li, J., Truta, T.M.: Enhanced p-sensitive k-anonymity models for privacy preserving data publishing. Trans. Data Priv. **1**(2), 53–66 (2008)
30. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. Int. J. Uncertainty Fuzziness Knowl.-Based Syst. **10**(5), 571–588 (2002)
31. Terrovitis, M., Mamoulis, N., Kalnis, P.: Local and global recoding methods for anonymizing set-valued data. VLDB J. **20**(1), 83–106 (2011)
32. Tian, H., Zhang, W.: Extending l-diversity to generalize sensitive data. Data Knowl. Eng. **70**(1), 101–126 (2011)
33. Wichmann, H.-E.E., Kuhn, K.A., Waldenberger, M., Schmelcher, D., Schuffenhauer, S., Meitinger, T., Wurst, S.H., Lamla, G., Fortier, I., Burton, P.R., Peltonen, L., Perola, M., Metspalu, A., Riegman, P., Landegren, U., Taussig, M.J., Litton, J.-E.E., Fransson, M.N., Eder, J., Cambon-Thomsen, A., Bovenberg, J., Dagher, G., van Ommen, G.-J.J., Griffith, M., Yuille, M., Zatloukal, K.: Comprehensive catalog of European biobanks. Nat. Biotechnol. **29**(9), 795–797 (2011)