Privacy-Preserving Data Warehousing

ARTICLE *in* INTERNATIONAL JOURNAL OF BUSINESS INTELLIGENCE AND DATA MINING · JULY 2015

2 AUTHORS, INCLUDING:

Benjamin Fabian
Humboldt-Universität zu Berlin
69 PUBLICATIONS **344** CITATIONS

SEE PROFILE

Privacy-Preserving Data Warehousing

Benjamin Fabian[a,*], Tom Göthling[a]

[a]*Institute of Information Systems*
Humboldt-Universität zu Berlin
Spandauer Str. 1, 10178 Berlin, Germany

Abstract

Data warehouses are an important element of business intelligence and decision support in many companies and inter-organizational data infrastructures. However, when personal information of individuals is concerned, it is critical to provide sufficient protection mechanisms in order to preserve privacy. In addition to classical access control, database anonymization is an important element of an encompassing strategy for privacy-preserving data storage. This article gives an overview on selected anonymization concepts and techniques and investigates if they are suitable for a data warehouse context. Furthermore, a process of privacy-preserving data integration and provisioning is presented and the impact of architecture, privacy criteria, and further parameter choices is discussed. Finally, we experimentally compare the impact of these parameters on data utility after anonymization in several experiments on multiple data sets and derive corresponding recommendations.

Keywords: Data Warehouse, Data Integration, Privacy, Anonymity

1. Introduction

Strong growth of the data integration market indicates that companies continuously consolidate their data into data warehouses in order to improve availability and create added value [1]. Because data warehouses usually store sensitive information such as customer or sales insights, they represent a potential target of privacy attacks. The impact of such attacks could be severe. In 2011, a privacy breach of the Sony Play Station network disclosed sensitive user data, including credit card information of 77 million users [2]; in 2013, the market data provider Bloomberg inadvertently allowed reporters to access restricted customer data [3]. Besides external adversaries and inadvertently published data, employees are also a potential threat to companies' data. Furthermore, even data that was neither stolen nor published inadvertently can reveal sensitive information. As a result of such privacy breaches, personal data privacy has increasingly caught attention of the public and leads European Data Protection authorities to encourage stricter legislations with regards to personal data processing [4]. Consequently, more and more companies realize the strategic matter of data privacy [5][6] and data protection has developed into a key issue of an enterprise IT strategy.

Data anonymization seems to be a reasonable approach to mitigate the impact of internal data thefts and to avoid inferences within published datasets. However, research proved that intuitive

*Corresponding author. *Email address:* bfabian@wiwi.hu-berlin.de. *Phone:* +49 30 20935662.
Email addresses: bfabian@wiwi.hu-berlin.de (Benjamin Fabian), tomgoethling@gmx.de (Tom Göthling)

methods of data anonymization, such as suppressing directly identifying information, are inappropriate for guaranteeing anonymity as multiple approaches exist to circumvent such mechanisms. Consequently, over the last decades multiple mechanisms have been developed to create anonymized datasets which guarantee a specific level of data privacy. However, most approaches are based on idealistic assumptions of data management and do not cover the setting of data warehouses. Thus, the application of these mechanisms in data warehouses remains an open research question.

This article will investigate how anonymization approaches can be applied to data warehouse scenarios. The goal is to identify critical process steps in data warehousing with regard to data privacy and provide architectural alternatives to manage them by adopting existing privacy concepts. Since data warehouses are mainly used to perform data analysis, the implications of architectural design decisions on data utility will be evaluated. Possibilities and limitations of the application of existing approaches in data warehouses are analyzed. Furthermore, this work will focus on data privacy mechanisms based on anonymization that can be applied in addition to access control [7] and further classical security measures [8].

The article is structured as follows. First, in section 2 work related to the general research area of security and privacy in data warehouses is discussed. Fundamental concepts of database anonymization and related work in this specific field are introduced in section 3. This includes mechanisms to sanitize data, utility metrics, possible attack models on static and dynamic data as well as privacy criteria in order to preserve data privacy. Section 4 provides insight into different data warehouse architectures. Critical process steps regarding privacy issues will be identified and exemplified with the help of two sample architectures. Afterwards, possible approaches for managing these process steps will be presented. Section 5 evaluates implications of the different design parameters on data utility based on a prototype which simulates different sample architectures. This is followed by a discussion of potential implementations of the results shown. Implications on data utility are evaluated by applying multiple utility metrics to the results of the simulation.

2. Related Work

Data warehouses are an important foundation for decision support in many commercial and other organizations and corresponding organizational and technical planning processes as well as the choice of the right architecture involve many practical challenges [9]. In particular, establishing information security issues for such integrated data stores with confidential content is a challenge in the face of sophisticated adversaries and heterogeneous attack vectors, moreover, when data is integrated and shared with several organizations such as partner firms or subcontractors [10] [11]. In such settings, classical access control mechanisms [12] need to be adapted to the multidimensional models of data warehousing [13], but they also need to evolve to cope with federated settings [14] and new technologies such as semantic data stores [15] [8], in-memory databases [16][17], and RFID-based information exchange [18].

Beyond access control, further protection mechanisms for confidentiality and privacy are needed as motivated by the examples of security breaches given in the Introduction. Here, privacy-preserving data mining [19] [20] could improve privacy of individual data subjects if analytical facilities are trustworthy. However, only data sanitization [21] can provide measures to protect privacy even in the face of potentially adversarial data warehouse providers or data consumers. One important area of research in data sanitization is anonymization, on which detailed literature will be provided in section 3. To the best of our knowledge, challenges and trade-offs of anonymization

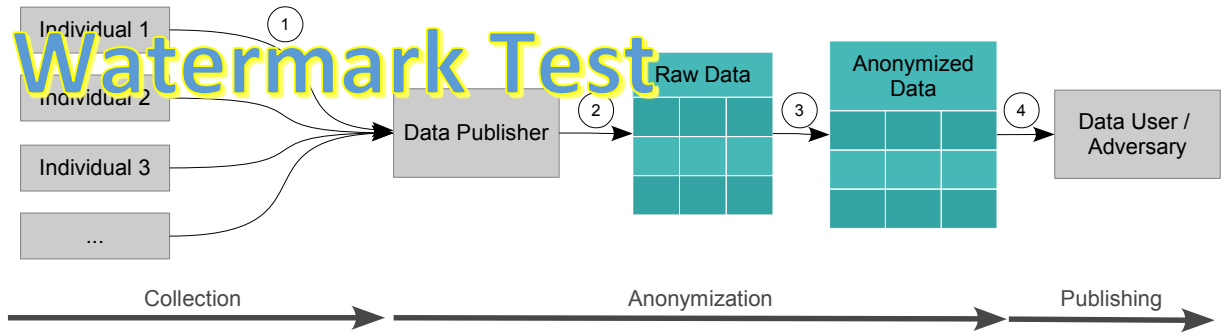

Figure 1: Process of privacy preservation in databases

in data warehousing have so far not received sufficient attention in the literature. The current article is motivated by this research gap.

3. Anonymization Process and Measures

3.1. Fundamentals and Process of Anonymization

As shown in Figure 1, a typical scenario of privacy preservation in databases consists of three phases – collection, anonymization, and publishing. The aim of the collection phase is to gather data (1) and to transform them into a usable format for data analysis (2). Usually, the collected raw data contains sensitive information. Hence, data publishers have to implement measures to protect this information against privacy disclosure. This is usually enforced by anonymization (3). In accordance with [22, 23, 24], anonymization can be defined as a process of converting data in order to hide the identity of each record owner, assuming that sensitive attribute values must be retained. In the next step, the anonymized dataset is published to data users (4). Before release, the data publisher has to decide what kind of access should be granted to data users. Principally, it can be distinguished between full access releases, where the whole dataset is available for data recipients, and query-based approaches, which only allow restricted queries [25].

In an untrusted publisher model, data publishers may attempt to extract sensitive information from the dataset [26]. As a consequence, sensitive information has to be protected before making the data available to the publisher because he represents a potential attacker. In contrast, the trusted publisher model assumes a trustworthy data publisher in a secure environment. This enables new possibilities to ensure data privacy because data publishers can operate on the raw data. In case of data integration scenarios, both models are realistic. For example, an integration project which operates within a single corporation will rather assume a trusted data publisher, whereas cross-company integration projects probably tend to use the untrusted model.

Moreover, data publishers have the choice of releasing aggregated or non-aggregated data [25]. Non-aggregated data provide information on the detail level of individuals [27] (for example, persons, products, or departments) and are also called *microdata*. Each row of a microdata table conventionally represents one individual unit of a given population. Each column contains values of an attribute of the individuals. According to [23], microdata tables contain the following types of attributes: *Identifier* attributes contain information that explicitly identify record owners, for example, social security number or name. *Quasi Identifier* attributes potentially identify record owners when combined, for example, gender, age, and ZIP code. *QI-groups* in a dataset are the

subsets of tuples that share the same values for the quasi identifiers. *Sensitive attributes* represent sensitive individual specific information that has to be protected against privacy disclosure, for example, diseases. *Insensitive attributes* contain neither identifying information nor hold other information which deserve protection, for example, metadata.

Depending on the intended use of data, data publishers either provide the entire dataset or only allow queries on the data without revealing it. In the former case, the data publisher has to anonymize data in a way that satisfies privacy goals. Unfortunately, it is not sufficient to only suppress identifying attributes because attackers may use quasi-identifier attributes to identify records representing an individual. Thus, data publishers have to apply advanced techniques that are presented in section 3.4. Query-based approaches, for example, SQL queries or statistical queries [28], require restrictions in order to ensure privacy of individuals [29]. Aggregated data are commonly provided in contingency tables, which contain count information about the frequency of attribute combinations and are used by statistical bureaus for releasing statistics on microdata [30]. In case of data integration systems, data publishers usually use microdata tables for flexible data handling and avoiding data loss caused by aggregation [31].

3.2. Mechanisms and Criteria for Anonymization

Data publishers usually have to manage two fundamental contrary goals. On the one hand, data users are interested in receiving as much detailed data as possible to achieve good utility. On the other hand, the privacy of individuals has to be preserved, which is usually related to a certain loss of information. To assess and manage this trade-off between privacy and utility, data publishers can utilize sanitization mechanisms, privacy criteria, and utility metrics. A *sanitization mechanism* reduces the precision of a dataset [25]. It is applied to a raw dataset and converts it into anonymized data. Sanitization mechanisms include suppression, generalization, and swapping of attribute values as well as more advanced techniques such as anatomization [32], the addition of noise or synthetic data, or multi-view releases.

Privacy criteria define properties that a sanitized dataset has to satisfy in order to guarantee a specific level of data privacy. A sanitized dataset which satisfies a specific privacy criterion is called a release candidate. The definition of a privacy criterion is mostly related to an attack model because an attacker's background knowledge strongly affects requirements on data sanitization. Privacy criteria can only guarantee a specific level of privacy. For example, if a privacy criterion requires that at least three tuples have equal quasi-identifier attribute values and different sensitive values, it prevents the identification of the record for a particular individual even if adversaries have knowledge about quasi-identifier values. However, such adversaries would be able to constrain the disease of an individual to three potential values. In order to quantify utility of anonymized datasets, *utility metrics* measure the loss of information caused by sanitization. If multiple release candidates satisfy a privacy criterion, utility metrics will be an appropriate instrument to select the most useful candidate.

3.3. Privacy Preservation for Dynamic Datasets

Most attack models discussed in the literature assume static datasets. For data warehouses, this assumption is not reasonable because datasets can change over time, sources can become unavailable, or additional sources deliver new information. Those changes on the dataset extend the capability of adversaries to infer sensitive information because information can be linked across different releases. The existing literature distinguishes between different application scenarios of

dynamic datasets that have different properties with respect to operations allowed and adversaries' background knowledge. *Multiple-Release Publishing* concerns scenarios where data publishers provide different sanitized projections of a given dataset. Privacy is threatened because adversaries may be able to link information between different releases. However, the tuples of the underlying microdata do not change between different releases. *Sequential-Release Publishing* handles scenarios where insert and/or delete operations are performed on the microdata. Hence, underlying data differ from release to release. *Collaborative Data Publishing* describes the situation where multiple data publishers provide different pieces of data to integrate them into a single dataset. The goal is that no data publisher should be able to infer more information than is available from his own and the final integrated dataset [24]. The original datasets remain stable. Multiple-release and collaborative data publishing do not cover situations where microdata is changed. Thus, if changes on data are possible or necessary, data publishers have to apply sequential-release publishing techniques to preserve privacy. For this reason, we will concentrate in the following on sequential-release publishing.

In addition to all attack models that can be applied to static datasets, dynamic datasets can also be the target of sequential attacks. An intuitive method to attack dynamic data is to compare attribute values of individuals through a sequence of releases (*Generalization-Linking Attack*). If records are generalized differently, for each attribute and each tuple, an attacker can select the less generalized value. As a result, the attacker can link information from several releases and consequently get more detailed information about an individual's quasi-identifier values. In a so-called *Exclusion Attack*, adversaries have background knowledge about which release contains which tuples and will be able to exclude possible sensitive attribute values by comparing different releases. By intersecting sensitive attribute values of each QI-group of a tuple across a sequence of releases, attackers can infer the sensitive attribute value of it. In case of recently added tuples, attackers use previous tuples to exclude sensitive attribute values. *Set-Theoretical Attacks* extend methods from the Exclusion Attack model and show that, even if adversaries only have knowledge about the values of quasi-identifier attributes of one individual as well as the release table at which the tuple was inserted, simple set-theoretical methods will enable them to reveal sensitive information. *Critical Absence Attacks* require background knowledge about the lifespan of at least one tuple which is a realistic assumption. It is easy to apply because adversaries only have to track a record over its lifespan. If delete operations are applied to a dataset, this attack represents an appropriate technique to extract sensitive information.

The previously considered attacks have the aim to associate a set of sensitive attribute values to an individual. Unlike these approaches, with *Value-Equivalence Attacks*, adversaries try to determine if two multisets of tuples are associated to the same multiset of sensitive attribute values [33]. Such attacks add a new perspective to the set of attack models because adversaries use dependencies between individuals to determine sensitive information. If data publishers provide samples of a dataset or parts of the dataset are available in other sources, this attack will become a notable threat to data privacy. *Possible World Exclusion Attacks* represent another possibility to attack privacy in scenarios where changes of some values are possible but specific sensitive values of tuples remain unchanged, for example, incurable diseases or criminal records. Such attacks are conducted by making assumptions on the linkage between an individual and his or her sensitive value. If the linkage is possible through the sequence of sanitized tables, adversaries can infer additional information [34]. Possible World Exclusion Attacks follow the principle of trial and error. Therefore, this attack is accompanied by high computational costs compared to other attack models.

Furthermore, it requires that some sensitive values remain linked to their individuals on a sequence of releases. Finally, *Background Knowledge Attacks* are possible if sensitive values are correlated to [illegible] Adversaries have the possibility to use general background knowledge about the sensitive values as well as sequential background knowledge resulting from changes of sensitive values for breaking privacy [35].

3.4. Privacy Criteria for Sequential Releases

Some intuitive methods exist to protect data on sequential release publishing. One possible idea is to anonymize and publish new records separately [36]. However, the major disadvantage of this approach is that releases become overgeneralized. Tuples from different updates cannot be combined to form a more specific QI-group. Thus, the approach suffers from bad data utility. Another approach is to apply privacy criteria for static datasets but, as shown in section 3.3, this approach is not appropriate to preserve individuals' privacy. An intuitive method to manage scenarios where delete operations are allowed is to ignore these operations [37]. However, this approach is not suitable for some application scenarios, for example, if deletions must be performed by legal regulations. Furthermore, data could suggest wrong implications. For example, some diseases such as flu epidemics occur in limited periods. If records are not deleted, data could suggest that patients suffer from flu even if this is not the case. Considering the disadvantages of these approaches, it becomes obvious that these mechanisms either are not appropriate to preserve privacy or suffer from bad data utility. In order to present a solution to this challenge, several major concepts have been introduced in the literature.

The first concept, *k-Anonymity Against Updates*, was proposed by [38] as a privacy criterion to maintain k-Anonymity [39, 40] with incremental updates which protects against Generalization-Linking attacks. k-Anonymity is defined as follows: Given a dataset T with quasi-identifier attributes q_1, ..., q_n; T is called k-anonymous, if every combination of values of q_1, ..., q_n occurs at least k times. The idea of k-Anonymity against updates is to avoid different generalizations on a sequence of releases in order to prevent adversaries from inferring QI-attributes of a record by comparing different releases. Although this approach guarantees k-Anonymity on a sequence of releases, it has various weaknesses with respect to privacy preservation. First, k-Anonymity is vulnerable to attacks against sensitive attribute values, for example, Homogeneity Attacks [41]. Second, the approach assumes a weak attack model with regard to background knowledge. For example, it is assumed that the attacker has no knowledge about which release contains which records. Furthermore, the criterion is restricted to insert-only scenarios. On the other hand, computational costs are low compared to other privacy criteria of sequential release publishing scenarios.

