Bulletin of the Technical Committee on

Data Engineering

March 2022 Vol. 45 No. 1

IEEE Computer Society

Letters

Opinions

Special Issue on Directions Towards GDPR-Compliant Data Systems and Applications

Conference and Journal Notices

The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TCDE web page is http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at
http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

Data is considered the most valuable asset but as its volume, complexity and implications continue to grow, the tech industry must face an unprecedented range of challenges from data management to ethical obligations and regulatory pressures.

This issue of the Data Engineering Bulletin, curated by Sebastia Schelter, endeavors to address some of these challenges. For the first time, we focus on regulations shaping how data is being collected, managed, and used in the tech industry. Several papers in this issue dive into questions originating from the "right-to-be-forgotten" postulated by GDPR. A concomitant technical challenge is how to focus resources on legitimate and valuable data to maximize the business impact. For example, Davidson et al.'s work on "Disposal by Design" used e-commerce to highlight challenges and opportunities in the realm of data regulation. Applications such as e-commerce data reduction, image archiving, and relational data sampling and aggregation open the door for further research in this domain.

This issue also features an opinion piece by Ihab Ilyas and Felix Naumann, who looked into the critical question of data and model observability. For years, data quality has been a key concern and a main priority for the tech industry, but the problems have become more elusive as the industry relies more and more on machine learning models. While the situation has given rise to a new tech segment pioneered by companies such as BigEye and Monte Carlo Data, the solutions are still primitive. Ilyas and Naumann's call for action opens a new chapter of data quality and data cleaning that understands the entire data processing pipeline.

Haixun Wang
Instacart

Letter from the Special Issue Editor

Software systems that store and process personal data have become ubiquitous over the last years and have enabled numerous economic, technological and scientific advances. Unfortunately, the benefits of data-driven analysis and decision making have also been accompanied by several negative developments. Examples include the increased surveillance capabilities of the state [3] and private companies [4], negative impact on economic inequality [2] and traumatic experiences for individuals [5]. As a reaction, many countries have started to regulate data storage and processing to guarantee and protect the rights of individuals. The most comprehensive such regulation is the General Data Protection Regulation (GDPR, `https://gdpr.eu`) issued by the European Union.

In this special issue on *Directions Towards GDPR-Compliant Data Systems and Applications*, we continue the ongoing discussion in the data management community [1] on how to redesign data systems and applications to be compliant with such regulation.

Data deletion as a first-class-citizen. The first three papers of this issue address an important question originating from the "right-to-be-forgotten" postulated by GDPR: *How can we design efficient data systems that support the timely deletion of data as a first-class citizen?* The first paper on *Disposal by Design* presents a vision for automating data disposal which takes into account processing constraints, regulatory constraints as well as storage constraints, and gives concrete examples from the e-commerce domain, including a suggestion of how to to find summaries of relational data with machine learning. The second paper on *Building Deletion-Compliant Data Systems* argues that the the requirement of timely deletion of user data is becoming central in modern data management scenarios. The authors present a new framework for building deletion-compliant data systems from a holistic perspective, analyse the requirements derived from the new policies, and propose changes in the application and the system layer of data management systems. The third paper called *Provenance-based Model Maintenance: Implications for Privacy* focuses on efficient data deletion in a machine learning context. In particular, the authors focus on the extremely challenging problem to refresh existing models after the removal of training samples, which is called "machine unlearning". They argue that GDPR regulations imply that the removed samples must be fully erased from the models so that they cannot be leaked to an adversary. The paper reviews two provenance-based solutions and shows how they can guard against "model inversion attacks", which reconstruct the removed training samples from the updated models after the unlearning process.

Efficient data processing under regulatory constraints. The subsequent two papers of this issue adress an orthogonal systems-related question originating from GDPR: *How can we design efficient data systems that comply with data processing regulations?* The fourth paper of this issue on *Navigating Compliance with Data Transfers in Federated Data Processing* presents work on novel systems and methods for federated data processing, where the processing of geo-distributed data is subjected to data transfer regulations. The authors showcase recent work on compliant geo-distributed data processing and present research challenges and opportunities in making federated data processing systems GDPR-compliant. The fifth paper called *Towards Privacy by Design for Data with STRM privacy* discusses the practical challenges of engineering teams to balance privacy and innovation, with respect to effort, data utility and computation costs. The authors argue that current approaches in scalable data systems often treat privacy as an access problem, which is at odds with important legal and design principles. Instead, the authors propose that engineering teams should shift their data privacy efforts to the point of data collection, and discuss an architectural setup for privacy-compliant stream processing applications, which is in production usage.

This work was supported by Ahold Delhaize. All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

[1] S. Supreeth, V. Banakar, M. Wasserman, A. Kumar, and V. Chidambaram. Understanding and Benchmarking the Impact of GDPR on Database Systems. Proceedings of the VLDB Endowment 13(7), 2019.

[2] V. Eubanks. Automating inequality: How high-tech tools profile, police, and punish the poor. St. Martin's Press, 2018.

[3] G. Greenwald. No place to hide: Edward Snowden, the NSA, and the US surveillance state. Macmillan, 2014.

[4] S. Zuboff. The age of surveillance capitalism: The fight for a human future at the new frontier of power. Profile books, 2019.

[5] V. Warmerdam. Beyond Broken - Horrible Remedies for Broken Recommenders. Published online at `https://koaning.io/posts/beyond-broken/`, 2021.

Sebastian Schelter

University of Amsterdam & Ahold Delhaize Research, The Netherlands

Data Errors: Symptoms, Causes and Origins

Ihab F. Ilyas, Felix Naumann
University of Waterloo, University of Potsdam

1 Introduction: Data Errors and their Root Causes

With the recent move towards data-centric AI, data quality is now playing an even bigger role in producing sound and reliable insights, predictions and analytics. While the data management community has been working on the problem of data cleaning for decades, the problem remains very much present. Most efforts have focused on error detection [13], attempting to leverage symptoms and the manifestations of these errors in data sets to locate and possibly repair them. Indeed, the last few years witnessed significant advances in automating error detection and repairing [20, 9, 15, 22] by probabilistically modelling dirty data sets, and reasoning about error detection and repair as structured prediction problems [21, 8]. In this opinion piece, we present our views on how to further advance the field of data cleaning, and go beyond treating the symptoms of the problem and understand what it takes to treat the causes and the sources of these anomalies and errors.

Tracking errors to their sources is not a new quest of the research community [5, 24, 7]. So why are we revisiting it now? The way current research currently reasons about the root causes of data errors is still, in our opinion, limited. *Tracking errors to sources* has often been framed as "computational" provenance that represents *what* was involved in computing a final data product and possibly *how* this product was computed. The main goal of these provenance-based error tracking systems is projecting errors detected in downstream applications all the way to upstream data sources, where they should be fixed [5]. While the principle is sound, multiple issues often complicate this approach. First, as data processing pipelines become more complex, with cascades of complex machine learning models, capturing this rich provenance information becomes harder, as most if not all of the input is involved in producing the output. Recent progress, however, has been made in tracking responsibility of training data, for example, in the predictions of complex models [18, 14, 10]. Second, even with advances in modelling the responsibility of input sources in the observations in output analytics reports, fixing the sources does not mean that errors have been fixed at their true "roots" – their point of creation; these raw data sources are often the results of other processes not modelled at all in the computational data pipelines, such as grading tasks of humans, sensor readings, and extraction scripts from logs and documents.

Hence, we argue that effective management of data errors and the next-generation data cleaning systems require a more profound understanding of the root causes data errors. These systems should: (1) distinguish between *why* errors occur and the processes that generated them in the first place, and *how* these errors manifest themselves as bad data symptoms (e.g., violations of integrity constraints and appearing as outlying values), and (2) explicitly model these data generation processes to allow for new process repair actions that go beyond fixing raw data sources.

2 Reasoning about the How: The Symptoms of Errors

In its most general form, information quality is defined as "fitness for use" [23], i.e., its definition focuses on the use case, the application, the *context* of the data at hand. Most, if not all, error detection methods make heavy use of this context. They search for and focus on the *symptoms* of poor data quality and ask the question *how* a particular data element is an error. We exemplify this insight with selected examples of traditional error detection problems and their solutions.

Outliers Already by definition, outliers can be recognized only in the context of other data elements – only these "inliers" make some other value an outlier. Typical outlier detection methods create a model to represent

normal/typical values and mark as outliers all those values that do not fit the model [2]. Whether an outlier is, in fact, an error is application-dependent and user-defined.

How is an outlier a data error? It is very different from all *other* data elements, suggesting it is not the intended value for this data item.

Constraint violations Constraints, such as key-constraints, dependencies, or denial constraints can be used to express the validity of a data instance. These rules are specified by experts or discovered with data profiling methods [1]. Rarely do they refer to individual data elements; rather they forbid the existence of some elements in the presence of others, such as a key-constraint denying any other record with the same key value. Discovering and cleaning such violations is an active research area [12].

How does a data element violate a constraint? It exists in the presence of some *other* data element. While this is not an error on its own, the collection of these values cannot be part of the intended correct data instance.

Duplicates Within a dataset, duplicate records are multiple different representations of the same real-world entity [16]. To clean a dataset, such erroneous duplicates must be detected and then merged or eliminated. Identifying a duplicated record is, by definition, possible only by regarding other records, i.e., only in the context of the entire relation. Typical approaches intelligently create duplicate candidate pairs and then determine their similarity to decide whether they are indeed duplicates [17].

How is a duplicate an error? It represents the same real-world object as some *other* data element, with possibly other types of errors causing a different representation.

Missing values Missing values are easy to detect when they appear as null values in databases or empty strings in files. In more complex cases, "disguised missing values" can be recognized only by regarding their context, which usually comprises the other values in the column [19]. The typical means to "clean" missing values is to impute their value, again based on their context, usually the non-missing values in a column.

How is a missing value an error? It is explicitly represented to indicate an error. However, the difficulty in the case of missing value relates to the interpretation of "null" as *we don't know the value, but we should* as opposed to schema issues, for example, a relational employees table with some employees who do not have middle names.

Data cleaning, as a means to alleviate the symptoms of poor data quality, is an established and important research and development field, relying heavily on the context of data and its use in applications. Next, we move backwards along the data processing pipeline to explore not these symptoms, but their causes.

3 Reasoning about the Why: The Causes of Errors

It is time to ask (and answer) the *why* question! Existing work in the area of data quality, error detection and data cleaning almost exclusively focuses on alleviating the symptoms, rather than removing the cause of the error. None of the methods asks *why* a particular value is missing, why duplicates exist in the data, why violations occur. Answers as to "why" include: faulty (human) data entry, such as missing entries, misplaced values, typos, and vandalism; faulty reading from sensors; missed or not-propagated updates; faulty computations; and misconfigured data pipelines.

While researchers and practitioners (and medical doctors) will acknowledge the truism that problems are best addressed at their source rather than treating their symptoms, the research community has not adequately addressed this opportunity possibly for several reasons:

- In some scenarios, once errors are detected, it is too late – fixing their cause is futile because the data was intended for a one-time use.
- Often, the creation of data is out of the control of the data engineers or data consumers: the data stems from an external source and the data creation can be influenced only through human intervention, such as communicating with the data owners or creators.
- Modifying or improving the data creation process is difficult or impossible, for instance due to technical or human limitations: sensors have an inherent error margin; humans are not infallible, etc.
- Data processing pipelines have become so complex, that treating the symptoms is the easier short-term goal with quick rewards.

Knowledge of the cause of an error and not only its symptom can improve cleaning methods and can help avoid such errors in the first place. To seize this opportunity, multiple challenges must be overcome:

- *Modelling* the processes and data generators (including humans) in the system, instead of modelling only the data and errors.
- *Detection* of data errors without context and detection of erroneous data processes.
- Extending the notion of *provenance* to include (possibly faulty) processes, computation (internal provenance), and data generation steps (external provenance). More on this in Section4.
- Designing of algorithms and systems to efficiently and effectively trace such extended provenance.
- Designing *repair* operations for such errors and processes, which need to reach beyond the mere deletion or replacement of data instances that is the currently common approach.

These challenges can be summarized as creating a more holistic view of data creation and consumption than is currently practiced. Especially the extended notion of provenance deserves a closer look in the next section.

4 True Data Provenance

Provenance is powerful tool for *tracking* data artifacts. In the context of this paper, one might think of it as a way to identify the *where* of the data error's story. Most practical and effective cleaning solutions follow a clean-and-evaluate lifecycle [11], which leverages the computational provenance of data analytics to track data errors to their sources, and attempts to provide explanations that lead to cleaning actions. This typical lifecycle is depicted in Figure 1.

Provenance and lineage systems focus on describing how the analytical *report views* are computed from the sources. For example, Scorpion [25], DBRx [5] and QFix [24] (and many other followup work) are solutions that trace back the tuples that contributed to the problems in the target to explain and help fix these errors at data sources. A recent survey summarizes the large body of work in debugging data-driven systems and explain what users see downstream from processing raw data [7]. As these processing pipelines become more complex with cascades of large machine learning models, tracing errors in final predictions back to their causes can be very challenging. However, there is recent progress that can help us reason about observations in model predictions and track them back to errors in training data [10].

The question becomes: *is explaining errors in final analytics or predictions in terms of data sources enough?* What we refer to as "raw data sources" are often cut off the processes that generated these data, such as the human grader that input that data, the extraction script that generated this data from a webpage, or a presentation of the complex data pipeline that ran in a different software stack and generated this source data. From our discussions

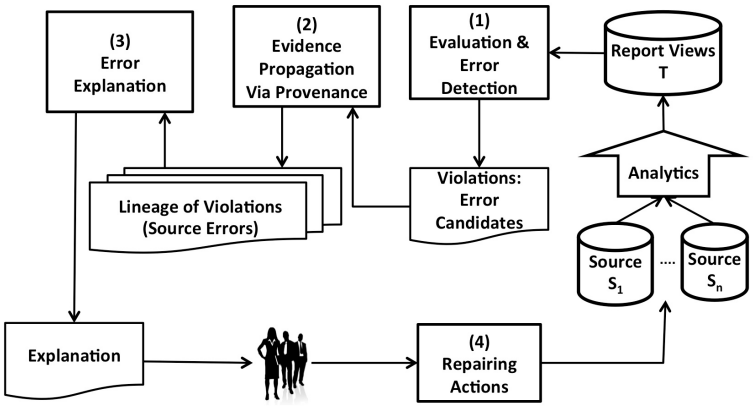


Figure 1: Clean-and-Evaluate Loop [11]

and involvement with large enterprises over the last decade, we argue that this decoupling is often due to two main reasons:

- *Difficulty of integrating data processes in provenance systems:* Representing the process that generated the data might require expressive (and hence complex) provenance systems. For example semiring-based provenance systems have been extended to capture information about external inputs (e.g., user choices), and to capture process executions [6]; and in the context of scientific workflows, the need for a control-flow driven workflow provenance model in contrast to the traditional data-driven execution provenance paradigm has been explored [4].

- *Loss of provenance continuity across systems:* We might be very careful in collecting and adequately presenting provenance information in the data pipelines we control. However, as the final data product (e.g., predictions, views, aggregates, or transformed data sets) get pushed to the downstream tasks, they are often treated as "source data" and downstream pipelines fail to consume the associated provenance information.

Understandably, these are hard problems to tackle and part of the challenge is not even technical and it involves standardizing data provenance representation across business units and different software stacks. However, this might suggest new research directions; for example, we might prefer developing simpler and less expressive provenance models that target interoperability and ease of propagation over representation power of the underlying computations. Another example is that propagating standard meta-data that ties data sources to central data governance and catalogs can be part of the integrity constraints and sanity checks. We suggest also extending meta-data representation of data sources to include *repair actions* that reference a controlled vocabulary or a *repairing ontology* tapping into the large body of work in work flow and business processes management.

5 Conclusion

To conclude, we suggest opening a new chapter of data quality and data cleaning that understands the entire data processing pipeline, in particular tracing it to the very beginning – the genesis of the raw data. We have pointed out the challenges, with a focus on a new view of data provenance.

Having discussed the *how* (symptom), the *why* (cause), and the *where* (via provenance), other questions about errors remain. We have only glossed over the question *what* is erroneous: an individual value, a row, a column, a table, or a process? Our general discussion allows these questions for data model beyond the relational, including tree or graph data, or even images, sound and video. When regarding data as it is created over time, we can ask

when the data error was introduced, and use data versions to understand the nature of the error [3]. The final question of *who* to blame, we leave to the management sciences.

References

[1] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. *Data Profiling*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2018.

[2] Charu C. Aggarwal. *Outlier Analysis*. Springer, 2013.

[3] Tobias Bleifuß, Leon Bornemann, Theodore Johnson, Dmitri V. Kalashnikov, Felix Naumann, and Divesh Srivastava. Exploring change - a new dimension of data analytics. *PVLDB*, 12(2):85–98, 2018.

[4] Anila Sahar Butt and Peter Fitch. A provenance model for control-flow driven scientific workflows. *Data Knowl. Eng.*, 131-132:101877, 2021.

[5] Anup Chalamalla, Ihab F Ilyas, Mourad Ouzzani, and Paolo Papotti. Descriptive and prescriptive data cleaning. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 445–456, 2014.

[6] Daniel Deutch, Yuval Moskovitch, and Val Tannen. Provenance-based analysis of data-centric processes. *VLDB Journal*, 24(4):583–607, 2015.

[7] Boris Glavic, Alexandra Meliou, and Sudeepa Roy. Trends in explanations: Understanding and debugging data-driven systems. *Found. Trends Databases*, 11(3):226–318, 2021.

[8] Alireza Heidari, Ihab F. Ilyas, and Theodoros Rekatsinas. Approximate inference in structured instances with noisy categorical observations. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 115 of *Proceedings of Machine Learning Research*, pages 412–421. AUAI Press, 2019.

[9] Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. Holodetect: Few-shot learning for error detection. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, page 829–846, 2019.

[10] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *CoRR*, abs/2202.00622, 2022.

[11] Ihab F. Ilyas. Effective data cleaning with continuous evaluation. *IEEE Data Eng. Bull.*, 39(2):38–46, 2016.

[12] Ihab F. Ilyas and Xu Chu. Trends in cleaning relational data: Consistency and deduplication. *Found. Trends Databases*, 5(4):281–393, 2015.

[13] Ihab F. Ilyas and Xu Chu. *Data Cleaning*. ACM, 2019.

[14] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the International Conference on Machine Learning*, volume 70, pages 1885–1894. PMLR, 2017.

[15] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. Raha: A configuration-free error detection system. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, page 865–882, 2019.

[16] Felix Naumann and Melanie Herschel. *An Introduction to Duplicate Detection*. Morgan & Claypool Publishers, 2010.

[17] George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas. *The Four Generations of Entity Resolution*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2021.

[18] Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracking gradient descent. *CoRR*, abs/2002.08484, 2020.

[19] Abdulhakim A. Qahtan, Ahmed Elmagarmid, Raul Castro Fernandez, Mourad Ouzzani, and Nan Tang. FAHES: A robust disguised missing values detector. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, page 2100–2109, 2018.

[20] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, August 2017.

[21] Christopher De Sa, Ihab F. Ilyas, Benny Kimelfeld, Christopher Ré, and Theodoros Rekatsinas. A formal framework for probabilistic unclean databases. In *Proceedings of the International Conference on Database Theory (ICDT)*, pages 6:1–6:18, 2019.

[22] Pei Wang and Yeye He. Uni-detect: A unified approach to automated error detection in tables. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 811–828. ACM, 2019.

[23] Richard Y. Wang and Diane M. Strong. Beyond accuracy: What data quality means to data consumers. *Management of Information Systems*, 12(4):5–34, 1996.

[24] Xiaolan Wang, Alexandra Meliou, and Eugene Wu. Qfix: Diagnosing errors through query histories. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 1369–1384. ACM, 2017.

[25] Eugene Wu and Samuel Madden. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6(8):553–564, 2013.

Disposal by Design

Susan B. Davidson
University of Pennsylvania
susan@cis.upenn.edu

Shay Gershtein
Tel Aviv University
shayg1@mail.tau.ac.il

Tova Milo
Tel Aviv University
milo@cs.tau.ac.il

Slava Novgorodov
eBay Research
snovgorodov@ebay.com

Abstract

The flood of data that has enabled breakthroughs in medicine, commerce, transportation, science and society also threatens to overwhelm our storage capacities and our privacy. Due to the volume of data and growth of regulations governing its maintenance and use, it is essential to develop automatic disposal techniques to manage this flood. We present a vision for automating data disposal – disposal by design – which takes into account processing constraints, regulatory constraints as well as storage constraints, and give three concrete examples which address aspects of this vision. Two of the examples address current needs in e-commerce, while the third suggests how to use machine learning to find summaries of relational data. We then discuss the research challenges that remain to provide a holistic solution to disposal by design.

1 Introduction

We are experiencing an amazing data-centered revolution in almost every aspect of our lives. Huge amounts of data are being generated, collected, transformed, integrated and analyzed, leading to breakthroughs in medicine, commerce, transportation, science and society. This data-centered revolution is fueled by the massive amount of data that is constantly being generated, and at the same time is threatened by the very same information flood. First, the size of our digital universe is growing exponentially, and it is estimated that, despite continuous advances in storage technology, the demand for storage will outstrip storage production *by an order of magnitude* by as early as 2025 [8]. If we do not learn how to effectively dispense with some of this data we will drown. Second, uncontrolled data collection endangers security and privacy, as recognized, e.g., by the recent EU Data Protection Regulation (GDPR) [1].

This problem has become even greater following the growth of COVID-19 data that has been collected and distributed. Data disposal policies must be systematically developed and enforced to benefit and protect organizations and individuals. Due to the volume of data involved and the growth of regulations/policies governing its maintenance and use, it is essential to develop automatic data disposal techniques that take these policies into account to control the information flood.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

A primary advantage of automating data disposal is the ability to focus resources on legitimate, valuable information, thereby enabling the creation of new products and solutions. The potential for cost-savings associated with reduced data volumes also provides an attractive solution to companies with tight budgets who cannot afford to store and maintain unlimited data. Furthermore, effective enforcement of data retention, deletion and privacy regulations allows legal and business requirements to be met.

However, it is obviously not enough to consider just policies when disposing of data: business processes must also be guaranteed to have the data or information available at the level of detail that they need. This leads to the use of ideas such as data sketching and summarization, if the data cannot be outright deleted. However, data sketching and summarization techniques have only been developed for specific aspects of the problem, mostly related to query answering over structured data [4, 6, 29, 2, 20]. We are still very far from a comprehensive solution that covers the entire data analysis pipeline (e.g. data cleaning, integration, sharing, and analysis, in addition to querying) and the breadth of data types (e.g. images and text, in addition to structured data). Every single data disposal solution therefore has to address, almost from scratch, the same tough challenges. Furthermore, these ad hoc solutions, even when successful, are application specific and rarely shareable.

Retaining the information hidden in the data while respecting storage, processing, and regulatory constraints is a therefore a major challenge. The difficulty stems from the separate, detailed requirements that each of these type of constraints entails. For instance, satisfaction of *processing constraints* is essentially an optimization problem where one needs to determine what data may be discarded and which summary of it to retain, if needed, so that data utilization is minimally harmed. In contrast, *regulatory constraints* tell us what information must be deleted (or kept for a specific time period [28]), and the challenge is to identify and discard (or retain) all the relevant data.

To pave the road for sustainable big data management, in this paper we set forth a research vision for automating data disposal which takes into account processing constraints, regulatory constraints, as well as storage constraints which we call *disposal by design*. To accomplish this vision, we must develop the formal scientific foundations for massive-scale data disposal. This encompasses the development of a formal model that captures all the diverse facets of data disposal: retention constraints, heterogeneous data, and data processing pipelines. It means developing reasoning capabilities over the data processing pipelines to ensure that they work over the retained, possibly summarized, data, and that this can be done in a dynamic manner as retention constraints, data, and pipelines are added, deleted or modified. Such a principled approach is essential for developing reusable solutions, and thereby sustaining the data-centered revolution that is transforming our lives.

We start in Section 2 by giving an architecture for our envisioned disposal by design (DbD) framework. In Section 3 we give three concrete examples which address subproblems of this framework in the context of e-commerce, image and relational data. We summarize some of the research challenges in Section 4, and conclude in Section 5.

2 Vision

The architecture of our envisioned framework is shown in Figure 1. DbD takes as initial input a set of *constraints*, C; a set of heterogeneous data sets D; and a set of data analysis pipelines, A. The constraints include data that must be retained (e.g. photos that must appear on a product page due to licensing agreements, or bank records that must be retained for five year), data that can or must be removed (e.g. due to GDPR regulations or bank records that are older than five years), as well as overall space constraints. The data is heterogeneous, and includes tables, images, text, etc. It is annotated with provenance, size and a notion of accuracy. The analysis pipelines could be ML packages, code, queries, workflows, etc, and are annotated with accuracy constraints on their input data. Note that the inputs to the analysis pipelines could be descriptive rather than prescriptive, i.e. there could be choices of different data (of various quality) that could be used as an input.

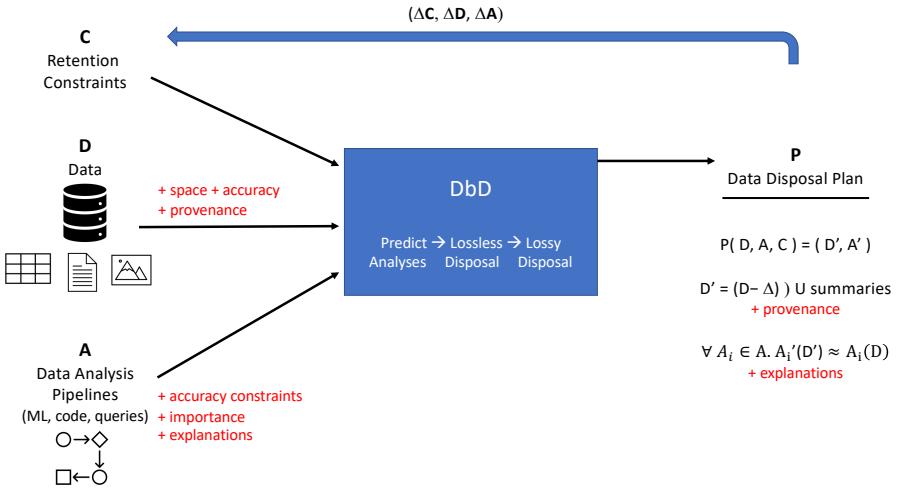


Figure 1: Architecture of DbD

The optimization goal used by DbD is some function of space and accuracy. One example of an optimization goal is to meet some overall space constraint while trying to satisfy the accuracy constraints of the data analysis pipelines. Another example is to guarantee that the accuracy constraints are satisfied and minimize the overall space.

In meeting the optimization goal within DbD, several strategies could be considered: First, attempting to predict future analyses and data usage to avoid disposing of data that might be needed in the future (see, e.g. [23, 12, 13]). Second, discarding any *unnecessary* redundant data that is not needed for any potential analysis pipelines or required by retention constraints. Similarly, unnecessary analysis pipelines that are not required by retention constraints and have not been used for a sufficiently long time could be discarded. We call this a "lossless" disposal since it disposes of data/analyses that are not needed. Third, making harder decisions about how to decrease the quality of potentially *necessary* data while still ensuring that the accuracy constraints on data analysis pipelines are met. We call this a "lossy" disposal since it degrades the quality of data and analyses.

The output of DbD is a data disposal plan, which includes a modified set of data sets, *D'*, in which some data have been removed (Δ) and possibly replaced by less accurate (but smaller) summaries, and a modified set of analysis pipelines, *A'*, in which some may have been removed, modified (e.g. to work over lower accuracy data), or added (e.g. in anticipation of future analysis needs). The output must guarantee that 1) the retention constraints *C* are met, and 2) the accuracy of the files of the data set is sufficient to meet the accuracy constraints on the data analysis pipelines. The output should also able to provide explanations, e.g. for how summaries were obtained and how they meet the accuracy requirements of relevant data analysis pipelines. Since retention constraints, data sets, and data analysis pipelines may be changed, added or removed over time, DbD must adaptively recompute over these changes (indicated by the loop back).

3 Examples of Specialized Instances

We mentioned in the introduction that there has been a lot of work on specific aspects of data disposal. In this section, we highlight two examples from e-commerce that deal with identifying a (bounded size) subset of items of highest utility, out of a large set of items: the first considers a catalog of items for sale [16] and the second deals with image data [9]. In these two cases, the focus is on (a non-traditional form of) data and the workloads are very simple (essentially, select queries). They also highlight how the inputs to DbD can be efficiently computed rather than by relying on humans. Interestingly, the setting in these two works is such that the non-selected items/images are not necessarily disposed off but may rather be stored, e.g., in a secondary storage. Nevertheless the algorithms are oblivious to whether or not a copy of the non-selected items is retained somewhere, and can be employed in both scenarios.

