SAP HANA goes private – From Privacy Research to Privacy Aware Enterprise Analytics

Stephan Kessler, Jens Hoff
SAP SE
Walldorf, Germany
{stephan.kessler, jens.hoff}@sap.com

Johann-Christoph Freytag
Humboldt-Universität zu Berlin
Berlin, Germany
freytag@informatik.hu.berlin.de

ABSTRACT

Over the last 20 years, the progress of information technology has allowed many companies to generate, integrate, store, and analyze data of unprecedented size and complexity. In many cases, this data is personal data and how it can be used is therefore subject to laws that depend on the specific countries and application domains. For example, the General Data Protection Regulation (GDPR) introduced in the European Union imposes strict rules on how personal data can be processed.

Analyzing personal data can create tremendous value, but at the same time companies must ensure that they remain legally compliant. Unfortunately, existing systems offer only limited or no support at all for processing personal data in a *privacy-aware* manner. Approaches that have emerged from the academic and industrial research environments need to be integrated into large systems (like enterprise systems) in a manageable and scalable way. In many IT environments, it is also desirable and necessary to combine and to integrate personal data with other (non-personal) data in a seamless fashion.

In this paper, we present the first steps that SAP has taken to provide its database management system SAP HANA with privacy-enhanced processing capabilities, referred to in the following as SAP HANA Data Anonymization. Various goals on both the conceptual and technical levels were followed with the aim of providing SAP customers *today* with an *integrated* processing environment for personal and non-personal data.

PVLDB Reference Format:
Stephan Kessler, Jens Hoff, Johann-Christoph Freytag. SAP HANA goes private – From Privacy Research to Privacy Aware Enterprise Analytics. *PVLDB*, 12(12): 1998-2009, 2019.
DOI: https://doi.org/10.14778/3352063.3352119

1. MOTIVATION

Since the beginning of this millennium, the progress of information technology has allowed many businesses, scientific institutions, and non-commercial entities to generate, to integrate, to store, and to analyze data of unprecedented size and complexity. Looking at companies like Google, Facebook, or SAP, the volume of data managed by their systems has increased by orders of magnitude. Often, such data volumes are automatically generated by a variety of input devices and sensors, such as mobile phones, health care devices, scanners, cameras, or RFID tags. In many cases, the data collected might relate to individuals and describe various aspects of their behavior, condition, or actions. This personal data can be used to make predictions about different aspects of the individuals in the future. This could include predicting which products they will buy, determining the candidate that they are likely to vote for in the next election, or the likelihood that they will develop a certain disease within a certain time.

Many commercial enterprises, governmental organizations, and research institutes have built *data lakes*. Particularly the business and health care domains store data about individuals as described above. This personal data is often protected by national or international laws, such as HIPAA [8], the European General Data Protection Regulation GDPR [16], and others [1]. Often, these legal restrictions discourage enterprises and organizations from making use of the personal data which they have collected. Therefore, these data lakes might be only used partially or not at all. Also, new privacy laws like GDPR, which came into effect in 2016, have strengthened the privacy rights of individuals, therefore further increasing the requirement on IT-based solutions for processing personal data in a legally compliant manner.

Since the end of the 90s, there has been a growing number of research activities that have produced concepts and research solutions to address various privacy challenges, particularly to enable the analysis of personal data while respecting the privacy of individuals.

To enable such analysis tasks, this paper considers approaches that anonymize data, see Recital 26 in the GDPR: "... This Regulation does not therefore concern the processing of such anonymous information, ..." [16].

For example, the company Aircloak provides a tool that implements Diffix [3] based on the anonymization principle [6]. However, their implementation acts as a database proxy and therefore lacks the tight integration with enterprise systems that is necessary for a holistic approach to privacy protection. On the other hand, Google's RAPPOR is an example for an application-oriented privacy solution when collecting personal data from users over the Inter-

Proceedings of the VLDB Endowment, Vol. 12, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3352063.3352119

net [5]. However, enterprise-ready implementations of such solutions are still hard to find and, in many cases, are not applied to production data.

The research prototype PINQ by Microsoft integrates Differential Privacy into Microsoft's LINQ runtime environment [4, 23], therefore closely resembling our own goals and approach [14]. Still, to the best of our knowledge, SAP HANA is the first data management *product* that addresses data privacy through an integrated and domain independent approach. Our solution goes beyond security measures such as authentication, authorization, and data masking/redaction [15].

In the enterprise world, a growing number of SAP customers increasingly want to analyze personal data without violating existing privacy laws. They are requesting solutions that operate in SAP's enterprise system in a seamless manner. For this reason, there is a need to integrate privacy-enhancing techniques into SAP's business data platform HANA. SAP HANA Data Anonymization (DA) allows us to provide our customers with an easy-to-use solution that addresses the technical and non-technical (legal, process-oriented) challenges of today's complex data management world in a holistic manner.

We emphasize that the deep integration of privacy-enhancing techniques into SAP HANA provides a maximum of data protection since data is anonymized and stored in one and the same place.

Based on our motivation, the main contributions of this paper are as follows:

1. We present our design and implementation approach to seamlessly integrating privacy-enhancing techniques into SAP's enterprise system, as available to our customers in SAP HANA 2.0 SPS03 since May 2018.

2. We describe our implementation framework, which is independent of the privacy-enhancing methods chosen, therefore allowing additional methods to be integrated in the future without much effort.

3. We demonstrate how our holistic approach paves the way for a trustworthy and reliable solution that allows enterprise customers to handle privacy-enhanced data.

The paper is structured as follows: Section 2 discusses the main challenges when implementing privacy-enhancing solutions into an existing data-centric enterprise system. In Section 3, we provide a conceptual overview of the solutions we selected for implementation in SAP HANA DA. Section 4 describes the current implementation and discusses its pros and cons while Section 5 shows how to apply it to a data set and evaluates its impact on accuracy and performance. Finally, Sections 6 and 7 summarize our contributions and upcoming enhancements in SAP HANA DA.

For the remainder of this paper, we use a running example to explain the anonymization workflow and demonstrate the approach taken in SAP HANA DA.

EXAMPLE 1. *ACME IT, our example company, maintains a human resource database (HRDB) within its enterprise system. It consists of one table that stores employee-related personal data such as salaries (see Table 1 for synthetic sample data). The privacy research literature calls a table that records personal data a microdata table.*

Data Scientists at ACME IT plan to access HRDB for analyzing salaries, for example, the correlations with demographic properties such as the time spent with the company or location. Due to privacy restrictions, they are not allowed to access the original data, but they still want to get an accurate analysis. They approach the human resource (HR) department, which is the responsible Data Controller, to get access to HRDB. In addition, they also need to inform ACME IT's Data Protection and Privacy Officer (DPPO), who is responsible for privacy compliance. The DPPO must agree on how HRDB can be accessed.

Providing data access for a Data Scientist requires mechanisms beyond those based on authorization. They need to be able to access the data without revealing individuals' details. Simply removing identifiers such as names in Table 1 does not offer any privacy protection because unique combinations of the remaining attributes might allow individuals to be re-identified. Therefore, we need privacy-enhancing methods such as anonymization to satisfy the requirements of both the stakeholders involved (that is, the Data Scientists and the employees). Also, it is in the Data Controller's interest to avoid data duplication while at the same time maintaining privacy protection in a way that is transparent to the DPPO. In the remainder of this paper, we explain how we satisfy the diverging requirements within SAP HANA DA.

Figure 1 illustrates the process of accessing personal data using a Privacy View and authorization. A privacy view is the concept we use to provide privacy-enhancing methods in SAP HANA DA, as defined and described later in the paper.