The second concept uses l-Diversity [41] as a baseline. Unlike k-Anonymity, l-Diversity takes sensitive attribute values of the QI-groups into account. While k-Anonymity can be attacked by homogeneity or background knowledge attacks, l-Diversity extends k-Anonymity by enforcing distinct sensitive attribute values within the QI-groups [41]. Formally, a QI-group q is called l-diverse if it contains at least l well represented values for the sensitive attribute S. A dataset T complies to l-Diversity if every QI-group is l-diverse. This definition is still flexible with respect to the exact definition of "well-represented". One straightforward choice is that each QI-group must contain at least l distinct sensitive attribute values. *Incremental l-Diversity* [36] is an update mechanism to prevent Exclusion as well as Set-Theoretical Attacks on update-only scenarios. It is assumed that adversaries have knowledge about quasi-identifier attributes and know which individuals are included in which release tables. The basic idea is that new tuples will be inserted only if they do

not compromise privacy of existing tuples. Thus, waiting lists temporarily store tuples which so fa[illegible] not be inserted [36]. Main advantages of this approach are low computational costs and an [illegible] to l-Anonymity against updates. However, it does not cover scenarios where delete or update operations are allowed.

The third concept, *m-Invariance* [37], guarantees privacy in scenarios that allow insert as well as delete operations. Adversaries are assumed to possess knowledge about the quasi-identifier attribute values of each tuple as well as knowledge about which release contains which tuples. Missing sensitive values allow adversaries to apply critical absence attacks [37]. Correspondingly, the basic idea of m-Invariance is to avoid such attacks by holding the *signature* of a tuple's QI-groups constant over its lifespan where the signature of a QI-group q is the set of distinct sensitive values. In order to document the lifespan of a tuple, each tuple t in a sequence of sanitized tables $\widehat{T_0}, ..., \widehat{T_n}$ is augmented with a timestamp attribute. The *generalized historical union* U_n contains all timestamped tuples for all publishing times. Furthermore, *m-Uniqueness* is defined in [37] which is similar to l-Diversity: A sanitized table $\widehat{T}$ is *m-unique* if each QI-group in $\widehat{T}$ contains at least m tuples and all tuples in the group have different sensitive values. Then, m-Invariance is defined as follows [37]: A sequence of sanitized tables $\widehat{T_0}, ..., \widehat{T_n}$ with $n \geq 1$ is *m-invariant* if both of the following conditions hold: (1) $\widehat{T_j}$ is m-unique for all $j \in [1, n]$, (2) for any tuple $t \in U_n$ with lifespan [x,y], $q_x(t), q_{x+1}(t), ..., q_y(t)$ have the same signature where $q_j(t)$ is the QI-group of t at time j.

One can show that if a sequence of sanitized tables $\widehat{T_0}$, ..., $\widehat{T}_{n-1}$ is m-invariant, then $\widehat{T_0}$, ...,$\widehat{T}_{n-1}$, $\widehat{T_n}$ will be also m-invariant if $\widehat{T_n}$ is m-unique and for any tuple $t \in \widehat{T}_{n-1} \cap \widehat{T_n}$, t's QI-groups have the same signature [37]. This property enables data publishers to create a new release table $\widehat{T}_{n+1}$, by only consulting T_n, T_{n+1}, and $\widehat{T}_n$, which significantly simplifies the process of anonymization. There are situations where maintaining m-Invariance is not possible [37]; this is mitigated by using synthetic counterfeits tuples which are augmented to existing QI-groups in order to preserve their signatures. Publishing the number of counterfeits tuples of each QI-group has no influence on data privacy. M-Invariance disables generalization-based attacks such as Exclusion Attacks and Generalization Linking Attacks as well as attacks that compare sensitive values such as Critical Absence Attacks or Set-Theoretical Attacks. However, [33] shows that m-Invariance is not able to protect data against Value-Equivalence Attacks. Moreover, it does not provide appropriate protection in scenarios which allow updates of values of existing tuples.

A further concept applies *Graph-based Anonymization*. In [33], the authors also proposed a graph-based approach to modify and extend m-Invariance so that Value-Equivalence Attacks become impossible. Since m-invariant tables already imply the existence of m-Value-Equivalence attacks, this approach has the goal to prevent e-Value-Equivalence Attacks with $e < m$. To prevent such attacks, tuples cannot always be published in QI-groups with each sensitive attribute value occurring only once because these QI-groups often expose correspondence structures. Instead, [33] propose to merge QI-groups sharing the same signature. In order to preserve data utility, they use anatomization to publish original quasi identifier attribute values. However, this modification is still not sufficient to protect data against e-Value-Equivalence Attacks.

Another concept is called *HD-Composition*. [34] proposes a generalization technique which applies l-Diversity to dynamic datasets. They assume a scenario where quasi-identifier attribute values as well as most sensitive attribute values of an individual can change over time. They further assume that individuals can be linked to more than one sensitive value. Sensitive values that once have been linked to an individual and can never be unlinked are called permanent sensitive values. In contrast, changeable sensitive attributes are called transient sensitive values. They further assume

that adversaries have knowledge about quasi-identifier attribute values of each individual as well as the [illegible] of a fraction of individuals. Note that for now, we have assumed that attackers have [illegible] The basic idea of HD-Composition is to focus privacy-preserving measures on tuples with permanent sensitive values, called *s-holder*, because these tuples create inferences across a sequence of releases. The approach of [34] uses tuples with transient sensitive values, called decoys, to cloak s-holders. Each decoy has the goal to cloak a specific permanent sensitive value. The key idea is that each QI-group must contain a specific number of decoys for each s-holder. In [34] it is shown that HD-Composition provides effective protection against Possible World Exclusion Attacks. Furthermore, it is an appropriate approach for scenarios where values of existing tuples can change. However, to apply this methodology sensitive values must be approximately uniformly distributed [35]. Another disadvantage results from the fact that transient sensitive values are not protected against sequential knowledge. Even if attackers have no guarantee that transient sensitive values of existing tuples remain unchanged, they may assume these properties about transient values which rarely change. In this case, privacy of transient values is not preserved. Thus, privacy protection only covers a specific setting.

The final major concept is *JS-Reduce* introduced by [35]. This is a privacy mechanism for scenarios where attackers have knowledge of quasi identifier attribute values of individuals, about probabilities of correlations between individuals and sensitive values as well as probabilities of sequences of sensitive values. Furthermore, they assume changing sensitive values. The JS-Reduce criterion is used to protect data against Sensitive Value Background Knowledge Attacks. The basic idea is to form QI-groups whose tuples have similar distributions with regard to sensitive values background knowledge. In [35] similarity of probability distributions is measured by applying the Jenson-Shannon Divergence [42]. To protect data, data publishers use sensitive value background knowledge and sequential background knowledge to compute posterior background knowledge. They further calculate revised background knowledge which intuitively calculates decision trees for each combination of individuals and sensitive values. These computations are used to create similar QI-groups with regard to revised background knowledge.

JS-Reduce can be combined with further privacy criteria such as m-Invariance or l-Diversity. Thus, it can provide a privacy guarantee even under strong assumptions with regard to adversary background knowledge. Unlike HD-Composition, JS-Reduce is also applicable if sensitive values are not uniformly distributed. However, compared to other privacy criteria, computational costs are very high, which leads to limited practicality especially for large datasets. Furthermore, Sensitive Value Background Knowledge Attacks require specific assumptions on correlations within the dataset. These assumptions do not hold in most application scenarios. In consequence, data publishers have to evaluate if protection against this attack is necessary.

3.5. Further Related Research

The literature also provides some other approaches to protect data in multiple release publishing scenarios. [43] showed that releasing additional count tables of combinations of values, for example, the number of tuples with gender attribute value "male" and "age < 30" could increase utility. Unfortunately, they also lead to threats in privacy. Therefore, [43] proposed a criterion to check whether such a marginal leads to privacy risks or not. If previously published releases of a dataset already exist, [44] introduced the concept of lossy joins to create additional releases. The basic idea is that joining different views of a dataset must create additional tuples which do not represent real world objects. Generalization has to guarantee that join attributes between views match more

than one record owner. The goal is that even if adversaries have knowledge about all releases, they should not be able to disclose privacy by joining them.

[45] suggests to base generalizations on the previous releases instead of the microdata. This leads to the property that new releases are not more specific than previous releases. Unfortunately, if values become more generalized in later releases, then data utility usually decreases as well. As mentioned in section 3.3, collaborative data publishing describes scenarios where different data owners merge their data with the goal that no data owner extracts additional knowledge than provided in the merged dataset. [46] propose an approach of a two party integration where both parties initially generalize their dataset to the most general values. Then, values are iteratively refined by both parties to find the best global specialization. The party which owns an attribute to be specialized provides the IDs of the QI-groups that were generated to the second party which also creates these QI-groups in its dataset. [47] and [48] introduced a solution for the same problem. In their approach, cryptographic techniques are used to exchange information. Both parties create k-anonymous tables. Then, intersections between QI-groups are evaluated. If an intersection of two QI-groups contains at least k tuples, joining both QI-groups will represent a QI-group in the global dataset. To prevent information disclosure by exchanging tuple IDs of QI-groups, cryptographic methods are applied to compute intersections of QI-groups.

3.6. Comparison of Criteria

Table 1 shows which of the earlier discussed major privacy criteria are suitable to preserve privacy against what type of attack. Since assumptions about background knowledge and data manipulation operations can differ between privacy criteria and attack models, attack model scenarios are assumed on assessing compatibilities of privacy criteria. Table 1 also presents the allowed data manipulation operations for each privacy criterion.

Table 1: Comparison of privacy criteria of sequential release publishing scenarios.
(Notation: GLA = Generalization-Linking Attack, EA = Exclusion, STA = Set-Theoretical, CAA = Critical Absence, VEA = Value-Equivalence, PWE = Possible World Exclusion, SVBK = Sensitive Value Background Knowledge; i = insert, d = delete, u = update.)

	Operations Allowed	GLA	EA	STA	CAA	VEA	PWE	SVBK
K-Anonymity ag. Updates	i	√	×	×	×	×	×	×
Incremental l-Diversity	i	√	√	√	×	×	×	×
M-Invariance	i/d	√	√	√	√	×	×	×
Graph-based Anonymization	i/d	√	√	√	√	√	×	×
HD-Composition	i/d/u	√	√	√	√	√	√	×
JS-Reduce	i/d/u	√	√	√	√	√	√	√

By only considering Table 1 alone, JS-Reduce appears to be the best privacy criterion for sequential release publishing because it provides a broad protection against multiple privacy attacks. However, data publishers additionally have to take two essential aspects into account. First, computational costs differ considerably between privacy criteria. For example, JS-Reduce requires computations of several probabilities for each tuple over all releases, whereas Incremental l-Diversity only needs to store previous generalizations. Data publishers have to evaluate which privacy criteria can be applied based on the available resources. Second, most of the attack models are only applicable to specific settings of data structures, background knowledge, and data manipulation

operations. For example, Critical Absence Attacks are only possible if delete operations are performed on the data. Data publishers need to analyze whether attack models constitute a real threat or may be applicable in a given setting.

Privacy Criteria that work in Multiple Release Publishing scenarios can be applied to achieve data privacy if the structure of the dataset changes across the published releases, for example, by addition of a new quasi-identifier attribute. However, they do not cover data manipulation of the underlying dataset. Privacy criteria that manage data privacy in Sequential Release Publishing scenarios cover data manipulation across multiple releases but do not allow for changes in data structure. Hence, privacy preservation in scenarios where data as well as its structure can change is still an open research question.

4. Data Warehouse Architectures and Privacy Preservation

4.1. Data Warehouse Architectures

The basic principle of data warehouse architectures is the replication of data from heterogeneous sources into a single database. Queries are executed directly on the database without involving any of the sources at the time of query processing. This approach offers a considerable advantage in response time compared to virtual integration. Storing the data within the data warehouse furthermore enables aggregation, manipulation, and cleansing of data. Data warehouses are also called Online Analytical Processing (OLAP) systems. Typical characteristics are high storage capacity, few but complex analytic queries with high data volume, no manipulation of existing records, periodical bulk inserts, a multidimensional data model, and business focus [49].

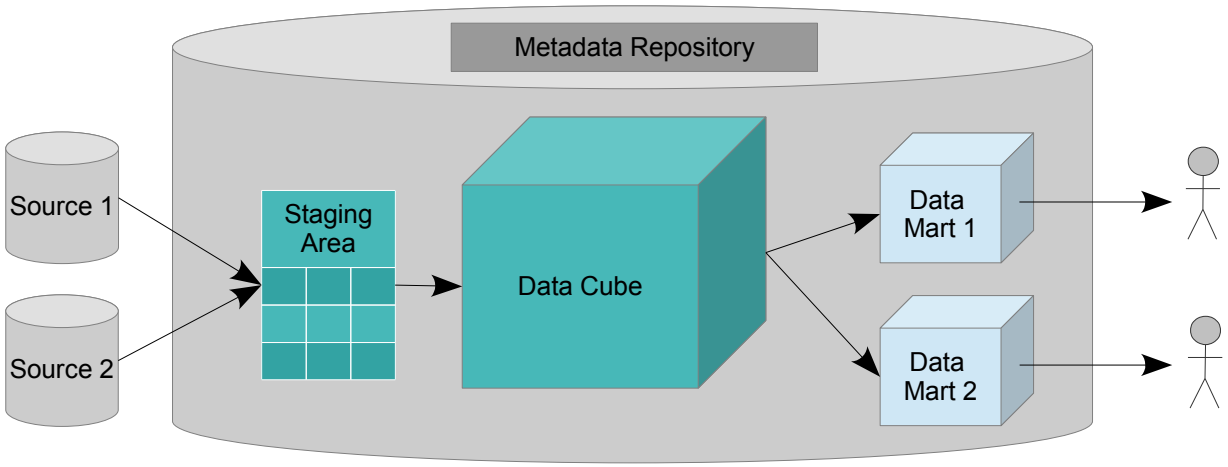

Figure 2: Data warehouse architecture

Figure 2 illustrates an exemplary architecture of data warehouses. The integration process of data warehouses is known as *Extract, Transform, Load* (ETL). The *extraction phase* includes all steps for extracting data from sources to the staging area, for example, executing export scripts, file parsing, or duplicate elimination [49]. In the *transformation phase*, data is transformed into the data structure of the data warehouse and stored in the staging area, which works as a buffer between sources and data cube, allowing for data cleansing and more efficient transformations. The multidimensional data cube represents the central storage component of a data warehouse

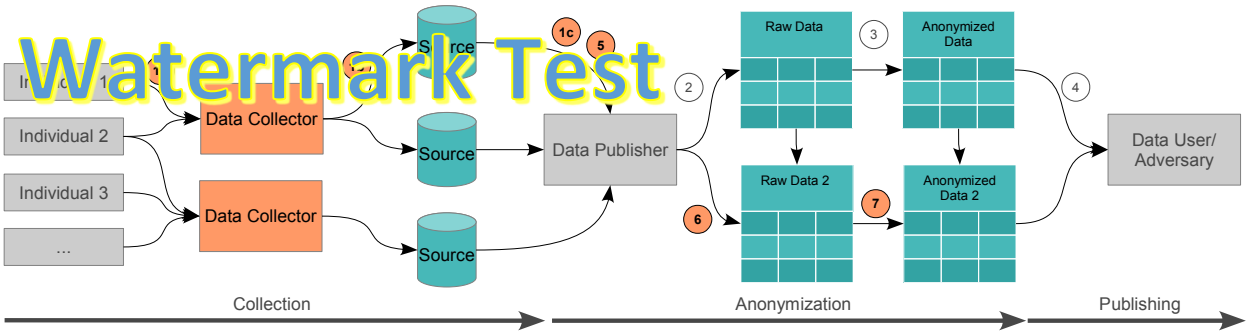

Figure 3: Process of privacy-preserving data provisioning in data warehouses

which holds the whole dataset. The *load phase* denotes the process of transferring data from the sources into the data cube. The metadata repository is used to store and manage metadata [50], for example, scripts, source descriptions, database schemes, view definitions, or version control information. Data marts provide pre-aggregated data from the data cube and are mainly used to model a specific aspect of the data.

4.2. Privacy Preservation in Data Warehouses

We assume a data warehouse scenario where data on individuals is processed. Because the sources of the data warehouse are often managed by different parties, the role of *data collectors* is introduced who gather data about individuals and provide them to the *data publisher*. Beside sanitizing and provisioning, the responsibilities of the data publisher involve data integration which includes merging and linking data from multiple sources, the integration of new sources as well as the management of source exclusions. Because these operations can lead to changes in the warehouse dataset, they represent a potential risk to individuals' privacy. Furthermore, the extended process has to consider that data integration is performed continuously. After provisioning of a release, new or changed data will be collected and integrated, which demands a new iteration of anonymization and data publishing and may also allow inferences between the sanitized datasets.

Figure 3 presents a process model of privacy-preserving data provisioning in data warehouses which extends the process from section 3.1. The extended model considers specific process steps in data integration scenarios. At the beginning, data collectors gather data on individuals (1a) and store them into one or more datasets (1b) which represent sources of the data warehouse. The data publisher gets access to these sources (1c) in order to integrate and transform them into a single dataset (2). Process steps 3 and 4 are equal to the process introduced in section 3.1. The first iteration is completed. The process does not end at this point because data collectors continue to gather data on individuals, which leads to changes in the sources. Furthermore, new sources can deliver additional data. Exclusion of sources also can lead to changes in the integrated dataset. The data publisher gets access to these changes (5) and applies them to the dataset (6). At the next step, the data publisher creates a new release by applying sequential release publishing techniques (7). Finally, the new release is published (4) and the next iteration starts at step 1a.