The third example that we present in this section illustrates some work in progress which focuses on (more traditional) relational data and (more complex) aggregate queries, and suggests a less traditional approach to summarization using deep learning.

3.1 E-Commerce Catalog Reduction

Many e-commerce platforms, such as eBay, serve as an intermediary between companies and consumers, receiving a commission per purchase. To increase sales, these platforms tend to offer as many items as possible. However, in many situations a reduced subset of the items should be offered for sale, e.g., when opening an express delivery branch, starting operations in a new region, or disposing of redundant items to improve data quality and decrease maintenance costs. In all these cases, it is imperative to select a reduced inventory that maximally covers consumer needs. In this problem (which we formalized in [16]), given a large set of items (D), a bound on the number of items that can be retained (C), and consumer preferences in terms of items popularity and suitability as alternatives (A), the goal is to select a reduced inventory that maximizes the likelihood of a purchase (that is formalized exactly based on A).

A naïve, yet popular, solution is to focus on the top-selling items. This however ignores the hidden relations between items and, in particular, the tendency of shoppers to buy, in the absence of an item they are looking for, a satisfying alternative. Instead, this problem can be modeled via a dedicated weighted directed graph, where the nodes are the items, their weights are the item's popularity (which is calculated based on provenance, i.e. the navigation patterns in the e-commerce platform website), and the weighted edges model to what extent an item may serve as a substitute for another (this can also be derived using provenance, via a statistical analysis of consumer data and purchase records that are available to all e-commerce platforms.). One can prove that this problem is NP-hard. Moreover, since in practical settings the overall number of items and the bound on the reduced item set are very large - in the order of magnitude of millions - a highly scalable algorithm is needed.

To solve this problem, we provide in [16] a highly parallelizable and scalable algorithm, that leverages our graph-based formulation of the problem, along with optimal approximation guarantees. Moreover, we have developed an end-to-end solution that fits the real-world e-commerce application and provide an extensive set of experiments demonstrating the efficiency and effectiveness of our solution.

Importantly, the model we defined for this problem is abstract and can be used to capture different use-cases outside of e-commerce settings. Each such application, however, requires a different method to derive the input. Specifically, one needs to assign a relative importance score to each item in the inventory, and to quantify the extent to which an item can serve as an alternative for another item.

3.2 Archiving Images in E-commerce

We now discuss an example of automating image selection that we are developing in collaboration with eBay, which is being tested for use in their product catalogs [9].

eBay has a huge archive of images of products some of which are used for display throughout a hierarchy of landing pages of product categories. For each page, there is a pre-defined subset of images that are relevant for the product category, out of which a small set are displayed. Each image may be relevant for a large number of different pages, and its value may differ between pages. Some of the images are required to appear on certain pages due to legal contracts (a *retention constraint*). Finally, the landing pages themselves may vary in importance, reflecting the relative popularity of product categories. To speed up the page display, it is desirable to maintain a smaller size image repository (which may be viewed as a bounded-size, fast-access cash), much smaller than the size of the full archive. The optimization problem is to find a "good" set of images that can fit in the bounded-size repository (cache) and meet the content and policy requirements of each of the landing pages.

In this application, *C* are simple constraints which state that some set of images must appear on landing pages due to legal contracts, as well as the size of the cache. *D* is the set of all photos in the archive; size and provenance are metadata associated with each photo. *A* represents the set of landing pages, each of which has a title (e.g., "Nike red shirts", "Samsung smartphones" or "shoes"). A landing page can be thought of as a simple query which identifies the set of all photos in the archive that could be used on the page.

Importantly, information associated with *D* and *A* can be automatically obtained. For each landing page, the relevance (accuracy) of an image for the page can be computed based both on the quality of the image (we are currently using an internal eBay ML model [10]) and the relevance score of the product represented in the image (which can be calculated using the product title and search engine retrieval score). The relative importance of a landing page can be calculated based on the landing page popularity, i.e. the number of visits in the last 90 days, normalized by sum of all visits to all pages.

An important property in evaluating the quality (goodness) of a solution is the diversity of the set of photos displayed on each page. Therefore, the quality of a solution for this application is not only assessed by the relevance of each individual image to its assigned landing pages, but also using more collective assessments on the subset of images, ensuring that each subset covers the spectrum of products and product aspects relevant for the page. To this end, we use a notion of *similarity* between photos, which can be calculated using cosine similarity between image embeddings. The score of a solution for each landing page is then not only computed based on the relevance scores of the selected subset, but is also based on the similarity of the most similar selected photo to each non-selected photo. This ensures that the selected subset is representative of the entire set of relevant photos, which naturally maximizes diversity and minimizes redundancies. The objective function then becomes the weighted sum of the scores for each landing page, where the weights are the relative importances of the pages as discussed above.

It can be shown that a solution to this problem cannot be approximated beyond a $(1 - 1/e)$ factor, unless $P = NP$, via a straightforward reduction from the Maximum Coverage problem. We nevertheless give in [9] an efficient algorithm with a tight worst-case approximation guarantee, based on the fact that the objective function can be proven to be nonnegative, monotone and submodular, and using an extension of the standard iterative greedy algorithm given in [33].

We have evaluated our image archival solution within eBay on several product categories (separately, as each category is assigned different analysts and has its own space constraint). Initial reports indicate that our solution significantly reduces manual work; the business analysts reported performing only a small number of modifications to the suggested solution, taking much less time than creating the solution from scratch.

3.3 Sampling and Aggregating Relational Data

In the first two examples, the core of the solution is based a formal definition of a problem with an algorithmic solution. However, another increasingly common approach to solving problems is based on machine learning. We therefore give a simple example using relational data and aggregate queries, and show how to "learn" the best subset of data/metadata to store. Note that this is very much work in progress, based on recent work in [24, 25, 26].

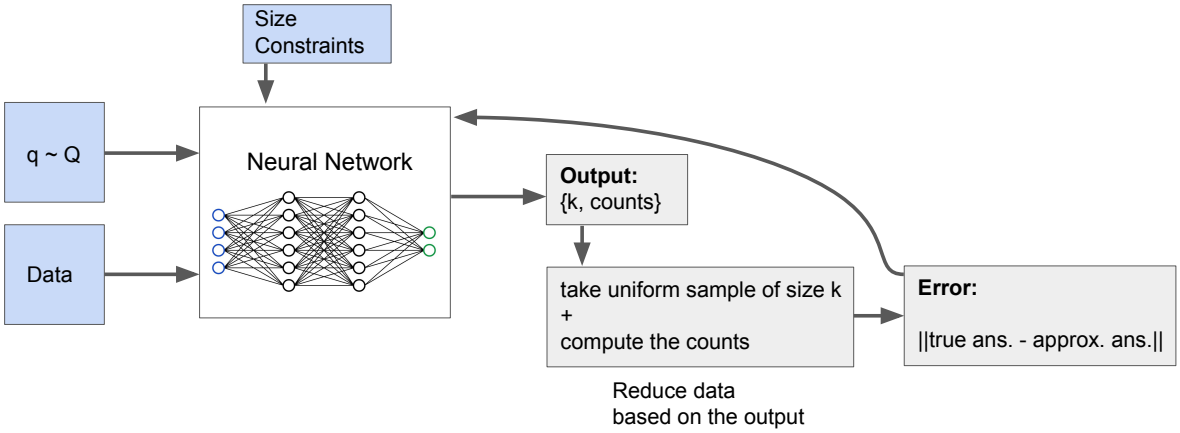


Figure 2: Architecture for learning the best subset of data/metadata to store

Given a set of relational data and input queries, the goal of our problem is to learn a subset of the data/metadata to store that takes significantly less space, such that it is still possible to compute, based on the retained information, approximate answers to an expected workload of queries with reasonable approximation errors.

In terms of the form of the retained data, we use a common approach: the retained information is a combination of (1) a uniform sample of the original data and (2) aggregates (i.e., results or partial results of aggregate selection queries) computed over the complete original data. There are several recent works proposing methods for deriving the best approximate answer to a query, given a sample and the aggregates (e.g., [24, 26, 27, 22]). Our focus, however, is on deciding, given an upper bound on the total space used by the retained data, how much of it should be used for each type of data (i.e., the sample or the aggregates), and which aggregates should be stored. Note that we do not need to decide which subset of the data should be stored in the sample, as it is a random uniform sample and the only relevant parameter is therefore its size.

To this end, we build a model for deriving a distribution Q on the expected queries by leveraging query workload information. This is achieved by a GAN-based solution [7], where you train a generator to produce artificial queries and a discriminator to distinguish between these queries and queries taken from a real-world query workload. The final distribution learned by the generator is Q.

We propose the following architecture, depicted in Figure 2, that is based on a deep learning approach: a learning model (neural network) receives as input the data, the space constraint (both are given in the initial setting), and a subset of sample queries generated over Q (these three components of the input to the network correspond to the three blue boxes in the figure). Given this input, the network then produces the size of the uniform sample, and the specific aggregates one should store (the box in the figure marked with the "Output" label). Given any such produced solution, the specified aggregates and a uniform sample of the specified size are aggregated, and their utility is evaluated by the resulting approximation errors on a large sample of queries (also sampled from Q). Note that the methods for computing the approximate answer uses only the reduced data. The model learns over time to minimize these errors, as marked by the back-arrow pointing from the "Error" box to the network. We note that the format of the output of the network (i.e. a sample along with a set of aggregates) is the same as the format of the input in the recent work of [26] for deriving approximate query answers over incomplete databases, and we also use the same evaluation methods for assessing the overall accuracy of the approximate answers (the query answers derived from the partial data) as used in their empirical analysis.

4 Research Challenges

While the examples in the previous section address specific aspects of disposal by design, we are still far from providing a holistic solution. In particular, the examples do not capture the complexity of workloads, variety of data, and dynamic aspects of a full solution. Thus in more complex settings, a solution would need to be redesigned from scratch. Therefore we argue that to solve the problem in a principled way, two scientific challenges must be addresses: designing a unified model, and dynamic incremental computation. We discuss each of these below, along with the enabling technologies that may be helpful.

4.1 A Unified Model

In order to develop a comprehensive solution to data disposal rather than a series of one-off solutions, we need a unified formal model for all types of data disposal. A prime motivation for using a unified model is that it will allow us to consider data sharing and replication between applications, as well as to re-use solutions between applications. As illustrated in Section 2, central to the model are notions of data coverage and quality, data usage and workload, storage and regulatory constraints, and evolution (of data, analysis pipelines and retention policies) over time.

Furthermore, we believe that the framework should be *declarative*. To understand the benefit of a declarative framework, consider the origins of modern relational database management systems [5]. A relational database is essentially a First-Order Logic machine which manages data. More precisely, a non-specialist can specify some needs declaratively, in first-order logic terms. The system then compiles such a “logical query" into an “algebraic query plan" that is optimized and evaluated. Relational systems thus perform automatic reasoning to handle queries and views, e.g., to rewrite queries into equivalent ones for optimization. Reasoning is also present in many other aspects of relational DBs, e.g., dependencies (logical formulas over the data that the system should enforce) and triggers (i.e. active rules). In relational systems, such reasoning is in some sense “hard-wired". For instance, several algebraic query plans may be possible for a given logical query. The system is in charge of verifying that the plans considered are indeed equivalent to the original query. To do that, the system assumes some laws governing the interaction of operations and implements an algorithm (some reasoning) to check that these laws are not violated. The laws are decided in advance and the reasoning is encoded in algorithms whose correctness has been proven in advance (e.g., the commutativity of joins).

Similarly, we want an intelligent interface between the data and its disposal process. For data disposal, logic is needed to reason about how data relates to analysis tasks, to describe data disposal and retention constraints and preferences/criteria (the equivalent to the “hard-wired" logic of relational systems) and to model analysis workloads and properties of data summaries. The laws governing data coverage and quality, data usage and workload, storage and regulation constraints, and evolution (of data and retention policies) over time must therefore be “first-class citizens” of the model rather than inflexible, preconceived laws (see, e.g., [30]). Furthermore, the reasoning behind data disposal must be able to take into account the specific context and application domain.

Two key enabling technologies may be helpful in developing this unifying model: data provenance, and sketching and summarization.

Data Provenance Provenance traces the source of information and the computational process it undergoes, and is critical for understanding data usage, for explaining query results and for assessing their validity [3, 17]. In our setting, provenance must further capture what data has been omitted and what kind of summary was retained for it. Such meta-data will allow for effective computation over the retained information and for explanation/justification of the obtained results. It will also allow the system to keep a record of the processing activities, as required by regulations such as GDPR [1].

Attaching provenance to individual data items is typically straightforward; what is more complex and needs to be studied is what provenance should be used to annotate summarized or omitted data. The difficulty stems from the fact that common provenance models often apply only to a fairly restricted set of declarative data manipulation operators (e.g. SQL) whereas data summarization/deletion strategies are typically given as general programs, possibly empowered by machine learning (ML) tools [29, 20], for which provenance is not yet well defined. A second, related, difficulty is to propagate such meta-data through the analysis process, which may be complex. Finally, an additional challenge is that the size of provenance may be very large [14, 1], and so effective disposal must be recursively applied to this meta-data as well.

Sketching and Summarization Much of the data that we keep around is redundant and can be discarded with no harm. Common examples are old document drafts and ancient emails dealing with non important issues. But in general, whether or not data should be kept, and at what level of granularity, depends on the expected analysis workload. (as well as the regulatory constraints). For a simple example, the analysis could compare multiple versions of the same document, in which case discarding old (legitimate) versions of documents without retaining some record of the changes between documents is harmful. A key challenge thus is to dynamically determine which data should/may be discarded and, if allowed, what summary should be kept for the deleted items, so that data utilization is minimally harmed.

Traditional techniques for data summarization, sketching and statistical/approximate query processing are mostly concerned with specific *query* operations and provide *accuracy guarantees w.r.t query answers*. But modern data analysis pipelines are far more complex, consisting of multiple analysis steps, including data integration, cleaning, restructuring, visualization, and machine learning (ML) [18]. Naturally, it would be desirable to build on existing technology when possible. For instance, in stream data processing, incoming data is summarized on the fly using powerful sketching techniques, then discarded all together [6]. Other, more comprehensive, summarization methods analyze the full dataset with techniques ranging from dimensionality reduction to compression-based data reduction methods and algorithms for clustering, data deduplication, redundancy elimination, and implementation of network (graph) summarization concepts [29, 21]. Each of these approaches, however, has been designed for a specific task and no single technique is guaranteed to always achieve superior results - performance depends on the type of data and its intended usage. A difficulty in assembling them together is that the summarization policies are often hard-coded and, consequently, are inflexible and difficult to combine and optimize. This is why it is important to have a declarative framework that captures the inputs described in Figure 1: retention constraints, data and accuracy metrics, and data analysis pipelines along with accuracy constraints governing the choice of resulting summary properties. This would serve as input for an engine that derives disposal policies that adhere to the needs (if possible), execute them, and efficiently run data analysis tasks over the retained information. Some early encouraging results on the use of declarative specification for data disposal have been presented in [31] in the context of OLAP queries over relational data. The challenge is to adapt these ideas to the execution pipelines of the modern world of big-data.

4.2 Computation and Optimization

A declarative specification describes *what* the requirements are in a disposal policy and what a query result should look like, but not *how* to efficiently derive or enforce the policy and evaluate data analysis pipelines over the retained information. Thus, we must support the framework by algorithms for policy derivation and enforcement and for query evaluation, and use optimization techniques to guarantee good performance. Furthermore, these techniques must take into account the feedback loop shown in Figure 1, suggesting the use of incremental methods (not illustrated in the examples of the previous section). Several enabling technologies are relevant here: preemptive computation, approximate query processing, incremental computation, incremental view maintenance, and machine learning.

Preemptive computation. Computing a comprehensive summary for deleted data may be a time consuming task especially when executed over large databases [29]. This may be problematic if postponed to the last minute when the storage overflows and data must be deleted. To overcome this, one must design methods for anticipating and preempting some of the summaries offline, e.g. in the form of a partial summary. The preempted computation could then be used for constructing a complete summary when data must be discarded. Similarly, to speed up the enforcement of privacy regulations such as GDPR, preemptive duplicate detection and object resolution may be used to identify replicated data that is relevant to entities that are required to be "forgotten".

Approximate query processing. The ability to provide approximate answers to queries, at a fraction of the cost of executing the query in the traditional way, has made approximate query processing extremely popular in big data applications. Sampling is a common tool in such systems. In query-time sampling, the query is evaluated over samples taken from the database at run time. Employing such techniques in our context requires extending the sampling mechanisms to sample also from the data summaries (which provide an aggregated description of the missing data elements) and, correspondingly, incorporating such samples in query evaluation. To achieve a sharper reduction on response time, some approximate query processing algorithms draw samples from the data in a pre-processing step, then use them to process incoming queries. An intriguing question is whether such precomputed samples can themselves serve as data summaries in our context, thereby allowing to discard some (or all) of the remaining items.

Incremental computation. Incremental computation is present in all dimensions of data reduction. As new data comes in and new disposal policies are employed, the retained information must be incrementally maintained. This includes incremental cleaning and integration of the new incoming data into the partial, retained information. It also includes maintenance of data summaries as new data arrives and other gets deleted. Updates should also be propagated to the provenance information used to explain the origin of the different query answers. Promising approaches to this problem include incremental view maintenance techniques [19], possibly combined with incremental machine learning [34, 35], as explained below.

Incremental view maintenance The retained information may be abstractly regarded as a *view* over the full (missing) data. Query evaluation over views has been extensively studied in the literature. Techniques for rewriting queries to be answered, as accurately as possible, using the views alone, are relevant to our context where queries may be evaluated only using the retained data [11]. Incremental view maintenance has also been extensively studied in the literature [19] and the results are relevant here; the relationship to incremental machine learning will also need to be investigated [35, 34]. A challenge particular to our setting, however, is that not only that the data is updated (as more data is accumulated) but also the view definition itself (what data is retained/summarized and how) may change as a result of the accumulated knowledge and/or changes in the workload and regulation.

Learning In our context, ML techniques are not only part of the analysis pipeline but may be employed to learn data access and usage patterns and, correspondingly, to derive effective retention policies complying with a given set of regulations. Deep learning has been used for various tasks in databases, including automatically identifying the "interesting" parts of data [12, 32] and designing effective indexing structures, and we suggest earlier in Section 3.3 how it could be used for summarization in the context of relational data. An added challenge in our context is that the retention policies and summaries may need to be dynamically adapted as more data comes in (or is disposed of) and the data analysis workloads change.

5 Conclusions

In this paper, we highlight the importance of developing a holistic solution to managing the flood of data that is enabling amazing breakthroughs in medicine, commerce, transportation, science and society while simultaneously threatening to outstrip storage production and endanger our privacy. We present a vision for managing this flood, called *disposal by design*, which takes into account the data, regulatory constraints, as well as the (potentially complex) data analysis pipelines that operate over the data and creates a data disposal plan in which data may have been removed and replaced by summaries, and analysis pipelines may have been modified to work over the modified data. At the core of this vision is a unified declarative model of data, constraints and processes, supported by algorithmic/learning techniques that develop the data disposal plan. We stress the importance of enabling technologies such as data provenance, sketching and summarization, preemptive computation, approximate query processing, incremental computation, incremental view maintenance, and machine learning. We believe that such a principled approach is essential for developing solutions that can be reused across applications, replacing ad hoc solutions that must be developed from scratch, and thereby sustaining the data-centered revolution that is transforming our lives.

Acknowledgment This work has been partially funded by the Israel Science Foundation, BSF - the Binational US-Israel Science foundation, Tel Aviv University Data Science center, and the Mortimer and Raymond Sackler Institute of Advanced Studies

References

[1] E. Ainy, P. Bourhis, S. B. Davidson, D. Deutch, and T. Milo. Approximated summarization of data provenance. In *Proc. of CIKM*, pages 483–492, 2015.

[2] M. Besta and T. Hoefler. Survey and taxonomy of lossless graph compression and space-efficient graph representations. *CoRR*, abs/1806.01799, 2018.

[3] J. Cheney, L. Chiticariu, and W. C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

[4] S. Chaudhuri, B. Ding, and S. Kandula. Approximate query processing: No silver bullet. In *Proc. of SIGMOD*, 2017.

[5] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

[6] G. Cormode. Data sketching. *Communications of the ACM*, 60(9):48–55, 2017.

[7] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[8] Dataage 2025 - the digitization of the world, seagate us. http://www.seagate.com/our-story/data-age-2025.

[9] Susan Davidson, Shay Gershtein, Tova Milo, Slava Novgorodov, and May Shoshan. PHOcus: Efficiently Archiving Photos. https://slavanov.com/research/phocus-demo.pdf.

[10] Arnon Dagan, Ido Guy, and Slava Novgorodov. An image is worth a thousand terms? analysis of visual e-commerce search. In *SIGIR*, 2021.

[11] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[12] Ori Bar El, Tova Milo, and Amit Somech. Automatically generating data exploration sessions using deep reinforcement learning. In *Proc. of SIGMOD*, pages 1527–1537, 2020.

[13] Ori Bar El, Tova Milo, and Amit Somech. Towards autonomous, hands-free data exploration. In *10th Conference on Innovative Data Systems Research, CIDR 2020, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings*. www.cidrdb.org, 2020.

[14] B. Glavic and G. Alonso. Perm: Processing provenance and data on the same data model through query rewriting. In *Proc. of ICDE*, pages 174–185, 2009.

[15] General Data Protection Regulation (GDPR). https://en.wikipedia.org/wiki/General_Data _Protection_Regulation.

[16] Shay Gershtein, Tova Milo, and Slava Novgorodov. Inventory reduction via maximal coverage in e-commerce. In *Proc. of EDBT*, pages 522–533, 2020.

[17] T. J. Green and V. Tannen. The semiring framework for database provenance. In *Proc. of PODS*, pages 93–99, 2017.

[18] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi. Big data and its technical challenges. *Communications of the ACM*, 57(7):86–94, 2014.

[19] C. Koch, D. Lupei, and V. Tannen. Incremental view maintenance for collection programming. In *Proc. of PODS*, pages 75–90, 2016.

[20] M. L. Kersten and L. Sidirourgos. A database system with amnesia. In *Proc. of CIDR*, 2017.

[21] Y. Liu, T. Safavi, A. Dighe, and D. Koutra. Graph summarization methods and applications: A survey. *ACM Comput. Surv.*, 51(3):62:1–62:34, 2018.

[22] Xi Liang, Stavros Sintos, Zechao Shang, and Sanjay Krishnan. Combining aggregation and sampling (nearly) optimally for approximate query processing. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1129–1141, 2021.

[23] Tova Milo and Amit Somech. Automating exploratory data analysis via machine learning: An overview. In David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 2617–2622. ACM, 2020.

[24] Laurel J. Orr, Samuel K. Ainsworth, Kevin G. Jamieson, Walter Cai, Magdalena Balazinska, and Dan Suciu. Mosaic: A sample-based database system for open world query processing. In *10th Conference on Innovative Data Systems Research, CIDR 2020, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings*. www.cidrdb.org, 2020.

[25] Laurel J. Orr, Magdalena Balazinska, and Dan Suciu. Entropydb: a probabilistic approach to approximate query processing. *VLDB J.*, 29(1):539–567, 2020.

[26] Laurel J. Orr, Magdalena Balazinska, and Dan Suciu. Sample debiasing in the themis open world database system. In David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 257–268. ACM, 2020.

[27] Jinglin Peng, Dongxiang Zhang, Jiannan Wang, and Jian Pei. Aqp++ connecting approximate query processing with aggregate precomputation for interactive analytics. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1477–1492, 2018.

[28] Data retention. https://en.wikipedia.org/wiki/Data_retention.

[29] M. H. Rehman, C. S. Liew, A. Abbas, P. P. Jayaraman, T. Y. Wah, and S. U. Khan. Big data reduction methods: A survey. *Data Science and Engineering*, 1(4):265–284, 2016.

[30] Subhadeep Sarkar, Jean-Pierre Banâtre, Louis Rilling, and Christine Morin. Towards enforcement of the EU GDPR: enabling data erasure. In *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), iThings/GreenCom/CPSCom/SmartData 2018, Halifax, NS, Canada, July 30 - August 3, 2018*, pages 222–229. IEEE, 2018.

[31] J. Skyt, C. S. Jensen, and T. Bach Pedersen. Specification-based data reduction in dimensional data warehouses. *Inf. Syst.*, 33(1):36–63, 2008.

[32] Amit Somech, Tova Milo, and Chai Ozeri. Predicting "what is interesting" by mining interactive-data-analysis session logs. In *Proc. of EDBT*, pages 456–467, 2019.

[33] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[34] Yinjun Wu, Edgar Dobriban, and Susan B. Davidson. Deltagrad: Rapid retraining of machine learning models. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10355–10366. PMLR, 2020.

[35] Yinjun Wu, Val Tannen, and Susan B. Davidson. Priu: A provenance-based approach for incrementally updating regression models. In David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 447–462. ACM, 2020.

Building Deletion-Compliant Data Systems

Manos Athanassoulis, Subhadeep Sarkar, Tarikul Islam Papon, Zichen Zhu, Dimitris Staratzis
Boston University

Abstract

Most modern data systems have been designed with two goals in mind – fast ingestion and low-latency query processing. The first goal has led to the development of a plethora of write-optimized data stores that employ the out-of-place paradigm. Due to their write-optimized design, out-of-place data systems perform deletes logically via invalidation, and retain the invalid data for arbitrarily long. However, due to the recent enactment of new data privacy regulations, the requirement of timely deletion of user data has become central. The right to be forgotten (in EU's GDPR), right to delete (in California's CCPA and CPRA), or deletion right (in Virginia's VCDPA) mandates that service providers persistently delete a user's data within a pre-set time duration. Logical deletion in out-of-place data systems, however, does not offer guarantees for timely and persistent deletion, and attempting to enforce it using existing tools leads to poor performance and increased operational costs.

In this paper, we present a new framework for building deletion-compliant data systems from a holistic perspective. We analyze the new regulations and the requirements derived from the new policies, and we propose changes in the application and the system layer of data management. We outline the new types of deletion requests that need to be supported, the query language modifications needed to be able to request for timely persistent data deletion, and the system-level changes needed to realize timely and persistent deletes. The proposed framework for deletion compliance lays the groundwork for a new class of data systems that can offer system-level guarantees for user data privacy. We present recent results spanning all layers of the framework: the requirements and the application layer target any database system, while the system layer discussion is geared towards out-of-place systems. Finally, we conclude with a discussion on next steps and open challenges on building deletion-compliant data systems.

1 Introduction

Data-intensive social and commercial applications and new advancements in domains like Internet-of-things, edge computing, 5G communications, and autonomous vehicles, generate a vast amount of *personal data* processed by several data companies [26, 40]. The increasing demand for efficient collection, storage, and processing of user data over the past two decades, has driven the development of data systems that can *sustain high ingestion rates* without compromising the ability to *access and analyze the data quickly*.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Out-of-Place Systems. The need for optimizing data ingestion while maintaining efficient data access has led to the prominence of the *out-of-place* paradigm, which fulfills these goals by minimizing the interference between reads and writes. Today, several commercial relational and array-based data stores [13, 34, 39, 46, 50, 53, 59, 71, 85] and NoSQL data stores [10, 12, 11, 32, 38, 42, 48, 79] have adopted the out-of-place paradigm.

Relational and Array-based Systems. Relational systems that buffer updates before applying them lazily on the base data, essentially, follow the out-of-place paradigm. The columnar and array data stores implemented by Vertica [59, 85], SciDB [68, 86], and TileDB [67, 88] use an in-memory storage component that stores incoming inserts, updates, and deletes out of place, and applies the changes lazily on the disk-resident data. Similarly, the state-of-the-art column-store system MonetDB [50] uses an in-memory positional index for incoming data [46], and SAP HANA uses a delta store per table to facilitate fast ingestion without affecting its read-optimized data layout [39]. Finally, several research-prototype systems use a separate delta store on faster storage (e.g., SSDs/NVM) to offer efficient access to incoming data [13, 14, 34, 53, 71].

NoSQL Systems. More than relational systems, production-grade NoSQL key-value stores predominantly employ the out-of-place paradigm, frequently based on the log-structured merge (LSM) paradigm. An LSM-tree is a heavily write-optimized out-of-place data structure that maintains several on-disk components, which can be viewed as several out-of-place delta stores [66, 31, 49, 62, 73, 92]. Key-value stores such as RocksDB [36, 38] and LevelDB [43] at Facebook, BigTable [24] at Google, X-Engine [48, 91] at Alibaba, Voldemort [61] at LinkedIn, DynamoDB [32] at Amazon, Cassandra [12], HBase [11], and Accumulo [10] at Apache, and bLSM [79] and cLSM [42] at Yahoo are based on the log-structured merge (LSM) paradigm. Other out-of-place architectures employed by NoSQL systems are B^+-tree, B^ϵ-tree, and fractal tree-based storage engines with *buffering support*, such as COLA [16], TokuDB [58], and B*e*rtFS [17, 51].