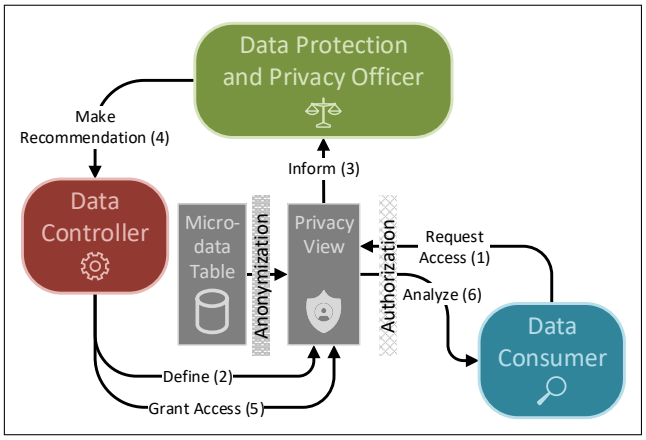


Figure 1: Process of providing access to personal data.

This simplified but realistic model shows the different stakeholders as introduced in Example 1, that is, the human resource department as the Data Controller, the DPPO, and the Data Scientists as Data Consumers:

- **Step 1:** The Data Consumer requests access to personal data for data analysis, thus requesting the definition of a Privacy View.
- **Step 2:** The Data Controller defines the Privacy View.
- **Step 3:** The DPPO is informed about how the Privacy View has been defined.

Table 1: Sample microdata from ACME IT's HRDB table. It consists of the key attribute *Tuple ID*, the identifier *Name*, several quasi-identifying attributes, and the sensitive attribute *Salary*, which records personal data.

Tuple ID	Name	Start Year	Zip Code	Gender	Education	Salary (in $)
1	Murril	1985	7204	m	HS-grad	139,051
2	Roosa	2005	4206	m	College	173,834
3	Jaquess	1989	6104	f	Assoc-acdm	122,419
4	Willinsky	2004	5107	m	College	158,726
5	Wheetley	2000	6004	m	Bachelor	137,879
6	Sander	2003	7204	m	Bachelor	158,248
7	Ishii	1988	6104	m	HS-grad	143,588
8	Szczurek	1987	4205	f	Prof-school	117,197
9	Nahass	1981	4106	f	9th-12th	181,546
10	Bargerstock	1995	5206	f	Doctorate	182,455
...	...	...	...	...	...	...

Key | Identifier | Quasi-identifiers | Sensitive Attribute

- **Step 4:** The DPPO recommends one of the following:
  1. Go back to Step 2 if he or she does not agree with the definition of the Privacy View.
  2. Continue with Step 5.
- **Step 5:** The Data Controller grants the Data Consumer access to the Privacy View.
- **Step 6:** The Data Consumer can perform the requested data analysis.

As presented in the following sections, our extensions in SAP HANA provide the necessary means to implement such a process in an enterprise environment. This approach guarantees that personal data is processed only by authorized Data Consumers in a privacy-enhanced fashion through Privacy Views that are under the control of the responsible Data Controller and DPPO.

2. PRIVACY CHALLENGES

When integrating a new concept into a large and complex software system like the SAP HANA database management system (DBMS), it is important to understand and to balance the various existing trade-offs this extension entails. In the following, we list the challenges and the goals that we wanted to achieve when extending SAP HANA with privacy-enhancing techniques.

Challenge 1. *Determining the appropriate level in the enterprise architecture for implementing privacy-enhancing techniques:* Rather than implementing a library that could be used by applications, or by providing a separate component, we decided to integrate privacy-enhancing techniques deeply within the DBMS kernel. Our choice allows us to combine the privacy extensions with other existing concepts such as views, therefore avoiding code replication and reduced system performance. At the same time, our extension seeks to keep the changes to the overall system environment to a minimum, with a maximum of flexibility and extensibility for future enhancements.

Challenge 2. *Transparent use of privacy-enhancing methods:* Despite the deep integration within the SAP HANA database engine, our approach should also be understandable to non-technical users such as DPPOs, who are responsible for ensuring privacy on the enterprise level. Therefore, we decided to base our implementation on well-known and well-researched privacy techniques developed in academia to ensure maximum transparency.

Challenge 3. *Choices between different privacy-enhancing techniques:* Based on the different privacy and utility requirements of applications and domains, users should have the flexibility to choose between several privacy-enhancing techniques and their parameters. For example, there are different requirements from a legal point of view when processing personal data in health care or within the marketing domain.

Challenge 4. *Follow SAP HANA's vision for data management:* Our implementation choice should be based on the data management principles that guided the implementation of SAP HANA so far. We therefore decided to perform privacy processing of existing data in real-time rather than storing privacy-enhanced data redundantly [21]. Our implementation choice ensures that current data is always accessed for privacy enhanced processing.

3. PRIVACY CONCEPTS IMPLEMENTED

Within the SAP ecosystem, many enterprise system applications and thus customers rely on SAP HANA as their central data management system. Therefore, extending SAP HANA with privacy-enhancing methods will be immediately beneficial to all of them, thus satisfying *Challenge 1.*

Furthermore, applying privacy-enhancing methods as soon as the data is fetched for processing on the *server side* reduces the risk of privacy breaches. In contrast, implementing privacy-enhancing methods in an application or at the client side leads to unmaintainable solutions with higher risk for privacy breaches.

To limit the number of new concepts, we decided to exploit the *view concept* of relational DBMSs to define and access anonymized data. By using views, data duplication

can be avoided and access to *up-to-date data* provided, thus addressing *Challenge 4*. The view concept is well-established and well-understood in DBMSs. Therefore, we immediately benefit from existing optimization and integration methods. When such a view is created, referred to as a ***Privacy View*** in the following, this initially only results in metadata being created in the DBMS to parameterize the chosen privacy-enhancing method. When queries are subsequently run on these *non-persistent* Privacy Views, SAP HANA DA decides at execution time whether the existing data anonymization still accommodates the specified privacy requirements if changes have been made to the corresponding source tables. If no changes have occurred, the result of anonymization is always the same, that is, it is *reproducible*.

In general, there are two approaches to performing data transformations for privacy protection:

- Transform the data first before querying the result, or
- Query the data first before transforming the query result.

In SAP HANA DA, we chose the former approach so that the result of anonymization is independent of the query, which is a more natural fit with the view concept. As such, a Privacy View acts like a membrane, ensuring that only anonymized data leaves the lowest layers of SAP HANA DA before queries perform data operations. We are convinced that our choice of using the well-known view concept for anonymized data is beneficial and comprehensible to SAP customers, thus addressing *Challenge 2*. In contrast, Aircloak [3, 6] and PINQ [14] follow the latter approach of anonymizing the result of queries.

Furthermore, the metadata generated during view definition allows SAP HANA DA to generate anonymization reports. Even users without extensive technical expertise, such as the DPPO, can understand the reports and the amount of privacy protection given by a Privacy View, therefore further supporting *Challenge 2*.

So far, the literature distinguishes between two classes of privacy-enhancing methods: grouping-based approaches and perturbation-based approaches [9]. *k-Anonymity* as described in [22] was the first approach based on grouping by distinct combinations of attribute values to prevent linkage attacks. Perturbation-based methods disguise sensitive data by adding noise to the original values. The most popular approach is Differential Privacy, as developed by Dwork et al. [4, 23], providing provable privacy guarantees. As of May 2018, SAP HANA DA supports both privacy-enhancing methods, that is, *k-Anonymity* and *Local Differential Privacy* [23], as discussed in Subsection 3.2. Both methods are well understood and well researched by the scientific community. Beyond that, Privacy Views together with the necessary metadata support future extensions for implementing almost any privacy-enhancing technique. Customers can choose between them by configuring the parameters of the view definition accordingly, thus satisfying *Challenge 3*.

Choosing the right privacy-enhancing technique together with the appropriate parameters could be simplified by DP-Comp, an extensible platform for implementing, evaluating, and comparing Differential Privacy methods targeting both developers and users in a prototypical environment [7].