Attack models on dynamic dataset usually base on sequential information. In addition to static data attack models, all process steps that change the dataset do affect data privacy. Several

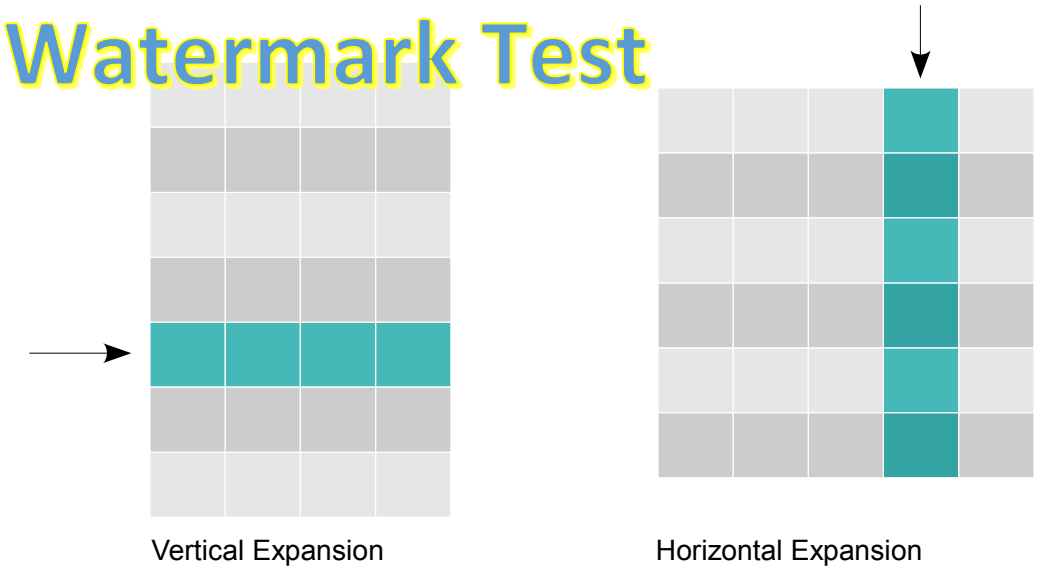

Figure 4: Vertical and horizontal expansion

activities can lead to changes in data and have to be monitored: inclusion of new sources, exclusion of existing sources, and data manipulation within the sources.

As shown in Figure 4, including new sources to a data warehouse principally can lead to a horizontal or a vertical expansion of the dataset. Intuitively, vertical expansions add new records to a dataset. The structure of the dataset remains unchanged. Thus, attack models as well as privacy criteria of sequential release publishing can be applied. Vertical expansions mainly occur if sources handle semantically similar data. In contrast, horizontal dataset expansion adds a new perspective to the dataset. Intuitively, in case of horizontal expansions, the dataset is extended by new attributes. Multiple release publishing could cover this scenario because new sources lead to additional views on the dataset. However, it does not consider changes of the underlying dataset. Thus, research on multiple release publishing can provide techniques to preserve individuals' privacy on horizontal dataset expansions but does not guarantee privacy on scenarios where datasets are expanded horizontally and vertically at the same time.

If sources for data integration become unavailable, for example, because they are going offline or are denying access, the integrated data will be affected in different ways. In the simplest scenario, the source will stop delivering new information but deletions of already integrated information are not necessary. In case of materialized integration, the data which was delivered by the source remains in the data warehouse. However, in some application scenarios, the data publisher must delete information which has been delivered by the source, for example, by legal regulation or copyright issues. Furthermore, virtual integration architectures are not able to store data of excluded sources. In these cases, similar to source inclusion, the dataset can be affected in two ways. If a source delivers a special aspect of information, the loss of information will be associated with deletions of existing attributes. We call this phenomenon horizontal reduction of the dataset. This kind of reduction will lead to a less detailed view on the existing records. However, future generalizations can lead to a refinement of quasi-identifier attribute values which may generate additional

inferences. In case of horizontal reduction of a dataset, adversaries may be able to infer additional knowledge out of the absence of records. Thus, excluding sources from a data warehouse could lead to privacy issues and has to be managed by the process to preserve privacy.

As data sanitization usually is performed to protect data against privacy attacks, it inherently affects individuals' privacy. Data publishers have to choose between a variety of privacy criteria and sanitization mechanisms. However, if data publishers use inappropriate sanitization mechanism or privacy criteria, adversaries have multiple possibilities to attack published datasets. Finally, even if data is protected against incremental updates, manipulating values of existing records also leads to privacy risks. For this reason, each operation which changes values of existing tuples has to be monitored with regard to privacy protection.

4.3. Point of Anonymization

Privacy techniques in data warehouses can be implemented at different locations. Depending on the general architecture and application scenario, data anonymization can be performed at following stages of the integration process which are presented in Figure 5.

Anonymized Sources. Sources which deliver already anonymized datasets represent the most restrictive possibility of anonymization because the data warehouse only works with sanitized data. As presented in Figure 5, anonymization is conducted before data integration. This leads to strong implications on data handling because the integration processing is performed on sanitized data. In case of horizontal expansion, joining the data becomes difficult or even impossible. Especially, if join attributes have been generalized, joining the data will lead to additional tuples which have no real world reference because generalizations lead to multiple join associations. If join attributes are not generalized, joining different sources will lead to a more detailed view on the data. In this case, the data publisher has to ensure that data still satisfies the selected privacy criterion. In case of vertical expansions, anonymized datasets can be merged into a single dataset. However, since the data publisher is not able to uniquely identify specific tuples, it is not possible to execute updates on existing tuples. Instead, the data publisher has to re-integrate data. The advantage of this approach is its applicability to the untrusted data publisher model. Since only sanitized data is stored, information disclosure by the data publisher would not lead to a privacy disclosure, but only if all sources provide data which satisfy the final privacy criterion.

Pre-Materialization Anonymization. Data can be anonymized in the staging area before it is stored in the cube. In this approach, data integration is performed by using original data. Since only anonymized records are stored persistently, security breaches do not reveal sensitive information, which represents a major advantage of this approach. However, similar to anonymized sources, after loading the data into the cube, the data integrator itself only has access to the anonymized dataset, which makes data management (e.g., update or delete operations) more difficult.

Post-Materialization Anonymization. In some scenarios, the data integrator needs to operate on the original data, for example, to verify deletion of individuals' records. In such scenarios, the data integrator has two possibilities. In the first approach, the original dataset as well as an anonymized dataset are stored in the data cube. Change operations are executed on the original dataset. Afterwards, a new anonymized release will be created. Since all data is available in the data warehouse, the application of anonymization mechanisms is straightforward. However, the data integrator has to manage two redundant datasets, which increases the workload of data managing (e.g., indexing) and leads to additional storage demand. In the second approach, the data integrator stores the original data and derives anonymized data marts which only represent

a special view on the data cube. However, this approach generates multiple anonymized datasets, which may lead to inferences between them. These dependencies have to be managed by the data integrator (e.g., by cache management). Furthermore, in both cases, privacy breaches on the data warehouse, for example, by hacking, will reveal the original dataset.

On-the-fly Anonymization. In virtual integration architectures, data is not stored by the data warehouse. Thus, if sources deliver non-anonymized data, anonymization has to be performed "on-the-fly". Similar to Pre-Materialization Anonymization, after integrating the data, anonymization techniques are applied to the dataset. At the next step, the anonymized dataset is provided to the data users. The data warehouse has to log sequential information which is required to apply selected privacy criteria to further releases. However, since data structure of virtual integration architectures can change from query to query, the complexity of anonymization processing strongly increases. This is aggravated by the fact that virtual integration architectures often operate on non-relational data structures such as semantic data. Since most privacy criteria assume relational data, data has to be mapped between different data models.

Query-based Anonymization. Unlike the approaches mentioned above, query based anonymization does not provide the entire dataset to data users. Instead, the data integrator provides answers. The result of each query is anonymized on-the-fly. Thus, for the data integrator it is not necessary to store any anonymized data. However, the data integrator has to restrict queries and manipulate query results to enforce data privacy. This approach has various advantages, especially in virtual integration scenarios, as the amount of data which has to be stored by data integrator can be kept small. Furthermore, there exist several approaches to anonymize query results, such as [51]. However, the information provided to data users is more restricted compared to approaches that provide the whole dataset. Furthermore, the influence of changing data on privacy preservation is still an open research question.

4.4. Further Considerations

Data Manipulation Operations. Before integrating data, software designers and data users have to define requirements with regard to data manipulations such as insert, update, or deletion of tuples, which are usually initiated by changes within the sources. Data publishers have to answer the question which data manipulations of sources have to be adopted by the data warehouse. Performing data manipulation operations after publication of records has fundamental implications on applicability of attack models and anonymization mechanisms.Data publishers can counteract this issue by using stronger privacy criteria that will increase computational costs or by accepting an increased privacy risk.

Source Exclusion Management. Basically, data publishers have two possibilities to manage the exclusion of sources. First, already included data remains in the dataset, which is only applicable for materialized integration scenarios. The exclusion of sources does not initiate delete operations on the dataset, which is a major advantage of this approach. However, this approach may be not possible in all application scenarios. Furthermore, it is possible that data quality decreases because records from excluded sources can become obsolete. In the second approach, already integrated data will be deleted in case of source exclusions. If only sanitized data is stored this approach significantly increases complexity. Furthermore, delete operations decrease the number of applicable privacy criteria.

Representation of Individuals. At the creation of a data warehouse, it has to be decided whether individuals are represented by a single or multiple tuples. Similar to data manipulation operations,

the decision has influence on possible attack models. For example, if an individual is represented by more than one record, attackers can use information from individual's tuples to infer sensitive information. Thus, if multiple records per individual are possible, the data integrator has to adapt its privacy mechanisms.

Privacy Criteria Selection. To preserve data privacy, the data integrator has to make assumptions about the possible attack models. Moreover, computational costs and applicability also need to be considered. As privacy criteria use different approaches of protecting data, different implications on data utility exist which depend on the characteristics of the data (e.g., distribution of values) and the general setting of the integration scenario (e.g., number of sources). Thus, the selection process will always be a tradeoff between complexity, utility, and privacy. Furthermore, privacy criteria usually have parameters to adjust the tradeoff between privacy and utility. Data publishers have the choice between lax or strict assignments for these parameters. The decision will affect the level of privacy preservation as well as data utility. Furthermore, an algorithm for applying the chosen privacy criteria has to be selected. As many privacy criteria have high complexity, in most cases heuristics are used to find an approximate optimal solution.

Number and Type of Sources. Considering the architecture of integrated systems, the number and type of sources have a major impact on architectural decisions. For example, if sources only allow restricted queries, materialized integration becomes impossible. Thus, data publishers have to decide which kind of sources to integrate into the data warehouse. The decision should include criteria such as size, availability, restriction, or update frequency. In most cases, developers will have less freedom of choice since the structure of sources usually drives the data warehouse architecture.

5. Experimental Evaluation

5.1. Goals and Preliminaries

In this section, implications of different anonymization approaches on data utility are evaluated. The goal of these experiments is to explore correlations between architectural decisions regarding anonymization and data utility in data warehouses. By simulating privacy preserving data provisioning in different sample architectures and measuring data utility of the results, differences are quantified. The architectures differ in design parameters that were discussed in section 4.3. By comparing results of different architectures, it is possible to evaluate implications of different parameters with respect to data utility. In order to avoid distortion of results caused by specific properties of a single dataset, the tests are performed on differently structured data sets.

Our newly developed test software is able to simulate different architectures by suppressing or providing information. The following aspects are important variables in such an evaluation. First of all, the ability to handle different *data manipulation operations* involves a fundamental design decision. Since the privacy criteria we selected do not support change operations, the experimental scenarios will concentrate on update-only and delete-update scenarios. Most privacy criteria assume that real world entities are represented by a single tuple, which is reasonable with data integration and therefore also assumed in the experiments. Furthermore, existing privacy criteria are not able to manage privacy on horizontal expansions of datasets in data warehouses. Thus, the test scenarios are limited to *vertical expansions*. As a result, the structure of data does not change if new sources are integrated. With data warehouse architectures, this represents a reasonable assumption because usually the central schema of the data cube remains stable.

In the following experiments, privacy criteria which enable changes of existing values are excluded for two reasons. First, both such criteria are accompanied by significant computational costs,

therefore, applicability to huge datasets will be restricted in many cases. Second, attack models which base on the changing of values and have high computational costs and are only applicable to specific settings that must be satisfied in most integration scenarios. Thus, these attack models currently do not seem to represent a major practical threat for large datasets. Since in many application scenarios it cannot ensure an appropriate level of privacy protection, k-Anonymity against updates is excluded from the experiment. In order to evaluate differences in utility preservation between less restrictive privacy criteria for insert-only scenarios and more-restrictive privacy criteria for insert/delete scenarios, *Incremental-l-Diversity* is selected as one privacy criterion for the experiments. M-Invariance and Graph-Based Anonymization both cover insert/delete operations. While m-Invariance uses generalization to sanitize data, Graph-Based Anonymization applies anatomization. Since Incremental-l-Diversity also uses generalization, *m-Invariance* is selected as the second sampling privacy criteria in order to avoid differences in data utility caused by different sanitization mechanisms.

Because the selected privacy criteria assume a provisioning of the full release, the approach of query-based anonymization will be excluded. Since On-The-Fly-Anonymization has its major advantage in virtual integration architectures, this point of anonymization is also not considered in the experiment. As a result, the experiments cover anonymized sources, pre-materialization anonymization, and post-materialization anonymization. If the integrated dataset results from only a few semantically similar sources, source exclusion or addition will affect a major part of the dataset. In the case of many sources, each representing a smaller proportion of the dataset, changes affect a smaller fraction of tuples. With sequential release publishing, data from further releases influences the process of anonymization in future releases. It is reasonable that different sizes of sources lead to different results with regard to data utility. Therefore, the experiments will compare scenarios containing few large sources as well as many small sources.

Furthermore, source datasets can have different statistical properties such as number of attributes or number of distinct sensitive values. To limit distortion of the experimental results due to specific statistical properties and to evaluate differences of the approaches depending on those properties, all sample architectures will be applied to multiple datasets. Because m-Invariance as well as Incremental-l-Diversity consider sensitive values during the anonymization process, it is reasonable to assume that the distribution and number of sensitive values can influence data utility. Moreover, m-Invariance as well as Incremental-l-Diversity use generalization to sanitize data. Thus, the number and range of quasi-identifier values potentially affect data utility of the resulting releases. In summary, the test datasets will differ in the distribution of sensitive values, the number of sensitive values, as well as the number and range of quasi-identifier attributes.

Because the selected privacy criteria use the sanitization method of generalization, generalization-based utility metrics represent an appropriate approach to quantify utility loss caused by anonymization. From the set of generalization based utility metrics, *Loss Metric* represents the most nuanced one and will thus be used in the experiment.

Any application of data mining algorithms is usually based on a preceding exploratory data analysis, often involving calculations of mean and standard deviation [52]. In order to evaluate the preservation of those statistics, queries with following structure will be executed on the datasets: `SELECT AVG(qi-attribute) WHERE sensitive-attribute = 'sen-attribute-value'`. The results from the original as well as from the sanitized datasets are compared to quantify information loss. Aggregate count queries also represent a commonly used way to analyze data [53]. For example, count queries are used to calculate correlations between attributes. Furthermore, the

application of many data mining algorithms, for example, the k-means clustering algorithm [54] requires the calculation of distances between tuples usually based on selection operations. In order to evaluate the applicability of these queries on the sanitized datasets, results from count queries of the following type will be compared between original and sanitized data: `SELECT COUNT(*) WHERE cond1 AND cond2`. In order to evaluate differences between complex and simple selection conditions, queries will differ in selectivity.

Probability-based metrics measure the preservation of statistical information, e.g., distributions. By combining count queries and the `GROUP BY` operator of SQL, it is possible to evaluate sensitive value distributions of the original as well as the sanitized dataset. By comparing both distributions, one can draw conclusions about the preserving of distributions. The distributions are computed using the following type of queries: `SELECT COUNT(*) WHERE qi-value1` $<$ `cond1 GROUP BY sen-values`. Based on these metrics, it will be possible to quantify following aspects of data utility: the degree of generalization and preservation of quasi-identifier attribute values, the preservation of count query information, and the preservation of distributions. The concrete application of utility metrics is described in section 5.5.

5.2. Experimental Scenarios

Figure 6 presents the sample architectures that will be tested in the experiments. All architectures will be applied to several scenarios that differ in the number and size of sources as well as the structural properties of the datasets (see Fig. 7). As a result, nine sample architectures will be applied to six scenarios, resulting in 54 test series.

Any selection of experimental data can be conducted in two ways: by using real datasets or by generating synthetic datasets. The major advantage of the first approach is that data represents a real application scenario. However, statistical properties are unchangeable. Furthermore, real datasets may require data cleansing, for example, because of missing values. Vice versa, synthetically generated datasets allow specification of statistical properties and guarantee good data quality but do not represent a concrete application. Since multiple datasets with different data structures will be tested, the synthetic data approach enables a more flexible definition of data properties. Thus, this approach will be used in the experiment. Datasets will differ in the properties of distribution of sensitive values, number of sensitive values, and number as well as range of quasi-identifier attributes. Dataset A represents data with fewer quasi-identifier attributes and uniformly distributed sensitive values. Dataset B differs from dataset A by the distribution of sensitive values as it contains dominating sensitive values. Dataset C represents data which contains more quasi-identifier attribute values than Datasets A and B. This property significantly increases potential combinations of generalizations. In case of smaller sources, five sources each containing 4,000 tuples are assumed, resulting in an integrated dataset of 20,000 tuples. In insert-only scenarios, newly added sources contain 4,000 tuples per source. In case of insert/delete scenarios, at each iteration, 4,000 tuples are deleted as well as inserted. Thus, the number of tuples remains stable. In scenarios which simulate many but small sources, 50 sources with 400 tuples each are assumed. Thus, the initial dataset also contains 20,000 tuples. At each iteration, 500 tuples are added to the dataset. If deletions are allowed, 500 tuples will be deleted per iteration.