Cloud-based Systems. Cloud-based systems naturally employ the out-of-place paradigm as they rely on the immutability of cloud storage. Hence, systems like Amazon Redshift [8, 44], Cloud Data Platform [28] at Snowflake, and Delta Lake [29, 30] at Databricks employ variations of the out-of-place paradigm in the interest of performance. Deletes and updates are initially performed logically and are gradually propagated to persistent media through periodic merging with base data.

Deletes in Out-of-Place Systems. A key property of out-of-place systems is that they treat deletes (and updates) similarly to inserts, i.e., instead of deleting (updating) entries in-place, they insert a new version of the entry to be deleted that *logically invalidates* the target entries. These special entries that are responsible for logical deletes are termed *delete markers* [59] or *tombstones* [36, 76].

Logical data deletion is a quintessential out-of-place operation, but it does not guarantee *purging* of the data under deletion within a definite timeframe. Rather, the data is marked as invalid; essentially, *not accessible* to external users. In practice, logically deleted entries are kept for arbitrarily long in the system, since the time to definitively delete the data (termed *persistent deletion*) depends on the state of the system, and not on when the user request expects the data to be deleted [76]. In fact, most out-of-place data stores are built with the underlying assumption of *perpetual data retention* in order to gain more insights from the user and organizational data [90], hence *timely persistent deletion* has not been part of their design goals. In addition to deletes, logical updates in out-of-place systems are applied lazily too, however, the implications of out-of-place deletes are critical in terms of the privacy regulations, and thus, are our focus.

1.1 Problem: The Privacy Concern

Cost of Logical Deletes. Logical deletes and updates in out-of-place systems boost ingestion performance, however, they come at a significant cost. In fact, when tasked with deleting user data persistently in a timely manner, out-of-place systems suffer both in terms of (a) data privacy protection and (b) the overall system performance. Such systems are designed to retain the logically invalidated data indefinitely, and the time required for persistent removal of the physical data entries depends on (i) the data layout, (ii) the data re-organization policy (e.g., node splitting/merging in B-trees, compaction in LSM-trees, consolidation in TileDB), (iii) the

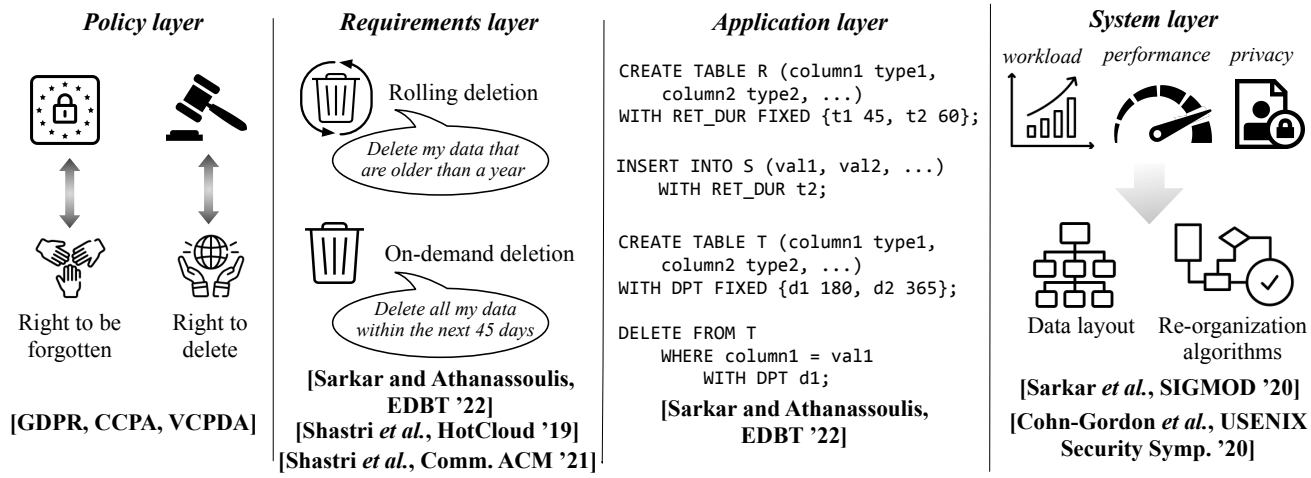


Figure 1: The four layers of deletion-compliant data systems.

design of the storage engine (such as the fanout of a tree and the size of a database), and (iv) the composition and distribution of the workload – factors that are *beyond the control of the application and system developers or administrators*. Thus, most out-of-place systems are unable to provide any latency guarantees for persistent deletion of user data [76].

The Legal Frontier. In recent years, a number of government-driven efforts across the globe unfolded, aiming to protect the privacy of user data and give back to the users the control of their personal data. On the legal side, regulations such as the EU's GDPR [26], California's CCPA [3] and CPRA [6], and Virginia's VCDPA [7] have been introduced, which mandate that data companies ensure *privacy through deletion* [83, 84]. GDPR's *right to be forgotten*, CCPA and CPRA's *right to delete*, and the *deletion right* in VCDPA particularly focus on *persistent deletion of user data on-demand and in a timely manner* [9, 35, 41, 52, 75, 83, 24, 89].

The Technological Roadblock. Treating *deletes* as *first-class citizens* is new for the data systems community, and it would require a significant amount of work to transform classical systems to be efficient deletion-wise. Even today, it continues to be a critical technological challenge for the biggest of data companies using state-of-the-art storage engines to demonstrate compliance with the deletion regulations and to efficiently delete user data on-demand [81, 29, 84]. To translate this into numbers, between January 2020 and January 2022, the penalties under GDPR paid by data companies amounted to more than $1B, which includes large contributions from companies such as Amazon ($877M), WhatsApp ($255M), Google Ireland ($102M), and Facebook ($68M), H&M ($41M), British Airways ($26M), and Marriot ($23M) [64, 70, 87]. Thus, to demonstrate compliance, many companies end up performing expensive database-wide consolidations periodically (e.g., every few weeks), to ensure timely persistent deletion of user data [76, 77]. Such operations are remarkably expensive in terms of time and money, cause undesirable latency spikes, and hence, should be avoided.

1.2 Deletion-Compliant Data Systems

In this paper, we present our vision and first results on designing data systems that ensure data privacy through timely and persistent deletion of user data. Existing efforts that attempt to delete user/application data on-demand suffer in terms of performance as the underlying data layout and data management mechanisms are ill-suited for the purpose. We identify the missing links, in terms of technological tools, both at the application level and the system level, and we propose a hierarchical framework that enables our vision of privacy through deletion in out-of-place data systems (Figure 1).

Roadmap. The privacy through deletion framework is a roadmap toward building deletion-compliant data systems. We begin by outlining the challenges associated with each layer of our vision, i.e., in the context of (i) translating the legal mandates to user requirements, (ii) expressing the user requirements through a declarative API, and (iii) realizing the application-level requirements at the system level. Next, we identify and categorize the different classes of user requests for deletes in light of the legal regulations. Based on this, we present the challenges associated with transforming the classes of deletion requests into application-level specifications, and we propose an SQL extension as an example that can be extended to other query languages to support deletion of user data *periodically* and *on-demand*. Further, we outline the design and tools necessary at the system level to support the application-level requirements. Finally, we conclude with a discussion on how the proposed framework drives us toward building deletion-compliant data systems, and what further research challenges remain open to fully realize this vision.

2 From Regulation to Practice

The legal landscape for data privacy has changed drastically over the past few years, and governments across countries, as well as across different states in the US, have enforced acts and regulations to control the consumption of user data by service providers and give back to the users the control of their personal data. Translating the new regulations to new *user-data privacy-compliant system behavior* still faces significant challenges. In this section, we present in more detail the requirements from the regulation point of view, and we showcase through three realistic scenarios the limitations of the state-of-the-art data systems when tasked to implement these requirements.

2.1 Regulations on Timely Data Deletion

While the new regulations propose an array of new requirements, we particularly focus on the legal policies concerning data retention and data deletion, with the objective of ensuring *privacy through deletion*.

Right to be Forgotten, *EU GDPR*. The General Data Protection Regulation (GDPR) has revolutionized the data privacy and security landscape across the European Union countries [26]. One of the fundamental changes introduced through the GDPR (over the older Data Protection Act (DPA) that it replaced), is the *right to be forgotten*, which empowers the users with the *right* to request a service provider to delete all their personal data persistently from its domain. Such deletion requests may be presented either up-front or on-demand. The service provider must comply with those requests, unless it has compelling reasons for acting otherwise (Art. 17(3)).

Right to Delete, *CCPA*, *CPRA*. The California Customer Protection Act (CCPA), which will eventually be replaced by the California Privacy Rights Act (CPRA) in 2023, allows the users/consumers in California to request from service providers to permanently delete all data personal to the user [3, 6]. Under CCPA and CPRA, the service providers must acknowledge such a user request within 10 days, and respond to the request within 45 business days [19]. Persistent deletion must be performed by removing the target data across all domains, barring archive and backup systems, along with data anonymization as required.

Right to Delete, *VCDPA*. Similarly to CCPA, the Virginia Consumer Data Protection Act (VCDPA) empowers users in Virginia to exercise their right to delete their personal data from a provider's domain [7]. VCDPA requires the service providers to serve a delete-request from a user within 45 business days [19].

Right to be Forgotten, *UK GDPR, DPA*. The UK GDPR, along with the Data Protection Act (DPA) 2018 provides the country's citizens with similar rights about personal data deletion as the EU GDPR. The users are allowed to express their deletion preference verbally or in writing, to which the service providers must respond within 30 days [1, 4].

Other Efforts. Among other countries, Argentina [23, 69], Singapore [25], India [55], Canada [5], and South Korea [18] have some implementation of the right to deletion as a part of their respective privacy protection acts.

2.2 Limitations of the State of the Art

In light of the deletion regulations, we now present three real-life scenarios to highlight why state-of-the-art data systems are ill-equipped to support deletes efficiently without hurting performance. We do so by identifying the missing links in different hierarchical levels of the proposed privacy-through-deletion framework. Below, we illustrate (i) that the users are unable to express their preferences about deleting their personal data, (ii) why it is difficult for application developers to support the deletion requests from the users, and (iii) why it is difficult to realize persistent deletes in a timely manner in live production servers.

Scenario 1: *Alice* is a user of a smart-home ecosystem, *HomeComp*, which provides real-time services including video surveillance, remote temperature, and illumination control. *Alice* enjoys the services of *HomeComp*, but concerned about her personal data privacy, she wants *HomeComp* to permanently delete all her data older than 30 days on a rolling basis.

The problem? Like most service providers, *HomeComp*'s data model is built around the assumption of perpetual data retention; deletion of user data needs a human-in-the-loop that performs the necessary actions. Thus, *HomeComp* does not allow its user to request for rolling timestamp-based data deletion.

Scenario 2: *StreamEra* is a company that provides real-time insights for data streams, and allows its users to request on-demand deletion of their personal data, as it is bound by the *right to be forgotten*. *StreamEra* uses an SQL-based wrapper on top of its storage layer.

The problem? While *StreamEra* wants to serve its users by ensuring timely persistent deletion of their personal data, SQL does not provide support for such an operation. Instead, the backend engineers implement the user-requested deletion functionality at the application level in an ad-hoc manner as it is not native to SQL.

Scenario 3: A cloud-based online data analysis company *ClouData*, stores user data using immutable files within its HTAP data store. *ClouData* is bound by the *right to be forgotten*, and thus, has to delete all user data that are older than D days.

The problem? As the data organization on disk is not based on the ingestion timestamp and aims to accelerate read queries, it uses the most frequently queried attribute to partition. Hence, the objects qualifying for a timestamp-based deletion may be dispersed within the data store. As in-place deletion is not supported due to immutability, state-of-the-art data stores periodically consolidate the entire data set to delete all invalid entries. Ensuring privacy via this approach is costly in terms of disk writes and overall accesses, and causes latency spikes leading to performance unpredictability.

Other Challenges of Logical Deletes. In addition to not complying with regulatory requirements, logical deletes may cause more hurdles. Specifically, by retaining invalidated data (that should not be used anymore), a data company:

1. Wastes storage space and energy on data that cannot exploit in any way. Further, data maintenance results in additional write amplification that wears off the underlying storage devices [15].
2. Risks that a security leak will reveal user data that users expect to be deleted [64].
3. Hurts read performance, as its data management layer uses metadata and indexes for all data regardless of whether they are invalidated [76].

3 Privacy Through Timely Deletion

We now outline our vision toward developing deletion-aware data systems, which by design, are capable of *deleting user data persistently, and in an efficient and timely manner*. Toward this, we introduce a new set of application level and system level tools that capture, transform, and realize the user-requirements for deletes.

Figure 1 shows our four-layered approach. The first step is the *policy layer*, implemented by the governments, that enact specific clauses to protect data privacy through deletion. The second layer is the *requirements layer* that translates the regulations into application requirements. Next, we have the *application layer* that proposes the necessary changes in query languages to allow applications to easily express their constraints. Finally, the *system layer* implements efficient means for data deletion and demonstrates regulation compliance.

3.1 Requirements Layer

Challenge. This layer analyzes the regulations from the policy layer and categorizes the various requests the user *should be* able to make on the application layer. The impact of the newly enacted policies is on various aspects including accountability (audit), security (protect data access), and right of access (efficient accessing) [81, 84]. In this work, we focus on *storage limitation* ("data should not be stored beyond its purpose"), the *right to be forgotten* ("find and delete groups of data") [81], and how to transform them into concrete requirements.

Types of Deletion Requests. We codify the two types of data deletion requests as requirements for (a) retention-driven *rolling deletion* and (b) *on-demand deletion* both with a timely constraint [76], as illustrated in the second part of Figure 1.

Retention-driven deletes. In cases that the purpose of storing the data has expired, a rolling deletion should take place, which will ensure that the underlying data management solutions persistently delete this data, based on a pre-set *retention duration*. This duration can be governed by legislation, the specific application, or even user preference, hence it has to be tunable. To abide by the policies, the data management layer has to permanently delete expired items within a specific timeframe, provided by the service-level agreement (SLA) between the user and the service providers.

Deletion on-demand. The regulations for deletes also allow users to submit on-demand deletion requests for any personal data, upon which the service provider has to delete user data persistently. On-demand deletion requests can be submitted through an API provided by the service provider, and upon submission, all data for a user are purged persistently within a threshold period. This threshold for persistent deletion is also set by the provider following the regulations and is agreed upon in the form of an SLA-clause.

3.2 Application Layer

Challenge. With the deletion-related regulations translated to deletion requirements, the next step is to transform them into a format that is interpretable by the application layer. The interface of data stores is typically declarative query languages (e.g., SQL, GraphQL, DMX, LINQ, and N1QL) that support expressing complex queries as well as inserting new data, updates, and deletes. The missing link here is that state-of-the-art query languages do not have support for data deletion based on retention and does not have a way to express the timely deletion requirements.

Extending SQL. Hence, to implement the deletion requirements, we propose an extension to SQL [74] that includes support for timely deletion both in the data definition (DDL) and the data manipulation (DML) parts of the language, as summarized in the third part of Figure 1. The objective of the SQL extension is three-fold.

1. To support *retention-driven deletion*, we augment both the `CREATE TABLE` and `INSERT INTO` statements so that a relational table can be associated with a number of options for specific time-to-live (TTL). Every data object is bound to a specific TTL according to the application SLA or to user preference.

2. To ensure *timely persistence of on-demand deletion requests*, we augment the `CREATE TABLE` and `DELETE FROM` statements to allow a relational table to support a predetermined set of timely deletion guarantees, and each deletion to select the level of service to which it adheres.

3. Finally, we extend the `CREATE TABLE`, `INSERT INTO` and `DELETE FROM` statements to support *arbitrary delete thresholds* for retention duration and deletion persistence.

Enabling retention-driven deletes. To support retention-driven deletes, we extend `CREATE TABLE` to allow an application developer to specify several levels of retention duration as a table property.

```
CREATE TABLE R (column1 type1, column2 type2, ...)
WITH RET_DUR FIXED (t1 <ret1>, t2 <ret2>, ...);
```

The above `CREATE TABLE` statement creates a table `R` that supports retention-based deletes with specific retention duration of `ret1`, `ret2`, etc, which are mapped to symbolic representations `t1`, `t2`, etc. In general, the `WITH RET_DUR` clause is an optional clause when creating a new table, and will be necessary only for tables that need to support deletes with predefined retention duration values. In such cases, each `INSERT` statement can use (up to) one of the predefined retention duration values, say `t1` to classify the specific object as one to be deleted after `ret1` time. For example, a table that is configured to support retention duration of 30 days and 60 days (`CREATE TABLE R (...) WITH RET_DUR FIXED (t1 '30 days', t2 '60 days');`), can only receive inserts with retention duration `t1` or `t2`. An ingestion without a retention period explicitly mentioned, is kept perpetually following the logic of a classical insert. The syntax of an insert, now, has the optional `WITH RET_DUR` clause as follows.

```
INSERT INTO R (val1, val2, ...) WITH RET_DUR t<i>;
```

Support for arbitrary retention duration. To support arbitrary retention duration, we further add the `ARBITRARY` keyword to both the `CREATE TABLE` and `INSERT` statements. The support for arbitrary retention duration is necessary particularly for systems in a distributed setting that replicate data across physical data stores in different geolocations, each bound by different regulatory requirements. The full syntax of the proposed SQL extension for retention-based deletion is below.

```
CREATE TABLE R (column1 type1, column2 type2, ...)
WITH RET_DUR {ARBITRARY | FIXED (t1 <ret1>, t2 <ret2>, ...)};

INSERT INTO R (val1, val2, ...) WITH RET_DUR { <t> | t<i> } ;
```

Note that having a *pre-defined set of retention duration values* provides more information to the system compared to allowing arbitrary duration. As a result, it *allows the system to better prepare* to offer efficient retention-driven deletes. Conversely, we expect that data stores that aim to support arbitrary retention duration will face increased system-level challenges.

Enabling timely on-demand deletion. We further propose to augment SQL to express timely on-demand deletion. To do so, we introduce the concept of *delete persistence threshold* (DPT) [76], which denotes the maximum delay between a logical delete and its persistence. Every relational table can be associated with several such thresholds that are defined from the legal constraints or based on user preference. Similarly to retention-driven deletes, we also extend SQL to support arbitrary DPTs when the DPTs are not specified *a priori*. Below, we outline the modifications to the DDL and DML parts of SQL to support on-demand timely deletion requests.

```
CREATE TABLE S (column1 type1, column2 type2, ...)
WITH DPT {ARBITRARY | FIXED (d1 <dpt1>, d2 <dpt2>, ...)};

DELETE FROM S WHERE (...) WITH DPT { <d> | d<i> };
```

Table S can support several DPTs (e.g., dpt1, dpt2) as long as the DPTs are pre-determined. Applications can trigger on-demand deletion with any such DPT through the DELETE command. Similarly to retention-driven deletes, timely persistent deletion of data on-demand is easier to handle from a storage engine if the DPTs supported are known *a priori* during the table creation.

Putting everything together. With the proposed SQL extensions, a relational table can now support multiple (pre-defined or arbitrary) thresholds for both retention-based and on-demand deletes, with the following CREATE TABLE statement.

```
CREATE TABLE T (column1 type1, column2 type2, ...)
WITH RET_DUR {ARBITRARY | FIXED (t1 <ret1>, t2 <ret2>, ...)};
WITH DPT {ARBITRARY | FIXED (d1 <dpt1>, d2 <dpt2>, ...)};
```

Note that typically retention-based deletes come from the *application requirements*, and on-demand deletion requests are issued *by the user*. However, in both cases, the deletes have to happen *timely* as per the regulatory requirements. The proposed changes in SQL are not enough to guarantee that the system will deliver on the need for timely and persistent data deletion. Rather, they create the interface for data systems so that need for timely deletion. Rather, they create the interface for data systems so that the users and applications can express the deletion requests which is enforced by the regulations. The data deletion *per se* is realized at the *system level*, and we will next discuss advances and challenges on that front.

3.3 System Layer

With the requirement analysis and the declarative interface in place, the users and the applications can express all the mandated deletion requests and the underlying system is now tasked with implementing them. Before discussing the challenges of implementing timely retention-based and on-demand deletes, we discuss the taxonomy of system-level deletes the application layer may initiate. In other words, we want to understand what delete patterns may be generated at the system layer.

3.3.1 Taxonomy of Deletes at the System Layer

The behavior of a low-level delete operation depends on (i) the logical organization and physical layout of the data, and (ii) the attribute based on which the deletion requests are issued. To better understand this *delete design space*, we classify different delete operations across two dimensions: (a) deletes on *primary* vs. *secondary* attributes, and (b) deletes based on a single value of the delete attribute, termed *point deletes*, vs. deletes on a range of the delete attribute, termed *range deletes* [76]. Table 1 summarizes the state of the art in out-of-place systems for different delete operations, their performance impact, and their at-large implications.

Delete Workloads	**Primary Deletes**		**Secondary Deletes**	
	Point	Range	Point	Range
State-of-the-art implementation	insert point tombstones (*logical*)	insert range tombstones (*logical*)	not supported	full-tree compaction
Point query path	search for key; stop if a tombstone is found	search for key; compare fetched key with the histogram (discard if invalidated)	N/A	N/A
Range query path	merge qualifying sorted runs; discard on the fly if TS exist	merge qualifying sorted runs; check each value against histogram	N/A	N/A
Implications	unbounded persistence latency high space amplification high write amplification	unbounded persistence latency high space amplification high write amplification severely affects read performance	N/A	huge latency spikes high write amplification superfluous reads from disk

Table 1: Implications of logical deletes on performance in state-of-the-art out-of-place data stores.

Primary Deletes. In out-of-place systems, data is ultimately organized based on a so-called *sort key* (for example, the key of the key-value pairs in an LSM-tree). The sort key often is the primary key of the database, hence, a majority of delete operations can be expressed as deletes based on the sort key, or *primary deletes*. Note that even deletes on other attributes may be preferable to be converted to primary deletes if there is a secondary index. Both point and range primary deletes use the notion of a *delete marker* or a *tombstone* that is inserted in the data collection (on the deleted sort key) and invalidates prior version of the key(s), and they are very common in real workloads [22]. Primary deletes can be triggered either by user activity (i.e., on-demand) or by automated processes (e.g., data migration).

Implications and Challenges. When an out-of-place system has files that contain tombstones, then both point and range query paths are affected. Specifically, since the system accesses files with decreasing age (i.e., the most recent ones first) when looking for a key (*point query*), it will also be looking for tombstones. If a tombstone is found, the search will terminate because all other (older) instances of that key are invalid. In the presence of range deletes, the implementation is more complex as it is hard to use per-file range delete tombstones [63]. Instead, a database-wide histogram of deleted ranges is maintained and every query compares against this histogram before proceeding [76]. When considering *range queries*, all the sorted files have to be merged and the deleted entries are discarded on the fly by comparing against the visited point and range tombstones [21].

The implications of primary deletes are multi-fold. First, while out-of-place systems support deletes, any deletion is logical, and there is no *a priori* bound on the delete persistence latency. Second, by maintaining both the tombstones and the invalid entries for arbitrarily long time, the systems pay in terms of increased space amplification. Thirdly, by reorganizing data including invalid entries and tombstones, we further pay in terms of increased write amplification. Note that space and write amplification [15, 36] are two fundamental sources of cost when deploying data system. Finally, while range tombstones are used to offer the range delete functionality, they are rather cumbersome and impact the read performance severely [21, 63].

Secondary Deletes. In some other cases, we may need to organize data based on a sort key, but we have a majority of deletes on a different attribute. Note that if we have individual deletes on a different attribute, the most prudent approach is to guarantee that we have a secondary index and transform a secondary delete in one (or more) primary point delete, hence in Table 1, we see the lack of support for point secondary deletes. However, in some cases, we may have long range deletes on a secondary attribute. For example, when working on a window of the most recent data we can repetitively delete data based on a timestamp. A similar case is the retention-based deletes introduced earlier.

Implications and Challenges. Secondary range deletes are not native in out-of-place systems, since the underlying data is organized based on the sort key. While converting them to a collection of point primary deletes might work in several cases, it will overload the system with tombstones. Instead, several systems opt to perform a full-database merging and re-writing periodically to fulfill any secondary range deletion constraints they might have to follow. This approach leads to significant write amplification, superfluous data accesses, and a large penalty in terms of latency spikes on the workload during this merging.

From Delete Requirements to Delete Types. The delete taxonomy at the system level helps us map the delete requirements to low-level data operations. A retention-driven deletion is typically modeled as a secondary range delete, and if the delete range has few objects (i.e., low selectivity), it can be implemented as a collection of primary point deletes. On the other hand, on-demand deletion is typically implemented as a primary point or range delete and there are several performance challenges to be addressed.

3.3.2 Realizing Timely Deletes

Timely data deletion while respecting the retention SLAs without hurting the system performance is a key challenge. The efficiency of deletion depends on the schema and the physical data layout, the data re-organization strategy, the workload, and the design of the storage engine. The right-most part of Figure 1 outlines the

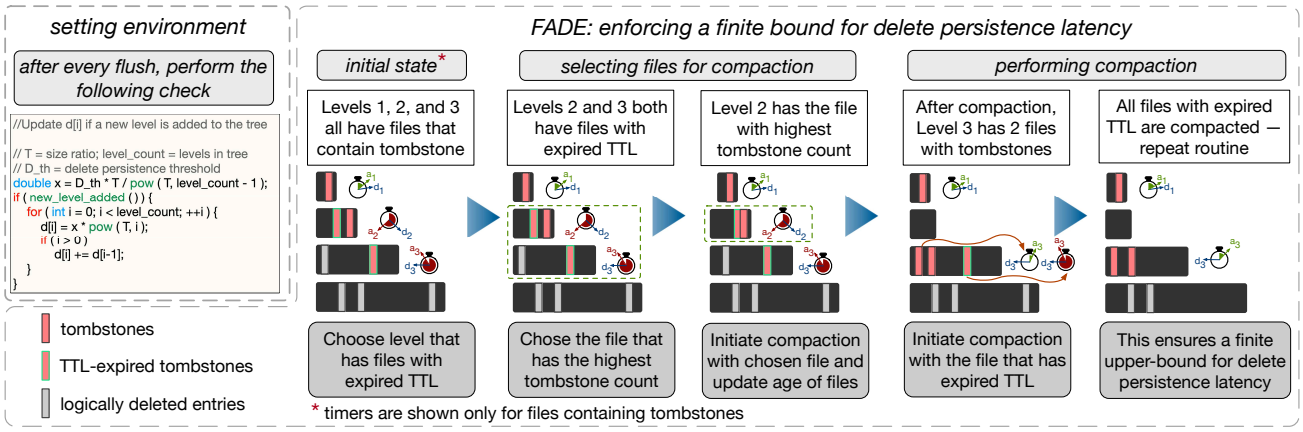


Figure 2: FADE persists tombstones within DPT, thus, improving overall performance.

design changes needed and the input parameters used to co-optimize the performance of a storage engine while ensuring timely delete persistence. We now discuss how both classes of deletes can be realized efficiently through modifications in the design of out-of-place storage engines, focusing on LSM-bases storage engines.

Realizing Primary Deletes. An LSM-based storage engine implements a primary delete by inserting a tombstone on the desired key (or key-range) with a DPT associated. The application-provided DPT is an indicator for how long a logical delete may live before its persistence, that is, before purging any invalidated versions of the key(s) under deletion. The tombstone representation is augmented with additional metadata, e.g., an extra byte to account for 128 possible different DPTs in the same table. During its natural course of data re-organization the storage engine checks for any data blocks (i.e., pages, files, or sorted runs) with an expired TTL and consolidates them to ensure timely persistence. This data consolidation in LSM-based data stores is called *compaction* [72, 77] and is the process of selecting some components of the database (files, sorted runs) to merge and discard invalid entries. At any point of time an LSM-based system has several files that may be compacted (it can be in the order of several thousands) so the decision which files to compact is a crucial decision. In general, the decision is based on read query metrics, however, in Lethe [76] we propose a new approach that prioritizes compactions of files depending on the age of the tombstones they contain. Specifically, we assign different TTLs based on the level of the underlying LSM-tree and when there is a tombstone with an expired TTL we select the file that contains it for compaction, as shown in Figure 2. A key decision is how to ensure that the multi-step merging of tombstones will always respect the application-defined DPT. This is ensured by assigning a different TTL to each tombstone after every compaction in a way that the sum of all its TTL amounts to the desired DPT.

Realizing Secondary Deletes. As we discussed above, several instances of secondary deletes can be realized as a collection of primary point deletes. However, when we are frequently tasked to deleted a range of values based on a secondary attribute, we can achieve something significantly better. In particular, a new weaved data layout between the original sort key and the (secondary) delete key can offer much more efficient and timely secondary range deletion while maintaining competitive read performance. The key idea is to create a nested data organization that alternates between organizing data based on the sort key (to facilitate good search performance) and based on the delete key (to allow for consecutive chunks of data to be deleted at a time).