Figure 2 illustrates our approach. The *Data Controller* is responsible for the microdata table R, which stores personal

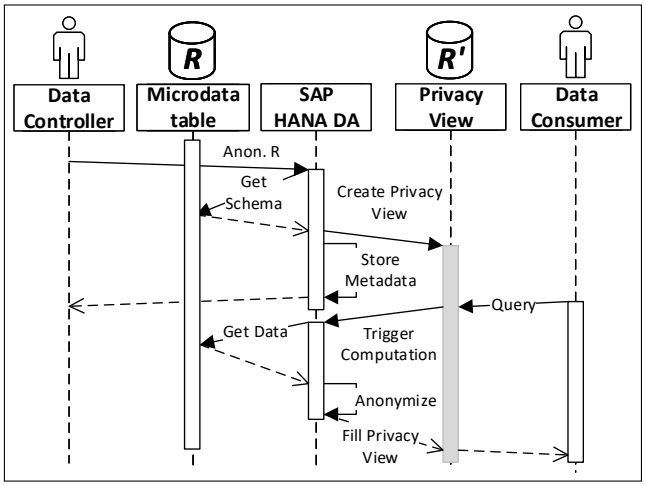


Figure 2: Anonymization sequence in SAP HANA DA.

(sensitive) data. A *Data Consumer* asks for permission to access R. However, the Data Consumer is only allowed to access an anonymized version of R. To provide anonymized access, the Data Controller uses SAP HANA DA to create a Privacy View R'. To do so, the DBMS retrieves the metadata of R, such as columns and data types involved. In addition, SAP HANA DA stores certain information, such as the anonymization method configured by the Data Controller. Authorization methods ensure that the Data Consumer can access R' but not R. Once the Data Consumer queries R', SAP HANA DA retrieves tuples from R and applies a privacy-enhancing method, as specified by the metadata of the Privacy View, before the query applies its operations.

In the following subsections, we briefly describe the conceptual modifications to *k-Anonymity* and *Local Differential Privacy* to adapt both to the specific requirements of SAP HANA DA. Section 4 then describes more details of the implementation.

3.1 k-Anonymity

We decided to implement *k-Anonymity*, one of the oldest and best understood privacy-enhancing techniques, as one of the two initial choices for anonymization into SAP HANA DA. Given an integer k, a set of quasi-identifying attributes (QIDs), and a corresponding collection of generalization hierarchies (GHs), *k-Anonymity* transforms the content of a microdata table into one whose QID values are generalized according to the GHs to achieve k-anonymous subsets of tuples. Thus, each subset contains at least k tuples with the same combination of QID values. We use *(global) full-domain recoding* for generalizing values of the QID attributes [11]. That is, all values of each QID attribute are generalized to the same level in the GH even if there exist disjoint subsets of tuples that would satisfy *k-Anonymity* at a lower level of the GH. As a consequence, we represent the levels of anonymization as a vector of generalization levels for fast processing in our implementation. Furthermore, we argue that the anonymized data resulting from global full-domain recoding is easier to grasp compared to other recoding schemes. Since the current version of SAP HANA DA is our first offering with anonymization features

in SAP's enterprise-level software, its use should be easily comprehensible for our customers.

EXAMPLE 2. *Creating in ACME IT's HRDB a k-anonymized view on employee data, one has to provide generalization hierarchies for all QIDs. For example, values of attribute Start Year range from '1977' to '2018'. They may be generalized to intervals '1976-1980', '1981-1985', '1986-1990', '1991-1995', '1996-2000', '2001-2005', and '2006+' on the first level, and to '*' on the second level. Note that '2006+' spans thirteen years, whereas the other intervals span only five years. Thus, with (global) full-domain recoding the values '1992' and '1996-2005' never appear together in the column Start Year. Figure 3 shows a tree-like representation of the generalization hierarchy.*

To satisfy *Challenge 4*, changes in a micro table must be reflected in the corresponding Privacy Views immediately (in real-time). Therefore we designed a strategy for *k-Anonymity* to handle those changes adequately:

- The query that accesses a newly defined Privacy View using *k-Anonymity* first triggers an action that determines the generalization level for each QID attribute to compute a k-anonymized instance for the Privacy View. The resulting generalization levels are fixed for future uses. Determining the minimal generalization level might be quite time consuming as it heavily depends on the size of the micro table, the number of QID attributes, and on other properties. We refer to Subsection 4.1.1 for more discussions.
- The first and all subsequent queries using this Privacy View then use the computed generalization levels to produce the anonymized instance for the Privacy View as long as *k-Anonymity* is satisfied. Transforming the original tuples into their anonymized form incurs little overhead; its computational complexity depends linearly on the number of records. (See Section 5.3 for details.)
- If updates to the corresponding microdata table cause a violation of *k-Anonymity*, then the Privacy View does not produce any resulting (anonymized) tuples. If further changes cause the instance of the Privacy View to satisfy *k-Anonymity* on the existing generalizations, then the Privacy View produces anonymized tuples again.

We chose such pragmatic strategy despite some drawbacks which we discuss further in Section 6.

3.2 Local Differential Privacy

The general goal of *Local Differential Privacy* (*LDP*) is to disguise individual attribute values [23]. For this reason, *LDP* adds noise to values of (numeric) attributes derived from a probability distribution. Choosing *LDP* satisfies *Challenge 2*. To instantiate *LDP* in SAP HANA DA, the DBMS expects a real-valued ε, a numeric target column *Noised Column*, and a *Sensitivity* value as parameters. Parameter ε determines the strength of the privacy protection and therefore the amount of noise added to the original attribute values. Parameter *Sensitivity* scales the added noise with respect to the existing value range of the *Noised Column*.

SAP HANA DA generates noise in a reproducible manner by drawing uniformly distributed random numbers p from the interval $[0, 1)$ and by transforming them to a Laplace distribution. Equation 1 shows the inverse cumulative Laplace distribution with an expectation value of 0 as used in SAP HANA DA:

$$F^{-1}(p) = \frac{\text{Sensitivity}}{\varepsilon} \operatorname{sgn}\left(p - \frac{1}{2}\right) \ln\left(1 - 2\left|p - \frac{1}{2}\right|\right). \quad (1)$$

Our implementation of *LDP* guarantees that inserting tuples into the microdata table does not change the (noised) values of the existing tuples in the Privacy View. Similarly, deleting tuples in the microdata table does not impact the remaining ones. Different queries accessing the same Privacy View access the same set of tuples with the same *LDP*-enhanced column always having the same values. Subsection 4.1 explains details of our implementation.

EXAMPLE 3. *Consider the tuple with Name value 'Roosa' stored in the HRDB (Table 1) which shows a Salary value of* \$173,834. *In a Privacy View using LDP this Salary value shows up as* \$918,159 *based on added noise. Consequently, all aggregate functions on Salary using this Privacy View will perform the computation using the privacy enhanced values. Single LDP-perturbed record values no longer hold any significance, however aggregates like averages still do.*

We emphasize that *LDP* fits more naturally our goals and our implementation context than DP for the following two reasons. First, Wang et al. [23] developed *LDP* to disguise *individual* attribute values first *before* they are accessed by query operators for evaluation. In contrast, DP disguises the result of queries, see Section 4. Second, *LDP* avoids the hassle and overhead of maintaining a privacy budget as in PINQ [14]. With every additional query the amount of noise added to the query results increases. Thus, utility decreases and at the same time there is no guarantee that the result is reproducible. Finally, with a limited privacy budget the number of queries that could be executed is limited. Nevertheless, PINQ provides an elegant extension of the LINQ environment with DP. In contrast, SAP HANA DA focuses on an extensible framework for including increasingly more complex anonymization methods in the future.