5.3. Algorithms

M-Invariance. The algorithm proposed by [37] is used for our implementation of m-Invariance. It consists of four phases: division, balancing, assignment, and split. In the *division phase*, existing

tuples are divided into buckets. Each bucket contains tuples which share the same signature. If tuples have been deleted from the datasets, some buckets can become unbalanced, i.e., some sensitive values occur more often than others. In the *balancing phase*, new tuples from insertions are used to balance these buckets. If no appropriate tuples are available, counterfeit tuples are used. The *assignment phase* assigns remaining tuples to new or existing buckets. The assignment of tuples is performed iteratively. At each iteration, a set of balanced tuples is assigned to a bucket. The goal is to create buckets which contain as much tuples as possible and have signatures that contain a small number of sensitive values in order to create nuanced generalizations and reduce the number of iterations. The challenge is to ensure that the remaining unassigned tuples enable further assignments at the following iterations. For this, [37] provide three conditions which have to be satisfied on selecting tuples for assignment.

The *split phase* is processed on each bucket. In this phase, buckets are split to m-unique QI-groups having optimal generalizations. Possible generalizations are compared by calculating the sum of generalization intervals. Small values indicate a precise generalization. The basic idea is to split buckets recursively into smaller buckets until each bucket becomes m-unique. For this, [37] propose the following processing. First, they divide buckets into groups of n tuples that have the same sensitive value. Tuples of each group are sorted by the first quasi-identifier attribute. Then the first tuple of each group is assigned to a new bucket B_1 and the others to another Bucket B_2 which determines a split scheme. [37] create further schemes by assigning the first 2 (3, 4, ..., $n-1$) tuples to B_1 and the others to B_2. This procedure is repeated for each quasi-identifier attribute, resulting in multiple split schemes. For each scheme, the length sum of generalization intervals is calculated. The split scheme which scores the minimal value is used to divide the bucket. However, this heuristic leads to high computational costs because many split schemes have to be created and scored. For the datasets that were used in the experiments, the heuristic has been adjusted by only testing those split schemes which create a small bucket containing at most three tuples of each sensitive value and a large bucket containing the others. This approach significantly improves runtime by only slightly decreasing Loss Metric values. Similar effects were found by limiting the maximum number of tuples with equal sensitive values in a bucket because the number of comparative operations between tuples significantly decreases. Thus, this parameter is limited to 50, which slightly deteriorates data quality but leads to a significant improvement of runtime.

Incremental-l-Diversity. In [36] the use of multi-dimensional generalization [55] is proposed for implementing l-Diversity. The major advantage of this approach is that no pre-defined generalization hierarchies are used, which considerably increases flexibility with sanitization, resulting in more precise generalizations. For this reason, the approach is used for implementation of l-Diversity. The l-Diversity algorithm [36] consists of two phases. In the first step, a set of tuples is recursively split at the median value of a quasi-identifier attribute until no further split is possible without violating the l-Diversity criterion. Since most datasets have multiple quasi-identifier attributes, multiple splits of a set of tuples exist. In the context of the experiments, the split which maximizes the minimum number of distinct sensitive values of the resulting partitions will be used. For example, if a split generates two groups of tuples each containing seven distinct sensitive values and a second split generates groups containing six and eight distinct sensitive values, then the first split will be selected because the minimum number of distinct sensitive values of the resulting groups is higher. The basic idea of this approach is to create partitions which enable additional splits. In the second step, the resulting partitions are generalized.

The insertion algorithm to enforce Incremental-l-Diversity, proposed by [36], consists of three

phases: Add, Insert, and Split. In the first phase, inserted tuples are partitioned into l-diverse QI-groups by the use of the algorithm described above. Each QI-group whose generalization does not overlap with existing QI-groups is inserted into the dataset. Tuples from the remaining QI-groups are inserted into a waiting list. At the next step, for each of the existing QI-groups, it is checked whether an l-diverse group of tuples exists in the waiting list which has an equal or more refined generalization. If so, the group of tuples can be inserted into the existing QI-group. Finally, if existing QI-groups satisfy the splitting criteria, they are split into two l-diverse QI-groups.

5.4. Simulation of Sample Architectures

Depending on the architecture, a data publisher's access to original data and to sequential information can vary. This leads to differences in data processing. This section will discuss different cases with varying assumptions on data processing. Furthermore, it will describe the simulations in the experimental software framework.

Case 1: Anonymized Sources – Insert-Only. In the case of anonymized sources the data publisher receives data which is already generalized. In this experiment it is assumed that sources use the same privacy criteria as the data warehouse. Thus, the integration process becomes simple because QI-groups from sources only need to be merged into a single dataset. This approach is simulated by creating different files of a dataset, each representing a single source. As shown in Figure 8, each file will be anonymized separately, which simulates the anonymization processing within the sources. At the next step, sanitized QI-groups are merged to a release table which is published to data users. As presented in Figure 9, QI-groups of the sources are equal to the QI-groups of the published dataset.

New sources are included by the same procedure. However, including updates that were executed within the sources becomes difficult because generalizations of existing tuples within the source can change by refinement of their QI-groups. Since the data publisher is not able to identify specific tuples in its anonymized dataset, it is not possible to transfer these changes directly to the data warehouse. Thus, changes within sources are applied by replacing already imported QI-groups of the source by their new QI-groups. Hence, at each iteration, sources have to provide their entire anonymized datasets to the data publisher. Because QI-groups only contain data of a single source, the data publisher can manage this process by inserting an internal attribute which identifies the source of each QI-group. This attribute enables the deletion of all QI-groups of a source in order to replace them by the new ones. If sources become unavailable, already imported records will remain in the dataset because insert-only environments disable deletions of tuples. Note that this case is based on rather idealistic assumptions. If those are not satisfied, further research will be required to manage the process of data integration and data anonymization. For example, if the data warehouse requires a stronger privacy level than the sources, the data publisher has to generalize tuples that have already been generalized before, which significantly increases the complexity of the process.

Case 2: Anonymized Sources – Insert/Delete. Case 2 is equal to Case 1, except that delete operations are transferred to the integrated dataset. Similar to insertions, it is assumed that sources execute deletions on their own datasets and push the whole dataset to the data publisher who replaces the old QI-groups from the sources with the new ones. In case of source exclusion, all QI-groups that were imported from the excluded source are deleted from the integrated dataset. Similar to Case 1, the use of an internal attribute that holds a QI-group's source could enable these deletions.

Case 3: Pre-Materialization Anonymization – Insert-Only. Compared to anonymized sources, the approach of Pre-Materialization Anonymization has the advantage that sources provide original data. As illustrated in Figure 10, the data publisher is able to create QI-groups by the use of tuples from different sources. However, since the data publisher only stores anonymized data, re-identification of specific tuples during future iterations is not possible. Sequential information, for example, about the signature of a tuple will be lost.

Intuitively, two approaches could solve this issue. First, sources could add a synthetic identifier attribute to the data. If the data publisher stores this key, he will be able to track a tuple over a sequence of releases to read sequential information and enforce privacy criteria. Second, sources could provide inserted tuples only. In this approach, the data publisher would merge new tuples from existing sources and tuples from new sources into a single dataset which is used to create new QI-groups. These groups would be added to the existing dataset. The advantage of the second approach is that privacy criteria of static data scenarios can be applied to anonymize data because QI-groups in the anonymized dataset do not change. Additionally, only updates need to be imported by the data publisher, which is an advantage if sources deliver large datasets. However, since all QI-groups remain stable, no refinement of existing QI-groups is possible. Since all sequential information is available in the first approach, anonymization algorithms can be applied in a straightforward way. Thus, the resulting releases would be equal to Case 5, but sources have to provide distinct identifiers, which is still an unsolved issue.

Therefore, the second approach is selected for this architecture. The simulation of this approach is visualized in Figure 11 and works as follows. Because data publishers can integrate data before anonymization, the resulting dataset is represented by one basis file which represents new tuples from the sources. Afterwards, anonymization algorithms are applied to generate new QI-groups. The release of the iteration is created by merging existing QI-groups and new QI-groups. Finally, the release is stored separately because QI-groups are needed to create further releases.

Case 4: Pre-Materialization Anonymization – Insert/Delete. Expanding Case 3 by delete operations significantly impedes data handling. Deletions can be initiated by two events: exclusion of sources and deletions within the sources. In case of source exclusion, deletions can be transferred by creating an additional attribute which identifies the source of a tuple. If a tuple is deleted, its QI-group will become unbalanced and the deletion has to be compensated, in the best case with a new tuple which fits QI-group's generalization. Otherwise, the generalization has to be expanded. If no tuple is available, the data publisher has to insert a counterfeit tuple. The handling of deletions within the sources becomes more difficult because data publishers are not able to identify specific tuples within the integrated dataset. Thus, it is not possible to delete specific tuples or deduce sequential information from the sanitized dataset. Data publishers require additional information from sources. Similar to Case 2, synthetic identifiers can solve this issue. In this case, data publishers can infer necessary sequential information by using the identifier and apply the anonymization. Hence, results would be identical to Case 6. If sources do not deliver synthetic identifier, the processing of deletions in this architecture remains an open research question.

Case 5: Post-Materialization Anonymization – Insert-Only. Because data publishers hold original data as well as sequential information, in post-materialization anonymization architectures, anonymization algorithms can be applied without additional information. The simulation of this architecture is visualized in Figure 12 and works as follows. Data publishers read and integrate data from sources into a single dataset. This dataset is represented by the base file. At the next step, the selected anonymization algorithm is applied to the dataset. In the following iterations,

new tuples and new sources are integrated into it. The new tuples are represented by an update file. Afterwards, the anonymization algorithm is applied to the integrated data.

Source Insert/Update/Delete. In case of m-Invariance, source exclusion is applied by simply deleting all tuples of the source. Deletions within sources can be executed in two ways. Either, the data publisher imports the entire datasets from the sources and detects deletions by comparing the data with previous imports or the sources provide IDs of tuples that have been deleted. Usually, data publishers will prefer the second approach because tuples can be deleted by selecting their IDs instead of comparing a large amount of data. Furthermore, the approach generates less network traffic. In the prototype, update files will include IDs of tuples that have been deleted. Since original data is available, deletions can be simply applied to the data warehouse.

5.5. Calculation of Utility Metrics

Intuitively, *Loss Metric* evaluates how many attributes have been generalized on average. For example, a Loss Metric value of 2.0 indicates that an average generalization covers the range of the domain of two attributes. In case of two quasi-identifier attributes, this value would imply that both attributes have been suppressed for all tuples. In case of ten quasi-identifier attributes, the same value would imply that the range of generalization averagely covers $\frac{1}{5}$ of the domain of all quasi-identifier attributes. It becomes obvious that Loss Metric values of datasets containing different numbers of quasi-identifier attributes are not comparable. In order to address this issue, in the context of this experiment, Loss Metric will be standardized by dividing it by the number of quasi-identifier attributes. As a result, the metric value will always range between zero and one; zero represents no generalization and one the suppression of all values.

The goal of testing count queries with different selection conditions is to evaluate how anonymization influences their selectivity. In the context of these experiments, it is assumed that count queries, which are executed on sanitized data, will only consider results whose generalization completely covers the query's conditions. For example, if a query condition is "`age`>40", then only generalizations with a lower bound of ">40" of the age attribute will be included. Information loss with count queries is called *Selectivity* for the remainder of this article and is quantified as follows: $Selectivity(CQ) = \frac{CQ(T_{san})}{CQ(T_{orig})}$, where $CQ(T_{san})$ is the result of the execution of a count query CQ on the sanitized dataset and $CQ(T_{orig})$ represents the query result on the original dataset. In the experiment, the following count queries will be used to quantify selectivity, where the first query represents a count with a simple selection condition while the second query is more complex:

1. `SELECT COUNT (*) WHERE qi-att1 >= 40 AND sen-att = sen-value1;`
2. `SELECT COUNT (*) WHERE qi-att1 >= 40 AND qi-att2=1 AND qi-att3 < 5000 AND sen-att = sen-value1;`

In case of queries that calculate the average of a generalized attribute, information loss is quantified as $AverageDeviation(Q) = \frac{AVG_{orig} - AVG_{san}}{D_{max} - D_{min}}$, where AVG_{orig} represents the result of a query Q which selects the average value of a specified attribute and AVG_{san} represents the result of the same query executed on the sanitized dataset. D_{max} and D_{min} are the maximum respectively minimum values of the domain of the attribute that was used to calculate the average. The result can be interpreted as the deviation of the average in percentage of the domain range. In the experiments the following query Q was used:

3. `SELECT AVG (qi-att1) WHERE sen-att = sen-value1;`

Finally, in order to evaluate the preservation of sensitive value distribution, the following query is executed on sanitized and original datasets:

4. `SELECT COUNT (*) WHERE qi-att >= 40 GROUP BY sen-att;`

For quantifying information loss in this case, first the share of each sensitive value in both datasets is computed by executing the query on the sanitized dataset and its underlying original data. Then, the *Absolute Error of Distribution* is calculated as follows. Given a dataset T containing tuples with sensitive attribute values $S = \{s_1, s_2, ..., s_m\}$ and a query Q of type four. Let c_i be the number of tuples with sensitive value s_i that occur in the result of executing Q on T. Let the share P_i of s_i of the query result be defined as $P_i = c_i / \sum_{j=1}^{n} c_j$. Then, *Absolute Error of Distribution* $= \frac{1}{n} \sum_{i=1}^{n} |P_i^{orig} - P_i^{san}|$, where P_i^{san} is the share of sensitive attribute s_i of the result of Q executed on a sanitized version of T and P_i^{orig} the analogue executed on the original T.

This metric gives a good indication about maintaining distributions in sanitized datasets. However, it does not consider the domain of sensitive attribute values. For example, if the share of each sensitive value of the original dataset is 0.2 on average, then an Absolute Error of Distribution of 0.001 indicates that the proportion of attributes is maintained very well. However, if the share of each sensitive value of the original dataset would be 0.0001 on average, then the same Absolute Error of Distribution value would indicate a distortion of distribution in the sanitized dataset. Hence, Absolute Errors of Distribution are not comparable between datasets which have different sensitive attribute domains.

The metric *Relative Error of Distribution* is introduced to solve this issue. The basic idea is to measure percentage deviation of shares instead of absolute deviation, which results in the following formula: *Relative Error of Distribution* $= \frac{1}{n} \sum_{i=1}^{n} (\frac{|P_i^{orig} - P_i^{san}|}{P_i^{orig}})$. This metric can be interpreted as the average deviation of shares of sensitive attribute values between the sanitized dataset and its underlying original data in percentage of the share of the sensitive attribute value in the original dataset. For example, a Relative Error of Distribution of 0.05 indicates that shares of sensitive values in the sanitized dataset differ five percent on average from shares of sensitive values in the original data.

5.6. Implementation

The programming language Java was used for implementing the experimental prototype. SQLite was adopted as relational database system which implements most SQL92 standards [56]. Furthermore, it allows the storing of databases within the memory. Test scenarios were performed on a 64-bit workstation running a 4×2.67Ghz Intel i5 CPU with physical memory limited to 8 gigabyte. The internal data handling of the prototype is organized by using Java objects as well as the in-memory database SQLite. The management of QI-groups and complex data structures, such as buckets or signatures, is performed by Java objects. All relational data is stored by the use of original values. Thus, the processes of anonymization and generalization have to be handled by Java objects within the application.

Database tables are represented by a class *Microdata* that manages operations on the dataset and converts Java operations and class methods into SQL queries. This includes operations like inserting and removing tuples, counting tuples of a specific sensitive value, or getting the distribution of values. The central variable of this class is a *SQLwrapper* object which manages a Java

Database Connectivity Driver (JDBC) to connect to the in-memory database management system (H2 [35]). The SQL wrapper class was adopted from the University of Texas Dallas Anonymization Toolbox (Java implementation [57]). Each Microdata instance represents a single table in the DBMS. Because m-Invariance and Incremental-l-Diversity have different requirements with regard to data handling, both algorithms are implemented by different classes.

5.7. Results for Selecting the Point of Anonymization

Figure 13 shows average Loss Metric values of different points of anonymization for varied datasets and privacy criteria. Each bar summarizes values from scenarios of many small sources and few large sources. It clearly shows that architectures which perform anonymization after data integration have significant advantages in data quality. In the case of m-Invariance, Loss-Metric values of pre-materialization anonymization and post-materialization anonymization are about 0.40, which means that 40 percent of each quasi-identifier attribute domain is generalized on average. For anonymized sources, this value increases to 0.49, indicating that the range of generalization is approximately 10 percent higher. The same holds true for Incremental-l-Diverse datasets.