This approach is implemented in the KiWi data layout [76] as shown in Figure 3. The core idea is that while the major components of the database (files) are organized based on the sort key, every file is composed of delete tiles that are internally organized based on the delete key, partitioning the data accordingly. Lastly, each data page is again organized on the search key to facilitate efficient in-memory search. The benefit for this weaved data layout is that in the case of secondary range deletes, we can discard entire groups of pages at a time, signaling the file system to reclaim this page instantly, essentially converting the secondary delete to a page reclamation action

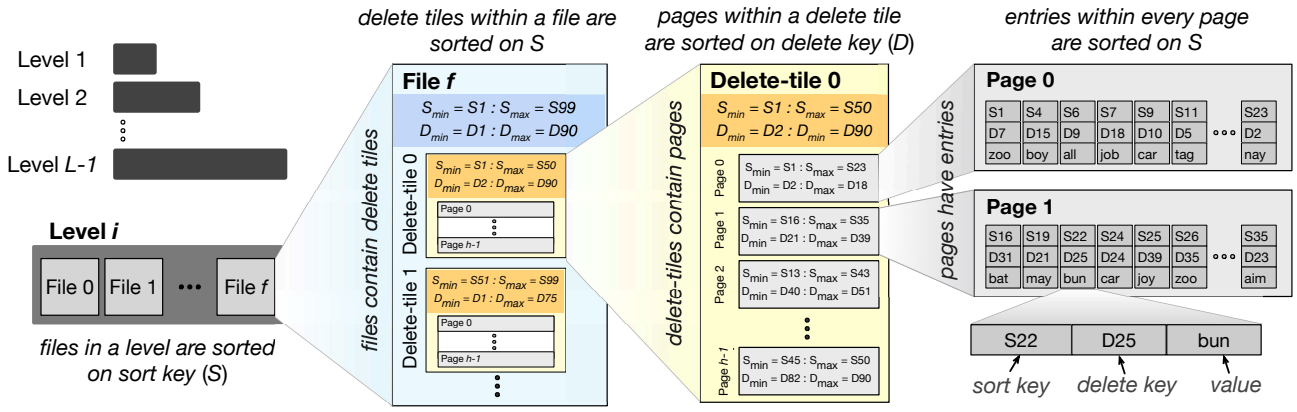


Figure 3: KiWi stores data in an interweaved fashion on the sort and delete key to facilitate efficient secondary range deletes while offering competitive read performance.

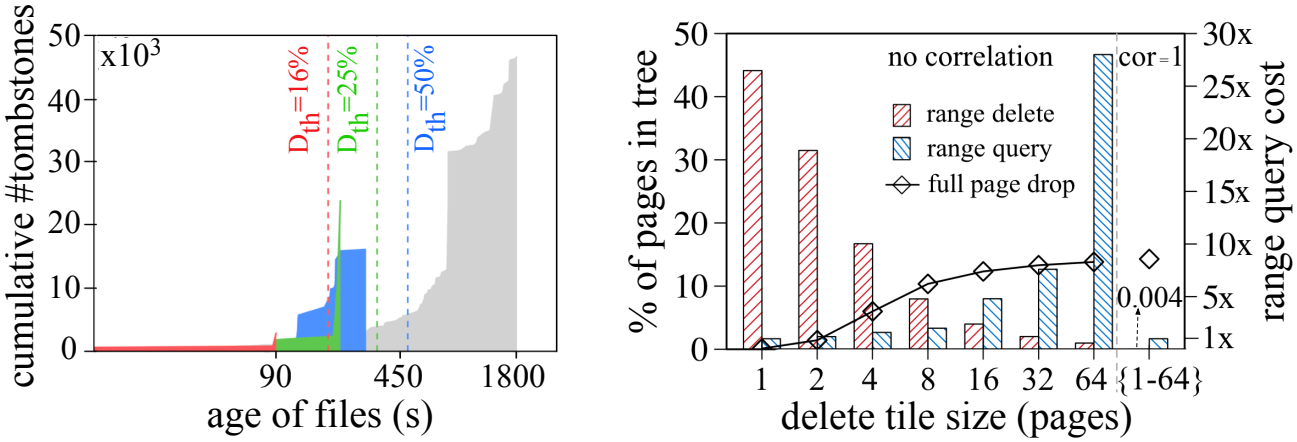


Figure 4: Lethe ensures timely persistence of logically invalidated data within LSM-based out-of-place data systems for both primary and secondary classes of deletes.

that has very low latency compared to a full database reorganization. In the worst case, we will have to in-place edit a few pages at the edge of the range, which is a tunable parameter that controls the maximum secondary deletion persistence latency as a tradeoff vs. read performance.

Evaluation. The approaches presented above for timely deletion were implemented as part of the LSM-based system Lethe [76], and achieved efficient timely deletion respecting predetermined guarantees. The left hand-side of Figure 4 shows the CDF of the tombstone age while varying the desired DPT to 16%, 25%, and 50% of the duration of the experiment. The colored areas correspond to the number of cumulative tombstones for the corresponding age on the x-axis, while the horizontal dotted-line is the desired DPT. We observe that Lethe was able to always deliver the requested DPT. The gray area corresponds to the age of tombstones of the state of the art, where no DPT is imposed and deletes are not persisted timely. Notably, we also measured that enforcing the desired DPT shows benefits in terms of access time because the amount of invalid data was reduced. Similarly, we saw benefits in space amplification, and only marginal cost increase in amortized write amplification.

The right hand-side of Figure 4 shows the fraction of fully dropped pages during a range delete as we vary the size of the delete tiles. We observe that the fraction of pages fully dropped increases with the delete tile size, allowing for efficient reclamation of the invalid data. Conversely, the read queries become more expensive as we allow for more page drops, so the ideal delete tile size should be tuned based on the workload.

Deletes in a Complex Data Model. The previous discussion focuses on handling deletes in a per-instance manner without considering multiple copies of the data in a more complex setting. Cohn-Gordon *et al.* [27] proposed the deletion framework *DELF* that ensures reliable data deletion from an online social network (OSN). *DELF* enables detection of inconsistent data deletion in OSNs and also facilitates data recovery in cases where user data was incorrectly deleted. Minaei *et al.* [65] proposed a framework for persistently deleting all instances of user data in presence of observers, thereby, ensuring privacy through timely content concealment and removal.

4 Challenges and Opportunities

In Section 3, we outlined the steps taken to realize the four-layered vision of delete-compliant data systems presented in Figure 1, however, there are still open questions and challenges for such systems which pose opportunities for further innovative systems research.

Device-level Deletion. Data management solutions rely on storage devices and treat them as black boxes. However, deleting data persistently at the device level and from data archives is an open technological challenge. Current endeavors in this direction are mainly focused on encryption-based solutions [54, 57, 60]. Nevertheless, retention-based deletes entail persistent deletion of a "quantum" of data (e.g., the data ingested in a day) posing the following challenges for encryption-based solutions. First, it is hard to *determine the encryption granularity* while minimizing the encrypt/decrypt overhead. Second, with several data streams for different users/applications (and thus, bound by different SLAs), it is hard to *manage the encryption keys efficiently and in a scalable manner*. Third, efficient and scalable *deletion from archives and backup stores on-demand* is hard to be supported by encryption-based deletion as the encrypt/decrypt cost and the fine encryption granularity adds prohibitive overheads. Finally, from a legislation point-of-view it is not yet clear whether encrypting and discarding the key is an accepted form of deletion.

When considering a system-level deletion similar approach to the one presented in Section 3.3 storage devices are essentially one more level of managing data at the physical layer and similar approaches have to be implemented in the file system or the file and data systems have to be developed in tandem.

Cloud-Level Deletion. Further, operating on the cloud, data systems use virtualized devices and object storage which is even more abstract hiding the details of how the low-level device and page management is taking place. Offering guarantees for timely data deletion in virtualized storage will require a similar multi-layered approach where the file system and the device firmware will expose knobs to allow the application on top to request specific page reclamation properties.

Deletion in Distributed/Federated Computing Environment. With more and more data stores being transformed to cloud-based stores, user data may be collected, processed, and stored across multiple domains, spread across different geographic locations [75]. With different geographic locations being bound by different privacy regulations, we need to design *solutions to ensure consistency for persistent deletion* of user data. Our intuition is that existing solutions for data stream-tainting [37], cross-domain data tracing [33, 47], and related data provenance solutions [20, 45] can be useful to address this problem.

Compliance Demonstration. Last but certainly not least, data systems have to be able to prove compliance when audited. The natural way to do so now is via log auditing, however, a more light-weight algorithmic way for providing this will benefit both systems and users. Inspecting logs and the underlying data is a time-consuming process and the long-term goal of the community should be to design system-level tools that can verifiably prove compliance with the privacy regulations. One interesting development in this direction is the evolution of security-driven operating systems, such as seL4 [56, 80]. Another approach that can be taken is to show that the codebase of a data system has the necessary code-paths for timely deletion via static and dynamic analysis. An open challenge is to develop static and dynamic analysis tools that can prove that a system deletes data respecting the timely deletion requirements set.

5 Conclusion

In this paper, we highlight that the recently enacted regulations mandate new data deletion requirements, requiring a new breed of data systems to support them. We show that existing state-of-the-art out-of-place systems are ill-equipped for this task, and we present a four-layered approach towards building the necessary infrastructure. We present recent work on that front, and we conclude by discussing several open research challenges.

Acknowledgments. This work was partially funded by National Science Foundation under Grant No. IIS-1850202 and a Facebook Faculty Research Award.

References

[1] Right to erasure. *https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/individual-rights/right-to-erasure/*.

[2] Regulation (EU) 2016/679 of the European Parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC. *Official Journal of the European Union (Legislative Acts)*, pages L119/1 – L119/88, 2016.

[3] California Consumer Privacy Act. *Assembly Bill No. 375, Chapter 55*, 2018.

[4] Data Protection Act 2018. *https://www.legislation.gov.uk/ukpga/2018/12/pdfs/ukpga_20180012_en.pdf*, 2018.

[5] PIPEDA in brief. *https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/pipeda_brief/*, 2019.

[6] The California Privacy Rights Act of 2020. *https://thecpra.org/*, 2020.

[7] Virginia Consumer Data Protection Act. *https://www.sullcrom.com/files/upload/SC-Publication-Virginia-Second-State-Enact-Privacy-Legislation.pdf*, 2021.

[8] Amazon. Redshift. *https://aws.amazon.com/redshift/*.

[9] M. L. Ambrose and J. Ausloos. The Right to Be Forgotten Across the Pond. *Journal of Information Policy*, 3:1–23, 2013.

[10] Apache. Accumulo. *https://accumulo.apache.org/*.

[11] Apache. HBase. *http://hbase.apache.org/*.

[12] Apache. Cassandra. *http://cassandra.apache.org*, 2021.

[13] M. Athanassoulis, S. Chen, A. Ailamaki, P. B. Gibbons, and R. Stoica. MaSM: Efficient Online Updates in Data Warehouses. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 865–876, 2011.

[14] M. Athanassoulis, S. Chen, A. Ailamaki, P. B. Gibbons, and R. Stoica. Online Updates on Data Warehouses via Judicious Use of Solid-State Storage. *ACM Transactions on Database Systems (TODS)*, 40(1), 2015.

[15] M. Athanassoulis, M. S. Kester, L. M. Maas, R. Stoica, S. Idreos, A. Ailamaki, and M. Callaghan. Designing Access Methods: The RUM Conjecture. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 461–466, 2016.

[16] M. A. Bender, E. D. Demaine, and M. Farach-Colton. Cache-Oblivious B-Trees. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 399–409, 2000.

[17] M. A. Bender, M. Farach-Colton, W. Jannen, R. Johnson, B. C. Kuszmaul, D. E. Porter, J. Yuan, and Y. Zhan. An Introduction to Bϵ-trees and Write-Optimization. *White Paper*, 2015.

[18] C. T. Brown and T. D. Manoranjan. South Korea Releases Guidance on Right to Be Forgotten. *https://www.lexology.com/library/detail.aspx?g=21be3837-0c43-4047-b8b5-9e863960b0b9*, 2016.

[19] G. A. Brown. Consumers' "Right to Delete" under US State Privacy Laws. *https://www.securityprivacybytes.com/2021/03/consumers-right-to-delete-under-us-state-privacy-laws/*, 2021.

[20] P. Buneman and W.-C. Tan. Data Provenance: What next? *SIGMOD Rec.*, 47(3):5–16, 2018.

[21] M. Callaghan. Deletes are fast and slow in an LSM. *http://smalldatum.blogspot.com/2020/01/deletes-are-fast-and-slow-in-lsm.html*, 2020.

[22] Z. Cao, S. Dong, S. Vemuri, and D. H. C. Du. Characterizing, Modeling, and Benchmarking RocksDB Key-Value Workloads at Facebook. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, pages 209–223, 2020.

[23] E. L. Carter. Argentina's Right to be Forgotten. *Emory International Law Review*, 27(1), 2013.

[24] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 205–218, 2006.

[25] W. B. Chik. The Singapore Personal Data Protection Act and an assessment of future trends in data privacy reform. *Comput. Law Secur. Rev.*, 29(5):554–575, 2013.

[26] Cisco. Cisco Global Cloud Index: Forecast and Methodology, 2016–2021. *White Paper*, 2018.

[27] K. Cohn-Gordon, G. Damaskinos, D. Neto, J. Cordova, B. Reitz, B. Strahs, D. Obenshain, P. Pearce, I. Papagiannis, and A. Media. DELF: Safeguarding deletion correctness in Online Social Networks. In *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, 2020.

[28] B. Dageville, T. Cruanes, M. Zukowski, V. Antonov, A. Avanes, J. Bock, J. Claybaugh, D. Engovatov, M. Hentschel, J. Huang, A. W. Lee, A. Motivala, A. Q. Munir, S. Pelley, P. Povinec, G. Rahn, S. Triantafyllis, and P. Unterbrunner. The Snowflake Elastic Data Warehouse. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 215–226, 2016.

[29] Databricks. Online reference. *https://databricks.com/*.

[30] Databricks. Table deletes, updates, and merges. *https://docs.databricks.com/delta/delta-update.html#delete-from-a-table*, 2021.

[31] N. Dayan, M. Athanassoulis, and S. Idreos. Monkey: Optimal Navigable Key-Value Store. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 79–94, 2017.

[32] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's Highly Available Key-value Store. *ACM SIGOPS Operating Systems Review*, 41(6):205–220, 2007.

[33] B. Demsky. Cross-application data provenance and policy enforcement. *ACM Transactions on Information Systems (TOIS)*, 14(1):6:1—-6:22, 2011.

[34] F. Deng, Q. Cao, S. Wang, S. Liu, J. Yao, Y. Dong, and P. Yang. SeRW: Adaptively Separating Read and Write upon SSDs of Hybrid Storage Server in Clouds. In *Proceedings of the International Conference on Parallel Processing (ICPP)*, pages 76:1—-76:11, 2020.

[35] A. Deshpande and A. Machanavajjhala. ACM SIGMOD Blog: Privacy Challenges in the Post-GDPR World: A Data Management Perspective. *http://wp.sigmod.org/?p=2554*, 2018.

[36] S. Dong, M. Callaghan, L. Galanis, D. Borthakur, T. Savor, and M. Strum. Optimizing Space Amplification in RocksDB. In *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*, 2017.

[37] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. D. McDaniel, and A. Sheth. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 393–407, 2010.

[38] Facebook. RocksDB. *https://github.com/facebook/rocksdb*, 2021.

[39] F. Färber, N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, and J. Dees. The SAP HANA Database – An Architecture Overview. *IEEE Data Engineering Bulletin*, 35(1):28–33, 2012.

[40] Gartner. Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. https://tinyurl.com/Gartner2020, 2017.

[41] M. Goddard. The EU General Data Protection Regulation (GDPR): European Regulation that has a Global Impact. *International Journal of Market Research*, 59(6):703–705, 2017.

[42] G. Golan-Gueta, E. Bortnikov, E. Hillel, and I. Keidar. Scaling Concurrent Log-Structured Data Stores. In *Proceedings of the ACM European Conference on Computer Systems (EuroSys)*, pages 32:1–32:14, 2015.

[43] Google. LevelDB. *https://github.com/google/leveldb/*, 2021.

[44] A. Gupta, D. Agarwal, D. Tan, J. Kulesza, R. Pathak, S. Stefani, and V. Srinivasan. Amazon Redshift and the Case for Simpler Data Warehouses. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1917–1923, 2015.

[45] S. S. Hasan, N. H. Sultan, and F. A. Barbhuiya. Cloud Data Provenance using IPFS and Blockchain Technology. In *Proceedings of the International Workshop on Security in Cloud Computing (SCC)*, pages 5–12, 2019.

[46] S. Héman, M. Zukowski, and N. J. Nes. Positional Update Handling in Column Stores. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 543–554, 2010.

[47] R. Herbster, S. DellaTorre, P. Druschel, and B. Bhattacharjee. Privacy Capsules: Preventing Information Leaks by Mobile Apps. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2016, Singapore, June 26-30, 2016*, pages 399–411, 2016.

[48] G. Huang, X. Cheng, J. Wang, Y. Wang, D. He, T. Zhang, F. Li, S. Wang, W. Cao, and Q. Li. X-Engine: An Optimized Storage Engine for Large-scale E-commerce Transaction Processing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 651–665, 2019.

[49] S. Idreos, N. Dayan, W. Qin, M. Akmanalp, S. Hilgard, A. Ross, J. Lennon, V. Jain, H. Gupta, D. Li, and Z. Zhu. Design Continuums and the Path Toward Self-Designing Key-Value Stores that Know and Learn. In *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*, 2019.

[50] S. Idreos, F. Groffen, N. Nes, S. Manegold, K. S. Mullender, and M. L. Kersten. MonetDB: Two Decades of Research in Column-oriented Database Architectures. *IEEE Data Engineering Bulletin*, 35(1):40–45, 2012.

[51] W. Jannen, J. Yuan, Y. Zhan, A. Akshintala, J. Esmet, Y. Jiao, A. Mittal, P. Pandey, P. Reddy, L. Walsh, M. A. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, and D. E. Porter. BetrFS: A Right-optimized Write-optimized File System. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, pages 301–315, 2015.

[52] M. L. Jones. It's About Time: Privacy, Information Lifecycles, and the Right to Be Forgotten. *Stanford Technology Law Review*, 16(2):54, 2012.

[53] W.-H. Kang, S.-W. Lee, and B. Moon. Flash as cache extension for online transactional workloads. *The VLDB Journal*, 25(5):673–694, 2016.

[54] R. Kissel, A. Regenscheid, M. Scholl, and K. Stine. Guidelines for Media Sanitization. *NIST Special Publication 800-88*, 2014.

[55] P. Kittane, I. S. Charles, A. Kamath, and G. Gokhale. Privacy and Data Protection – India Wrap 2020. *The National Law Review*, XI(162), 2021.

[56] G. Klein, J. Andronick, K. Elphinstone, G. Heiser, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. seL4: formal verification of an operating-system kernel. *Communications of the ACM*, 53(6):107–115, 2010.

[57] B. Köppel and S. Neuhaus. Analysis of a hardware security module's high-availability setting. *IEEE Security Privacy*, 11(3):77–80, 2013.

[58] B. C. Kuszmaul. A Comparison of Fractal Trees to Log-Structured Merge (LSM) Trees. *Tokutek White Paper*, 2014.

[59] A. Lamb, M. Fuller, and R. Varadarajan. The Vertica Analytic Database: C-Store 7 Years Later. *Proceedings of the VLDB Endowment*, 5(12):1790–1801, 2012.

[60] B. Li and D. H. C. Du. TASecure: Temperature-Aware Secure Deletion Scheme for Solid State Drives. In *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*, pages 275–278, 2019.

[61] LinkedIn. Voldemort. *http://www.project-voldemort.com*.

[62] C. Luo and M. J. Carey. LSM-based Storage Techniques: A Survey. *The VLDB Journal*, 29(1):393–418, 2020.

[63] A. Madan and A. Kryczka. DeleteRange: A New Native RocksDB Operation. *https://rocksdb.org/blog/2018/11/21/delete-range.html*, 2018.

[64] R. McKean, E. Kurowska-Tober, and H. Waem. DLA Piper GDPR fines and data breach survey: January 2022. *https://www.dlapiper.com/en/us/insights/publications/2022/1/dla-piper-gdpr-fines-and-data-breach-survey-2022/*, 2022.

[65] M. Minaei, M. Mondal, P. Loiseau, K. P. Gummadi, and A. Kate. Lethe: Conceal Content Deletion from Persistent Observers. *Proceedings on Privacy Enhancing Technologies (PoPET)*, 2019(1):206–226, 2019.

[66] P. E. O'Neil, E. Cheng, D. Gawlick, and E. J. O'Neil. The log-structured merge-tree (LSM-tree). *Acta Informatica*, 33(4):351–385, 1996.

[67] S. Papadopoulos, K. Datta, S. Madden, and T. Mattson. The TileDB Array Data Storage Manager. *Proceedings of the VLDB Endowment*, 10(4):349–360, 2016.

[68] Paradigm4. Online reference. *https://www.paradigm4.com/*.

[69] D. Pardo. First Decision on the "Right to be Forgotten" in Argentina. *https://scholarlycommons.law.emory.edu/cgi/viewcontent.cgi?article=1097&context=eilr*, 2020.

[70] D. Piper. GDPR Data Breach Survey 2020. *https://www.dlapiper.com/en/us/insights/publications/2020/01/gdpr-data-breach-survey-2020/*, 2020.

[71] M. Sadoghi, K. A. Ross, M. Canim, and B. Bhattacharjee. Exploiting SSDs in operational multiversion databases. *The VLDB Journal*, 25(5):651–672, 2016.

[72] S. Sarkar, K. Chen, Z. Zhu, and M. Athanassoulis. Compactionary: A Dictionary for LSM Compactions. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2022.

[73] S. Sarkar and M. Athanassoulis. Dissecting, Designing, and Optimizing LSM-based Data Stores. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2022.

[74] S. Sarkar and M. Athanassoulis. Query Language Support for Timely Data Deletion. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, 2022.

[75] S. Sarkar, J.-P. Banâtre, L. Rilling, and C. Morin. Towards Enforcement of the EU GDPR: Enabling Data Erasure. In *Proceedings of the IEEE International Conference of Internet of Things (iThings)*, pages 1–8, 2018.

[76] S. Sarkar, T. I. Papon, D. Staratzis, and M. Athanassoulis. Lethe: A Tunable Delete-Aware LSM Engine. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 893–908, 2020.

[77] S. Sarkar, D. Staratzis, Z. Zhu, and M. Athanassoulis. Constructing and Analyzing the LSM Compaction Design Space. *Proceedings of the VLDB Endowment*, 14(11):2216–2229, 2021.

[78] M. Schwarzkopf, E. Kohler, M. F. Kaashoek, and R. T. Morris. Position: GDPR Compliance by Construction. In *Selected Papers from VLDB Workshop on Polystore Systems for Heterogeneous Data in Multiple Databases with Privacy and Security Assurances (POLY)*, volume 11721 of *Lecture Notes in Computer Science*, pages 39–53, 2019.

[79] R. Sears and R. Ramakrishnan. bLSM: A General Purpose Log Structured Merge Tree. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 217–228, 2012.

[80] SeL4. Online reference. *https://sel4.systems*.

[81] A. Shah, V. Banakar, S. Shastri, M. Wasserman, and V. Chidambaram. Analyzing the Impact of GDPR on Storage Systems. In *Proceedings of the USENIX Conference on Hot Topics in Storage and File Systems (HotStorage)*, 2019.

[82] S. Shastri, V. Banakar, M. Wasserman, A. Kumar, and V. Chidambaram. Understanding and Benchmarking the Impact of GDPR on Database Systems. *Proceedings of the VLDB Endowment*, 13(7):1064–1077, 2020.

[83] S. Shastri, M. Wasserman, and V. Chidambaram. The Seven Sins of Personal-Data Processing Systems under GDPR. In *Proceedings of USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2019.

[84] S. Shastri, M. Wasserman, and V. Chidambaram. GDPR anti-patterns. *Communications of the ACM*, 64(2):59–65, 2021.

[85] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. R. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. Zdonik. C-Store: A Column-oriented DBMS. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 553–564, 2005.

[86] M. Stonebraker, J. Duggan, L. Battle, and O. Papaemmanouil. SciDB DBMS Research at M.I.T. *IEEE Data Eng. Bull.*, 36(4):21–30, 2013.

[87] Tessian. 25 Biggest GDPR Fines So Far (2019, 2020, 2021, 2022). *https://www.tessian.com/blog/biggest-gdpr-fines-2020/*, 2022.

[88] TileDB. Online reference. *https://tiledb.io*.

[89] A. Tsesis. The Right to be Forgotten and Erasure: Privacy, Data Brokers, and the Indefinite Retention of Data. *Wake Forest Law Review*, 48:51, 2014.

[90] Z. Whittaker and N. Lomas. Even years later, Twitter doesn't delete your direct messages. *https://techcrunch.com/2019/02/15/twitter-direct-messages/*, 2019.

[91] L. Yang, H. Wu, T. Zhang, X. Cheng, F. Li, L. Zou, Y. Wang, R. Chen, J. Wang, and G. Huang. Leaper: A Learned Prefetcher for Cache Invalidation in LSM-tree based Storage Engines. *Proceedings of the VLDB Endowment*, 13(11):1976–1989, 2020.

[92] Q. Zheng, C. D. Cranor, D. Guo, G. R. Ganger, G. Amvrosiadis, G. A. Gibson, B. W. Settlemyer, G. Grider, and F. Guo. Scaling Embedded In-Situ Indexing with DeltaFS. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 3:1–3:15, 2018.

Provenance-based Model Maintenance: Implications for Privacy

Yinjun Wu, Val Tannen and Susan B. Davidson
University of Pennsylvania
{wuyinjun, val, susan}@seas.upenn.edu

Abstract

With the increasing need for Machine-learning-as-a-service (MaaS) online systems, effectively maintaining and reusing machine learning models in light of changes to the underlying data has become a big concern. In particular, it is extremely challenging to refresh existing models after the removal of training samples, which is called "machine unlearning". Addressing this challenge not only requires an efficient solution, but must comply with emerging privacy issues, e.g. GDPR, which implies that the removed samples must be fully erased from the models so that they cannot be leaked to an adversary. We review two provenance-based solutions, PrIU and DeltaGrad, and show how they can guard against "model inversion attacks", which reconstruct the removed training samples from the updated models after the unlearning process. Since PrIU and DeltaGrad support a limited class of models, we envision a system that can unlearn general *models in an efficient and secure manner and outline possible technical challenges for building this system.*

1 Introduction

The problem of incrementally updating model parameters after the deletion of a small set of training samples has attracted increasing attention in machine learning over the past few years. It arises in applications such as refreshing model parameters after sensitive training samples are removed (the *GDPR* issue [1]), reducing bias in statistics [2], and quantifying uncertainty [3].

It is also used for quantifying the importance of a training sample using measures such as the *Data Shapley value* [4]. A key step in evaluating this type of measure is to *remove* a subset of training samples and calculate the updated ML model parameters. The most straightforward way to do this is to reconstruct the ML model from scratch after the samples have been deleted. However, recalculating from scratch is prohibitively expensive, especially when the training data is frequently updated, and so the question is whether the model can be updated in real time.