4. FROM CONCEPTS TO IMPLEMENTATION

This section presents the extensions to SAP HANA with *k-Anonymity* and *LDP* by discussing several implementation details. Figure 4 shows the main components and dependencies of our privacy enhanced architecture while Figure 5 shows the sequence of operations when queries access a Privacy View.

One of our main goals for the extension of SAP HANA was to minimize the changes in the system and to keep the use of privacy-enhancing methods as simple as possible for the user. Therefore, we integrated both *k-Anonymity* and *LDP* for the first version of our extension as a configurable operator into the SAP HANA Calculation Engine (CE) [19]. In SAP HANA CE, a user models a data analysis task (data cube) as a data flow graph represented by a directed acyclic graph (DAG) with nodes as operators. The privacy-enhancing methods are integrated as new operator nodes – called *privacy operators* in the sequel – into the data flow

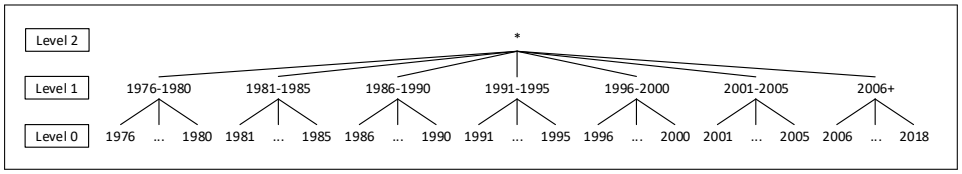


Figure 3: Generalization hierarchy for attribute *Start Year*.

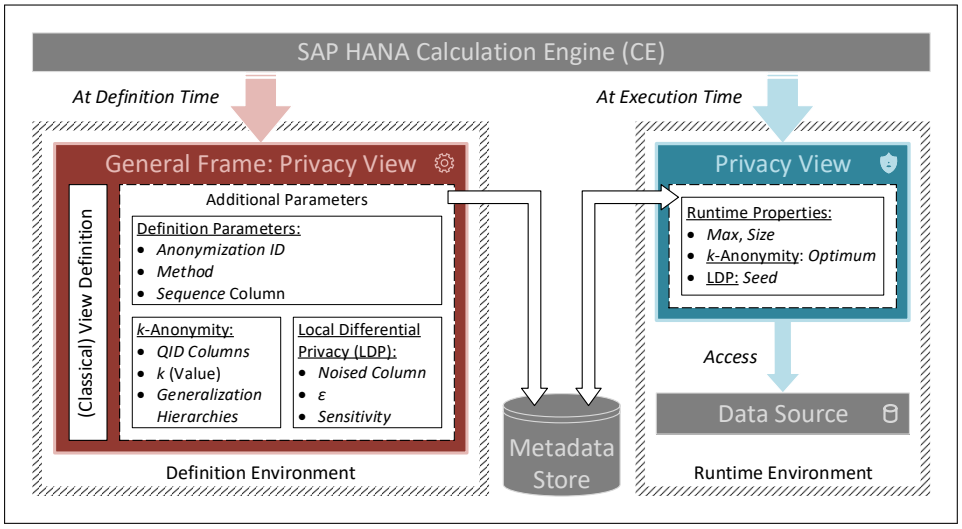


Figure 4: Current architecture of anonymization features in SAP HANA DA.

graph taking an arbitrary subgraph as input and producing a set of anonymized tuples as output.

The SAP Web IDE for SAP HANA contains a graphical tool that allows users to parameterize operator nodes including the newly defined privacy operators. Only in the context of this paper and for simplicity reasons, we call a data flow graph in SAP HANA CE a *Privacy View* if it consists of one privacy operator as root node and a single operator node accessing one microdata table. In general, SAP HANA CE enables users to define more complex queries (DAGs) consisting of multiple anonymization nodes, joins, and unions.

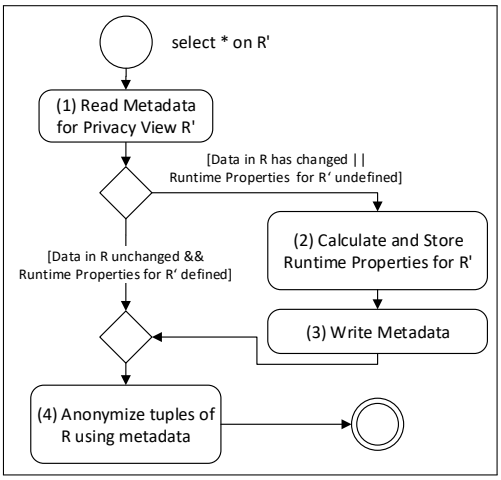


Figure 5: Activity Diagram for a query when accessing Privacy View R' defined on R.

Our extension in SAP HANA DA fully supports the concept of Privacy Views with different anonymization methods as described in Section 3. Most importantly, the metadata for a Privacy View contains all relevant information to transform tuples of a microdata table R into an anonymized set of tuples R'. Conceptually, we distinguish between *Definition Parameters* and *Runtime Properties*. The Data Controller specifies the *Definition Parameters* to create a Privacy View in SAP HANA DA. At execution time, SAP HANA DA determines the *Runtime Properties* dynamically when accessing the definition of a Privacy View for calculating an anonymized tuple set.

Figure 5 illustrates the decision flow of SAP HANA DA supporting the described approach: When a query is executed on R', SAP HANA DA reads the relevant metadata consisting of the *Definition Parameters* as well as *Runtime Properties*, i.e., Step 1. When accessing R' for the first time, the metadata only consists of the *Definition Parameters*. If no *Runtime Properties* exist, SAP HANA DA calculates them before generating the anonymized tuple set for R', i.e., Step 2. If the data of R has changed, SAP HANA DA determines if the changes cause a violation of the defined privacy criterion. The calculation step might be computationally expensive, thus we take any effort to execute this step as infrequently as possible.

Once SAP HANA DA has determined all necessary *Runtime Properties*, they are stored as metadata in the database, i.e., Step 3. Finally, SAP HANA DA is ready to transform the set of tuples of R into a set of anonymized tuples of R' according to existing parameters and properties, i.e., Step 4.

Since this decision flow is independent of the specific privacy-enhancing methods, we explain in the following subsec-

tions how this framework supports the implementation of both, *k-Anonymity* and *LDP*.

4.1 Parameterizing Privacy Views

Besides having common parameters and properties both anonymization methods need their individual sets of *Definition Parameters* and *Runtime Properties*. Currently, there are three *Definition Parameters* common to both methods: *Anonymization ID* identifying a Privacy View, *Method* specifying the privacy-enhancing method used, and *Sequence Column* naming a column in the Privacy View for storing tuple identifiers (TIDs) for every tuple in the view. SAP HANA DA uses those to check for possible view changes after updates on the microdata table.

To detect view changes SAP HANA DA requires the two *Runtime Properties Max* and *Size*. *Max* stores the maximum TID value of the *Sequence Column* while *Size* records the number of tuples of the microdata table accessed. Both values are sufficient to detect if tuples have been inserted, deleted, or updated. The DBMS can detect deletions or insertions on the microdata table based on property *Size*. Updates are detected by changes of the value of *Max*. Although these properties are sufficient to detect changes, they do not identify details of the changes, i.e., which exact tuple or value got changed. Since the content of the Privacy View is always recalculated it is still easy to check whether or not these changes result in violations of the privacy-enhancing method defined by parameter *Method*.

SAP HANA DA always checks these two *Runtime Properties* in Step 1 whenever a query accesses a Privacy View. Those properties determine if Steps 2 and 3 as shown in Figure 5 must be executed or not.

4.1.1 Parameterizing k-Anonymity

Creating a k-anonymous Privacy View requires the following additional *Definition Parameters*: *QID Columns*, i.e., the list of QID attributes, the requested level of anonymity k, and a list of *Generalization Hierarchies* (GHs), i.e., one generalization hierarchy for every QID attribute.