Results from other metrics confirm this tendency. The median Relative Error of Distribution of anonymized sources is 50 percent higher in comparison to the other architectures. Furthermore, for anonymized sources the Selectivity of count queries is significantly lower. Thus, anonymized sources are only preferable if the integrator uses the untrusted data publisher model or if anonymized sources are unavoidable. If data is integrated before anonymization, the algorithms are applied to larger datasets. If more tuples are available for building QI-groups by generalization, the probability of finding tuples with similar quasi-identifier attribute values increases. Therefore, it is possible to create QI-groups with a small range of generalization.

The results also indicate that data utility between post- and pre-materialization anonymization does not differ significantly for insert-only scenarios. However, in case of Incremental-l-Diversity, pre-materialization integration has the advantage that all tuples can be published immediately because each iteration of data insertions is handled separately. This approach does not allow an adversary to infer sequential knowledge because each QI-group remains stable. However, since refinements of existing QI-groups are disabled, no improvements of bad generalizations are possible. Furthermore, as described in section 4.3, post-materialization has several advantages over pre-materialization anonymization.

5.8. Results for Selecting Privacy Criteria

Experimental results indicate that the selection of privacy criteria has strong influence on data utility. In insert-only environments, Incremental-l-Diversity has advantages in Loss Metric, selectivity of count queries, Relative Error of Distribution, and preservation of average. As shown in Figure 13, Loss Metric values of incremental-l-diverse releases are much lower than of m-invariant releases. In average, Loss Metric values are smaller by 18.9 percentage points (median 19.5) compared to m-invariant releases. Differences in selectivity are even larger. Figure 14 presents average selectivity of m-invariant and incremental-l-diverse releases. Whereas m-invariant release tables enable data user to select 41 percent of tuples of simple count queries and 5.3 percent of complex count queries on average, incremental-l-diverse tables averagely reach selectivity rates of 97.6 percent respectively 49.4 percent.

Furthermore, results from average queries show that incremental-l-diverse tables better maintain average values. As visualized by Figure 15, which shows average deviations of average-queries

of both privacy criteria for varied datasets, incremental-l-diverse releases have smaller deviations on average than m-invariant releases. Results from Dataset B indicate that if data is not uniformly distributed, m-invariant releases remain stable to maintain average values. However, even if Incremental-l-Diversity has advantages in data utility, a major disadvantage is that a high number of tuples is inserted to the waiting list instead of the release, which negatively influences data refreshment. This may be an issue in application scenarios which require current values. Section 5.11 presents a detailed view on this issue. Overall results reveal that the selection of privacy criteria significantly influences data utility in several ways. M-Invariance will guarantee that all tuples are inserted immediately, while Incremental-l-Diversity delivers better data utility.

5.9. Implications of Delete Operations

To evaluate influences of delete operations on data utility, scenarios with delete operations are compared to the same scenarios without these operations. The results indicate that, in insert/delete scenarios, data utility depends on the distribution and structure of sensitive attribute values as results strongly differ between the different datasets. According to the experimental findings, architectures which use post-materialization integration can handle delete operation without significantly losing data utility compared to equal scenario settings without delete operations. However, in case of Dataset C, deletions lead to a loss in data utility as Loss Metric values and Relative Error of Distribution numbers increase. Furthermore, selectivity decreases compared to the insert-only scenario. This behavior can be explained by the high number of distinct sensitive values that lead to small buckets, which in turn decreases the probability to find similar tuples to build a QI-group.

Unlike post-materialization anonymization, anonymization at the sources does not preserve data utility on delete operations. As shown in Figure 16, in case of uniformly distributed sensitive values, delete operations in sources lead to increased Loss Metric values; the same holds for selectivity of count queries. Surprisingly, this effect reverses if the sensitive attribute values are non-uniformly distributed. In this case, data utility improves if delete operations are performed. However, in all cases, data utility only changes up to a specific level and then remains stable. These effects can be explained by taking into account that anonymized sources operate on small datasets in this experiment. In consequence, dissimilar tuples initially must form QI-groups. If only few tuples are inserted at each iteration, the probability to compensate a deletion by new tuples will be low because of the high number of distinct sensitive values. In the result, counterfeit tuples will be inserted which leads to a refinement of QI-groups. After some iterations of insertions, tuples continuously find partners for building QI-groups and replace counterfeits. At the same time, new counterfeits are inserted resulting in a stabilization of Loss Metric values. In case of pre-materialization anonymization, deletions lead to decreasing data utility. This is caused by fixed generalizations of QI-groups. Because original quasi-identifier attribute values are not available after data integration, generalization ranges remain stable or grow if inserted tuples, which are used for compensation of deletions, expand the generalization range. Thus, this approach leads to a continuing deterioration of QI-group generalizations, which has negative influence on all measured utility metrics.

Overall, delete operations lead to different effects on data utility depending on data structure and architecture. Figure 17 presents the ranges of Loss Metric values across all iterations of varied points-of-anonymization in insert/delete scenarios. Wide ranges indicate that utility decreases over a sequence of iterations. In all scenarios, post-materialization anonymization architectures deliver

the best level of utility. Results further indicate that pre-materialization anonymization leads to bad data quality if delete operations are possible. The major reason for this is the missing option of finding existing QI-groups in anonymized data. Architectures also suffer from bad data quality. However, the difference between insert-only scenarios and insert/delete scenarios is smaller than with post-materialization anonymization.

5.10. Implications of Data Structure

Figure 13 shows that Dataset A achieves best results in most cases. There are two major reason for this effect. First, Dataset A contains a small number of distinct sensitive values which are uniformly distributed; the probability of inserting a tuple with sensitive attribute value s will be $\frac{1}{d}$ where d is the number of distinct sensitive values. Assuming that one tuple with sensitive value s is deleted and in parallel 1000 tuples are inserted, the expected number of inserted tuples with sensitive value s will be $1000 * \frac{1}{d}$. For Dataset A (d=22) approximately 43 tuples with sensitive value s are expected whereas the set of inserted tuples of Dataset C (d=150) would only contain six tuples of this sensitive value on average. Thus, since Dataset A has more possibilities of compensating deletions, the probability of finding a tuple which is similar to a deleted tuple's QI-group is higher than for Dataset C. This leads to a better data quality in the long term. Second, a smaller number of quasi-identifier attribute values leads to better generalizations because tuples need to be similar in fewer attributes. In consequence, similar tuples can form a QI-group which leads to small generalization ranges. However, in case of Incremental-l-Diversity, a small number of quasi-identifier attributes leads to more overlapping generalizations because the total number of possible combinations is smaller compared to datasets with many quasi-identifier attributes.

5.11. Effects of Inserting Synthetic Data

Beside utility metrics, consistency and currentness represent additional dimensions of data quality. Synthetic data such as counterfeits or expired (but not deleted) tuples affect utility because they can lead to wrong implications. Delete operations on m-invariant datasets produce such counterfeits. Figure 18 shows the median number of counterfeits tuples of varied points of anonymization and datasets of scenarios of many small sources. One the one hand, results show that architectures which anonymize data after integration need less tuples to compensate deletions. On the other hand, significant differences between the datasets exist. In case of few large source and Dataset A, pre-materialization integration as well as post-materialization integration architectures produce no counterfeit tuples.

Many application scenarios require current data. For example, retail companies require current sales figures in order to manage the delivery of goods. Waiting lists used to enforce Incremental-l-Diversity counteract this requirement as they insert multiple tuples by a time shift. Thus, waiting lists also represent a data quality issue. Figures 19 and 20 exemplary show that Incremental-l-Diversity has limitations in publishing insertions at the time of their occurrence. The number of published tuples grows very slowly compared to the overall number of inserted tuples. Instead, most tuples are inserted into the waiting list. The reason is that QI-groups are only extended if an l-diverse group of tuples from the waiting list matches a QI-group's generalization. This is a restrictive hurdle as QI-groups are refined from iteration to iteration. Thus, a large number of tuples is needed to find fitting l-diverse groups in the waiting list. If time-shifted insertions are not acceptable, this behavior represents a major disadvantage of Incremental-l-Diversity.

5.12. Discussion

The experiments showed that several parameters and design decisions of the anonymization process have significant implications on data utility. It becomes apparent that architectures which integrate data before anonymization achieve a better data utility. Thus, the architecture of anonymized sources only has advantages if the data warehouse is considered insecure or if the untrusted data publisher model is used. If delete operations within the data warehouse are required, post-materialization anonymization has clear advantages in data utility compared to pre-materialization anonymization because it allows for a refinement of existing QI-groups. Anonymized sources also suffer from bad data utility. Not least, the number of counterfeit tuples is significantly lower if post-materialization anonymization is used.

The privacy criterion of Incremental-l-Diversity has considerable advantages in data utility compared to m-Invariance but is accompanied by several limitations. First, the criterion is not applicable to insert/delete scenarios. Expired tuples are not deleted, which possibly leads to inconsistencies. Second, waiting lists lead to a delayed insertions of most tuples, thus, currentness of data is affected. However, pre-materialization anonymization architectures are able to avoid waiting lists because QI-groups are not changed after being published. Unlike Incremental-l-Diversity, m-Invariance is able to manage delete operations. However, if specific sensitive attribute values do not occur in further releases, counterfeit tuples will artificially preserve this value and the published dataset may provide wrong implications to the data user. The structure of the data also has an important influence on post-anonymization utility. Uniformly distributed sensitive values enable a higher data utility of m-Invariant releases. In contrast, Incremental-l-Diversity is less dependent on data structure, which results in stable utility metric values even if datasets change.

In summary, no single approach emerges as preferable in general. Instead, the best setting to manage privacy preserving data warehousing strongly depends on tradeoffs and the specific requirements of the application scenario.

6. Conclusion

The goal of this research was to evaluate the applicability of database privacy concepts to data warehousing. Adversaries have multiple possibilities to extract sensitive information even if identifying attributes are suppressed in a dataset. Also when updated anonymized datasets are provisioned by a data warehouse, inferences could be used by adversaries to apply advanced attack models. In particular, sequential release publishing techniques are in general suitable for data warehouses. The selection of privacy criteria is accompanied by a trade-off between data utility, restrictions on data handling, computational costs, and the level of protection. Data publishers have to analyze their setting in order to identify possible threats and select appropriate protection mechanisms. Furthermore, several architectural alternatives for managing anonymization were discussed. The integration of new sources, the exclusion of sources, data manipulation operations, and data sanitization were identified as privacy affecting process steps. Here, the point of anonymization represents another major design decision of data anonymization in data warehouses.

With the help of our newly-developed experimental framework, the anonymization processing in different data warehouse architectures was simulated. By applying multiple utility metrics to the results, differences in data utilities were identified. The results do not show any generally preferable architecture. With respect to the point of anonymization, however, architectures which integrate data before sanitization have advantages in data utility. There further exist considerable

differences between privacy criteria. The less restrictive privacy criterion of Incremental-l-Diversity has advantages over m-Invariance in most scenarios and utility metrics. However, most of the instantiations are insufficient for delay. Thus, if current data is required by the application scenario, this criterion has a significant disadvantage. Furthermore, Incremental-l-Diversity is not able to handle delete operations. Moreover, we showed that delete operations within the data warehouse significantly increase the complexity of data handling and limit architectural possibilities. Finally, the statistical properties of the data, for example, the distribution of sensitive attribute values also impact data utility. The results indicate that incremental-l-diverse releases are less dependable on the properties of a dataset than m-invariant releases.

Our work also identifies several issues that need to be further researched. Semantic data structures, which are commonly used in virtual integration, lead to additional issues of privacy preservation. Because of changing sources and flexible integration, semantic data often do not possess a static structure. Since privacy criteria of sequential release publishing work on static data structures, their application for semantic data is still an open issue. Furthermore, the application of privacy concepts to dynamic semantic data also requires additional research. Moreover, many privacy criteria require sequential information, for example, previous generalizations of a tuple. Since virtual integration architectures do not materialize data, the handling of this information should be considered in future research. Not least, if sources do not satisfy certain assumptions, for example, enforcing the same privacy level as required in the data warehouse, further concepts and approaches will be required for efficient anonymization.

In summary, even if our results so far are limited to materialized integration scenarios, our work motivates that existing privacy concepts can in fact be fruitfully adopted to enforce data privacy in data warehouses. The simulations revealed different properties of possible design approaches regarding data utility and limitations of data handling which could serve as a guideline for practical applications.

References

[1] IDC Corporate . Data management and analytics solutions drive the adoption of application development and deployment software, according to IDC. 2012. URL http://www.idc.com/getdoc.jsp?containerId=prUS23811012.

[2] Baker LB, Finkle J. Sony PlayStation suffers massive data breach. 2011. URL http://www.reuters.com/article/2011/04/27/us-sony-stoldendata-idUSTRE73P6WB20110427.

[3] Rushe D. Bloomberg editor in chief: reporters' access to client data 'inexcusable'. 2013. URL http://www.guardian.co.uk/media/2013/may/13/bloomberg-editor-chief-clients-data.

[4] Federal Commissioner for Data Protection and Freedom of Information . Protect Privacy! European Data Protection Conference ends with appeal. 2013. URL http://www.bfdi.bund.de/EN/PublicRelations/PressReleases/2013/08_EuropeanDataProtectionConferenceEndsWithAppeal.html.

[5] IDC Corporate . Data privacy protection becomes strategic it priority, according to recent IDC energy insights survey of central and eastern european utilities. Article 48365; 2011. URL http://idc-cema.com/eng/about-idc/press-center/.

[6] IDC Corporate . Business and IT Priorities in the Western European Professional Services Industry: IDC study finds that efficiency and flexibility are center stage. 2013. URL http://www.idc.com/getdoc.jsp?containerId=prIT23968613.

[7] Fabian B, Kunz S, Konnegen M, Müller S, Günther O. Access control for semantic data federations in industrial product-lifecycle management. Computers in Industry 2012;63(9):930–40.

[8] Fabian B, Kunz S, Müller S, Günther O. Secure federation of semantic information services. Decision Support Systems 2013;55(1):385–98.

[9] Ariyachandra T, Watson H. Key organizational factors in data warehouse architecture selection. Decision Support Systems 2010;49(2):200 –12.

[10] Kunz S, Fabian B, Marx D, Müller S. Engineering policies for secure interorganizational information flow. In: 15th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011. 2011, p. 438–47.

[11] Sarathy R, Muralidhar K. Secure and useful data sharing. Decision Support Systems 2006;42(1):204 –20.

[12] Kunz S, Evdokimov S, Fabian B, Stieger B, Strembeck M. Role-based access control for information federations in the industrial service sector. In: Proc. 18th European Conference on Information Systems (ECIS 2010), Pretoria, South Africa. 2010,.

[13] Fernández-Medina E, Trujillo J, Villarroel R, Piattini M. Access control and audit model for the multidimensional modeling of data warehouses. Decision Support Systems 2006;42(3):1270–89.

[14] Evdokimov S, Fabian B, Kunz S. Challenges for access control in knowledge federations. In: Proceedings of the International Conference on Knowledge Management and Information Sharing (KMIS 2009), Madeira, Portugal. ISBN 978-989-674-013-9; 2009, p. 224–9.

[15] Kunz S, Brecht F, Fabian B, Aleksy M, Wauer M. Aletheia – improving industrial service lifecycle management by semantic data federations. In: Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. 2010, p. 1308–14.

[16] Loos P, Lechtenbörger J, Vossen G, Zeier A, Krüger J, Müller J, et al. In-Memory-Datenmanagement in betrieblichen Anwendungssystemen. Wirtschaftsinformatik 2011;53(6):383–90.

[17] Loos P, Lechtenbörger J, Vossen G, Zeier A, Krüger J, Müller J, et al. In-memory databases in business information systems. Business & Information Systems Engineering 2011;3(6):389–95.

[18] Evdokimov S, Fabian B, Günther O, Ivantysynova L, Ziekow H. RFID and the internet of things: Technology, applications, and security challenges. Foundations and Trends® in Technology, Information and Operations Management 2011;4(2):105–85.

[19] Agrawal R, Srikant R. Privacy-preserving data mining. ACM Sigmod Record 2000;29(2):439–50.

[20] Zhu D, Li XB, Wu S. Identity disclosure protection: A data reconstruction approach for privacy-preserving data mining. Decision Support Systems 2009;48(1):133 –40.

[21] Amiri A. Dare to share: Protecting sensitive knowledge with data sanitization. Decision Support Systems 2007;43(1):181–91.

[22] Cox L. Suppression methodology and statistical disclosure control. Journal of the American Statistical Association 1980;75(370):377–85.

[23] Dalenius T. Finding a needle in a haystack or identifying anonymous census records. Journal of Official Statistics 1986;2(3):329–36.

[24] Fung B, Wang K, Chen R, Yu P. Privacy-preserving data publishing: A survey of recent developments. ACM Computing Surveys 2010;42(4).

[25] Chen BC, Kifer D, LeFevre K. Privacy-Preserving Data Publishing. Boston, MA, USA: Now Publishers Inc.; 2009.

[26] Gehrke J. Models and methods for privacy-preserving data analysis and publishing. In: Proceedings 22nd International Conference on Data Engineering (ICDE). IEEE; 2006,.

[27] United Nations Statistical Commission and Enonomic Commission for Europe . Terminology on Statistical Metadata; 2000. URL http://ec.europa.eu/eurostat/.

[28] Denning D, Denning P, Schwartz M. The tracker: A threat to statistical database security. ACM Transactions on Database Systems (TODS) 1979;4(1):76–96.