From the perspective of a database researcher, this problem seems very similar to the well-studied problem of *materialized view maintenance* [5, 6] (see Figure **??**). In materialized view maintenance, we have input relations over which a view is constructed using relational algebra operators (left side of the figure). In machine learning (right side of the figure), the analogy to input relations is the training data, and the operations forming the "view" (the model) is the learning algorithm. The question is whether techniques that have been developed for materialized view maintenance can be used for what we will call *model-maintenance*.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

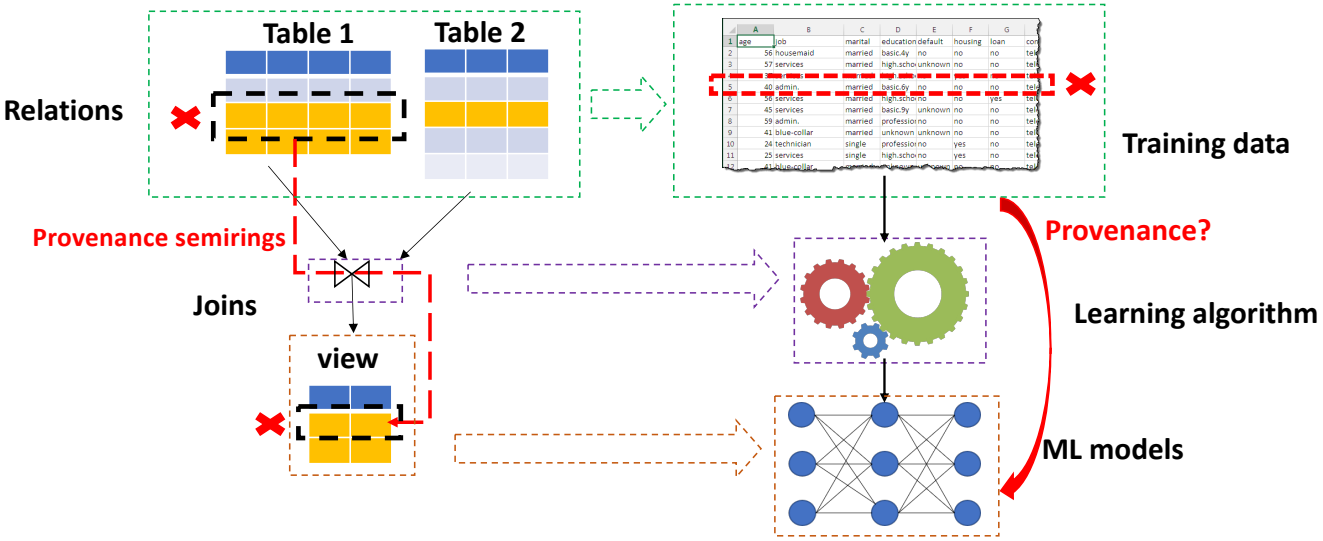


Figure 1: Parallels between materialized view maintenance and model maintenance

One very effective technique for materialized view maintenance is based on provenance, in particular the *provenance semiring* framework [7]. In this setting, by tracing the use of input tuples for each tuple in the view using *provenance polynomials*, the deletion of an input tuple can be propagated to a view tuple by seeing how and if it is used in the view tuple's provenance polynomial.

Of course, a machine learning algorithm is much more complicated than a relational algebra expression. However, recent work has extended the provenance semiring framework to linear algebra operations [8], opening the door to using provenance to reason over ML algorithms based on those operations, such as *linear regression* and *logistic regression* in which non-linear operations are linearized using piecewise linear interpolation [9]. In this paper, we show how provenance can be used for incrementally updating machine learning models for linear or logistic regression in our PrIU system [9]. In particular we show how provenance information carried by the linear operations can be cached during the model training process and then reused for speeding up the model maintenance. For more complex models which use non-linear operations, provenance is not yet defined. However, building on the ideas used in PrIU of caching essential intermediate results, we therefore show how caching can be used to incrementally update more complex models (such as neural networks) in our DeltaGrad system [10].

We then explore the connection of our provenance-based technique to other *machine unlearning* techniques (e.g., [11]). A major concern in machine unlearning is that the unlearned model may suffer from *model inversion attacks* [12], in which the adversary is able to restore the deleted data items (a.k.a *private* data) from the resulting model either using just the model (a *black box* attack) or using auxiliary information such as the model type, model parameters, or even the remaining training samples (a *white box* attack). In contrast to other machine unlearning techniques, we show that a benefit of our provenance-based technique is that it can avoid such attacks with low overhead and without loss of prediction performance.

The remainder of this paper is organized as follows: In Section 2 we give background information on provenance semirings and deletion propagation. In Section 3 we discuss how to extend these ideas to incremental model maintenance for linear and logistic regression models in our system PrIU, and for more complex models in our system DeltaGrad. We then discuss the problem of model inversion attacks in Section 4, and show how our framework can be used to guard against them.

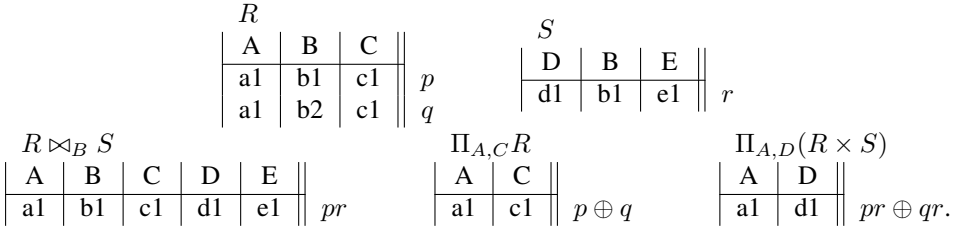


Figure 2: Provenance propagation

2 Background

We start by describing the provenance semiring framework in relational databases, and how provenance can be used to incrementally update views. We then discuss how the framework has been extended to include linear algebra operations

Provenance Semirings. In the provenance semiring framework [7], input data to a query is annotated with provenance tokens which are propagated through the algebraic operators performed on the data, e.g. select, project, join, union, and more recently aggregation [13]. As the provenance annotations propagate, they are combined using two abstract operations. The provenance operation $\otimes$ signifies *joint use* of data. For example, when tuple t_1, respectively t_2, are annotated with provenance p, respectively q, then the tuple obtained by joining t_1 and t_2 is annotated with provenance $p \otimes q$. The other provenance operation, $\oplus$, signifies *alternative use* of data. For example, when tuple t_1, respectively t_2, are annotated with provenance p, respectively q, such that t_1 and t_2 project to the same tuple t, then the resulting tuple t is annotated with provenance $p \oplus q$. In addition, an abstract "zero-polynomial" 0_{prov} is used to annotate *absent* tuples and an abstract "unit-polynomial" 1_{prov} is used to annotate "neutral" data whose provenance is ignored (it's of no interest in a specific analysis). If P is the mathematical space of provenance annotations, standard equivalences of positive relational algebra imply that $(P, \oplus, \otimes, 0_{prov}, 1_{prov})$ is a *commutative semiring* [7]. It follows that the expressions obtained as provenance is propagated through a query can be put in standard polynomial form, where the indeterminates (the variables) are provenance tokens that annotate the tuples in the data given as input to the query. Therefore, in the semiring framework provenance is captured by *provenance polynomials*.

Example 1: As an example, suppose we have two tables, R and S, whose tuples are annotated with provenance tokens p, q, and r, as shown in Figure 2. In the same figure we show the provenance polynomials in indeterminates p, q, r that are produced by provenance propagation in the outputs of queries $R \bowtie_B S$, $\Pi_{A,C}R$, and $\Pi_{A,D}(R \times S)$. As is customary in algebra, we omit the multiplication-like symbol $\otimes$ when we write polynomials in commutative indeterminates.

Deletion Propagation. One of the benefits of the semiring framework is that deletions of tuples in the input tuples propagate straightforwardly to deletions (or modifications of provenance) of output tuples by partially evaluating provenance polynomials using familiar algebraic rules.

When an input tuple is deleted, the effect on the output can be efficiently calculated by setting its token to 0_{prov} in the output's provenance polynomials, signifying absence. For example, if the first tuple in R, (a1, b1, c1) with provenance token p, were deleted, then the provenance of the (only) tuple in $R \bowtie_B S$ would become $0_{prov} \otimes r = 0_{prov}$, indicating that it no longer appears – intuitively, both tuples are needed for that tuple to be present. On the other hand, the tuple in $\Pi_{A,C}R$ would still appear but with a different provenance $0_{prov} \oplus q = q$ – intuitively, at least one of p and q is needed for the result to be present. Similarly, deleting the

first tuple in R will still leave the tuple (a1,d1) in the output of $\Pi_{A,D}(R \times S)$, but its provenance would become $(0_{prov} \otimes r) \oplus (q \otimes r) \ = \ 0_{prov} \oplus (q \otimes r) \ = \ qr$.

Extension to Linear Algebra Operations Recently, the semiring framework has been extended to include basic linear algebra operations: matrix addition and multiplication [8]. In this extension, the provenance polynomials play the role of abstract scalars and an abstract version of multiplication with scalars plays the role of annotating matrices (in particular, vectors) with provenance. Matrices form a non-commutative, many-sorted ring. The structure obtained by combining the ring of matrices with multiplication by scalars from the semiring of provenance polynomials is a *semialgebra* [8]. This framework is flexible enough that for input matrices we can annotate both rows (samples) and columns (features) with arbitrary provenance tokens.

As an example, suppose that p,q,r,s are provenance tokens that annotate four different samples in a training dataset. We denote the multiplication with scalars (i.e., the annotation with provenance polynomias) by the abstract operation $*$. Using the work in [8], our methods will show that vectors of interest (such as the vector of model parameters) can be expressed with provenance annotated expressions such as:

$$\mathbf{w} = (p^2 q * \mathbf{u}) + (qr^4 * \mathbf{v}) + (ps * \mathbf{z}) \tag{1}$$

Here, **u**,**v**,**z** are numerical vectors signifying contributions to the answer **w** and they are annotated with the provenance polynomials p^2q, qr^4, ps.

Now suppose the sample (input row) annotated with r is deleted while those annotated p,q,and s are retained but we decide not to track them anymore. As we did in the paragraph on deletion propagation, we can express the updated value of **w** under this deletion by setting r to 0_{prov} which signifies absence, and p,q,s to 1_{prov}, which signifies "neutral" presence in Equation (1). Again, the resulting expressions can be simplified using familiar algebraic manipulations. As expected, $0_{prov} \otimes r^4 = 0_{prov}$ as well as $0_{prov} * \mathbf{v} = \mathbf{0}$ (the all-zero vector). Moreover, $1_{prov} \otimes 1_{prov} = 1_{prov}$ and $1_{prov} * \mathbf{z} = \ \mathbf{z}$. It follows that under this deletion $\mathbf{w} = \ \mathbf{u} + \mathbf{z}$.

3 Overview of PrIU and DeltaGrad

We start with preliminaries on Stochastic Gradient Descent (SGD) before showing the explicit use of provenance in PrIU and the implicit use of provenance in DeltaGrad for incrementally updating machine learning models in DeltaGrad.

3.1 Preliminaries on SGD

By assuming that SGD is used for model training, PrIU and DeltaGrad can incrementally update the "gradients" at each SGD step. Suppose the training dataset is $D_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ and model parameter is **w**, at the step t of SGD **w** is computed by evaluating the gradients on a randomly sampled mini-batch of the training dataset, i.e.:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \cdot \text{Grad}(\mathbf{w}_t; \mathscr{B}_t), \tag{2}$$

where $\text{Grad}(\mathbf{w}_t; \mathscr{B}_t)$ represents the gradient evaluated on a mini-batch $\mathscr{B}_t$.

Suppose a subset of training samples, R, is removed from the training dataset. Then to compute the updated model parameter, Equation (2) is modified as:

$$\mathbf{w}_{t+1}^U = \mathbf{w}_t^U - \eta_t \cdot \text{Grad}(\mathbf{w}_t; \mathscr{B}_t - R) \tag{3}$$

where $\mathbf{w}^U$ represents the updated model parameter and $\mathscr{B}_t - R$ represents the remaining samples in $\mathscr{B}_t$ after the removal of R.

Note that both Grad$(\mathbf{w}_t; \mathscr{B}_t)$ in Equation (2) and Grad$(\mathbf{w}_t; \mathscr{B}_t - R)$ in Equation (3) can be regarded as the integration of two components: the model parameter $\mathbf{w}_t$ (or $\mathbf{w}_t^U$) and the data part (the sample $(\mathbf{x}_i, y_i)$). For example, for linear regression models, the loss function $F(\mathbf{w}_t; (\mathbf{x}_i, y_i))$ using L2 regularization is

$$F(\mathbf{w}_t, (\mathbf{x}_i, y_i)) = (y_i - \mathbf{x}_i^\top \mathbf{w})^2 + \frac{\lambda}{2}\|\mathbf{w}_t\|^2,$$

Using this, Grad$(\mathbf{w}_t; \mathscr{B}_t)$ and Grad$(\mathbf{w}_t; \mathscr{B}_t - R)$ can be rewritten as:

$$\text{Grad}(\mathbf{w}_t; \mathscr{B}_t) = (\lambda \mathbf{I} + \frac{2}{|\mathscr{B}_t|}\sum\nolimits_{i \in \mathscr{B}_t} \mathbf{x}_i \mathbf{x}_i^\top)\mathbf{w}_t - \frac{2}{|\mathscr{B}_t|}\sum\nolimits_{i \in \mathscr{B}_t} \mathbf{x}_i y_i, \tag{4}$$

$$\text{Grad}(\mathbf{w}_t; \mathscr{B}_t - R) = (\lambda \mathbf{I} + \frac{2}{|\mathscr{B}_t - R|}\sum\nolimits_{i \in \mathscr{B}_t - R} \mathbf{x}_i \mathbf{x}_i^\top)\mathbf{w}_t^U - \frac{2}{|\mathscr{B}_t - R|}\sum\nolimits_{i \in \mathscr{B}_t - R} \mathbf{x}_i y_i \tag{5}$$

The data part in Grad$(\mathbf{w}_t; \mathscr{B}_t)$ (Equation (4)) consists of two terms:

$$D_1(\mathscr{B}_t) = \sum\nolimits_{i \in \mathscr{B}_t} \mathbf{x}_i \mathbf{x}_i^\top,\ D_2(\mathscr{B}_t) = \sum\nolimits_{i \in \mathscr{B}_t} \mathbf{x}_i y_i.$$

Similarly, the data part in Grad$(\mathbf{w}_t; \mathscr{B}_t - R)$ (Equation (5) can be expressed as:

$$D_1(\mathscr{B}_t - R) = \sum\nolimits_{i \in \mathscr{B}_t - R} \mathbf{x}_i \mathbf{x}_i^\top,\ D_2(\mathscr{B}_t - R) = \sum\nolimits_{i \in \mathscr{B}_t - R} \mathbf{x}_i y_i.$$

3.2 PrIU

As described above, by decomposing the gradient formulas Grad$(\mathbf{w}_t; \mathscr{B}_t)$ or Grad$(\mathbf{w}_t; \mathscr{B}_t - R)$ into a data part and a model parameter part, we can *capture the provenance of the data part to track how the changes of the data part lead to the updates of the model parameters at each SGD iteration*.

Specifically, for the above example, suppose each training sample $(\mathbf{x}_i, y_i)$ is given a unique provenance token, p_i. Then we can generate the following provenance-aware formula for $D_1(\mathscr{B}_t)$ and $D_2(\mathscr{B}_t)$:

$$\text{Prov}(D_1(\mathscr{B}_t)) = \sum\nolimits_{i \in \mathscr{B}_t} p_i^2 * \mathbf{x}_i \mathbf{x}_i^\top,\ \text{Prov}(D_2(\mathscr{B}_t)) = \sum\nolimits_{i \in \mathscr{B}_t} p_i^2 * \mathbf{x}_i y_i$$

To obtain the values of $D_1(\mathscr{B}_t - R)$ and $D_2(\mathscr{B}_t - R)$, we can set the provenance token p_i to 0_{prov} for each $i \in R$ to zero out the removed training samples, and set the other provenance tokens to 1_{prov}, i.e.:

$$D_1(\mathscr{B}_t - R) = \sum\nolimits_{i \in \mathscr{B}_t - R} \mathbf{x}_i \mathbf{x}_i^\top = [\sum\nolimits_{i \in \mathscr{B}_t - R} p_i^2 * \mathbf{x}_i \mathbf{x}_i^\top]_{p_i = 1_{prov}} + [\sum\nolimits_{i \in \mathscr{B}_t \bigcap R} p_i^2 * \mathbf{x}_i \mathbf{x}_i^\top]_{p_i = 0_{prov}}$$

$$D_2(\mathscr{B}_t - R) = \sum\nolimits_{i \in \mathscr{B}_t - R} \mathbf{x}_i y_i = [\sum\nolimits_{i \in \mathscr{B}_t - R} p_i^2 * \mathbf{x}_i y_i]_{p_i = 1_{prov}} + [\sum\nolimits_{i \in \mathscr{B}_t \bigcap R} p_i^2 * \mathbf{x}_i y_i]_{p_i = 0_{prov}}$$

This can be implemented by reusing the cached terms, $D_1(\mathscr{B}_t)$ and $D_2(\mathscr{B}_t)$ and subtracting the terms corresponding to the removed samples in R, i.e.:

$$D_1(\mathscr{B}_t - R) = \sum\nolimits_{i \in \mathscr{B}_t - R} \mathbf{x}_i \mathbf{x}_i^\top = D_1(\mathscr{B}_t) - \sum\nolimits_{i \in \mathscr{B}_t \bigcap R} \mathbf{x}_i \mathbf{x}_i^\top$$

$$D_2(\mathscr{B}_t - R) = \sum\nolimits_{i \in \mathscr{B}_t - R} \mathbf{x}_i y_i = D_2(\mathscr{B}_t) - \sum\nolimits_{i \in \mathscr{B}_t \bigcap R} \mathbf{x}_i y_i$$

This is considerably more efficient than recomputing $D_1(\mathscr{B}_t - R)$ and $D_2(\mathscr{B}_t - R)$ from scratch if the size of R is much smaller than the total number of training samples.

So far, we have discussed how to efficiently update Grad($\mathbf{w}_t; \mathscr{B}_t$) to Grad($\mathbf{w}_t; \mathscr{B}_t - R$) after the removal of a training sample set for SGD formulas where the data part and the model parameters are *linearly combined*. However, in general cases the data part and the model parameters in Grad($\mathbf{w}_t; \mathscr{B}_t$) can be integrated in an arbitrary way, making it difficult to use provenance. For example, for *logistic regression with L2 regularization* the loss function, $F(\cdot)$, and Grad($\mathbf{w}_t; \mathscr{B}_t$), can be instantiated as:

$$F\left(\mathbf{w}_t, (\mathbf{x}_i, y_i)\right) = \ln(1 + \exp\{-y_i \mathbf{w}_t^\top \mathbf{x}_i\}) + \frac{\lambda}{2}\|\mathbf{w}_t\|^2 \tag{6}$$

$$\text{Grad}(\mathbf{w}_t; \mathscr{B}_t) = \lambda \mathbf{w}_t - \frac{1}{|\mathscr{B}_t|} \sum\nolimits_{i \in \mathscr{B}_t} y_i \mathbf{x}_i (1 - \frac{1}{1 + \exp\{-y_i \mathbf{w}_t^\top \mathbf{x}_i\}}) \tag{7}$$

in which the model parameter $\mathbf{w}_t$ and the data part are combined with non-linear operations (i.e., exp and division). To be able to use provenance for rapid updates, we therefore linearize the non-linear operations in Equation (7) using *piecewise linear interpolation*, which leads to the following "Linearized Grad($\mathbf{w}_t; \mathscr{B}_t$)":

$$\text{Linearized Grad}(\mathbf{w}_t; \mathscr{B}_t) = \lambda \mathbf{w}_t - \frac{1}{|\mathscr{B}_t|} (\sum\nolimits_{i \in \mathscr{B}_t} a_{i,t} \mathbf{x}_i \mathbf{x}_i^\top \mathbf{w}_t + \sum\nolimits_{i \in \mathscr{B}_t} b_{i,t} y_i \mathbf{x}_i) \tag{8}$$

in which the two coefficients, $a_{i,t}$ and $b_{i,t}$, are generated after piece-wise linear interpolation is applied. As a consequence, Equation (8) shares a similar form with Equation (4), hence data provenance can also be used in Equation (8) for incrementally updating logistic regression models. Note that due to the linearization operations, the incrementally updated model parameters may not be the same as the ones retrained from scratch. However, we can prove that the difference between the two model parameters (quantified by L2 norm) is very small (see detailed analysis in [9]).

3.3 DeltaGrad

General machine learning models such as neural networks can be arbitrarily complex, making it challenging to separate the data part from the model parameters in the gradient expression. Therefore, instead of *explicitly* employing data provenance as in PrIU we proposed a second method, DeltaGrad, which updates model parameters by *implicitly* leveraging data provenance.

We start by rewriting the SGD update rule in Equation (3) as follows:

$$\mathbf{w}_{t+1}^U = \mathbf{w}_t^U - \eta_t \cdot \text{Grad}(\mathbf{w}_t^U; \mathscr{B}_t - R) = \mathbf{w}_t^U - \eta_t \cdot [\text{Grad}(\mathbf{w}_t^U; \mathscr{B}_t) - \text{Grad}(\mathbf{w}_t^U; R)] \tag{9}$$

In this update rule, after the removal of R instead of directly evaluating the gradient on the remaining samples in $\mathscr{B}_t$, i.e., $\mathscr{B}_t - R$, we subtract the gradient on the removed samples (i.e., $\mathscr{B}_t \bigcap R$) from the gradient on the full mini-batch (i.e., the term Grad($\mathbf{w}_t^U; \mathscr{B}_t$)). It is worth noting that Grad($\mathbf{w}_t^U; \mathscr{B}_t$) has the same form as Grad($\mathbf{w}_t; \mathscr{B}_t$) in Equation (2) except for the differences on the dependent model parameters (i.e. $\mathbf{w}_t$ and $\mathbf{w}_t^U$ resp.). We therefore *cache the term, Grad($\boldsymbol{w}_t; \mathscr{B}_t$) during the model training phase*, i.e. the evaluation of Equation (2), so that it can be reused for accelerating the evaluation of Grad($\mathbf{w}_t^U; \mathscr{B}_t$).

Specifically, we estimate Grad($\mathbf{w}_t^U; \mathscr{B}_t$) by estimating the difference between Grad($\mathbf{w}_t; \mathscr{B}_t$) and Grad($\mathbf{w}_t^U; \mathscr{B}_t$), which can be computed using the Cauchy mean value theorem:

$$\text{Grad}(\mathbf{w}_t^U; \mathscr{B}_t) - \text{Grad}(\mathbf{w}_t; \mathscr{B}_t) = \mathbf{H}([\mathbf{w}_t, \mathbf{w}_t^U]; \mathscr{B}_t) \left(\mathbf{w}_t^U - \mathbf{w}_t\right)$$

where $\mathbf{H}([\mathbf{w}_t, \mathbf{w}_t^U]; \mathscr{B}_t)$ represents the Hessian matrix integrated between $\mathbf{w}_t$ and $\mathbf{w}_t^U$ given a mini-batch $\mathscr{B}_t$. Since the explicit evaluation of the Hessian matrix is extremely time-consuming, we adapt the L-BFGS algorithm [14] to approximately evaluate the Hessian-vector product $\mathbf{H}([\mathbf{w}_t, \mathbf{w}_t^U]; \mathscr{B}_t) \left(\mathbf{w}_t^U - \mathbf{w}_t\right)$. In this modified version of the L-BFGS algorithm, the input consists of the vector $(\mathbf{w}_t^U - \mathbf{w}_t)$, the history model parameters, $\mathbf{w}$ and $\mathbf{w}^U$

from previous SGD iterations, as well as the corresponding gradients (i.e., $\text{Grad}(\mathbf{w}_t; \mathscr{B}_t)$ and $\text{Grad}(\mathbf{w}_t^U; \mathscr{B}_t)$) from those iterations. As a result, $\text{Grad}(\mathbf{w}_t^U; \mathscr{B}_t)$ is approximately evaluated with the following formula:

$$\text{Grad}(\mathbf{w}_t^U; \mathscr{B}_t) \approx \text{Grad}(\mathbf{w}_t; \mathscr{B}_t) + g^{\text{L-BFGS}}(\mathbf{w}_t^U - \mathbf{w}_t, \{(\mathbf{w}_{t_r}^U, \mathbf{w}_{t_r}, \text{Grad}(\mathbf{w}_{t_r}^U; \mathscr{B}_{t_r}), \text{Grad}(\mathbf{w}_{t_r}; \mathscr{B}_{t_r}))\}_{r=1}^m) \quad (10)$$

in which $g^{\text{L-BFGS}}(\cdot)$ denotes the L-BFGS algorithm and the history model parameters, $\mathbf{w}_{t_r}$, and the corresponding gradients, $\text{Grad}(\mathbf{w}_{t_r}; \mathscr{B}_{t_r})$, are regarded as the "implicit" provenance.

Note that the computation of $\text{Grad}(\mathbf{w}_t^U; \mathscr{B}_t)$ can lead to approximation errors. Therefore, in order to guarantee that the updated model parameters calculated by this approximate gradient are not far away from the expected ones, we compute $\text{Grad}(\mathbf{w}_t^U; \mathscr{B}_t)$ from scratch in the first few SGD iterations and periodically compute it from scratch afterwards. It can be shown that the updated model parameters calculated in this way are very close to the ones retrained from scratch [10].

4 Security concern: Model inversion attacks

In the previous section, we showed how provenance can be used both explicitly (in the case of linear and logistic regression) and implicitly (in the case of more complex models such as neural networks) to incrementally update machine learning models. Going beyond incremental updates, we now discuss how a provenance-based framework can be used to defend against an emerging security concern: *model inversion attacks*.

4.1 Preliminaries

The recently established General Data Protection Regulation (GDPR) guidelines [1] state that users have the right to have private data items removed from the entities storing those items. However, this is not as simple as just deleting the data items: If they have been used as training data in state-of-the-art machine learning systems, the effect of these data items must also be "erased" from the models that these systems have learned and rely on. Otherwise, the systems may be subject to *model inversion attacks*, a type of attack in which the adversary is able to restore the private data items from the machine learning models without accessing the data items [12]. Therefore, the ultimate goal of *unlearning a machine learning model* is to give an updated model in which the private data items have been "forgotten", i.e. the model behaves as if the private data items never appeared in the training set thus safeguarding against model inversion attacks. Therefore, in addition to *efficiency* (i.e. the time to update the model) and *performance* guarantees (i.e. the predictive power of the updated model), an ideal unlearning algorithm should guard against model inversion attacks.

It is worth noting that, depending on the adversary's knowledge, the vulnerability of the model can vary. Generally speaking, there are two model inversion attack settings: *black-box* and *white-box* [12]. In the black-box setting, the adversary can only use the model output to launch the attack, whereas in the white-box setting the adversary can also use auxiliary information such as the model type, model parameters, or even the remaining training samples (in the extreme case). It is therefore much more challenging to defend against white-box attacks than black-box attacks.

4.2 Vulnerability of current machine unlearning methods

Current machine unlearning methods fall into one of several different categories: 1) Retraining-based methods; 2) Methods based on differential privacy; and 3) One-step update methods. We give an overview of each, and then compare them as well as our provenance-based approach with respect to *efficiency* (i.e. the time to update the model), *performance* (i.e. the predictive power of the updated model), and their ability to guard against model-inversion attacks. A summary of this comparison can be found in Table 2.

Retraining-based methods One straightforward machine unlearning strategy is to retrain the models from scratch, which can fully erase the removed training samples from the models and maintain the model prediction performance. However, this is very inefficient when frequent unlearning requests occur. To mitigate this, [11] proposes to shard the training set into smaller partitions, construct one local model for each partition and only retrain the local model if a deletion request hits the corresponding partition. Note that this method is not efficient if training samples from each partition are removed simultaneously, leading to the reconstruction of all the models.

Differential privacy-based methods. These methods, [15, 16], build on the classical notion of differential privacy [17]. The goal is to update the model in a single step, and then add some carefully designed random noise to the incrementally updated model so that it is indistinguishable from the one retrained from scratch, to which the same level of random noise has been added. In particular, [15] leverages the following Newton update mechanism for updating linear models (e.g., linear regression models and logistic regression models):

$$\mathbf{w}_*^U = \mathbf{w}_* + \mathbf{H}^{-1}(\mathbf{w}_*; D_{\text{remaining}}) \cdot \text{Grad}(\mathbf{w}_*; R), \tag{11}$$

in which $\mathbf{H}^{-1}(\mathbf{w}_*; D_{\text{remaining}})$ represents the inversed Hessian matrix (i.e. the second order gradient) on the remaining training set, $D_{\text{remaining}}$. Then $\mathbf{w}_*^U$ is perturbed with some randomly drawn noise vector $\mathbf{b}$ to hide the gradient information of the removed training samples. Despite the efficiency and perfect privacy guarantees of this type of solution, they suffer from the fact that the added noise may hurt the model prediction performance [18, 19].

One-step update methods. The second type of method incrementally updates the model in one-step but does not introduce extra noise [20, 18]). Specifically, these solutions can be represented by the following abstract formula:

$$\mathbf{w}_*^U = \mathbf{w}_* + G(R, D_{\text{remaining}}) \tag{12}$$

in which G is a function taking the removed training set R and the remaining training set $D_{\text{remaining}}$ as arguments. For example, Equation (12) could be the one-step Newton update (Equation (11)) in which the function G could be expressed as:

$$G(R, D_{\text{remaining}}) = \mathbf{H}^{-1}(\mathbf{w}_*; D_{\text{remaining}}) \cdot \text{Grad}(\mathbf{w}_*; R)$$

As observed in [21], the product between the inverse of the Hessian matrix, $\mathbf{H}^{-1}(\mathbf{w}_*; D_{\text{remaining}})$, and the vector $\nabla F(\mathbf{w}_*, R)$ in the above formula can be effectively evaluated using conjugate gradients [22] or the stochastic estimation method of [23]. To further speed up the above computations, by leveraging the fact that R is far smaller than $D_{\text{remaining}}$, $\mathbf{H}^{-1}(\mathbf{w}_*; D_{\text{remaining}})$ could be regarded as the low-rank updates on $\mathbf{H}^{-1}(\mathbf{w}_*; D_{\text{remaining}} + R)$, which is the inverse of the Hessian matrix on the full training set and thus can be cached beforehand. According to [18], such low-rank updates could be effectively computed by employing the Sherman-Morrison-Woodbury formula [24]. Note that the models incrementally updated in this manner are also very close to the retrained ones [21].