Conceptually, a tree represents a GH with '*' as root node and leaf nodes representing the values of the attribute in the microdata table. All leaf nodes have the same distance to the root node. SAP HANA DA stores every GH as a set of paths from the leaf node (original value) to the root node, see Figure 6c. The levels of generalization are computed and fixed during the first access to a Privacy View and stored as the *Runtime Property Optimum*. A vector of integers specifies the generalization level in the corresponding generalization hierarchies for each QID attribute.

SAP HANA DA determines the generalization level for all QID attributes with the least information loss using algorithm FLASH developed by Kohlmayer et al. [10] which is based on the INCOGNITO algorithm [11]. FLASH is currently the fastest known algorithm for this NP-hard optimization problem. Although we do not describe details of this algorithm in this paper we emphasize the following aspects of our implementation of FLASH:

- We encode all values of all QID attributes (always assumed categorical) as integer values using a dictionary-based approach. Thus, the integer-encoded table is kept in memory as an integer column for each QID (column store) during the execution of FLASH. Similarly, all hierarchy entries are also encoded as integer values via dictionary entries. Therefore, the complete in-memory computation of the FLASH algorithm uses integer comparisons and arithmetic, thus drastically improving performance during data generalization.

- As described in [10] FLASH computes a generalization using a specific metric to minimize the information loss. Computing the metric depends only on the generalization hierarchies rather than on the data itself thus making the check of a candidate solution for *k-Anonymity* more efficient.

- SAP HANA DA creates a generalization lattice for all QID attributes and their combinations while computing the optimal generalization level for each QID attribute during the first access to a Privacy View. Due to the monotonous metric used for the FLASH algorithm, successor nodes (predecessor nodes) of (non-) k-anonymous nodes are (non-) k-anonymous as well, respectively. The resulting generalization levels are stored as *Runtime Property Optimum* as part of Step 2 in Figure 5.

We emphasize that we implemented *all* optimizations as described in [10] such as projection, roll-up, and history. Still, executing the FLASH algorithm might take more than a few minutes to compute the optimal generalization levels for a given set of QID attributes as part of the first query. However, subsequent queries scale very well as we show in the evaluation (Section 5.3).

4.1.2 Parameterizing Local Differential Privacy

When using *LDP*, SAP HANA DA needs parameters *Noised Column*, i.e., the name of the column to which to add noise, as well as ε and *Sensitivity* as additional *Definition Parameters*. Furthermore, it uses the additional *Runtime Property Seed* to generate noise together with the value of the *Sequence Column* for each value of the *Noised Column* in a *reproducible* manner. It determines the *Seed* value with the first use of a Privacy View in Step 2 in Figure 5. As of May 2018, we restrict the definition of each Privacy View to a single *Noised Column* for pragmatic reasons. In principle, our approach for implementing *LDP* is extensible to an arbitrary number of attributes.

4.2 Transforming Microdata Tables

After setting the *Runtime Properties* (Step 2), SAP HANA DA is ready to generate a set of anonymized tuples (Step 3). In general, the anonymization step relies only on the stored *Definition Parameters* and *Runtime Properties* and is *reproducible*: For the same parameters, properties, and the same microdata table R as input, SAP HANA DA returns the same anonymized result.

For *k-Anonymity*, the anonymization step uses the stored *Generalization Hierarchies* as well as the stored *Optimum* to perform the anonymization of tuples. For each value v of a QID attribute A, SAP HANA DA uses level l stored in *Runtime Property Optimum* as an index into the generalization hierarchy of A to fetch the generalization value v' that replaces v in A.

When using *LDP*, SAP HANA DA determines for a record with *Sequence Column* entry i the i-th random number p from the interval $[0, 1)$. Equation (1) maps p onto a value which is then added as noise to the i-th value of attribute

Noised Column. We use the stored *Seed* to compute random numbers in a reproducible manner with low computational overhead that is constant per tuple.

5. MAKING IT ALL WORK

This section shows how to define a Privacy View and how to anonymize data of HRDB of Example 1 using SAP Web IDE, SAP's web-based development environment that includes the CE definition and execution environment. The dataset for the evaluation consists of 100,000 tuples with the schema of Table 1 to evaluate the utility of the anonymized results. Based on the HRDB in Table 1 we created a new calculation scenario by defining and configuring a Privacy View, i.e., an operator, via the SAP Web IDE modeler. Subsequently, we apply a projection operator to remove key attribute *Tuple ID* and identifier *Name* from the output view. Both operators are depicted as CE nodes in Figure 6a.

5.1 Using k-Anonymity

Figure 6b shows how to configure the *Definition Parameters* for *k-Anonymity* as outlined in Subsection 4.1.1. In our scenario, we select attribute *Tuple ID* as *Sequence Column*, define all attributes labelled *Quasi-identifiers* in Table 1 as *Quasi Columns* (see "Column Name" in Figure 6b; they result in the *Definition Parameter QID Columns*), choose the desired anonymization level k (= 10 for our scenario), and specify a generalization hierarchy for every *QID Columns* value stored in serialized form as *Definition Parameter Generalization Hierarchies* (see "Column Hierarchy" of "Quasi Columns" in Figure 6b).

For example, attribute *Gender* contains values 'f' (female) and 'm' (male), both of which are completely suppressed to '*' on the first generalization level. Obviously, generating the different generalization hierarchies requires detailed domain knowledge to capture the correct semantics, as shown in Figure 6c. Different generalizations might lead to different results for *k-Anonymity* with different information loss.

As a fist step of building a generalization hierarchy, SAP HANA DA imports all values of the corresponding attribute of a table into the Hierarchy Editor of SAP Web IDE. Those form the leaves nodes at the lowest level of the hierarchy. Only then the Hierarchy Editor allows a user to define additional generalizing values for higher levels of the hierarchy. For example, both values 'Bachelor' and 'College' in the microdata table (Level 0) generalize to 'Undergraduate' at Level 1 which then generalizes to 'Higher Education' at Level 2, together with other values. Level 3 becomes the most generalized level using value '*' by convention. Figure 3 already showed the generalization hierarchy for attribute *Start Year*. The generalization hierarchy for attribute *Zip Code* – not shown in the paper – suppresses the last two digits on Level 1, the last three digits on Level 2, and all (four) digits on Level 3. With all *Definition Parameters* in place, the Privacy View is deployed by a button-click making it accessible within the SAP Web IDE environment.

Table 2 shows the 10-anonymized version of (part of the) HRDB of Table 1. QID attributes *Start Year*, *Zip Code*, *Gender*, and *Education* are generalized to levels $(1, 3, 0, 0)$, respectively, with level numbers indicating the distance from leave nodes (Level 0) in the corresponding generalization hierarchy.

Figures 7 and 8 show histograms for average *Salary* by *Start Year* generated directly with the integrated analysis tool in SAP Web IDE. Figure 7 shows the histogram computed from the microdata table in Table 1 while Figure 8 shows the histogram computed from accessing the Privacy View based on 10-Anonymity.

Table 2: ACME IT's HRDB data anonymized using 10-Anonymity with *Start Year*, *Gender*, *ZIP Code*, and *Education* as QIDs.

Start Year*	Zip Code*	Gender	Education	Salary (in $)
1981-1985	*	m	HS-grad	139,051
2001-2005	*	m	College	173,834
1986-1990	*	f	Assoc-acdm	122,419
2001-2005	*	m	College	158,726
1996-2000	*	m	Bachelor	137,879
2001-2005	*	m	Bachelor	158,248
1986-1990	*	m	HS-grad	143588
1986-1990	*	f	Prof-school	117,197
...	...	...	...	...