[29] Denning D, Schlörer J. A fast procedure for finding a tracker in a statistical database. ACM Transactions on Database Systems 1980;5(1):88–102.

[30] Citteur C, Willenborg L. Public use microdata files: Current practices at national statistical bureaus. Journal of Official Statistics 1993;9(4):783–94.

[31] Leser U, Naumann F. Informationsintegration. Heidelberg: dpunkt-Verlag; 1st ed.; 2007.

[32] Xiao X, Tao Y. Anatomy: Simple and effective privacy preservation. In: Proceedings 32nd International Conference on Very Large Data Bases (VLDB). 2006, p. 139–50.

[33] He Y, Barman S, Naughton J. Preventing equivalence attacks in updated, anonymized data. In: Proceedings 27th International Conference on Data Engineering (ICDE). IEEE; 2011, p. 529–40.

[34] Bu Y, Fu A, Wong R, Chen L, Li J. Privacy preserving serial data publishing by role composition. Proceedings of the VLDB Endowment 2008;1(1):845–56.

[35] Riboni D, Pareschi L, Bettini C. JS-Reduce: Defending your data from sequential background knowledge attacks. IEEE Transactions on Dependable and Secure Computing 2012;9(3):387–400.

[36] Byun J, Sohn Y, Bertino E, Li N. Secure anonymization for incremental datasets. In: Proceedings 3rd VLDB International Conference on Secure Data Management. Springer; 2006, p. 48–63.

[37] Xiao X, Tao Y. M-Invariance: Towards privacy preserving re-publication of dynamic datasets. In: Proceedings 2007 ACM SIGMOD International Conference on Management of Data. ACM; 2007, p. 689–700.

[38] Pei J, Xu J, Wang Z, Wang W, Wang K. Maintaining k-anonymity against incremental updates. In: Proceedings 19th International Conference on Scientific and Statistical Database Management (SSBDM). IEEE; 2007,.

[39] Samarati P, Sweeney L. Generalizing data to provide anonymity when disclosing information. In: Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 1998), Seattle, WA, USA. 1998,.

[40] Samarati P. Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 2001;13(6):1010–27.

[41] Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M. l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) 2007;1(1):1–52.

[42] Lin J. Divergence measures based on the shannon entropy. IEEE Transactions on Information Theory 1991;37(1):145–51.

[43] Kifer D, Gehrke J. Injecting utility into anonymized datasets. In: Proceedings 2006 ACM SIGMOD International Conference on Management of Data. ACM; 2006, p. 217–28.

[44] Wang K, Fung B. Anonymizing sequential releases. In: Proceedings 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM; 2006, p. 414–23.

[45] Sweeney L. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 2002;10(05):557–70.

[46] Wang K, Fung B, Dong G. Integrating private databases for data analysis. In: Proceedings 2005 IEEE international Conference on Intelligence and Security Informatics (ISI). Springer; 2005, p. 23–41.

[47] Jiang W, Clifton C. Privacy-preserving distributed k-anonymity. Data and Applications Security XIX 2005;:924–.

[48] Jiang W, Clifton C. A secure distributed framework for achieving k-anonymity. The VLDB Journal 2006;15(4):316–33.

[49] Bauer A, Günzel H. Data-Warehouse-Systeme. dpunkt-Verlag, Heidelberg; 2nd ed.; 2004.

[50] Chaudhuri S, Dayal U. An overview of data warehousing and OLAP technology. ACM Sigmod Record 1997;26(1):65–74.

[51] Dwork C. Differential privacy: A survey of results. In: Theory and Applications of Models of Computation. Springer; 2008, p. 1–19.

[52] Fayyad U, Piatetsky-Shapiro G, Smyth P. From data mining to knowledge discovery in databases. AI Magazine 1996;17(3):37–54.

[53] LeFevre K, DeWitt D, Ramakrishnan R. Workload-aware anonymization. In: Proceedings 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM; 2006, p. 277–286.

[54] MacQueen J. Some methods for classification and analysis of multivariate observations. In: Proceedings 5th Berkeley Symposium on Mathematical Statistics and Probability; vol. 1. California, USA; 1967, p. 281–97.

[55] LeFevre K, DeWitt DJ, Ramakrishnan R. Mondrian multidimensional k-anonymity. In: Proceedings 22nd International Conference on Data Engineering (ICDE). IEEE; 2006,.

[56] Newman C. SQLite. Sams Publishing, Indianapolis; 1st ed.; 2004.

[57] University of Texas at Dallas, Data and Privacy Lab, Department of Computer Science . UTD Anonymization Toolbox. 2013. URL http://cs.utdallas.edu/dspl/cgi-bin/toolbox/.

Biographical Notes

- PD Dr. Benjamin Fabian is a senior researcher and former visiting professor at the Institute of Information Systems, Humboldt-Universität zu Berlin. His research focuses on medium to global-scale data exchange, data analysis, cloud technology as well as security and privacy in the era of Big Data and the Internet of Things. He holds a Diplom degree in Mathematics from the Free University of Berlin and a Ph.D. in Information Systems from HU Berlin. He completed his Habilitation in 2013 on Security and Privacy in Federated Information Services at HU Berlin.

- Tom Göthling holds a Master degree in Information Systems from Humboldt-Universität zu Berlin. His research interests focus on data warehousing and privacy preservation in databases. Furthermore, he is an expert in business intelligence and business IT alignment. He currently works as Business Analyst for Berlin Airports.

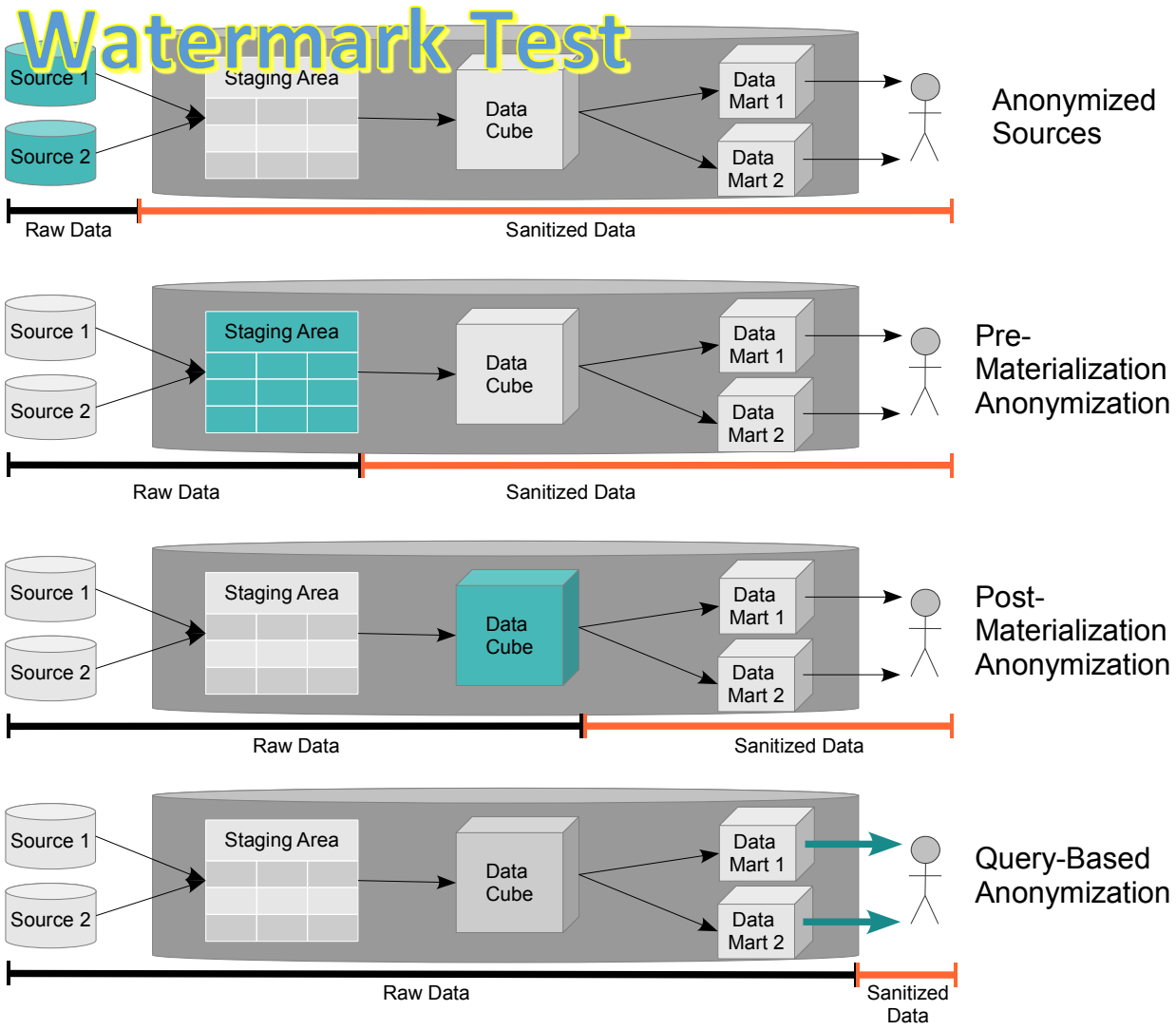

Figure 5: Points of anonymization in data warehouse architectures

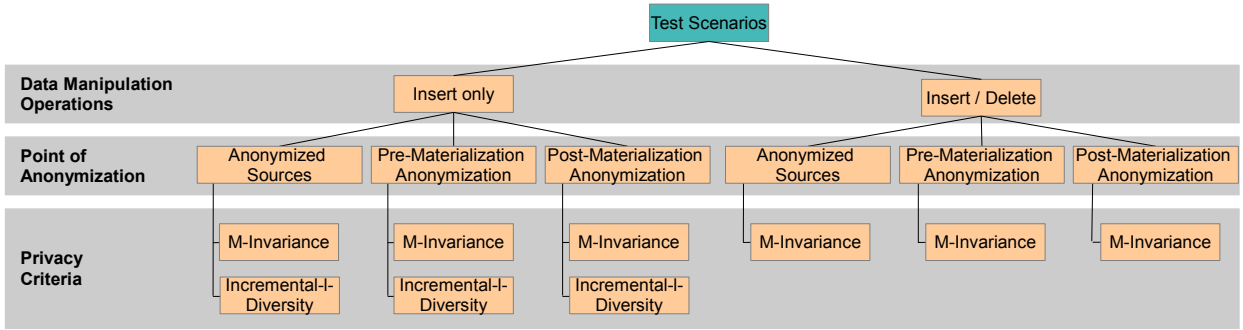

Figure 6: Sample architectures of the experiment

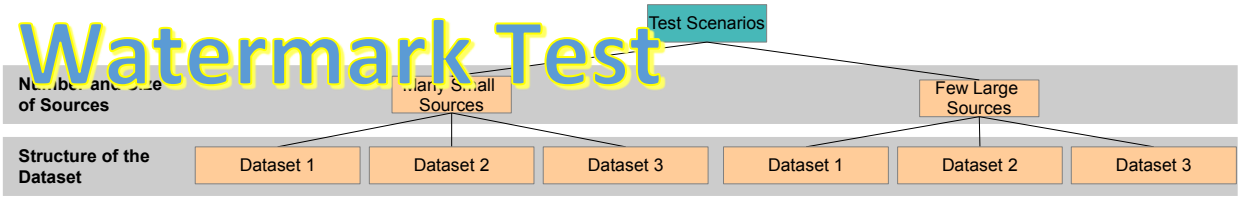

Figure 7: Test scenarios

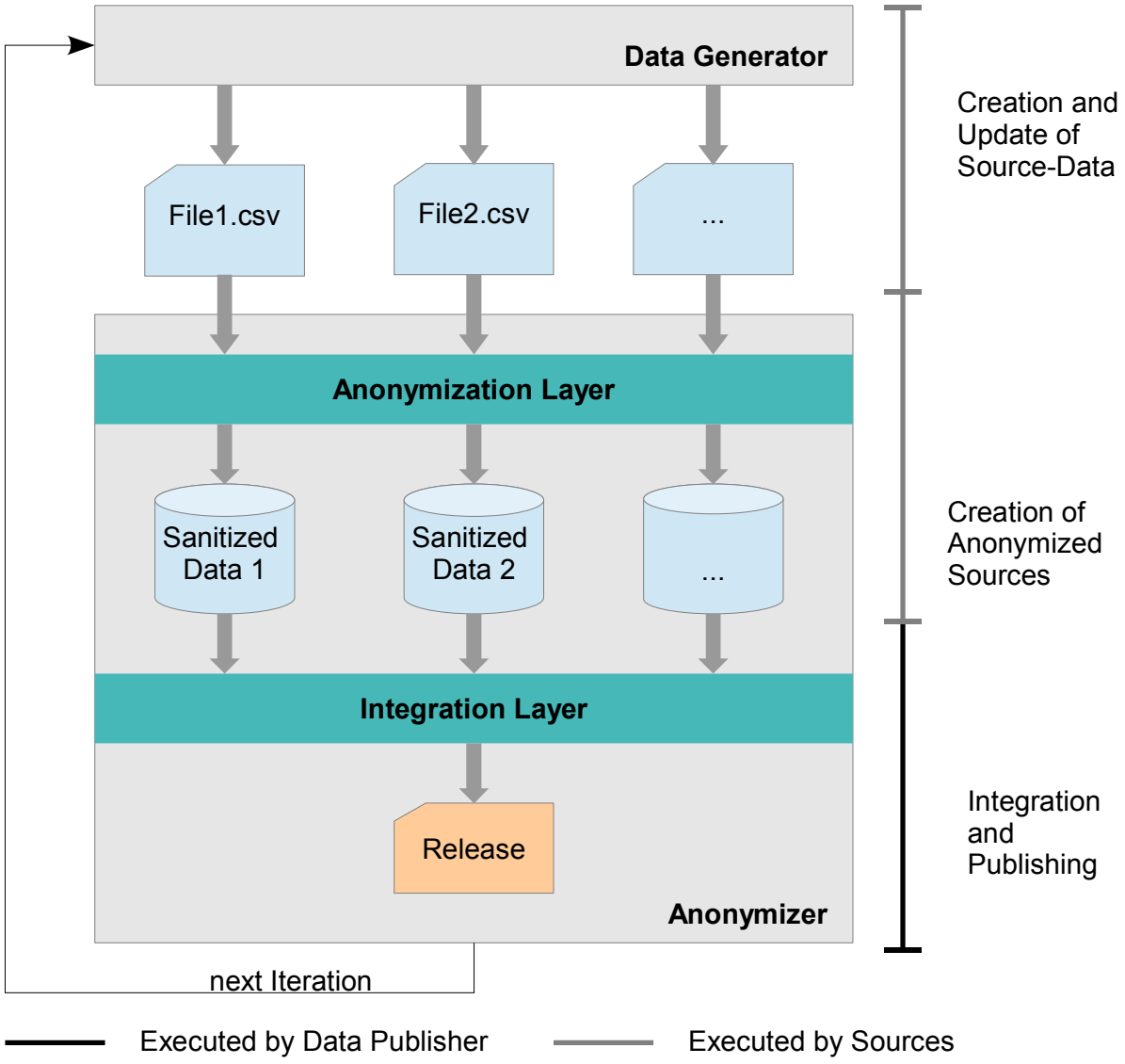

Figure 8: Simulation of Cases 1 and 2 (Anonymized Sources)

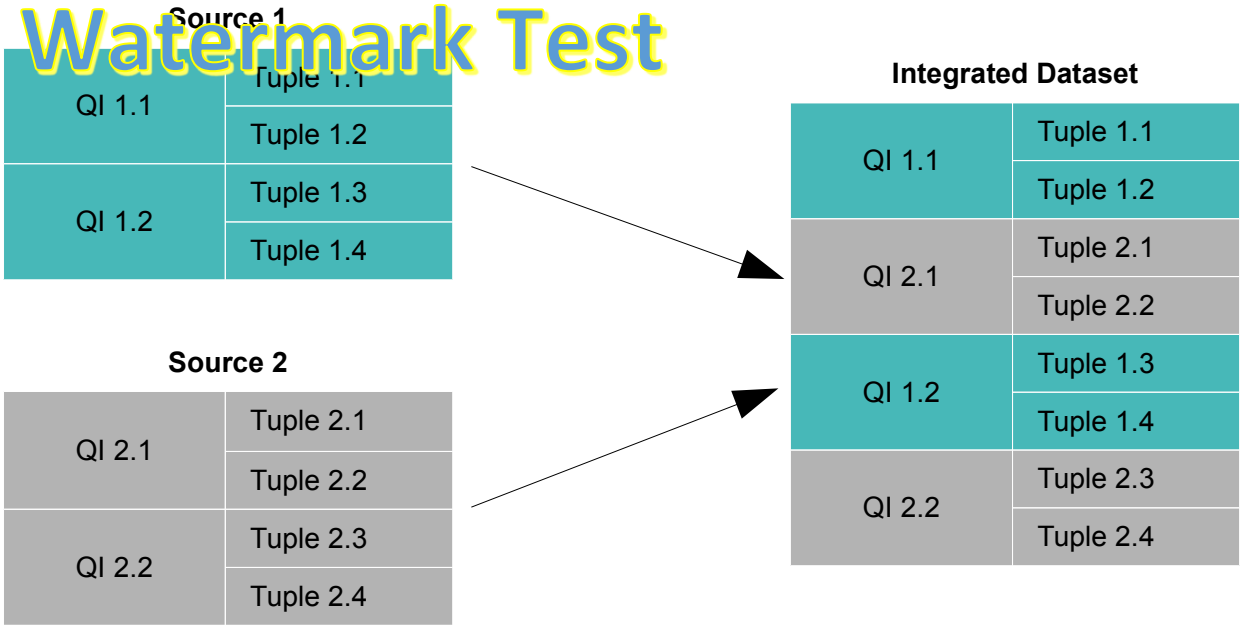

Figure 9: Creation of QI-groups in Cases 1 and 2 (Anonymized Sources)