Despite the efficiency and predictive performance, this type of method suffers from model inversion attacks. In what follows, we describe at least two scenarios in which in which model inversion attacks can occur.

Scenario 1 Consider the following extreme scenario where everything about the model except for the removed sample set R is revealed to the adversary, which includes the remaining training samples $D_{\text{remaining}}$, the original model parameter, $\mathbf{w}_*$ and the incrementally updated model parameter, $\mathbf{w}_*^U$). However, in this case, the value of $G(R, D_{\text{remaining}})$ could be obtained by calculating the difference between $\mathbf{w}_*$ and $\mathbf{w}_*^U$, and thus R could be reconstructed by solving the following optimization problem:

$$\text{argmin}_{R'} \|G(R', D_{\text{remaining}}) - G(R, D_{\text{remaining}})\| = \text{argmin}_{R'} \|G(R', D_{\text{remaining}}) - (\mathbf{w}_*^U - \mathbf{w}_*)\| \tag{13}$$

It is worth noting that this extreme scenario could indeed occur in practice. First of all, different versions of models could be accessed by the adversary simultaneously. For example, machine learning models are increasingly used inside DBMSs, e.g., learned indexes [25] and learning-based query optimizers [26]. The models are typically constructed by taking the data in the DBMS as the training dataset. Due to updates on the underlying data, the models could also be updated, thus leaving different copies of the model snapshots, which could be logged inside the DBMS. Plus, due to the reproducibility requirements, the input training samples might also be stored for later use, which thus might be accessed by the adversary.

Scenario 2 In what follows, let us consider a more practical scenario where the knowledge of the adversary is limited. Specifically, we assume that only the updated model $\mathbf{w}_*^U$ is revealed to the adversary, which is similar to the standard assumption of the white-box attack. Then given $\mathbf{w}_*^U$, we assume that there is a strong white-box model inversion attack tool that the adversary could employ to reconstruct the remaining training samples, $D_{\text{remaining}}$ [12]. Note that in this scenario, the original model parameter $\mathbf{w}_*$ is not available to the adversary. Therefore, it is not enough to directly employ Equation (4.2) for reconstructing the removed training sample set, R. Instead, the adversary could **jointly** construct $\mathbf{w}_*$ during the derivation of R by leveraging the dependence of $\mathbf{w}_*$ on R (recall that $\mathbf{w}_*$ is trained on the full training set $D_{\text{remaining}} + R$). This could be formalized as the following bi-level optimization problem:

$$\begin{aligned} &R = \text{argmin}_{R'} \|G(R', D_{\text{remaining}}) - (\mathbf{w}_*^U - \mathbf{w}_*)\|, \\ &\text{where } \mathbf{w}_* = \min_{\mathbf{w}} \{F(\mathbf{w}; D_{\text{remaining}}) + F(\mathbf{w}; R')\} \end{aligned} \tag{14}$$

which could be effectively solved by using the optimization method proposed in [27].

Provenance-based approach. In contrast, both of our methods, PrIU and DeltaGrad, can resist the above scenarios. To see this, consider the following abstract formula representing how PrIU and DeltaGrad incrementally update the model:

$$\text{Grad}(\mathbf{w}_t^U; D_{\text{remaining}}) \approx \text{Prov}(\mathbf{w}_t^U; D_{\text{remaining}} + R) - \text{Grad}(\mathbf{w}_t^U; R) \tag{15}$$

in which,

$$\text{Prov}(\mathbf{w}_t^U; D_{\text{remaining}} + R) \approx \text{Grad}(\mathbf{w}_t^U; D_{\text{remaining}} + R)$$

Assuming that the the provenance term $\text{Prov}(\mathbf{w}_t^U; D_{\text{remaining}} + R)$ and the removed training sample set R are not accessible, then the adversary must solve the following formula to recover R:

$$\text{Grad}(\mathbf{w}_t^U; D_{\text{remaining}}) \approx \text{Grad}(\mathbf{w}_t^U; D_{\text{remaining}} + R) - \text{Grad}(\mathbf{w}_t^U; R) \tag{16}$$

which, however, holds for an arbitrary set of samples R' instead of simply holding for R. As a consequence, the best that the adversary can do is to randomly guess what R is.

Table 2: Comparison of state-of-the-art machine unlearning methods

	Privacy	Prediction performance	Efficiency
Retraining from scratch	✓	✓	
Partition-based Retraining [11]	✓	✓	
Differential-privacy-based methods	✓		✓
One-step-update methods		✓	✓
Provenance-based methods	✓	✓	✓

A summary of the comparison between current machine unlearning strategies can be found in Table 2. We next show how our provenance-based method can be used to avoid model-inversion attacks while achieving high prediction performance and efficiency.

4.3 Adding provenance support for secure machine unlearning

In future work, our goal is to develop a machine unlearning system that can achieve *real-time and secure updates on general machine learning models* such that the updated models are *almost identical to the retrained models*, thus not hurting the prediction power. The strategy to achieve this goal could vary depending on the threat model. However, we expect that in the worst scenario mentioned in Section 4.2, adding provenance support to the machine unlearning system is essential.

Specifically, we plan to generalize the idea of PrIU and DeltaGrad for general neural network models, such that the footprint of the removed training samples is erased from the model parameters collected across all the SGD iterations, and the updated model parameter at each SGD iteration is almost identical to the one for retraining the model from scratch. Given this, despite the knowledge of the adversary about the model and the remaining training samples, it would be almost impossible to reconstruct the removed training samples since the remaining information on the updated training trajectory does not include the removed training samples. This can therefore bring a better security guarantee than the One-step-update methods (as introduced in Section 4.2) in which the function G still encodes information about the removed samples, leading to the leakage of those samples.

Figure 3 illustrates how the envisioned provenance-enabled model unlearning system would work. In this figure there are two main components: the "Training loop component" and the "Updating loop component". In the "Training loop component" necessary provenance information is collected during the training process on the neural network models, where SGD is assumed to be the default training method (similar to PrIU and DeltaGrad). The dominant provenance information would be different versions of model snapshots (i.e., the model parameters and the computed gradients) captured at each iteration. Such provenance information would then be stored in secure storage, which would then be retrieved for incrementally computing the updated model in the "Updating loop component" after sensitive training data is deleted. As analyzed in the previous subsection, this could fully erase the footprint of the removed training samples from the model parameters at each SGD step, thus guarding against the model inversion attacks.

Discussion Note that there are several existing solutions that also aim to delete the footprint of the removed sensitive training samples from the entire training trajectory. For example, [28] proposes an efficient way to incrementally update K-means models through caching and reusing the information of all the clusters at each training iteration for model updates. Such cached cluster information could be therefore considered as provenance information. However, to facilitate effective unlearning, some necessary modifications are applied to the K-means models, which may degrade the model prediction performance.

Challenges and research opportunities Several challenges remain for developing a provenance-enabled machine unlearning system for general neural network models. First of all, general neural network models are non-convex, which is beyond the model classes that PrIU or DeltaGrad support. Therefore, we would envision that a new provenance-based unlearning method is necessary to support incremental updates on general non-convex neural network models. Inspired by PrIU and DeltaGrad, the models incrementally updated in this way could be approximately close to the retrained models, but with rigorous theoretical guarantees on the smallness of the approximation errors. One potential idea to design this new unlearning method is to relax the strong-convexity assumption on the model class in DeltaGrad such that general non-convex models could be handled. The main bottleneck is the strong dependency of DeltaGrad on the L-BFGS algorithm, in which the strongly convex objective functions are essential. We notice that this assumption has been relaxed in many extended versions of the L-BFGS algorithm (e.g., [29]), which could be potentially adapted for extending DeltaGrad.

In addition, due to the high complexity of state-of-the-art neural neural networks, the model snapshots at each SGD iteration could be extremely large, incurring a prohibitively high overhead for the entire unlearning system. For example, the ResNet18 network for vision tasks has around 11 million parameters [30] while the GPT-3 model [31] for natural language processing tasks has around 175 billion parameters. The problem would become

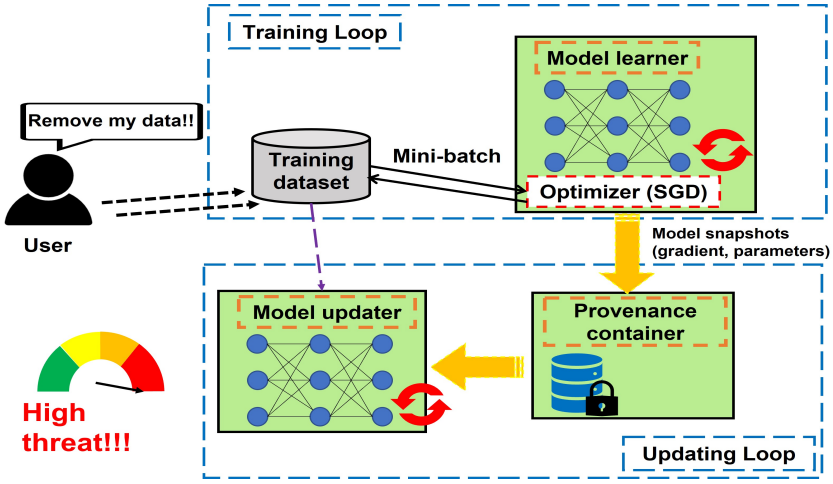


Figure 3: Provenance-enabled model unlearning system

even worse if their parameters and gradients were required to be repetitively recorded in the provenance-enabled model unlearning system. As a consequence, it is essential to build a version control system on large neural network models such that the model snapshots could be effectively captured, stored and retrieved.

One expected characteristic of such a version control system is that the models from different versions (i.e. from different SGD iterations) could be compressed to reduce the storage overhead. Ideally, the compression should be lossless so that the variations between different versions can be captured without adding extra approximation errors after those models are uncompressed during the model update phase. Otherwise, it could potentially amplify the approximation errors brought by the unlearning algorithm itself, thus hurting the quality of the unlearned models.

The version control problem for machine learning models has been recently studied in [32]. Specifically, the difference between the models from different SGD iterations is calculated first, which can be represented by a set of matrices. Then each element of each matrix, i.e., one float number, is approximately represented with k-byte (k=16 or 8) integers through the quantization operations. This is then followed by compressing the higher-order bits of the quantized representations across different versions of the models throughout the SGD iterations, which can yield significant savings (see the experiments of [32]). However, since the quantization operations can produce approximation errors, as discussed above, when this solution is applied to the machine unlearning pipeline such errors may lead to significant deviations of the incrementally updated models from the retrained ones.

Another requirement for the provenance-enabled model unlearning system, as shown in Figure 3, is that the collected provenance information be cached in secure storage. Otherwise, the adversary could easily reconstruct the removed training samples from the cached provenance of those samples by using the attack paradigm presented in Section 4.2. Finally, it is worth noting that the chance of the the security threat mentioned in Section 4.2 actually occuring may be quite low in practice due to the limited knowledge of the adversary on the models. For instance, in typical online systems, the deployed machine learning models are released as an open API [1] where the adversary only has black-box access, meaning that only the model output given one input sample can be obtained through the API. In this scenario, it might be inappropriate to use provenance-based unlearning methods due to their relatively high overhead with respect to other unlearning methods. It would be interesting to explore the applicability of existing unlearning methods under different threat assumptions and rank them based on their risk of leaking the removed training samples.

[1] see e.g., Google prediction API: https://cloud.google.com/ai-platform/prediction/docs/reference/rest/v1/projects/predict

5 Conclusions

In this paper, we reviewed our provenance-based techniques, PrIU and DeltaGrad, for incrementally updating machine learning models and showed the connection to incrementally updating database views. We then studied the privacy implications of machine unlearning techniques, and analyzed the capability of PrIU and DeltaGrad as well as other state-of-the-art unlearning techniques on defending against the model inversion attack. Our analysis reveals that provenance is essential for the unlearning process to guard against this type of attack without hurting performance and the model prediction power. Based on this observation, we envision a provenance-based unlearning system, which could effectively unlearn general machine learning models in a secure manner. We also outlined critical technical challenges and potential solutions, paving the way towards building such systems.

Acknowledgements

This material is based upon work that is in part supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0047. Partial support was provided by NSF Awards 1547360 and 1733794. Tannen's work at the National University of Singapore was supported in part by the Kwan Im Thong Hood Cho Temple/Avalokiteśvara.

References

[1] P. Voigt, and A. Von dem Bussche The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

[2] M. H. Quenouille, Notes on bias in estimation. *Biometrika*, 43 (3/4), pp.353-360, 1956.

[3] D. N. Politis, and J. P. Romano and M. Wolf Subsampling. *Springer Science & Business Media*, 1999.

[4] A. Ghorbani, and J. Zou. Data shapley: Equitable valuation of data for machine learning. *In International Conference on Machine Learning*, pp. 2242-2251. PMLR, 2019.

[5] A. Gupta, and I. S. Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Eng. Bull.*, 18(2), pp.3-18., 1995

[6] A. Labrinidis, and Y. Sismanis. View Maintenance, pages 3326–3328. *Springer*, Boston, MA, 2009.

[7] T. J. Green, G. Karvounarakis, and V. Tannen, Provenance semirings. *In Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* pp. 31-40, 2007.

[8] Z. Yan, V. Tannen, and Z. G. Ives. Fine-grained Provenance for Linear Algebra Operators. In *TaPP*. 2016.

[9] Y. Wu, V. Tannen, and S. B. Davidson. Priu: A provenance-based approach for incrementally updating regression models. *In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 447-462. 2020.

[10] Y. Wu, E. Dobriban, and S. B. Davidson. DeltaGrad: Rapid retraining of machine learning models. *In International Conference on Machine Learning*, pp. 10355-10366. PMLR, 2020.

[11] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot. Machine unlearning. *In 2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141-159, 2021.

[12] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. *In Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1322-1333. 2015.

[13] Y. Amsterdamer, D. Deutch, and V. Tannen. Provenance for aggregate queries. *In Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 153-164. 2011.

[14] A. Mokhtari, and A. Ribeiro. Global convergence of online limited memory BFGS. *The Journal of Machine Learning Research* 16, no. 1 (2015): 3151-3181.

[15] C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten.Certified Data Removal from Machine Learning Models. *In International Conference on Machine Learning*, pp. 3832-3842. PMLR, 2020.

[16] S. Neel, A. Roth, and S. Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. *In Algorithmic Learning Theory*, pp. 931-962. PMLR, 2021.

[17] C. Dwork, and A. Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* 9, no. 3-4 (2014): 211-407.

[18] Z. Izzo, M. A. Smart, K. Chaudhuri, and J. Zou. Approximate data deletion from machine learning models. *In International Conference on Artificial Intelligence and Statistics*, pp. 2008-2016. PMLR, 2021.

[19] S. Zanella-Béguelin, L. Wutschitz, S. Tople, V. Rühle, A. Paverd, O. Ohrimenko, B. Köpf, and M. Brockschmidt. Analyzing information leakage of updates to natural language models. *In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 363-375. 2020.

[20] A. Golatkar, A. Achille, and S. Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. *In European Conference on Computer Vision*, pp. 383-398. Springer, Cham, 2020.

[21] P. W. Koh, and P. Liang. Understanding black-box predictions via influence functions. *In International Conference on Machine Learning*, pp. 1885-1894. PMLR, 2017.

[22] J. Martens.Deep learning via hessian-free optimization. In *ICML*, vol. 27, pp. 735-742. 2010.

[23] N. Agarwal, B. Bullins, and E. Hazan. Second-order stochastic optimization in linear time. *stat* 1050 (2016): 15.

[24] J. Sherman, and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics* 21, no. 1 (1950): 124-127.

[25] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis. The case for learned index structures. *In Proceedings of the 2018 International Conference on Management of Data*, pp. 489-504. 2018.

[26] R. Marcus, P. Negi, H. Mao, C. Zhang, M. Alizadeh, T. Kraska, O. Papaemmanouil, and N. Tatbul. Neo: A Learned Query Optimizer. *Proceedings of the VLDB Endowment* 12, no. 11.

[27] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng. Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting. *Advances in Neural Information Processing Systems* 32 (2019): 1919-1930.

[28] A. Ginart, M. Guan, G. Valiant, and J. Zou. Making AI Forget You: Data Deletion in Machine Learning. *Advances in neural information processing systems* (2019).

[29] D. Li, and M. Fukushima. A modified BFGS method and its global convergence in nonconvex minimization. *Journal of Computational and Applied Mathematics* 129, no. 1-2 (2001): 15-35.

[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.

[31] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan et al. Language Models are Few-Shot Learners. *arXiv e-prints (2020)*: arXiv-2005.

[32] H. Miao, A. Li, L. S. Davis, and A. Deshpande. Towards unified data and lifecycle management for deep learning. *In 2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pp. 571-582. IEEE, 2017.

Navigating Compliance with Data Transfers in Federated Data Processing

Kaustubh Beedkar
TU Berlin
kaustubh.beedkar@tu-berlin.de

Jorge Quiané
TU Berlin & DFKI
jorge.quiane@tu-berlin.de

Volker Markl
TU Berlin & DFKI
volker.markl@tu-berlin.de

Abstract

In an increasingly digital world, compliance with data regulations play an important role. More and more individuals are rapidly getting concerned with the way their data is being stored and processed by organizations. Therefore, it is crucial that data processing be subjected to regulatory obligations at its core. Yet, achieving compliance with data regulations requires the entire data processing pipeline to be revisited to embrace data policies as first-class citizens. In this paper, we present our work on novel systems and methods for federated data processing, where the processing of geo-distributed data is subjected to data transfer regulations. We showcase our work on compliant geo-distributed data processing and present research challenges and opportunities for a federated data processing system to make compliance truly its first-class citizens.

1 Introduction

Federated data processing has been a standard model for virtual integration of disparate data sources, where each source upholds a certain amount of autonomy. While early federated technologies resulted from mergers, acquisitions, and specialized corporate applications, recent demand for decentralized data storage and computation in information marketplaces[32]) and for geo-distributed data analytics [33, 22, 14] has made federated data services an indispensable component in the database market. Cloud providers such as AWS, Google, and Microsoft have also adopted distributed query capabilities within their products to support federated data processing.

Running analytics in a federated environment mainly relies on distributed query processing frameworks, such as those based on data integration systems (e.g., [25]) and/or multi-database systems (e.g.,[3, 31]). At high-level, a distributed query processing framework provides a unified query interface to query distributed and decentralized data. It transparently translates a user-specified query into a so-called query execution plan. To do so, a query optimizer considers distributed execution strategies (involving distributing query operators like join or aggregation across compute nodes), communication cost between compute nodes, and introduces a global property that describes where, i.e., at which site, processing of each plan operator happens. For example, a two-way join query over data sources in Asia, Europe, and North America may be executed by first joining data in North America and Europe and then joining with the data in Asia. As one can notice, federated queries implicitly ship data (i.e., intermediate query results) between compute sites. While several performance aspects, such as

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

bandwidth, latency, communication cost, and compute capabilities have received great attention, the federate nature of data processing has been recently challenged by data transfer regulations (or policies) that restrict the movement of data across geographical (or institutional) borders or by any other rule of data protection that may apply to the data being transferred between certain sites. European directives, for example, regulate transferring only certain information fields (or combinations thereof), such as non-personal information or information not relatable to a person. Likewise, regulations in Asia may also impose restrictions on data transfer. Non-compliance to such regulatory obligations has attracted fines in the tune of billions of dollars[10]. It is, therefore, crucial to consider compliance with respect to legal aspects when analyzing federated data.

Nevertheless, complying with regulations when transferring data is a big challenge today. One has to expand the capabilities of modern federated data processing systems to aid data controllers (i.e., entities that determine what data and how the data should be processed) and data processors (i.e., entities that provide data storage and processing capabilities) in navigating compliance with data transfers. In particular, we need (a) a declarative language for expressing data transfer rules, (b) to revisit query rewriting and optimization techniques to translate user queries transparently into compliant query execution plans, and (c) to revisit query execution to support decentralized data processing across heterogeneous compute nodes.

In this paper, we outline the main system challenges required for complying with data transfer obligations in the context of federated data processing. We showcase our work on compliant data processing, which offers limited capabilities in navigating compliance. We then discuss our current research endeavors that overcome prior limitations and discuss open problems and research challenges.

2 Problem Scope & Challenges

We start by giving a birds-eye view of federated data processing systems (FDPS) and then outline aspects of data regulations that affect the transfer of data between national (or institutional) borders. We then discuss the challenges that we set out to address in order to expand the capabilities of FPDS to navigate compliance with data transfers.

2.1 A Brief Recall on Federated Data Processing

An FDPS consists of three major components as illustrated in Figure 1: a data interface, a query optimizer, and a query processor. The data interface provides end-users (e.g., data analysts or data administrators) the ability to query and process data that is stored across distributed data stores in a unified manner. The data interface upon receiving a user query (e.g., SQL) parses and translates it into a framework-specific internal structure (e.g., a logical query plan). The query optimizer then rewrites the logical query plan into a query execution plan (QEP). It extends a single-site data processing across distributed compute notes. To do so, it considers communication costs between compute nodes and introduces a global property that describes where, i.e., at which site processing of each plan operator happens. The query processor then "orchestrates" the actual execution of the query, which results in the transfer of (intermediate) query data between compute nodes. In geo-distributed environments, compute nodes are located across national (or institutional) borders. In this context, the transfer of data between sites may result in non-compliance to data transfer regulations.

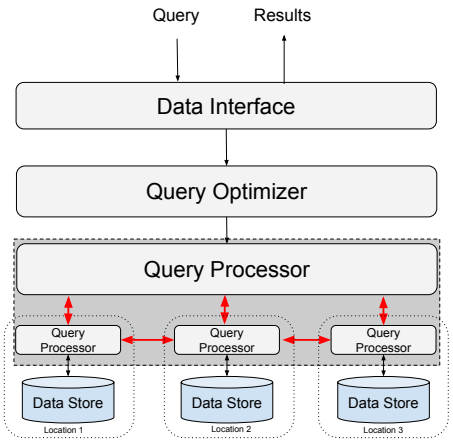


Figure 1: Federated Data Processing

2.2 Primer on Data Transfer Regulations

Data regulations, such as EU's General Data Protection Regulation (GDPR) [1] or California Consumer Privacy Act (CCPA) [2] significantly affect how data is stored, processed, and transfer. In this section, we aim to understand regulations from the perspective of data transfers. To achieve that, we analyze GDPR articles that regulate the transfer of data across national borders[1].

GDPR articles 44–50 explicitly deal with the transfer of data across national borders. Among these, we identified two articles and one recital wherein the legal requirements for transferring data fundamentally affect FDPS components.

Article 45: Transfers on the basis of an adequacy decision. The article dictates that transfer of data may take place without any specific authorization, e.g., when there is adequate data protection at the site where data is being transferred or when data is not subjected to regulations (i.e., when the data does not follow the definition of personal data as in Article 4(1)).

Article 46: Transfers subject to appropriate safeguards. This article prescribes that (in the absence of applicability of Article 45) data transfer can take place under "appropriate safeguards". Based on the European Data Protection Board (EDPB) recommendations that supplement transfer tools, *pseudonymisation* of data (as defined under Article 4(5)) is considered as an effective supplementary method.

Recital 108: Transfers under measures that compensate lack of data protection. Data after adequate *anonymization* (i.e., when resulting data does not fall under Article 4(1) and as described in Recital 26) does not fall under the ambit of GDPR and therefore can be transferred.

Discussion. Based on the above regulations, we observe that depending on the data and to where that data is being transferred, we can classify data transfer regulations into:

- *No restrictions on transfer.* Some data maybe allowed to be transferred unconditionally, and some to only certain locations.
- *Conditional restrictions on transfer*. For some data, only derived information (such as aggregates) or only after anonymization, can be transferred to (certain) locations.
- *Complete ban on transfer*. Some data, no matter whatsoever, must not be transferred outside.

2.3 Compliance by Design: Research Challenges

Our overreaching goal is to develop methods and systems that aid *data controllers* (entities that control what data and how the data should be processed) and *data processors* (entities that processes data on behalf of a controller) by providing appropriate safeguards within FDPS such that transfer—as a result of federated data processing—of data across borders complies to regulatory obligations described above.

Declarative Data Transfer Rules. The first and foremost challenge to achieve compliance by design is to have declarative languages for specifying data transfer regulations. Doing so is not trivial as regulations affect data differently depending on its type, it's processing, and the location where it is processed (or transferred). For example, regulations may apply to an entire dataset, parts of it, or even information derived from it. Furthermore, datasets are heterogeneous in their data models (e.g., graphs, relational, and textual). Therefore, devising a declarative language where one can specify data constraints in an easy and effective manner is far from being simple.

[1] We note that compliance to GDPR aspects, such as collecting, securing, storing, deleting are beyond the scope of our current focus (see discussion in Section 5).

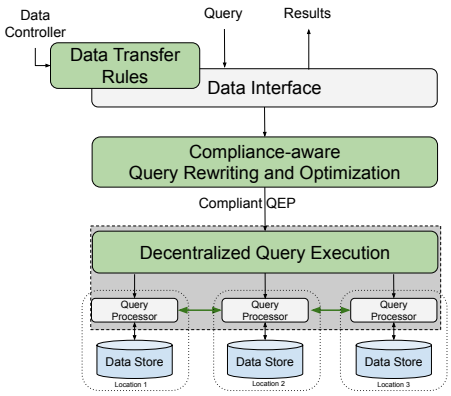


Figure 2: Compliant FDP

Compliance-aware Query Rewriting and Optimization Once a user specifies the data policies on her data, she then needs effective and efficient ways to process federated queries in a manner that the processing is compliant. Achieving so is challenging as we need to extend query rewriting and optimization capabilities. A query optimizer should be able to transparently translate logical query plans into compliant query execution plans by "injecting" operations that transform data such that the transformed data prescribes to regulations affecting its transfer.

Decentralized Query Execution Lastly, we also need to revisit query processors than can execute a compliant query plan across distributed (potentially heterogeneous) compute nodes. In contrast to current approaches that employ a mediator-based execution, we need meta-execution engines, which can delegate query execution to disparate compute nodes and de-centrally execute the query, without involving itself in the query execution. Achieving so is crucial to make sure that data processing happens on the locations prescribed by the query optimizer, irrespective of the location of the processor.

3 Compliance-aware Query Rewriting and Optimization

We start by giving the overall idea of our approach for achieving compliant data processing. We then discuss our research on supporting compliance assuming relational workloads before outlining open problems and current research directions going beyond the relational world.

3.1 Overall Idea

A crucial aspect in adhering to data transfer regulations is to transform the data in a way that renders it suitable to be transferred across borders. From the perspective of a FDPS, transforming (intermediate) data before shipping it to another compute location can be considered as performing additional *masking operations* on the data. To illustrate this, consider a two-way join query that access data stored across Europe, North America, and Asia. Figure 3 (left) illustrates a logical query plan for this query where orange boxes denote cross-border operations (e.g., join operators) that require inputs from two or more sources and blue circles denote other query operators (e.g., map, filter, or aggregate). On the right, we illustrate a corresponding execution plan, where the optimizer decides at which site the processing of each plan operator should happen (e.g., both cross-border operators must be performed in North America). The optimizer also rewrites the query by reordering the query operators (e.g., selection pushdowns) and by "injecting" data masking operators (shown by red boxes) as a means to provide appropriate safeguards for cross-border data transfers. In this example, both cross-border operations happen in North America, and data from EU and Asia is transformed by a masking operator before being shipped to North America.

3.2 Navigating Compliance in the Relational Paradigm

In our current approach, we confine to processing of data that is stored in geo-distributed SQL databases and propose query rewriting and optimization techniques that preserve the query semantics. In more detail, we focus on data transfer rules that can be adhered to by data masking via relational operations (e.g., project, aggregate, or filter) such that the resulting compliant QEP retains the query semantics, i.e., the output of the query should be the same as if there were no data transfer constraints. For instance, a projection operator can mask certain

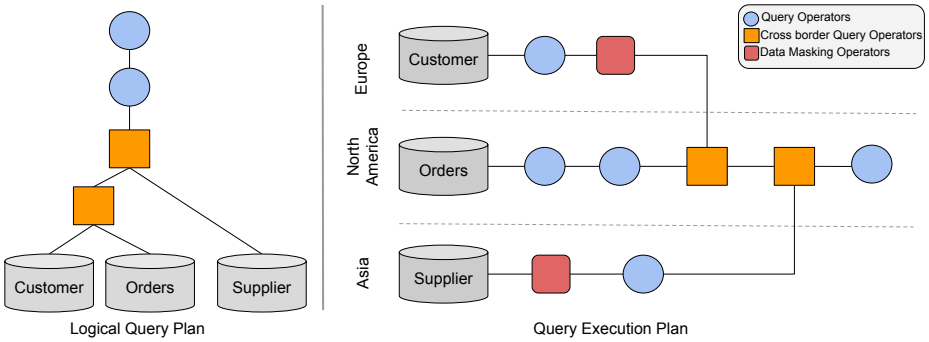


Figure 3: Illustration of Query Rewriting and Optimization: (left) Logical query specified by user and (right) execution plan derived by the query optimizer after injecting masking operators.