Obviously, the histogram granularity is coarser in Figure 8 due to the privacy setting $k = 10$. However, we claim that the histogram based on the 10-anonymized data preserves the underlying trend of the salary data, i.e., the highest salaries occur before 1986, salaries are above $100,000 until 1996 before lower salaries occur (we underline that the interval '2006+' spans a wider range than previous intervals).

EXAMPLE 4. *Such analysis – possibly together with additional information – could be highly valuable for members of the HR department at ACME IT since they allow them to predict how salary values for new employees might develop in the forthcoming years. Without anonymization they might not have been allowed to access the salary data at all.*

We are aware that the minimal example chosen for illustration purposes only, does not reflect real-world anonymization setups with many more QIDs that are subject to generalization.

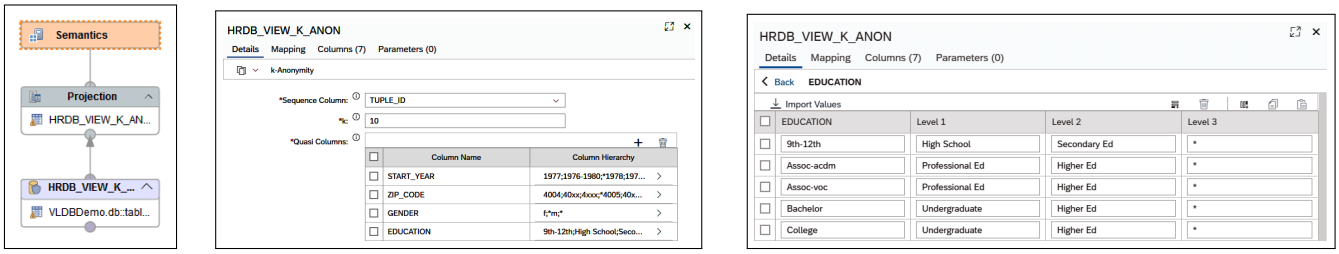
5.2 Using Local Differential Privacy

Figure 9 shows the definition of a Privacy View using *LDP* by configuring the *Definition Parameters* as described in Subsection 4.1.2.

Table 3 contains the results of applying *LDP* to attribute *Salary*. Adding noise to each *Salary* value of every record generates new salary values that show no relationship to their original values – some of the values might even be negative, for example see the fifth tuple in Table 3. We emphasize that only aggregate functions *SUM* and *AVERAGE* generate meaningful results on attributes with noised values in this scenario.

EXAMPLE 5. *Members of the HR department at ACME IT might compare aggregated salary values across different demographic groups based on the QID columns Start Year, Gender, Zip Code, and Education.*

Although data for Table 1 is synthetic and does not relate to any particular real scenario/persona, it represents a realistic case for an HRDB in enterprise systems. In our example, the relation consists of 100,000 employee tuples with four QID attributes and the sensitive attribute *Salary*



(a) DAG. **(b)** Defining a Privacy View with *k-Anonymity*. **(c)** Hierarchy Editor for generalizing attribute *Education*.

Figure 6: *k-Anonymity* Modelling in the SAP Web IDE.

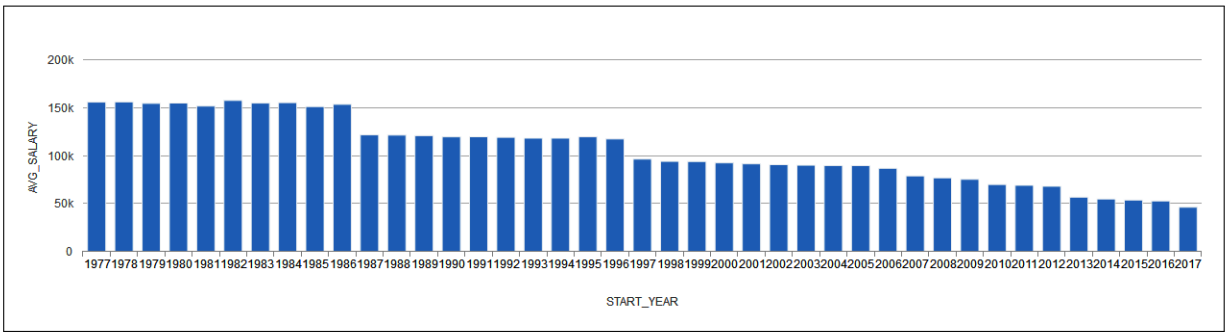


Figure 7: Histogram for average of *Salary* by *Start Year* computed on the original data of Table 1.

Table 3: ACME IT's HRDB anonymized via *LDP* with *Salary* as *Noised Column* containing sensitive information.

Start Year	Zip Code	Gender	Education	Salary* (in $)
1985	7204	m	HS-grad	145,437
2005	4206	m	College	918,159
1989	6104	f	Assoc-acdm	226,855
2004	5107	m	College	139,277
2000	6004	m	Bachelor	-196,900
2003	7204	m	Bachelor	194,303
1988	6104	m	HS-grad	38,070
1987	4205	f	Prof-school	171,990
...	...	...	...	...

with salary values that are typical in a worldwide acting IT-company. In general, the *Sensitivity* value depends on the spread of the sensitive attribute values, i.e., the difference between the maximum and the minimum value. In our example, the spread of the *Salary* values in the microdata table is $165,961.

Much in the spirit of DPComp [7], we performed a quantitative analysis on how data noised via *LDP* still preserves its utility – in our example the *Salary* pattern w.r.t. *Start Year* – for different values of ε. As a baseline, we compute the average *Salary* based on the microdata table R, denoted by $\bar{s}_R$, and average *Salary* values for the *Start Year* intervals 1977–1986, 1987–1996, and 1997–2006. We ignore the interval 2007–2017 since it does not show a clear trend.

Similarly, we compute these averages using the *LDP* Privacy View R' with the settings $\varepsilon = 0.1$, 0.5, and 0.8, indicated by the subscripts R' and ε, respectively. We compute relative errors of these averages for R' with respect to the microdata table R, $\delta\bar{s}_\varepsilon = |\bar{s}_R - \bar{s}_{R',\varepsilon}| / \bar{s}_R$. The lower the (relative) errors the higher the accuracy of the *LDP* approach. Since the added noise depends on the seed for the random number generator chosen in the *LDP* algorithm, we repeated the experiments 100 times with different seeds to determine the distribution of results. Table 4 shows the average results, indicated by $\varnothing$, of our experiments. Figure 10 shows the distribution of the relative error.

For $\varepsilon = 0.1$ the relative error in the first two time intervals is much larger than the overall relative error. As a consequence, this setting for ε is likely to miss the pattern for the first two groups (which are the most pronounced). For some samples the error of 15%, respectively 30%, will result in averages that do not distinguish between the two groups (1977–1986 and 1987–1996), thus making a useful analysis impossible. For $\varepsilon = 0.5$ the average values for every range of *Start Year* are similar to those computed on the microdata table and on the 10-anonymized Privacy View. This observation is not surprising for 100.000 tuples: The generated Laplace-noise using Equation (1) cancels out for such aggregates over large subsets. For $\varepsilon = 0.8$ all errors are the lowest as expected, *Salary* patterns can still be recognized in the anonymized data.

The accuracy evaluation shows the effects of ε on the data set: The lower ε, the lower the accuracy (Figure 10). Still, the use case shows that sufficient utility is left to reveal the trend in the data correctly, e.g., less than 10% error for all experiments with an ε of 0.5 or larger.

Our brief analysis shows the challenge in choosing an appropriate value for ε to keep a sensible balance between privacy protection and application utility. Furthermore, the above analysis is exemplary and does not replace a more detailed analysis to determine the impact of ε on the usefulness of the noised values in a particular application context.