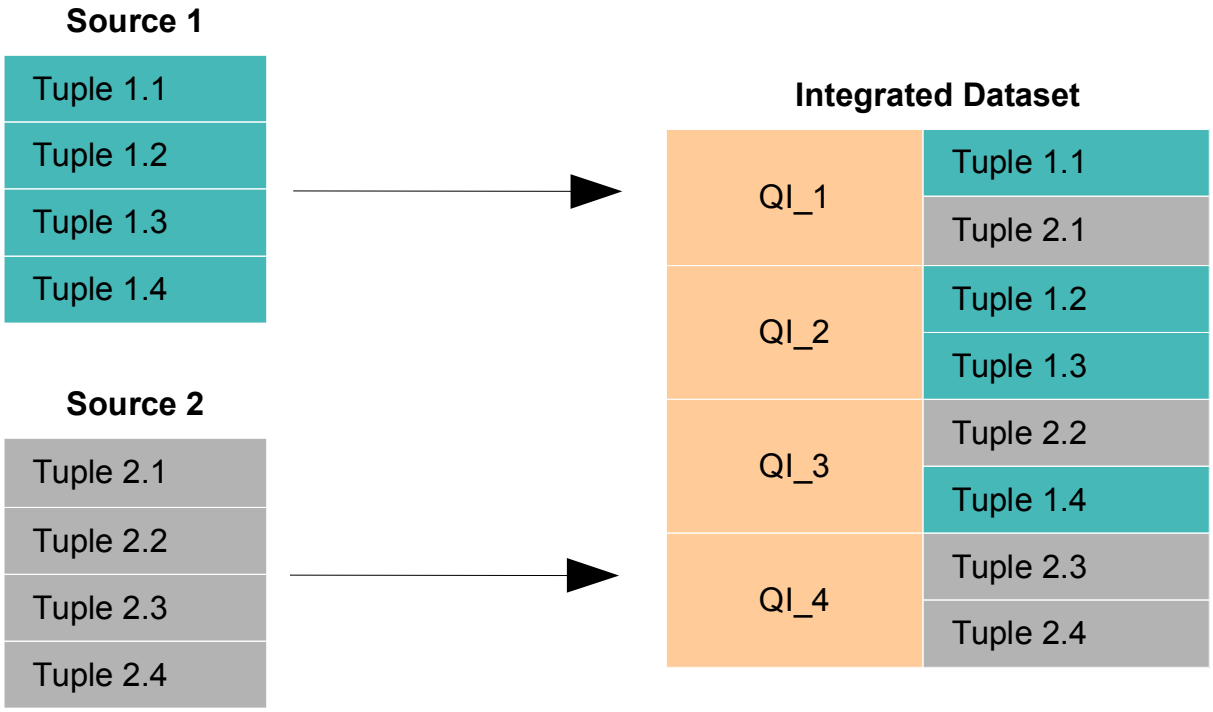

Figure 10: Creation of QI-groups in Cases 3 – 6 (Pre/Post-Materialization)

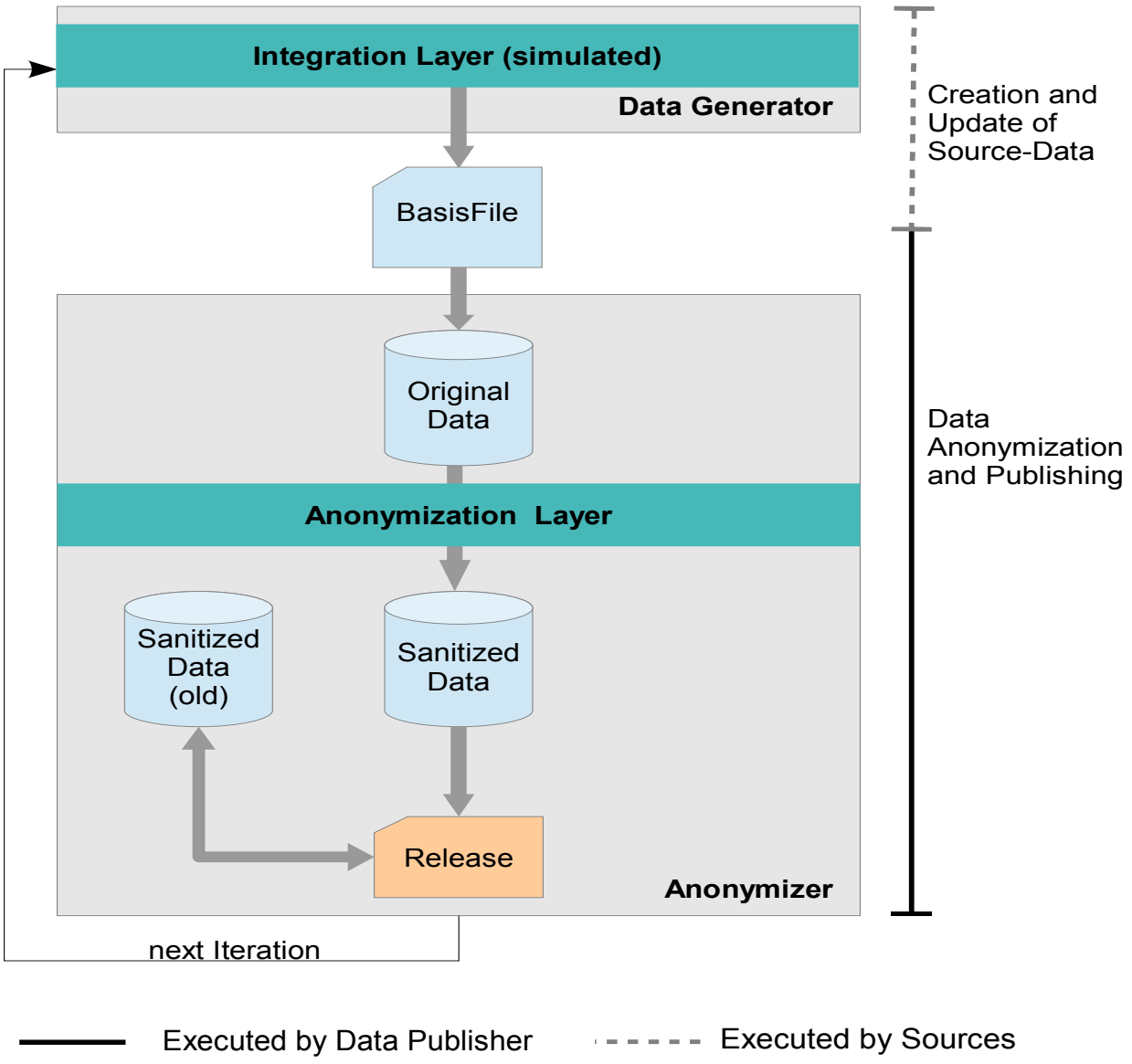

Figure 11: Simulation of Cases 3 and 4 (Pre-Materialization)

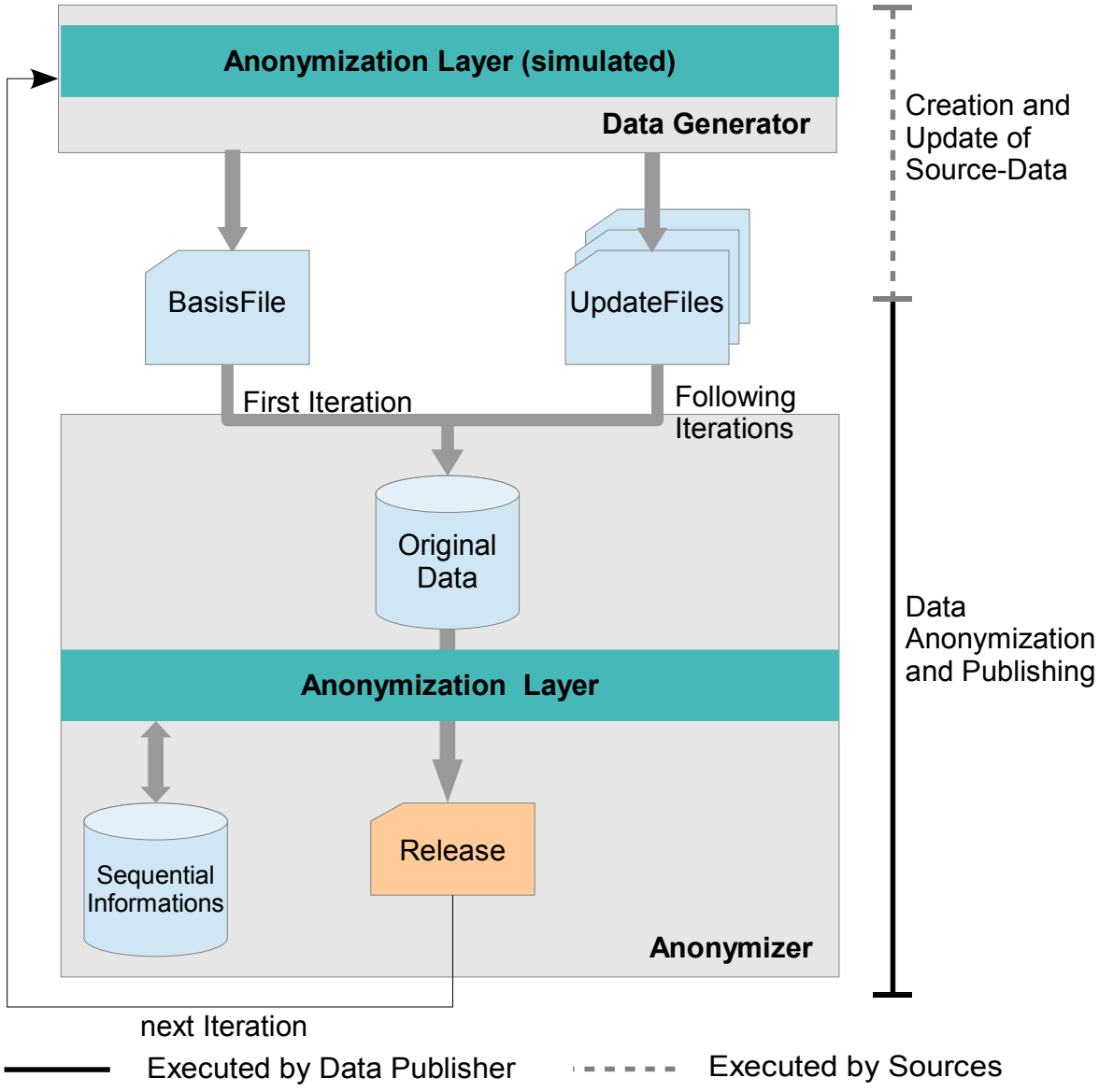

Figure 12: Simulation of Cases 5 and 6 (Post-Materialization)

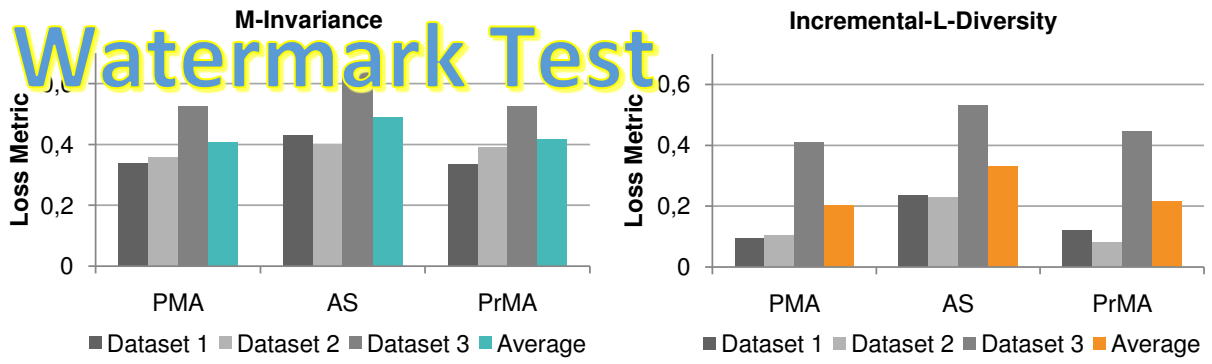

Figure 13: Average Loss Metric (PMA: Post-Materialization; AS: Anonymized Sources; PrMA: Pre-Materialization)