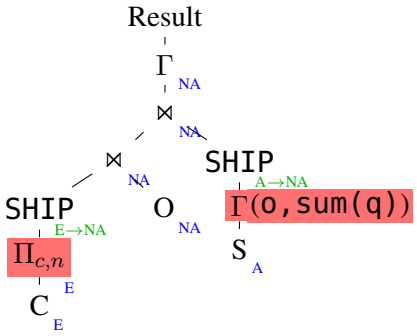
columns by projecting them out before the (intermediate data) is transferred to another location, and when the masked columns are not required by the query later.

Let us illustrate this approach by expanding upon the previous example. Consider the following schema for the Customer, Orders, and Supplier tables along with data transfer rules that apply to data at each location.

`Customer`	(**c**ustkey, **n**ame, **a**cctbal, **m**ktseg, **r**egion)	Customer data can be transferred outside only after suppressing account balance information
`Orders`	(**c**ustkey, **o**rdkey, **t**otprice)	Only aggregated Orders data can be transferred to Asia and an order's price cannot be transferred to Europe
`Supply`	(**o**rdkey, **q**uantity, **e**xtprice)	Only aggregated Supply data for orders' quantity and extended price from Asia can be transferred to North America.

Furthermore, consider a query Q_{ex}

```
SELECT C.name, SUM(O.totprice), SUM(S.quantity)
FROM Customer AS C, Orders AS O, Supply AS S
WHERE C.custkey=O.custkey AND O.ordkey=S.ordkey
GROUP BY C.name
```



The query plan on the right shows a compliant QEP as derived by our query optimizer (discussed below). Here the `SHIP` operator describes the point where intermediate results are communicated between two sites and Γ denotes the aggregation operator. Annotations corresponding to each operator describe where processing of the plan operator should happen. Observe, that executing such a plan will not violate any of the above rules: it performs both join operations in North America, masking data via the projection operator $\Pi_{c,n}$ suppresses the account balance information of Customers (before the data is shipped from Europe to North America) and via the aggregation operator Γ(`o,sum(q)`) suppresses the orders' quantity (prior to shipping data from Asia to North America), as desired by the rules.

Policy Expression Language. One of the challenges in automatically translating user-specified queries into compliant QEPs is to first integrate rules into the query optimization framework. For this, we have developed a policy expression language that provides a simple and intuitive (SQL-like) syntax to specify *which* and *where*

data are allowed to be transferred. We basically define two kinds of policy expressions: basic and aggregate expressions. A basic expression is of the form of a Select-Project query that can specify restrictions pertaining to certain tables, rows, and/or columns. An aggregate expression is of the form of a Select-Project-GroupBy query and further allows specifying restrictions pertaining to the transfer of aggregated information.

Basic Expressions. Basic expressions allow specifying shipping of certain rows and columns of a table to another location and have the following syntax:

```
ship attribute list from table to location list where condition list
```

This expression specifies cells, i.e., rows and columns, of a table to be transferred without affecting the query semantics.[2] The specified cells from the table in the **from** clause (i) belong to both columns in the **ship** clause and tuples that satisfy the predicates in the **where** clause, and (ii) can be transferred to locations in the **to** clause. Intuitively, if a subquery accesses only the specified cells, then its output can be transferred to locations specified in the expression. Consider the data transfer rule from the above example, which does not allow for shipping the account balance information of customers outside Europe. Suppose the rule also allowed for shipping customer's mktsegment and region information to North America for commercial customers. We can use the following two policy expressions:

```
ship custkey,name from Customer C to Asia, North America
ship mktseg, region from Customer C to North America where mktseg='commercial'
```

Aggregate Expressions. For certain data, transfer rules only allow shipping of aggregated information. For these cases, we have aggregate expressions that allow us to specify aggregations over columns. The syntax of an aggregate expression is given as:

```
ship attribute list as aggregates aggregate types from table to location list
where condition list group by attribute list
```

In the above syntax, the list of attributes in the **ship** clause specifies cells of columns that should be aggregated before being transferred to locations in the location list. The **as aggregate** clause specifies aggregation functions that should be used to aggregate specified cells. As before, the specified cells must belong to columns in the attribute list for the tuples that satisfy the predicate in its **where** clause. Lastly, the **group by** clause specifies lists of grouping attributes for which the specified cells can be grouped by zero, one, or more attributes from its attribute list. Consider again the Customer data from the above example and assume that account balance information can be transferred only after aggregating. A possible expression is:

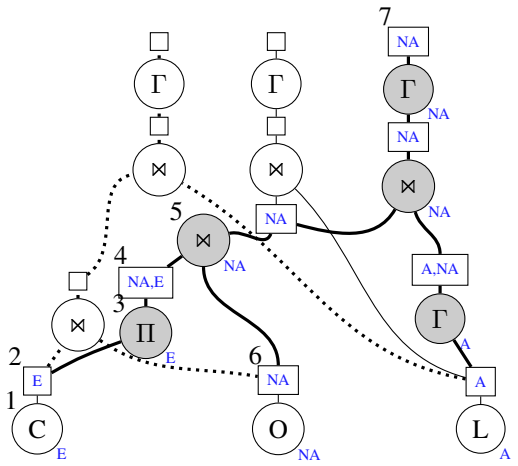
```
ship acctbal as aggregates sum, avg from Customer C to * group by mktseg, region
```

The above expression specifies how values of the acctbal column of the Customer table can be transferred outside. In particular, it specifies that (i) acctbal should be aggregated via the functions SUM or AVG and (ii) the cells of the acctbal column can be grouped by mktsegment and/or by nationkey. For example, output of the queries $\mathcal{G}_{\text{sum}(acctbal)}(C)$ and ${}_{region}\mathcal{G}_{\text{avg}(acctbal)}(C)$ can be transferred to all locations, whereas of $\mathcal{G}_{\text{sum}(acctbal)}(\sigma_{name='abc'}(C))$ and $\Pi_{acctbal}(C)$ cannot be transferred at all.

Compliance-based Optimizer. Now, in the query optimization phase, our optimizer aims to determine if a query is legal (i.e., its execution does not lead to violating data transfer rules) and to automatically generate an optimal compliant plan. We follow a two-phase optimization process that comprises plan annotation and site selection. The plan annotator receives a logical plan as input and outputs an annotated QEP. An *annotated QEP* is an optimized logical plan in which each plan operator is annotated with a set of compliant sites (i.e., sites where the execution of the operator will not violate any dataflow constraint). The site selector then uses dynamic programming to find the optimal placement of query plan operators taking data shipping cost into account.

[2]For exposition, we restrict to expressions over a single table. This is not a limitation: one can specify a policy expression over more than one base table. In this case, the condition list in the **where** clause of the expression must contain the join predicate.

More specifically, we adapt the Volcano optimizer generator [12] to generate the plan annotator. Our adaptations allow us to produce an annotated plan by enumerating the plan space by applying algebraic equivalence rules in a top-down fashion and filter compliant ones by applying our annotation rules in a bottom-up fashion. To do so, we treat geo-locations associated with base tables as "interesting properties" and propagate these properties bottom-up via annotation rules. Our annotation rules are based on the structure of the subplans and make use of a lightweight mechanism to evaluate data transfer rules. Our policy evaluator allows for easy integration of policy expressions into the annotation process. In particular, during plan enumeration, it determines to which cross-borders sites the output of operators can be shipped. The figure on the right illustrates the annotation process for our running example. Here the plan with the dotted lines shows the initial logical plan. The plan with thick solid lines shows the annotated plan which the annotator outputs. The letters in the square boxes denote sites to which the output of an operator can be shipped to and letters below each operator denote sites where each plan operator can be executed. For example, the project operator (node 3) must be executed in Europe but its output (node 4) can be shipped to North America and Europe. It is easy to see that the plan with thick solid lines translates to the compliant plan shown above. For a more detailed description of our optimizer, we refer readers to [6], which also gives a more formal treatment and proof of correctness.



3.3 Compliance Beyond the Relational Paradigm

Data masking via relational operations and our above query rewriting and optimization techniques have inherent limitations. They limit the whole gamut of compliant QEPs. As conditional restrictions on data transfers (as discussed in Section 2.2) may allow transferring of pseudonymized and/or anonymized data, it is important to expand the scope of masking functions to beyond relational operations.

Advanced Masking Functions. Based on European Commission's Opinions on Anonymization Techniques [21], we consider the following masking operations in addition[3].

Masking function	*Description*
Suppression	*Similar to Projecting out, replaces a value with a generic value (e.g., 'xxx')*
Pseudonymisation	*Replaces one value with another, s.t., new value has no logical relationship to the original (e.g., 'abc' to 'xyz')*
Blurring	*Alters a value by partial suppression (e.g., 'abc' → 'aXX')*
Generalization	*Generalizes a value using a predefined domain hierarchy (e.g., '23' → '20–25')*
Shuffling	*Replaces existing values with values from the same column*
Noise Addition	*Alter accuracy of (numeric) attributes*

Interleaving masking operators with query operators. A direct consequence of masking via non-relational operations is that: (1) it may no longer be possible to preserve query semantics. To illustrate this, consider that when transferring employee records, the age attribute is generalized by a function f that translates numeric attributes to certain range intervals (e.g., $f(23) = 20 - 25$). Such a masking may change the data type (e.g., `int`

[3]We note that this is not an exhaustive list of masking functions that we plan to support

to `int4range`) leading to change in query semantics. As another example, now consider that the age attribute is masked via suppression. In this case too, the resulting records will contain fewer attributes than that desired by the query. (2) Naive interleaving may affect robustness of the data masking. For example, consider a masking function f that blurs zipcode (e.g., $f(12345) = 123xx$). In this case a filter predicate p on zipcode (e.g., $p \equiv$ zipcode=12345) evaluated before masking may lead to possible singling out of an individual records. To this end, our current work explores the following research questions.

Q1 *How to interpose masking functions with query operators such that the resulting data is still anonymized?* For example, for scalar and univariate masking functions (e.g., blurring), we can substitute filter predicate by an UDF filter where the UDF is the masking function (e.g., we can rewrite p as zipcode=$f(12345)$.

Q2 *How to minimize information loss by "injecting" the right masking functions?* For example, by exploiting the fact that noise addition preserves aggregates (such as SUM() and AVG()), we can use masking via noise addition for aggregate queries instead of masking by blurring.

General Purpose Dataflow Programs. Many data analytics tasks are expressed as directed (a)cyclic graphs (DAG) composed of second order functions (e.g., map). To this end, we are investigating how advanced masking function can be composed with dataflow operators. For this, we plan to leverage our prior work on dataflow optimizations [13] and investigate effective and efficient ways to support (iterative) DAG programs.

4 Decentralized Query Execution

We now turn our attention to the execution of compliant QEPs. We first discuss why current FDPS fall short of executing compliant QEPs. We then present key challenges we need to tackle before presenting our approach.

State-of-the-art & Limitations. State-of-the-art FDPS (such as Presto [25]) mostly follow a mediator-based approach [16]. Although, such an approach (as illustrated by the figure on the right) is useful for performing data analytics across heterogeneous compute nodes, it does not lend itself to executing compliant QEPs. This is because *cross-database operations* (i.e., operators requiring inputs from multiple databases) are performed by the mediator's execution engine. This leads to the added complexity in ensuring compliance (due to the centralized processing by the mediator) to data transfers during query execution. To see this, recall the example of Section 3.2, and consider that now Orders data from North America can be transferred to Asia but nothing can move out from Asia. Under these constraints, a possible compliant plan is to first ship the data (after masking via projection operator as before) from Europe to North America, perform the first cross-database operation ($R \equiv C \bowtie O$) in Europe, and the second cross-database operation ($R \bowtie S$) in Asia. Current FDPS cannot execute such multi-site queries. Another limitation arises when metadata (or statistics) cannot be freely shared across sites, which leads to bad QEPs.

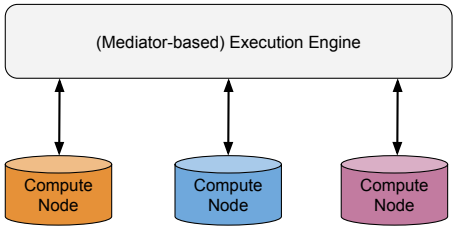


Figure 4: Centralized FDP

Challenges. In contrast to mediator-based approaches, we need a fully-decentralized approach (i.e., without any central entity in the execution pipeline) to execute queries over geo-distributed compute nodes. This, however, poses several challenges:

1. System Interoperability: A key aspect in achieving a fully-decentralized query execution is the ability to communicate (intermediate) data between underlying data processing systems. A key challenge, therefore, is to make systems interoperable without affecting their autonomy, even without them noticing.

2. Fast Data Transfer: As a consequence of lack of systems' interoperability, moving data among different data processing systems incur a high cost (e.g., we might need to export data from one DBMS and import that data in another DBMS). The challenge resides in enabling "native" data transfers (e.g., reading a relation from a DBMS directly from its binary store) without affecting the autonomy of the underlying processing systems.

3. Incompatible Data Formats: Different systems work on different internal data formats, which adds additional complexity and non-trivial cost of storing and converting data from one format to another during data transfers. The main question to answer is: does an intermediate data representation exists that can speed up the conversion from any source data format to any target data format?

4. Query Optimization: Last but not the least, optimizing queries without having a "centralized" access to data systems makes query optimization across geo-distributed data processing systems non-trivial. For example, data processing systems have their own cost models, which impedes in determining global cost of executing queries.

Our Approach. We have developed a meta-execution engine, which enables executing federated queries in a decentralized fashion. Figure 5 illustrates our overall approach. In contrast to a mediator-based approach, we follow what we refer to as a delegator-based approach. The Meta-execution engine, delegates the entire query execution to underlying data processing systems. This avoids a need to have any central entity in the execution pipeline, and data transfer only happens among compute nodes following annotations as prescribed by the compliance-aware query optimizer.

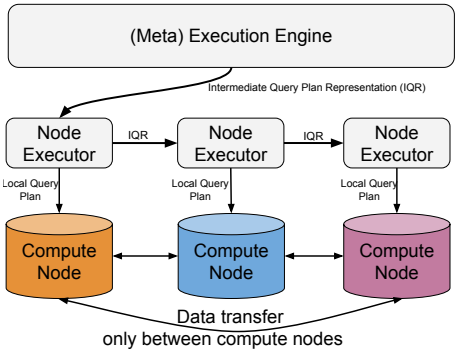


Figure 5: Decentralized FDP

In more detail, the meta execution engine first globally optimizes a query (with limited available metadata) and translates the plan into an intermediate representation (IQR). The IQR is agnostic to underlying data processing platforms and encapsulates query semantics comprising local subplans (i.e., processing that should be limited to certain locations) and inter-operator (cross-database) communication endpoints as well as mechanics of data movement between different systems. The node executors are local meta execution engines that are responsible for (a) translating and optimizing a local-subplan into platform specific query plan (e.g., to a Spark program), (b) communicating the IQR to other (relevant) node executors, and (c) facilitating the data movement between systems either by leveraging underlying system's capabilities (see below) or by creating suitable data "pipes". It is important to note that the node executors themselves do not execute any part of the query but only delegate execution to underlying systems.

In our current prototype [5], we support such decentralized query execution over multiple RDBMSes (including PostgreSql, MariaDB, MySQL, Hive, and DB2). To do so, the local node executors leverage the SQL/MED standard to communicate intermediate query data between systems. More specifically, while translating into a local DBMS specific program, the node executor first registers external tables (i.e., tables corresponding to the output of remote subplans) as local tables. This enables achieving a completely decentralized query execution.

Current Research Directions. We are currently investigating how to process data using geo-distributed heterogeneous compute platforms. For example, how can we execute a multi-site query using a PostgreSql database at one site and a Spark Cluster at another site? This is crucial to support compliance for non-relational workloads. To this end, our work includes expanding capabilities of cross-platform data processing systems (such as Apache Wayang [3]) to support geo-distributed compute sites. We are also investigating suitable common data formats such as Parquet, Protocol Buffers, or Avro, which makes inter-system data transfers efficient.

5 Related Work

We now relate the ideas presented in this paper to prior work on compliance and federated data processing.

Compliance. With growing concerns over data privacy and enforcement of regulations, several works have looked into various legal contexts within with data processing systems must be designed. With respect to GDPR compliance, most works have focused on data subjects' rights as a large proportion of GDPR governs data storage. In this regard, [20] analyzed various aspects of GDPR including deletion, indexing, monitoring and logging, and access control, and how they impact database systems. [11, 29] studied the performance of GDPR compliant systems. Their work mainly considers rights of data subjects (e.g., customers) and provide a benchmark to evaluate data processing aspects including metadata indexing, deletion, access control, and encryption. [19, 18] proposed an architectural vision for a database that natively supports auditing, deletion, and user consent management. [28] and [27] examine how GDPR affects the design and operation of modern computing systems. [24] and [34] studied supporting restrictions based on "purpose"-based access control. [26] presents an analysis on the impact of GDPR on storage systems. [15] presents a vision for Software-Defined Data Protection, for which they propose leveraging recent advances in Software-Defined Storage (e.g., FPGA-based key-value stores) to achieve compliance at the storage level. In the context of dataflow processing, [30] investigated supporting of data subject's privacy request (for access, deletion, and objection) by adopting causal snapshot consistency. [4] focuses on auditing GDPR compliance based on logs. All the above work can be seen as complementary to our work. While the aforementioned works focus more on data subject's rights, we focus more on the actual processing of the data wherein we consider regulations pertaining to the movement of data. Perhaps, closest to our work on compliance-aware query optimization is that of [23] and [9, 8]. While these works only focus on the operator placement problem based on privacy or user-specified constraints, we additionally consider rewriting queries by extending and interleaving query operators with data masking functions.

Federated Data Processing. Our work is also related to earlier works on multi-database query processing [17] and more recent work on cross-platform systems [3] and polystores [7]. It differs from the former in that we focus on heterogeneous compute infrastructures and from the latter in that we target geo-distributed environments and a facilitate a decentralized query execution.

6 Conclusion

Growing concerns on data privacy and usage preclude data transfers across national (or organizational) borders. It is therefore crucial that data processing in federated environments be compliant with data transfer regulations. In this paper, we have analyzed data transfer regulations from the perspective of GDPR and discussed key research challenges for including compliance aspects in federated data processing. We presented our approach for compliant geo-distributed data processing that focuses on relational workloads. We also discussed how query rewriting and optimization techniques must be extended to support data masking and outlined open problems and research directions to support workloads beyond relational workloads. We have advocated the need for decentrally executing queries and presented challenges and research directions to support data analytics over geo-distributed and heterogeneous compute infrastructures.

Acknowledgements This work is funded by the German Ministry for Education & Research as BIFOLD – "Berlin Institute for the Foundations of Learning and Data" (01IS18025A and 01IS18037A).

References

[1] Regulation (eu) 2016/679 of the European parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. 59:1–88, 2016.

[2] California consumer privacy act. California civil code, section 1798.100, June 28. 2018.

[3] Divy Agrawal, Sanjay Chawla, Bertty Contreras-Rojas, Ahmed Elmagarmid, Yasser Idris, Zoi Kaoudi, Sebastian Kruse, Ji Lucas, Essam Mansour, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, Saravanan Thirumuruganathan, and Anis Troudi. Rheem: Enabling cross-platform data processing: May the big data be with you! *Proc. VLDB Endow.*, 11 (11): 1414–1427, 2018.

[4] Debois S. Arfelt E., Basin D. Monitoring the GDPR. In *ESORICS*, pages 681–699, 2019.

[5] Kaustubh Beedkar, David Brekardin, Jorge-Anulfo Quiané-Ruiz, and Volker Markl. Compliant geo-distributed data processing in action. *Proc. VLDB Endow.*, 14 (12): 2843–2846, 2021.

[6] Kaustubh Beedkar, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. Compliant geo-distributed query processing. In *ACM SIGMOD*, page 181–193, 2021.

[7] Jennie Duggan, Aaron J. Elmore, Michael Stonebraker, Magda Balazinska, Bill Howe, Jeremy Kepner, Sam Madden, David Maier, Tim Mattson, and Stan Zdonik. The bigdawg polystore system. *SIGMOD Rec.*, 44 (2): 11–16, 2015.

[8] Nicholas L. Farnan, Adam J. Lee, Panos K. Chrysanthis, and Ting Yu. PAQO: A preference-aware query optimizer for postgresql. *Proc. VLDB Endow.*, 6 (12): 1334–1337, 2013.

[9] Nicholas L. Farnan, Adam J. Lee, Panos K. Chrysanthis, and Ting Yu. PAQO: Preference-aware query optimization for decentralized database systems. pages 424–435, 2014.

[10] GDPR Enforcement Tracker. `https://www.enforcementtracker.com/`.

[11] GDPRbench. `https://www.gdprbench.org/gdpr`.

[12] Goetz Graefe and William J. McKenna. The volcano optimizer generator: Extensibility and efficient search. In *ICDE*, pages 209–218, 1993.

[13] Fabian Hueske, Mathias Peters, Matthias J. Sax, Astrid Rheinländer, Rico Bergmann, Aljoscha Krettek, and Kostas Tzoumas. Opening the black boxes in data flow optimization. *Proc. VLDB Endow.*, 5 (11): 1256–1267, 2012.

[14] Chien-Chun Hung, Leana Golubchik, and Minlan Yu. Scheduling jobs across geo-distributed datacenters. In *SoCC*, page 111–124, 2015.

[15] Zsolt István, Soujanya Ponnapalli, and Vijay Chidambaram. Software-defined data protection: Low overhead policy compliance at the storage layer is within reach! *Proc. VLDB Endow.*, 14 (7): 1167–1174, 2021.

[16] Vanja Josifovski, Peter Schwarz, Laura Haas, and Eileen Lin. Garlic: A new flavor of federated query processing for db2. In *SIGMOD*, pages 524–532, 2002.

[17] Donald Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32 (4): 422–469, 2000.

[18] Tim Kraska, Michael Stonebraker, Michael Brodie, Sacha Servan-Schreiber, and Daniel Weitzner. *SchengenDB: A Data Protection Database Proposal*, pages 24–38. 2019.

[19] Tim Kraska, Michael Stonebraker, Michael Brodie, Sacha Servan-Schreiber, and Daniel J. Weitzner. Datumdb: A data protection database proposal. In *Poly'19 co-located at VLDB 2019*, 2019.

[20] Jayashree Mohan, Melissa Wasserman, and Vijay Chidambaram. Analyzing gdpr compliance through the lens of privacy policy. In Vijay Gadepally, Timothy Mattson, Michael Stonebraker, Fusheng Wang, Gang Luo, Yanhui Laing, and Alevtina Dubovitskaya, editors, *Poly'19 co-located at VLDB 2019*, 2019.

[21] Opinion 05/2014 on Anonymisation Techniques. `https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf`.

[22] Qifan Pu, Ganesh Ananthanarayanan, Peter Bodik, Srikanth Kandula, Aditya Akella, Paramvir Bahl, and Ion Stoica. Low latency geo-distributed data analytics. In *SIGCOMM*, pages 421–434, 2015.

[23] Guido Salvaneschi, Mirko Köhler, Daniel Sokolowski, Philipp Haller, Sebastian Erdweg, and Mira Mezini. Language-integrated privacy-aware distributed queries. *Proc. ACM Program. Lang.*, 3 (OOPSLA), 2019.

[24] Malte Schwarzkopf, Eddie Kohler, M. Frans Kaashoek, and Robert Morris. Position: GDPR compliance by construction. In *Poly'19 co-located at VLDB 2019*, 2019.

[25] Raghav Sethi, Martin Traverso, Dain Sundstrom, David Phillips, Wenlei Xie, Yutian Sun, Nezih Yegitbasi, Haozhun Jin, Eric Hwang, Nileema Shingte, and Christopher Berner. Presto: Sql on everything. In *ICDE*, pages 1802–1813, 2019.

[26] Aashaka Shah, Vinay Banakar, Supreeth Shastri, Melissa Wasserman, and Vijay Chidambaram. Analyzing the impact of GDPR on storage systems. In *USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 19)*, 2019.

[27] Supreeth Shastri, Melissa Wasserman, and Vijay Chidambaram. The seven sins of Personal-Data processing systems under GDPR. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*, 2019.

[28] Supreeth Shastri, Melissa Wasserman, and Vijay Chidambaram. Gdpr anti-patterns: How design and operation of modern cloud-scale systems conflict with gdpr, 2019.

[29] Supreeth Shastri, Vinay Banakar, Melissa Wasserman, Arun Kumar, and Vijay Chidambaram. Understanding and benchmarking the impact of gdpr on database systems. *Proc. VLDB Endow.*, 13 (7): 1064–1077, 2020.

[30] Jonas Spenger, Paris Carbone, and Philipp Haller. Wip: Pods: Privacy compliant scalable decentralized data services. In *Poly 2021 and DMAH 2021*, page 70–82, 2021.

[31] Rebecca Taft, Irfan Sharif, Andrei Matei, Nathan VanBenschoten, Jordan Lewis, Tobias Grieger, Kai Niemi, Andy Woods, Anne Birzin, Raphael Poss, Paul Bardea, Amruta Ranade, Ben Darnell, Bram Gruneir, Justin Jaffray, Lucy Zhang, and Peter Mattis. Cockroachdb: The resilient geo-distributed sql database. In *ACM SIGMOD*, page 1493–1509, 2020.

[32] Jonas Traub, Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. Agora: Bringing together datasets, algorithms, models and more in a unified ecosystem [vision]. *SIGMOD Rec.*, 49 (4): 6–11, 2021.

[33] Ashish Vulimiri, Carlo Curino, Brighten Godfrey, Konstantinos Karanasos, and George Varghese. Wanalytics: Analytics for a geo-distributed data-intensive world. In *CIDR*, 2015.

[34] Lun Wang, Joseph P. Near, Neel Somani, Peng Gao, Andrew Low, David Dao, and Dawn Song. Data capsule: A new paradigm for automatic compliance with data privacy regulations. *CoRR*, abs/1909.00077, 2019.

Towards Privacy by Design for Data with STRM privacy

Bart van Deenen
strmprivacy.io

Pim Nauts
strmprivacy.io

Robin Trietsch
strmprivacy.io

Bart Voorn
strmprivacy.io &
University of Amsterdam

Abstract

Both societal and regulatory pressure (GDPR) increasingly challenge organizations and engineering teams to balance privacy and innovation. Striking this balance can be costly in terms of effort, data utility and computation costs. Moreover, current approaches in scalable data systems often treat privacy as an access problem, which is at odds with important legal and design principles. A plethora of privacy preserving- and enhancing- technologies are available, yet their adoption in production data systems still faces challenges. In particular, their focus is often on narrow use cases such as external data sharing, on mostly existing data sets, rendering them unusable in real-time data architectures. In this paper we argue engineering teams should "shift left" with their data privacy efforts, to the point of data collection. We show how privacy challenges in production architectures can be addressed without compromising speed, data quality or privacy. We provide a detailed yet practical explanation of an architectural set-up that allows users to launch privacy streams in seconds.

1 Introduction

If the industrial revolution ran on fossil fuel, modern-day organizations run on data. From governments to hospitals, manufacturing to digital marketplaces: data systems are everywhere and have quickly become a core asset and capability to deploy for any modern organization in order to innovate. With"data" driving both considerable value creation and grounded concern about the impact to private individuals, legislators developed and implemented new regulations like the General Data Protection Regulation (GDPR) [26] in the European Union to foster innovation without foregoing individual rights to privacy [7] .

But even in *data-native* organizations, precisely this "balancing" of privacy and data driven innovation is not an easy task for data engineers, privacy officers and data governance experts alike. While the first foundational work on *Privacy by Design* from Ann Cavoukian dates back to the 90s [9], building modern, scalable data systems around privacy principles is still a nascent field due to a lack of (technical) standards and a prolonging gap between the policy perspective and operational reality of data systems [1]. This gap leads to a very real "Cost of Privacy" to organizations beyond just legal costs [5]. Structuring privacy operations in organizations, at scale, requires time and effort. Increased coordination across departments, unclarity around requirements, longer development cycles, additional staffing for e.g. RTBF requests [16] and "privacy paralysis"[1] all mean organizations incur additional- and opportunity costs. For example, a structured approach to privacy is often

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] the observation that taking a next step with application development is delayed or even refrained from due to privacy concerns

lacking, and the GDPR requirements around data collection and user opt-ins lead to reverse-engineering the legal ground and related purpose for collecting and processing data, with engineering teams often pulled into lengthy legal discussions on purpose. Moreover, requirements like strong anonymization pose a practical challenge to data utility and -usability for (data) engineering teams.