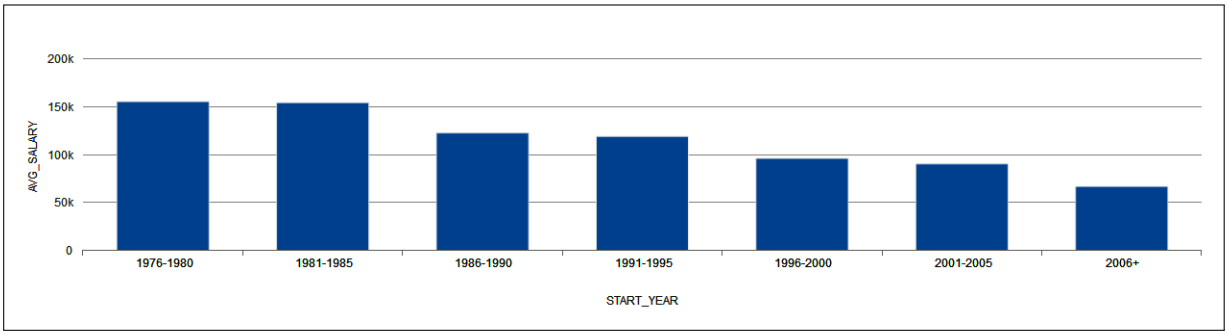


Figure 8: Histogram for average of *Salary* by *Start Year* computed on the 10-anonymized data of Table 2.

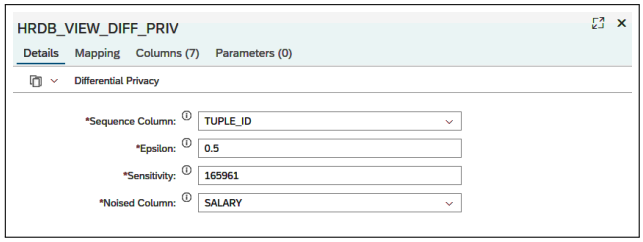


Figure 9: Definition of a Privacy View with *LDP*.

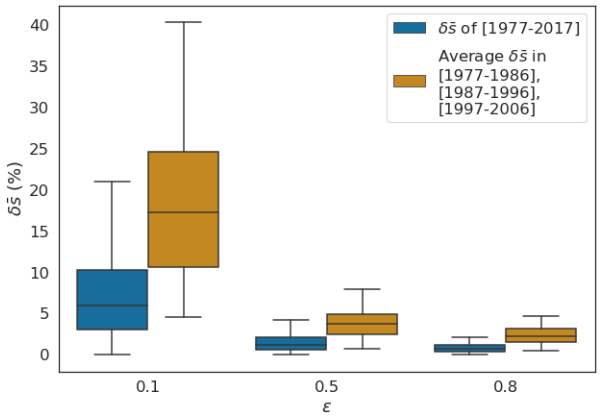


Figure 10: Relative error on averages of *LDP*-noised *Salary* data in given intervals.

We emphasize that SAP HANA CE allows users to combine definitions of multiple Privacy Views using *k-Anonymity* or *LDP* to handle successive anonymization of different (possibly overlapping) subsets of columns.

5.3 Impact on Performance

Besides the accuracy, we evaluate the performance impact of the implementation described in Section 4 on (*SELECT*) queries on the HRDB data described in Section 5. The evaluation omits the first query execution which imposes additional, but only one-time, latency. This performance evaluation was run on a 64 Core *Intel Xeon E5-2683 v4 2.10GHz* system with an allocation limit of 32*GB* for SAP HANA 2.0 SPS03. We increased the number of tuples in the Privacy View in steps of 10,000 and measured query runtime relative to 10,000 tuples, see Figure 11. The results confirm the discussion in Section 4: The time of querying a Privacy View is linear w.r.t. to the number of tuples accessed. In our example, *LDP* adds random noise to only one column while *k-Anonymity* modifies four QID columns, thus resulting in longer query times.

Table 4: Quantitative analysis of the *LDP*-enhanced *Salary* data for 100 experiments.

Interval	1977–2017	1977–1986	1987–1996	1997–2006
$\bar{s}_R$	\$81,857	\$154,740	\$119,603	\$91,376
$\varnothing\bar{s}_{R',0.8}$	\$81,943	\$154,172	\$120,112	\$91,182
$\varnothing\delta\bar{s}_{0.8}$	0.8%	4.0%	2.1%	1.1%
$\varnothing\bar{s}_{R',0.5}$	\$81,867	\$155,863	\$118,616	\$91,421
$\varnothing\delta\bar{s}_{0.5}$	1.4%	6.7%	3.1%	2.0%
$\varnothing\bar{s}_{R',0.1}$	\$82,732	\$163,516	\$118,984	\$92,003
$\varnothing\delta\bar{s}_{0.1}$	7.4%	30%	15%	10.3%

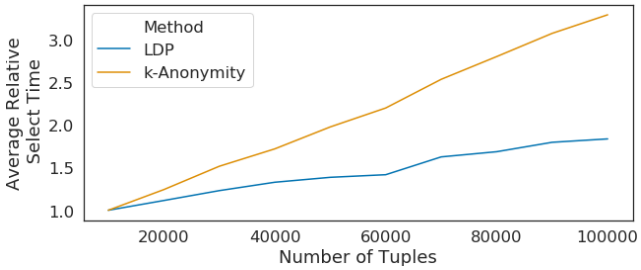


Figure 11: Relative times for executing queries.

5.4 DPPO View of Anonymization Scenarios

Figure 12 shows how the DPPO accesses a Privacy View of SAP HANA DA for assessing its privacy settings. The information presented allows the DPPO to verify a common understanding on the set of QID attributes as well as the generalization hierarchies in case of *k-Anonymity*. We already emphasized the importance of such feature at the beginning of this paper in Example 1 and *Challenge 2*.

6. ASSESSMENT AND CONTRIBUTIONS

This paper presents the first steps on how to integrate privacy-enhancing methods into SAP HANA in a seamless manner which makes it natural for applications to use these new features. Our approach anonymizes data *as early as*

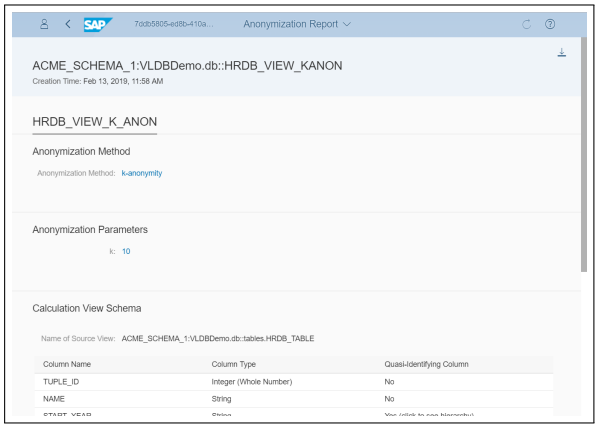


Figure 12: DPPO view of the *k-Anonymity* Scenario.

possible by applying privacy-enhancing methods quite close to the storage layer of SAP HANA. Our approach reduces the risk of privacy breaches at the same time avoiding data replication. Additionally, we designed and implemented a flexible framework that allows us in the future to extend SAP HANA DA with additional privacy-enhancing features.

Adding an operator for protecting privacy to the already existing ones in the CE system provides an elegant approach for both the Data Controller and the Data Consumer when creating and using a Privacy View, respectively. Section 5 clearly demonstrates the *ease of use* of our solution in the enterprise systems world.

Furthermore, SAP HANA DA is capable of *bridging the gap between technology and legal requirements*. It enables a DPPO to monitor Privacy Views when managing the access to personal data. Currently, our implementation in SAP HANA DA meets the challenges as defined in Section 2.

For the remainder of this section we discuss possible future enhancements, for that matter we closely interact with our customers to gather additional requirements they consider important from their perspective when extending the system with additional privacy-enhancing features.