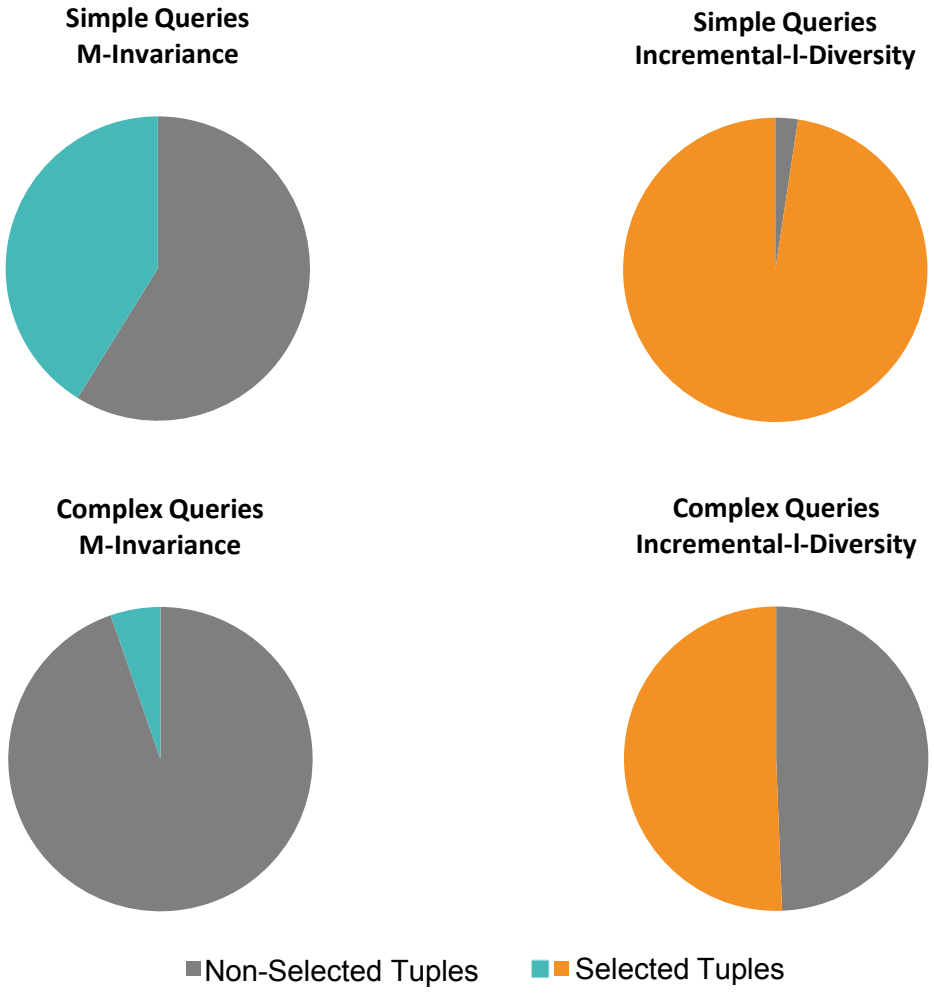

Figure 14: Average selectivity of m-invariant and incremental-l-diverse releases

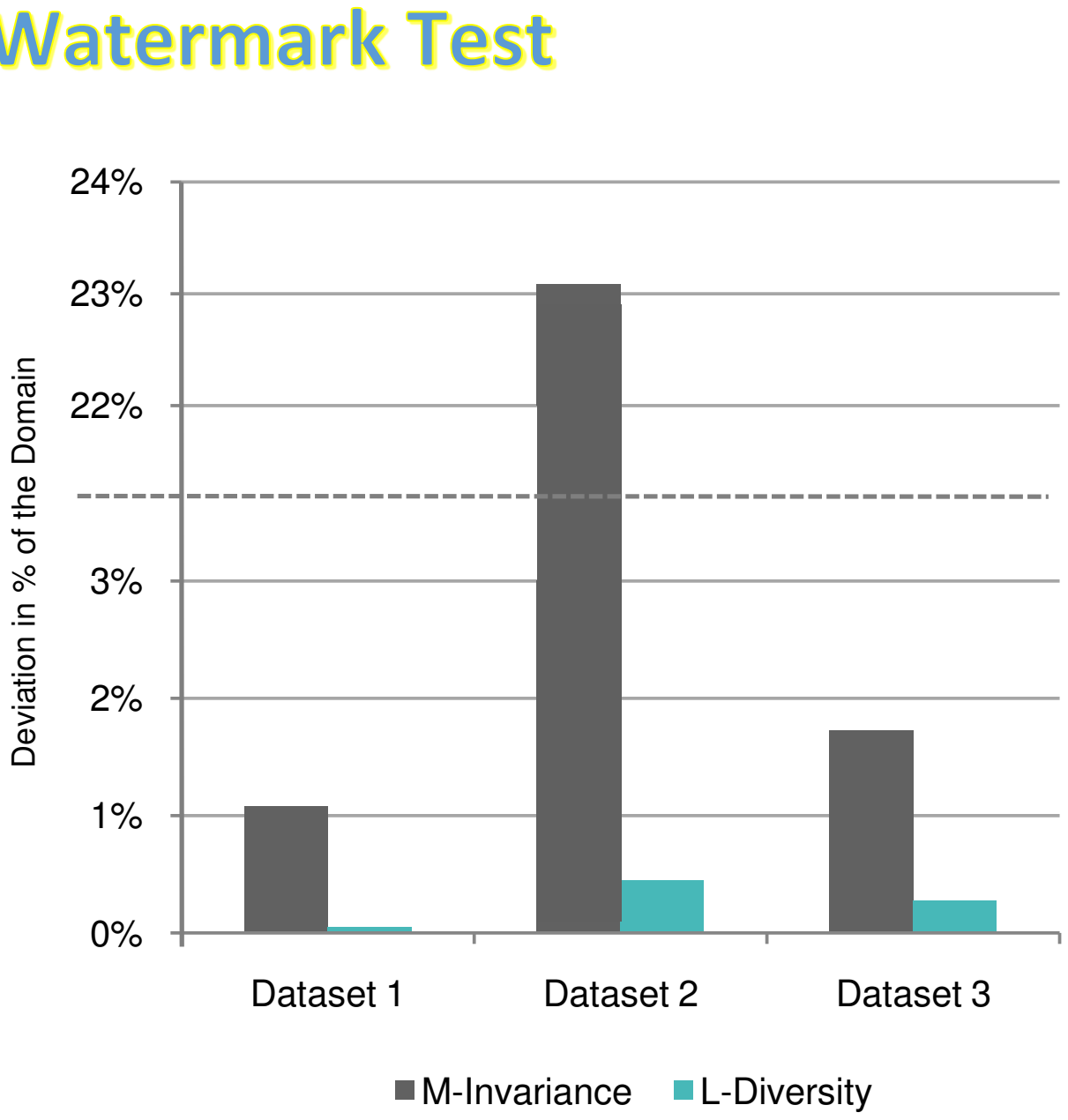

Figure 15: Deviation of average-queries

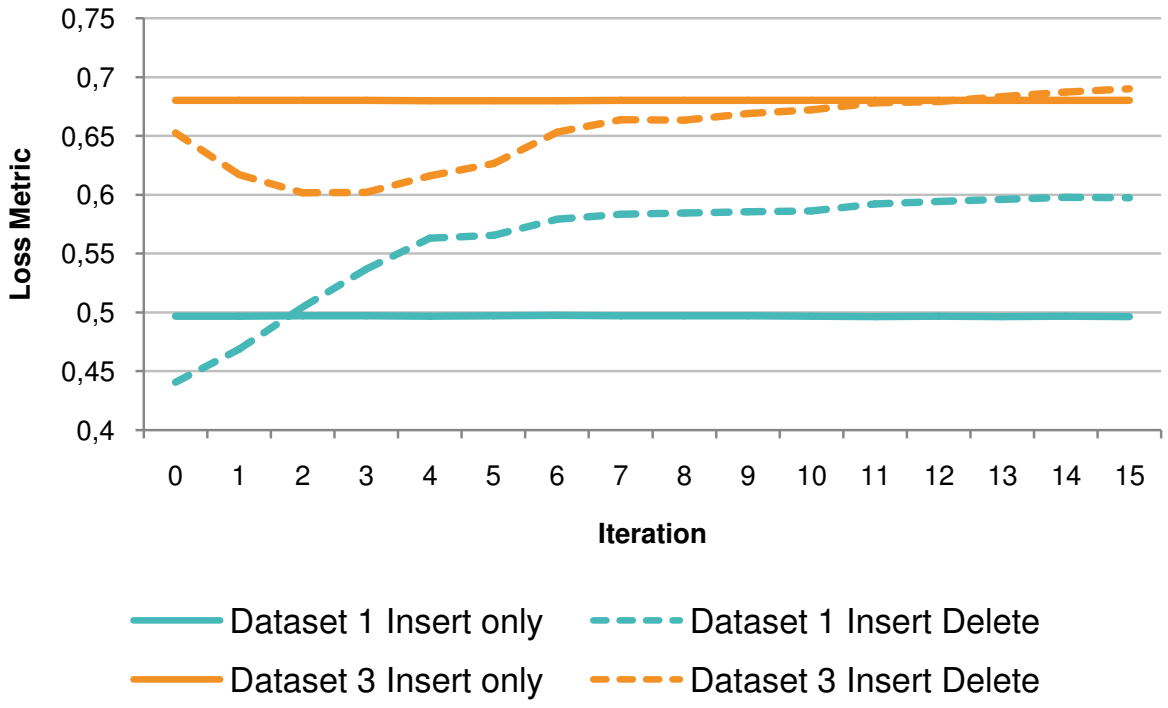

Figure 16: Anonymized sources: Loss Metric development on small-source scenarios

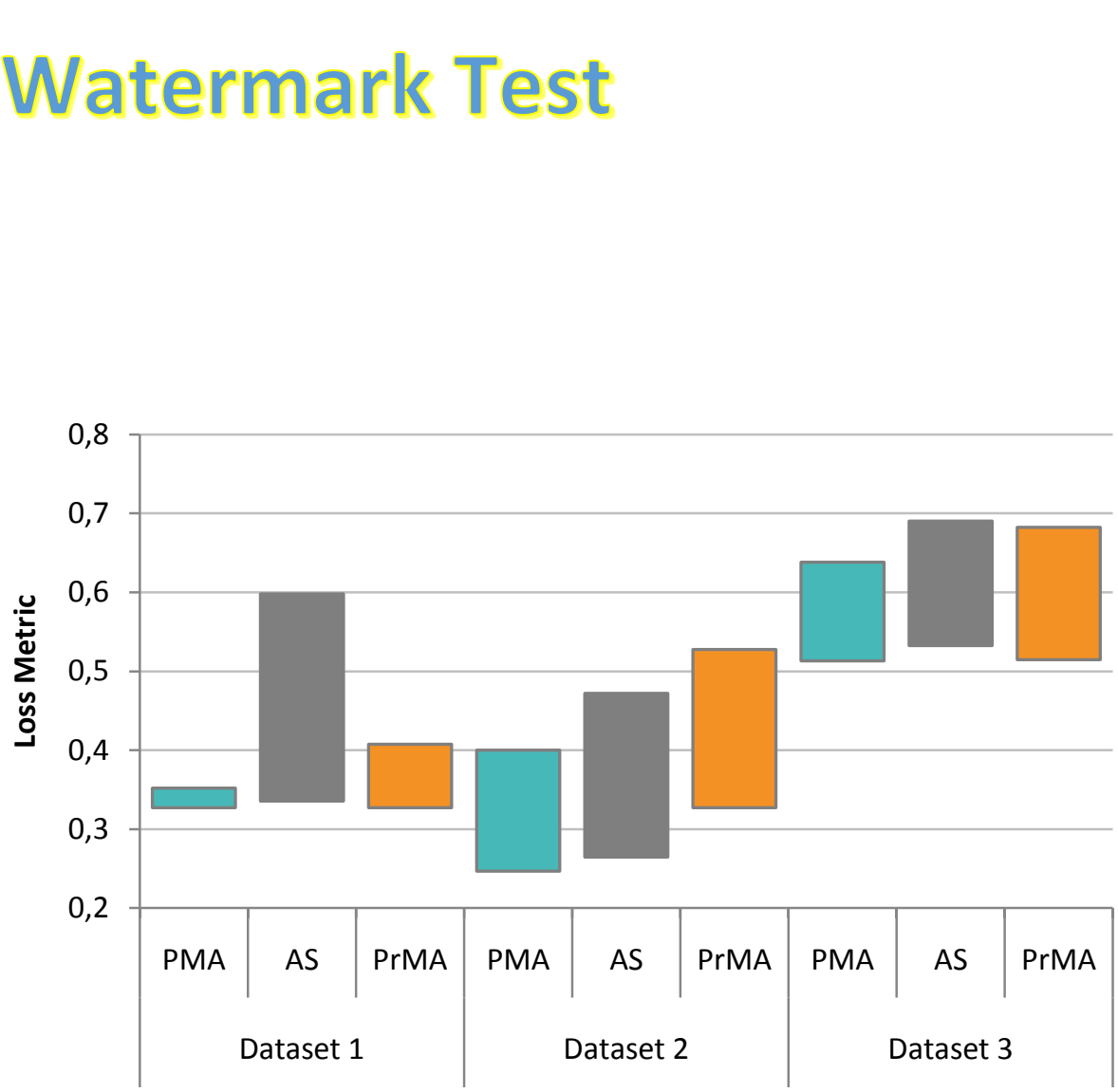

Figure 17: Ranges of Loss Metric across iterations of varied scenarios (PMA = Post-Materialization Anonymization; AS = Anonymized Sources; PrMA = Pre-Materialization Anonymization).

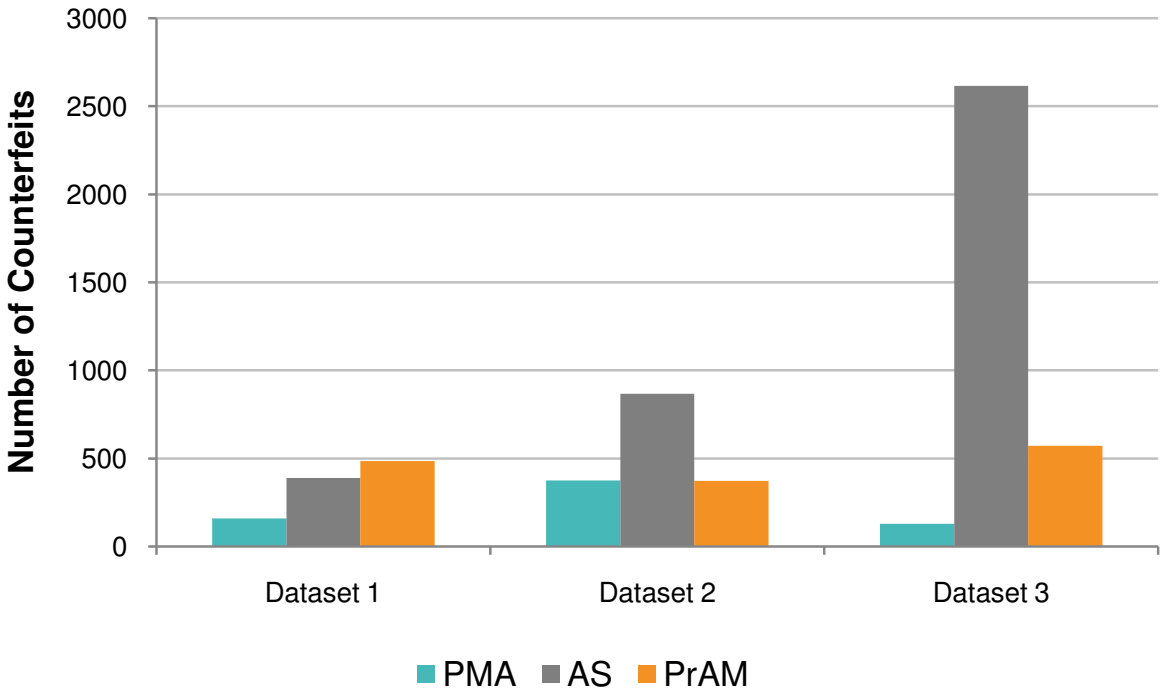

Figure 18: Median number of counterfeit tuples in m-invariant releases

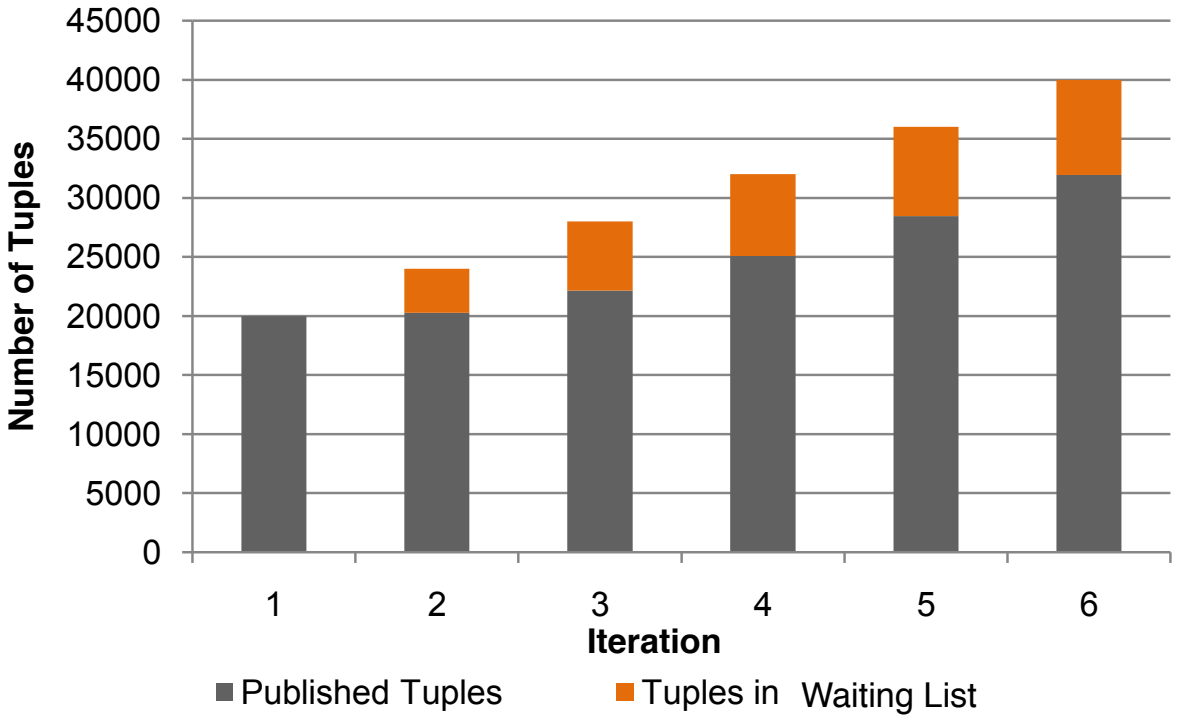

Figure 19: Case 1 – Development of waiting lists (Dataset A – large sources)

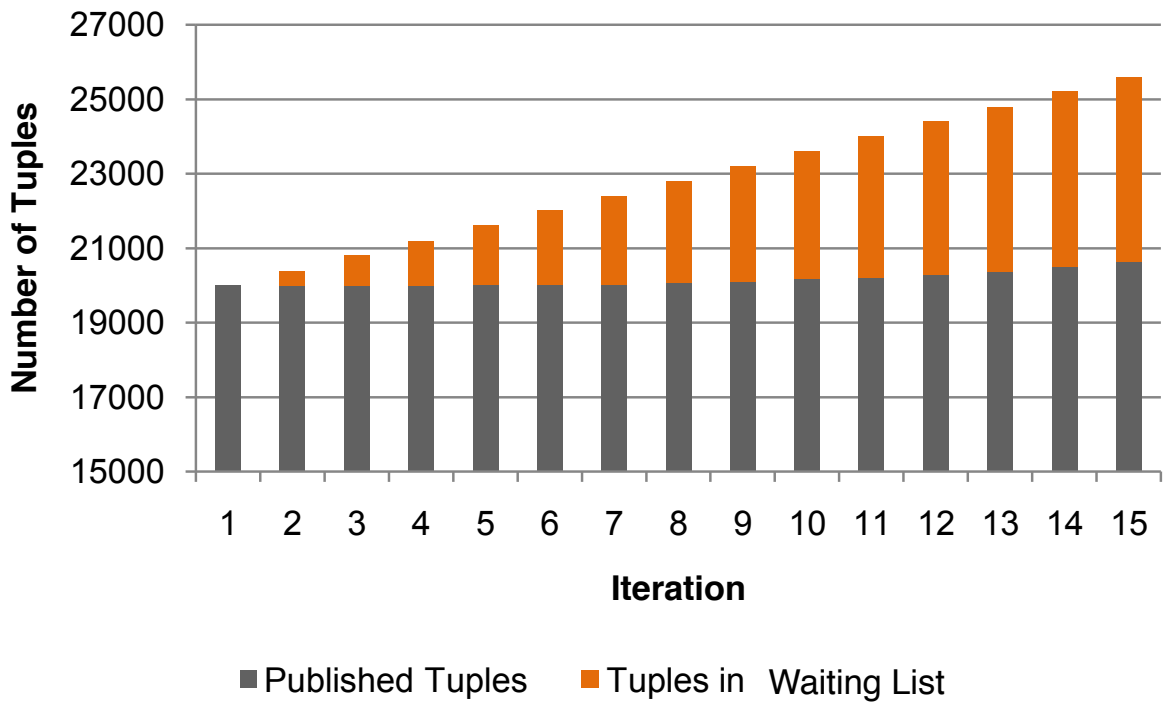

Figure 20: Case 1 – Development of waiting lists (Dataset A – small sources)