Despite this apparent *Cost of Privacy*, available solutions are still limited. We see the gap between the policy perspective and real-world systems is most apparent in the way privacy is embedded in many data architectures. Often, privacy is treated as an access problem on data that is already collected, stored and used[2], not as a first principle of systems design (often without traceability to the collection ground and collection purpose). While this is convenient as it requires limited re-architecting of existing systems, we believe this is an "early efficiency win [that causes] late stage headaches" [5]. Moreover, it is at odds with (if not in violation of) legislative directions. This often leads to (i) organizations accepting an increased risk profile or (ii) refraining from engaging in new use cases (and so innovation) altogether. All in, the current approach to dealing with privacy often poses another real cost and risk to organizations, both financial and in reputation.

1.1 Privacy Enhancing Technologies

One of the domains where these challenges are being (partly) addressed, is in Privacy Enhancing or Privacy Preserving Technologies (PET's) [21]. PET's allow data practitioners to transform sensitive data into usable input for consumption or support detecting sensitive data [5], limiting risk of unsolicited data consumption. Yet, they have important shortcomings in the practical sense: they often center around external data sharing as the dominant use case, are computationally heavy, best suited to aggregate data operations only and/or rely on "complete" data sets. This makes them better suited to use cases where data is already collected and stored. Hence, PET's provide a solution for a limited set of use cases and can still lead to non-compliance with privacy regulations if data was not properly collected or bounded to data purpose [5].

1.2 Towards Privacy by Design for data

To bridge the gap from the realities of privacy professionals and data (system) engineers to available technologies, we propose an architecture to encode privacy in data that helps to structure privacy efforts, and a system to implement this at lower real-world costs. Both the architecture and system come from a set of clear principles, derived from legal and privacy by design requirements:

1.2.1 Requirements

- ***Explicit contracting*** - define both data shape (the schema) and privacy implications as metadata (including key legislative concepts such as collection ground) in data contracts that describe each datapoint.
- ***Enforce transformation*** - define inside the contracts, set per-field privacy transformations, such as de-identification or anonymization, for sensitive (PII[3]) data fields, so that data collection and processing are tightly coupled.
- ***Shift left*** - Enforce these data contracts at ingestion, so data is (i) always bound to purpose and (ii) welded to the original collection ground.
- ***Retain utility*** - transform data in such a way that the trade-off between privacy and utility is limited or even removed.

[2] e.g. the way many modern data warehouses deal with privacy is through role-based access

[3] Personally Identifiable Information, such as a name or email

- ***Dedicated distribution*** - provide a dedicated interface for every specific collection ground through privacy streams, so that applications and/or downstream users are guaranteed to only receive (i) necessary and (ii) properly transformed data.
- ***Interoperable*** - the system should naturally complement existing architectures and accommodate easy integration and deployment (aligned with industry standards).
- ***Cheap and Fast*** - extend interoperability by aiming to add minimal additional latency to existing systems and limit computational cost of operations.

In effect, a solution should allow data teams to build with confidence, cut costly coordination time, and enable them to quickly deploy a contemporary (streaming) data infrastructure with a focus on high quality data at low effort at lower legal and reputation risks.

2 Limiting the Cost of Privacy: privacy, utility and design challenges

In this section we identify the challenges that surround building scalable data systems from privacy forward, and the design principles that should shape them. We will approach these as drafting the list of requirements for building these *Privacy by Design* data systems.

2.1 Privacy challenges

2.1.1 Key requirements from GDPR

While often viewed as a complex legislation, the core of what GDPR seeks to achieve is straight forward: protection of individuals' private data. A few key concepts set the boundaries for responsible (and lawful) data behavior, such as accountability (you are accountable for the data you control), and data subject rights (what an individual can require from a controller). We will use these as a treasure map in hunting for requirements.

Data minimization and consent Of key interest to our challenge are the governing principles on how to collect and use data: data minimization and consent obligations. Simply put, they state one should not collect more data than necessary for a purpose, and can only collect data under *either* explicit consent for that purpose or a justified cause[4], like legitimate interest or contract fulfillment (e.g. an e-commerce marketplace simply needs your home address to deliver your order).

In practice, these legal grounds are oftentimes liberally interpreted. Historical data – i.e. data that has been collected in the past and stored in e.g. a data warehouse – is often persisted without storing the purpose under which it was acquired. This is clearly problematic in the privacy sense: it is (nearly) impossible to determine the purpose of a data point after the data collection took place.

It becomes even more interesting once we acknowledge the dynamic nature of consent. For example, how do we handle changing preferences of end-users, and how is consent linked to data over time? By collecting purpose up-front, and ensuring that this purpose stays with the data, two problems are handled instantly: purpose limitation and consent [24]. This yields the first challenge we should take into account:

Req. 1: ***We should encode how a data point was obtained and for what purpose***

Purpose limitation As data in practice is often collected once and then up- and recycled often, there is another key principle to take into account: purpose limitation. In the official text, GDPR article 5 section 1(b) [26], we see the following:

[4]Consent, purpose and legal ground are used interchangeably

> *Personal data shall be collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with those purposes; further processing for archiving purposes in the public interest, scientific or historical research purposes or statistical purposes shall, in accordance with Article 89(1), not be considered to be incompatible with the initial purposes ('purpose limitation').*

This is directly at odds with the common practice of hoarding data into a warehouse. If the purpose of data collection is not clear, the legal ground cannot be determined and consent cannot legally be obtained [24]. While the legalese and exact wording of what purposes can be considered legal is far outside of the scope of many data engineering professionals, this is a clear area of interest for designing systems around privacy: data usage without determining if it is legal to use for a certain purpose is already violating this very principle. In practice we see this judgement is often made by a data consumer team without (sufficient) knowledge of Personal Data regulations. This paves the way to our second requirement:

Req. 2: ***Data consumers should only receive data they are entitled to, given the consent of the data-owner, for the specific purpose of the data consumer*** [5]

So e.g. while a data-team building a recommender system has a vested interest in knowing what a user does, they do not need to know who the user is - and so could do with any random value representing the user as long as its consistent over time for that specific person or user account (provide consent or legal ground is present!).

2.1.2 Privacy by Design, the principles

Our next source of system requirements are the original Privacy by Design principles laid out by Ann Cavoukian in her capacity as the Information and Privacy Commissioner of Ontario [9].

The first principles[6] revolve around timing - when privacy comes into play in both designing systems and the flow of data. Clearly, it is only private by *design* if "characterized by proactive rather than reactive measures, [anticipating and preventing] privacy invasive events before they happen". Moreover, the "maximum degree of privacy" is in warranting the protection of personal data in "any given" data system (aligning with our first requirement: encode privacy into data). This suggests a profound shift of the point at which privacy becomes part of the data flow:

Req. 3: ***Privacy should be embedded right where data is collected.***

More recent views on best practices like Bhajaria's (2022) [5] also suggest that both classifying and tagging data before it hits data pipelines (i.e. at ingest) mitigates privacy risks more optimally.

2.2 Data utility

2.2.1 Bringing structure to data improves data quality

Even before privacy comes into play, many data practitioners will recognize the challenge of data quality in and of itself. In organizations with lots of operational legacy systems (which does not imply they are "old"!), it is common to see the creative re-use of fields/columns in a data set. Dealing with confusing field names, undocumented changes in lineage and empty fields are common practice in the real world [4]. Moreover, data consumers generally have *no direct control* on what is accepted as valid data, and will often have to resort to heuristics to determine what to do with a certain data field.

[5] processor in GDPR parlance

[6] 1. Proactive not Reactive; Preventative not Remedial, 2. Privacy as the Default Setting, 3. Privacy Embedded into Design

Think of this as *data debt* or negative interest building inside data flows. Adding and enforcing more structure in data will improve data utility over the long run considerably as it addresses this kind of real-world challenges (see for an overview of data quality frameworks [10]. In any case, we should strive to:

Req. 4: ***Make structure explicit and retain as much of the data quality and utility as privacy permits.***

2.2.2 The opportunity of privacy transformations for data utility

One of the key challenges of applying privacy regulations and principles to data systems lies in the (supposed) limitations they impose on data utility. Any personal data collected and used is subject to the GDPR provisions, limiting collection ground and purpose - e.g. you cannot simply use an acquired e-mail for marketing purposes, as used to be common practice. This creates another practical cost, as different data types need to be shielded from general use inside systems.

There are however circumstances under which collected data may be used for different purposes than originally requested: in anonymized form, with "data rendered anonymous in such a way that the data subject is not or no longer identifiable." (Recital 26 of [26]). When done properly, and this is an extremely high standard many organizations fail to achieve (Article 29 Working party, in [14]), this even places personal data outside of GDPR, hence allowing to retain utility. This is especially attractive for pattern-driven tasks such as aggregated analytics and applied machine learning that do not rely on identification of the data subject persé.

Along that axis is a much lighter interpretation of what it means to de-identify data: pseudonymization, "the processing of personal data in such a way that the data can no longer be attributed to a specific data subject without the use of additional information." (Recital 4(5) of [26]). Hence, by separating the necessary link between a data point (such as a click event) and the data subject, GDPR allows data to be used with much more degrees of freedom (even beyond the original purpose limitation!) [14]. In summary, applying lighter or heavier de-identification methods helps to retain the data utility many believe is lost under GDPR.

Req. 5: ***Immediately transform data in such a way that the trade-off between privacy and utility is limited or even removed***

2.3 Technical design challenges

Building and maintaining scalable data systems requires deep knowledge of a wide set of technologies and domains. On top of that, adding privacy does not simplify these already complex designs and systems. A great example of the complexity GDPR brings to systems design is fulfilling *Right To Be Forgotten* (RTBF) requests:

Under RTBF, a data subject can request a data controller to deliver or remove their (personal) data. This has the potential to be a daunting and expensive task, as legally obtained personal data can be *everywhere* in or even outside an organization, from local copies to data shared externally with vendors. In order to completely fulfill an RTBF request, one has to conduct either a full table scan on every day of data for each request in any location, or build and update an index of the users present in every data file or storage, including derivatives of personal data. Both are expensive operations, even when automated. We would have to repeat this at least every 30 days (minus operation time), as that is the response window GDPR permits. RTBF is just one of many challenges (and opportunities) GDPR imposes. Various challenges on Integration, Architecture, and Performance exist.

2.3.1 Architecture

One of the first challenges to discuss is in the (dominant) architecture underlying many data systems: batch processing, where (often millions) of rows of data are created and processed at set intervals.

But if data "exists" already, we run into a clear challenge for adhering to the aforementioned principles or to be able to *shift [privacy] left* to the point of data collection. The presence of personal data in raw form needs to be prevented up-front. A sensible way to achieve this is by shifting data collection to an event-based approach. If we capture and treat each datapoint individually, e.g. through following the exchanging API's design, we can enrich and add metadata at much higher resolution. A stream processing-based[7] event gateway allows for coupling specific metadata to each and every single data point (the event) and so allows to enforce "embedding" privacy.

Although a good starting point for privacy by design systems, creating (and maintaining) an end-to-end streaming architecture with data enrichment centered around strict requirements is technically challenging. Technical difficulties and limited maturity to fully benefit from real-time architectures in data consumption has caused many companies to avoid or even abandon them. Executing transformation on a data stream is technically more demanding in both development and reliability as compared to batch processing, and standards (both technical and implementation) are not as widespread. When applied to business challenges in the right way, processing and enrichment of streaming data can create highly valuable real-time insights, for example into customer behavior (see for an example [12] and the remainder of the Dec 2015 Bulletin) and real-time machine learning use cases [27].

In order to conduct the type of transformations necessary for privacy processing, popular tools such as Apache Kafka, Apache Calcit e(streaming SQL), Kafka Streams or Apache Flink can be leveraged. While powerful, in our experience they are also characterized by a steep learning curve, and are non-trivial to bring into production (in both platform maintenance and usage), especially at scale. Other popular components and solutions (self-hosted or managed) generally provide the necessary strict event formats and support for stream processing, but are agnostic to use cases or specific application domains and therefore require a lot of engineering on top of the bare metal solutions. For instance, retaining data purpose like data subject consent is still left to the data consumers.

2.3.2 Integration

Another key challenge when building for privacy lies in the integration to existing data technology. Enterprises and SMEs are often (deeply) invested in existing data architectures, and are likely unwilling or not capable of switching technologies very easily. As mentioned, when designing for privacy, the goal is to prevent having to deal with privacy *after* data collection, preferably agnostic to the configuration and architecture of existing data systems [5] that handle data downstream.

To many engineers, even when leveraging cloud or OSS[8] building blocks, designing a platform for data processing from privacy forward is a daunting prospect, precisely because the aforementioned gap between the legal and policy perspective persists. However, drawing upon our experience, it is possible to achieve Privacy by Design and implement its principles by framing it as an augmentation instead of a replacement challenge . In fact, many basic building blocks are provided by various technologies that have proven themselves over the past years (e.g. like Apache Kafka and processing engines). We therefore argue that evaluating how to augment and evolve an existing stack is likely a more effective strategy than re-architecting everything for privacy.

Moreover, an event-based approach provides the opportunity to simply aggregate and transform data to match existing downstream (batch) processes at low cost, while retaining the metadata necessary for encoded privacy. This limits scope of implementing such systems: it would simply precede existing processes instead of replacing them and is valuable even when there's no immediate need or value to obtain and consume data at (near) real-time latency.

[7] Often referred to as streaming or real-time data

[8] Open Source Software

2.3.3 Performance

The last key challenge to point out is in system performance. While end-to-end stream enrichment with meaningful data can bring value and suits the privacy challenges well, these opportunities can be offset by decreased systems performance (e.g. increases in system latencies can cause decreased conversion in online retail [2]. A lot of the value of enrichment occurs if the result of the enrichment is available as soon as possible. Hence, latency is critical to take into account when preventing raw form PII data at the moment of collection.

A major factor in the performance of such systems is how they respond to increased scale. Bottlenecks are highlighted quickly when data processing systems are put under high load. All sorts of challenges should be taken into account when designing for scalability of data processing systems: distribution of the data, average size, different types of events, and many other factors. Orchestration platforms, such as Kubernetes, allow for fine-tuning in order to deal with the requirements of scalability and flexibility.

Systems can scale in two ways: vertical (more resources) and horizontal (more instances). Designing systems for horizontal scalability is more complex, as applications should be designed stateless and resilient to restarts, but often cost effective as dynamically scaling instances keeps idle resources to a minimum. All these factors (architectural, integration and performance challenges) add up to our final requirement:

Req. 6: ***Complement existing data systems, focus on scalability and aim to add as little latency and computational costs.***

3 From Requirements to Solution: STRM Privacy

The aforementioned requirements are the boundaries for a Data System designed for Privacy. The core premise of privacy regulations like GDPR is that every data item should have a legal ground to gather and process it. The only way to guarantee proper processing of the data item is to tie the owner and permissions based on legal grounds directly and irrevocably into the data item. Our solution adds a standard set of attributes to *every data item* to achieve this.

Structure is imposed before any data is transferred through a data contract[9] defining the data shape, the privacy implications and the data validations. As can be seen in figure 1, a central Event Gateway receives data from many different applications, where a data contract is enforced, and *Personal Data Attributes* are encrypted[10]. Basic data validation is performed upon ingest, resulting in an encrypted data stream (where only the PII fields in data items are encrypted). It should only be possible to decrypt these attributes provided the data collection purpose and *specific usage or consumption purpose* match. By encrypting on ingest, and only putting encrypted *privacy safe* data items in long-term storage, the long-term storage does not become data *toxic waste*.

While taking purpose into account, a decrypter step then creates derived privacy streams, including only events with the required consent for that stream as decrypted. If a user did not provide consent for their data being used for a specific purpose, the data does not end up as private data in these streams.

We apply encryption methods beyond just obfuscating underlying field values. Through key rotations on links inside the data items, privacy transformations such as masking or anonymization can be performed in real-time. Combined with the purpose and consent binding, we can split the incoming data and create purpose-driven data interfaces that downstream applications or teams can consume without privacy concerns.

The encryption mechanism will assign the same encryption key to the same value of a specific data item field. As the same value for a field in the data item yields the same cipher-text value, the integrity between events is retained. With a time-based encryption key rotation (i.e. the privacy algorithm) we can consider the resulting encrypted data stream to be *not privacy sensitive anymore*. Next to that, in case a data owner requests to *be*

[9] for more information on data contracts: `https://docs.strmprivacy.io/docs/latest/concepts/schemas-and-contracts/`

[10] through a symmetric encryption algorithm, with a fixed initialization vector

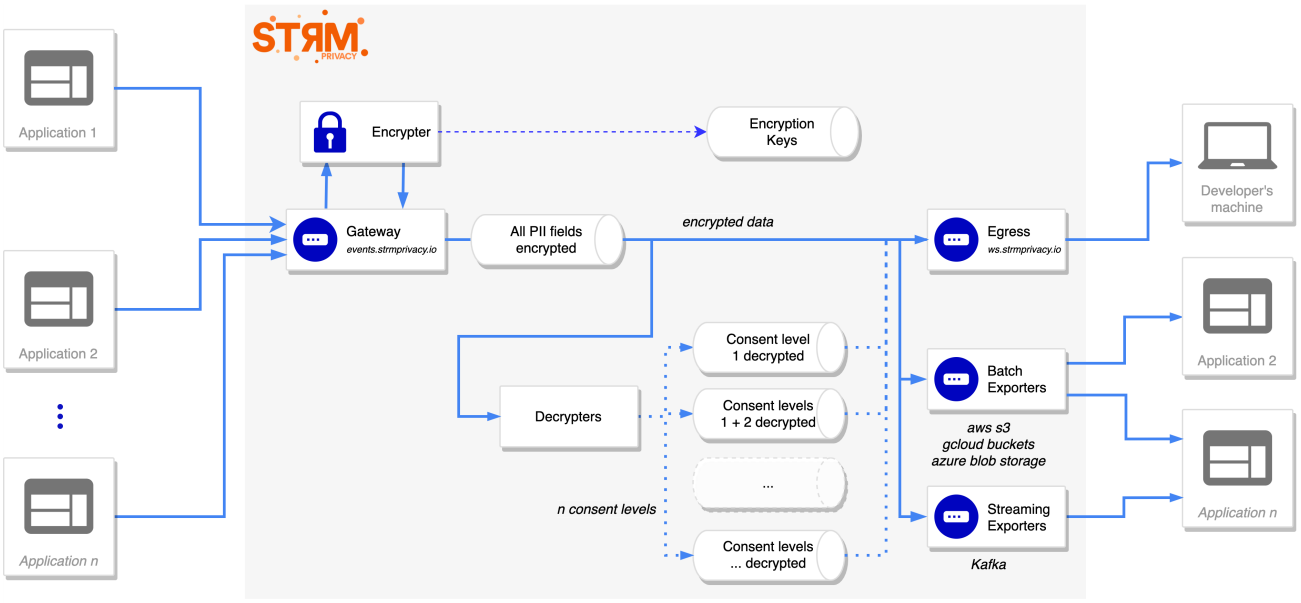


Figure 1: STRM Privacy High Level Architecture

forgotten, throwing away the encryption key destroys all personality dimensions inside the data, but helps to retain the utility of the non-sensitive fields. This process is also known as crypto-shredding [11].

In sum, our architecture allows to encode privacy into a data point from the moment of collection onward, enforcing that personal data is only available when the purpose allows it, while retaining important utility of the data. Our documentation [25] provides a deep dive into all the various components that drive this architecture.

4 Related Work

At this point, we positioned the need and guiding principles for building real-world data systems from privacy forward. We have described how the architecture of STRM Privacy follows these principles through an *event-based* component, a *privacy* component, and a *data quality* component.

Evidently, specific technologies exist for the various components such an architecture requires. First, for data processing in streaming fashion, there are various established technologies, such as Apache Kafka, Kafka Streams, Apache Beam, and many other Open Source and commercial solutions (see e.g. [15] or [19]).

Secondly, we emphasize that specific solutions in the realm of Privacy Enhancing Technologies are on the rise that warrant privacy through different approaches (see for an overview [21]). Their goal generally is to destroy identification in data, while maintaining the same (statistical) characteristics to sufficiently represent the original data to conduct operations (see for example earlier IEEE publications [17] or [28]). For example, solutions include Synthetic Data Generation, obfuscating data on existing data sets, and data encryption at rest. Some challenges remain, e.g. with obfuscation, one should ensure that the obfuscated data is not susceptible to linkage attacks combining obfuscated data with other data to de-anonymize the data [20]. An important limitation of these approaches is that they are currently not able to accommodate real-time transformations and embedding in operational data systems [3], and often require existing data sets that might be collected in violation.

Finally, *data quality* is considered to be of pivotal importance for any downstream application [8],[6]. Hence, both traditional ETL tooling and novel ML supported techniques (see for example [13]) enable both validating and enriching data at ingest or even in transit. Metadata management is an important adjacent field, which could further enrich *data usability* in data pipelines [22], although scalability of such systems is still challenging [23].

5 Conclusion

In this paper we argued that building data systems for privacy in real-world production settings should be guided by a core set of principles derived from legislative and *Privacy by Design* frameworks. This enables closing an existing gap between the policy perspective of privacy and the operational reality of data systems.

Existing *Privacy Enhancing Technologies* focus mostly on heavy computational tasks, such as synthetics, at the cost of latency and compute power. They are generally best suited to existing data sets, not in-flight data. Critically, data teams should not treat privacy as an access problem and shift their data privacy efforts to the point of data collection (i.e. "shift left") to better position for compliance. When employing data contracts to encode privacy and tight coupling to privacy transformations, scalable data systems that retain performance and data quality are achievable. They can be compatible with progressive data system philosophies (i.e. the Modern Data Stack or Data Mesh [18]). In effect, this lowers the *Cost of Privacy* through better data usability, reduced compliance risks and improved coordination between engineering teams and legal departments.

To the best of our knowledge, no other solution exists that focuses on encoding privacy inside data, combined with proven technologies for streaming infrastructure to minimize impact in latency and performance to complement existing data systems. Apart from a better modus operandi for data systems in general, such systems can drive valuable innovation like real-time machine learning while respecting end-user privacy (and thus privacy regulations), and bring privacy by design for data to a much wider audience. We happily invite the community to further extend the collective body of knowledge, by providing feedback or iterations to our work.

References

[1] Serge Abiteboul and Julia Stoyanovich. Transparency, fairness, data protection, neutrality: Data management challenges in the face of new regulation. *Journal of Data and Information Quality (JDIQ)*, 11(3):1–9, 2019.

[2] Akamai. Akamai Online Retail Performance Report. `https://www.akamai.com/newsroom/press-release/akamai-releases-spring-2017-state-of-online-retail-performance-report`, 2017.

[3] Jason W Anderson, Ken E Kennedy, Linh B Ngo, Andre Luckow, and Amy W Apon. Synthetic data generation for the internet of things. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 171–176. IEEE, 2014.

[4] Mahmoud Barhamgi and Elisa Bertino. Special issue on data transparency—data quality, annotation, and provenance. *Journal of Data and Information Quality (JDIQ)*, 14(1):1–3, 2022.

[5] N. Bhajaria. *Data Privacy: A Runbook for Engineers*. Manning, 2022.

[6] Eric Breck, Neoklis Polyzotis, Sudip Roy, Steven Whang, and Martin Zinkevich. Data validation for machine learning. In *MLSys*, 2019.

[7] Mind the Bridge. Gdpr and beyond recommendations from a transatlantic perspective, Dec 2018.

[8] Li Cai and Yangyong Zhu. The challenges of data quality and data quality assessment in the big data era. *Data science journal*, 14, 2015.

[9] Ann Cavoukian. Privacy by design: The 7 foundational principles, 1997.

[10] Corinna Cichy and Stefan Rass. An overview of data quality frameworks. *IEEE Access*, 7:24634–24648, 2019.

[11] Andrea Compton. Assured deletion in the cloud. `https://www.cs.tufts.edu/comp/116/archive/fall2014/acompton.pdf`.

[12] Maosong Fu, Sailesh Mittal, Vikas Kedigehalli, Karthik Ramasamy, Michael Barry, Andrew Jorgensen, Christopher Kellogg, Neng Lu, Bill Graham, and Jingwei Wu. Streaming@ twitter. *IEEE Data Eng. Bull.*, 38(4):15–27, 2015.

[13] Stefan Grafberger, Julia Stoyanovich, and Sebastian Schelter. Lightweight inspection of data preprocessing in native machine learning pipelines. In *Conference on Innovative Data Systems Research (CIDR)*, 2021.

[14] Looking to comply with gdpr? here's a primer on anonymization and pseudonymization. `https://iapp.org/news/a/looking-to-comply-with-gdpr-heres-a-primer-on-anonymization-and-pseudonymization/`.

[15] Haruna Isah, Tariq Abughofa, Sazia Mahfuz, Dharmitha Ajerla, Farhana Zulkernine, and Shahzad Khan. A survey of distributed data stream processing frameworks. *IEEE Access*, 7:154300–154316, 2019.

[16] Paulan Korenhof, Jef Ausloos, Ivan Szekely, Meg Ambrose, Giovanni Sartor, and Ronald Leenes. Timing the right to be forgotten: A study into "time" as a factor in deciding about retention or erasure of data. In *Reforming European data protection law*, pages 171–201. Springer, 2015.

[17] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd international conference on data engineering*, pages 106–115. IEEE, 2007.

[18] Inês Araújo Machado, Carlos Costa, and Maribel Yasmina Santos. Data mesh: Concepts and principles of a paradigm shift in data architectures. *Procedia Computer Science*, 196:263–271, 2022.

[19] Erum Mehmood and Tayyaba Anees. Challenges and solutions for processing real-time big data stream: A systematic literature review. *IEEE Access*, 8:119123–119143, 2020.

[20] Martin M Merener. Theoretical results on de-anonymization via linkage attacks. *Trans. Data Priv.*, 5(2):377–402, 2012.

[21] Jules Polonetsky and Tim Sparapani. A review of the privacy-enhancing technologies software market. *IEEE Security Privacy*, 19(6):119–122, 2021.

[22] Pegdwendé Sawadogo and Jérôme Darmont. On data lake architectures and metadata management. *Journal of Intelligent Information Systems*, 56(1):97–120, 2021.

[23] Harcharan Jit Singh and Seema Bawa. Scalable metadata management techniques for ultra-large distributed storage systems–a systematic review. *ACM Computing Surveys (CSUR)*, 51(4):1–37, 2018.

[24] Jordi Soria-Comas and Josep Domingo-Ferrer. Big data privacy: challenges to privacy principles and models. *Data Science and Engineering*, 1(1):21–28, 2016.

[25] STRM Privacy documentation. `https://docs.strmprivacy.io/`.

[26] European Union. General Data Protection Regulation. `https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN`, 2016.

[27] Eugene Yan. Real-time machine learning for recommendations. `https://eugeneyan.com/writing/real-time-recommendations/`, 2021. Post published on 10 Jan 2021.

[28] Bin Zhou, Jian Pei, and WoShun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM Sigkdd Explorations Newsletter*, 10(2):12–22, 2008.

It's FREE to join!

TCDE

tab.computer.org/tcde/

The Technical Committee on Data Engineering (TCDE) of the IEEE Computer Society is concerned with the role of data in the design, development, management and utilization of information systems.

- Data Management Systems and Modern Hardware/Software Platforms
- Data Models, Data Integration, Semantics and Data Quality
- Spatial, Temporal, Graph, Scientific, Statistical and Multimedia Databases
- Data Mining, Data Warehousing, and OLAP
- Big Data, Streams and Clouds
- Information Management, Distribution, Mobility, and the WWW
- Data Security, Privacy and Trust
- Performance, Experiments, and Analysis of Data Systems

The TCDE sponsors the International Conference on Data Engineering (ICDE). It publishes a quarterly newsletter, the Data Engineering Bulletin. If you are a member of the IEEE Computer Society, you may join the TCDE and receive copies of the Data Engineering Bulletin without cost. There are approximately 1000 members of the TCDE.

Join TCDE via Online or Fax

ONLINE: Follow the instructions on this page:

www.computer.org/portal/web/tandc/joinatc

FAX: Complete your details and fax this form to **+61-7-3365 3248**

Name _____

IEEE Member # _____

Mailing Address _____

_____

Country _____

Email _____

Phone _____

TCDE Mailing List

TCDE will occasionally email announcements, and other opportunities available for members. This mailing list will be used only for this purpose.

Membership Questions?

Xiaoyong Du
Key Laboratory of Data Engineering and Knowledge Engineering
Renmin University of China
Beijing 100872, China
duyong@ruc.edu.cn

TCDE Chair

Xiaofang Zhou
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD 4072, Australia
zxf@uq.edu.au

IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1314

Non-profit Org.
U.S. Postage
PAID
Los Alamitos, CA
Permit 1398