Currently, we leave it up to the Data Consumer to decide which privacy-enhancing method and which parameter settings are acceptable for the purpose of an application. In general, there exists only limited experience on how to balance privacy requirements and utility on data sets, since utility requirements predominantly depend on the application [9].

We are aware that *k-Anonymity* has shortcomings, such as the possibility of sensitive attribute disclosure. Those caused the research community to develop more sophisticated approaches, among them *l*-Diversity [13] and *t*-Closeness [12]. Despite these known weaknesses, *k-Anonymity* has become quite popular and has widely been used in various application domains, e.g., in medical research, due to easy comprehension [17, 2, 10].

In Section 3.1, we outlined that SAP HANA DA does not return any results in case *k-Anonymity* is violated after changes in the microdata table. While the current solution might be inconvenient for Data Consumers, it satisfies *Challenge 2*: Any violations of the guaranteed privacy parameters are not acceptable for a Data Owner, Data Controller, or DPPO. It is well known that *k-Anonymity* is not resilient against those changes. Therefore the research community developed the concept of *m*-Invariance to manage changes on the microdata table safely without privacy breaches [24]. Since *m*-Invariance is computationally expensive, we currently explore alternative approaches with similar privacy guarantees. In the current version of SAP HANA DA, we rely on access control measures on (Privacy) Views to restrict access, thus mitigating the risks of privacy breaches.

In the first release, we achieved our goal to provide an enterprise-ready implementation that meets the challenges of Section 2. The current implementation of SAP HANA DA leaves room for extensions, but is sufficiently powerful and usable for SAP's customers by applying *k-Anonymity* or *LDP* when accessing personal data. We enable our customers to extract new insights from their data and to gain experience in applying anonymization methods. Based on their experience we are convinced to lay a solid foundation of privacy enhanced data processing in enterprises for the future.

7. SUMMARY AND OUTLOOK

We presented an overview on SAP HANA DA, SAP's holistic approach of making privacy-enhancing features available in SAP's enterprise system. As of May 2018, we extended SAP HANA with two anonymization techniques, *k-Anonymity* and *LDP*. For our first version – called SAP HANA DA in this paper – we decided to keep the extension understandable for Data Owners, Data Consumers, and DPPOs rather than implementing all features that are technically possible. Our implementation allows users of SAP HANA DA to build applications with privacy protection and to give us feedback for future enhancements. Using the list of challenges as presented in Section 2 helped us to set priorities and preferences in our implementation efforts and to decide which features to select and which ones to leave out for the first version of SAP HANA DA.

Technically, we see potential for further improvements and extensions. In particular, we strive to find a more viable solution for *k-Anonymity* when changes of microdata tables cause the corresponding Privacy View not to return any result. Although *m*-Invariance offers a solution [24] we see the need for an alternative approach that is less computationally expensive.

Additionally, we would like to integrate privacy-enhancing methods better with the business tasks and models as part of SAP's enterprise system. Although SAP HANA DA currently provides a solution to include a DPPO into a monitoring process there exist additional requirements that we want to support in a future version. Such support might also include extensions that balance utility and the level of privacy protection in a given application context. For this problem, additional research and real-word experience for a better understanding and for better solutions is required.

As of April 2019, we extended the first version of SAP HANA DA with *l*-Diversity as third anonymization method and introduced optional parameters for better controlling anonymization results. Additionally, Privacy Views can now also be created with extensions to SQL. For *k-Anonymity* and *l-Diversity*, the new release simplifies the configuration of hierarchies by integrating existing ones defined via SQL functions or (SAP HANA) Hierarchy Views. Furthermore, the system provides means for evaluating data quality and other properties of Privacy Views.

We refer interested readers to the GitHub repository [20] to download the SAP Web IDE sample project to define and to use Privacy Views together with the sample data of the examples presented in Section 5. Anyone can download and install the SAP HANA Express Edition for free from the Web [18] to run the examples discussed in this paper.

8. ACKNOWLEDGEMENTS

We would like to thank Thomas Seufert, Kaweh Amoi-Teleghani, Kai Morich, Aleks Aleksic, Andrea Kristen, Johann Boehme, and Holger Mack as members of the team for their dedicated work on the first version of SAP HANA DA. Also, we thank Ponnahi Singam for carefully reading the manuscript.

In remembrance of Daniel Schneiss, we dedicate this paper to him. He put trust and support into the SAP HANA DA project from its very beginning.

9. REFERENCES

[1] CNIL. Data Protection around the World. https://www.cnil.fr/en/data-protection-around-the-world.

[2] F. K. Dankar and K. El Emam. The Application of Differential Privacy to Health Data. In *Proc. of the Joint EDBT/ICDT Workshops*, pages 158–166, 2012.

[3] Diffix. Anonymisation with Diffix: how it works. https://aircloak.com/solutions/diffix-en/.

[4] C. Dwork. Differential Privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12. Springer-Verlag, 2006.

[5] U. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA*, pages 1054–1067, 2014.

[6] P. Francis, S. P. Eide, and R. Munz. Diffix: High-Utility Database Anonymization. In *Privacy Technologies and Policy - 5th Annual Privacy Forum, Vienna, Austria*, pages 141–158, 2017.

[7] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, D. Zhang, and G. Bissias. Exploring Privacy-Accuracy Tradeoffs using DPComp. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 2101–2104. ACM, 2016.

[8] HIPPA. Health Insurance Portability and Accountability Act of 1996. https://www.congress.gov/bill/104th-congress/house-bill/3103.

[9] S. Kessler. *Privacy-enhancing methods for time series and their impact on electronic markets*. PhD thesis, KIT-Bibliothek, Karlsruhe, 2015. http://dx.doi.org/10.5445/IR/1000052458.

[10] F. Kohlmayer, F. Prasser, C. Eckert, A. Kemper, and K. A. Kuhn. Flash: Efficient, stable and optimal k-anonymity. In *2012 International Conference on Privacy, Security, Risk and Trust, PASSAT, Amsterdam, Netherlands*, 2012.

[11] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16*, 2005.

[12] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115, April 2007.

[13] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), Mar. 2007.

[14] F. McSherry. Privacy Integrated Queries. In *Proc. of the 2009 ACM SIGMOD International Conference on Management of Data*. ACM, June 2009.

[15] Oracle. Database Advanced Security Administrator's Guide. https://docs.oracle.com/en/database/oracle/oracle-database/18/asoag/asopart2.html, 2016.

[16] E. Parliament and E. Council. General Data Protection Regulation. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN.

[17] F. Prasser, F. Kohlmayer, H. Spengler, and K. A. Kuhn. A Scalable and Pragmatic Method for the Safe Sharing of High-Quality Health Data. *IEEE J. Biomedical and Health Informatics*, 22(2):611–622, 2018.

[18] SAP. SAP HANA Express Edition. https://developers.sap.com/topics/sap-hana-express.html.

[19] SAP. SAP HANA Modeling Guide. https://help.sap.com/viewer/e8e6c8142e60469bb401de5fdb6f7c00/2.0.03/en-US/a9ab474a16d34e56bd72572b9a598216.html.

[20] SAP. SAP HANA Web IDE sample project. https://github.com/SAP/hana-data-anonymization-vldb-demo.

[21] V. Sikka, F. Färber, W. Lehner, S. K. Cha, T. Peh, and C. Bornhövd. Efficient transaction processing in SAP HANA database: the end of a column store myth. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD, Scottsdale, AZ, USA, May 20-24*, pages 731–742, 2012.

[22] L. Sweeney. Achieving k-Anonymity Privacy Protection using Generalization and Suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):571–588, 2002.

[23] T. Wang, J. Blocki, N. Li, and S. Jha. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium, USENIX Security, Vancouver, BC, Canada, August 16-18*, pages 729–745, 2017.

[24] X. Xiao and Y. Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China*, 2007.