Poly'19 Workshop Summary: GDPR

Michael Stonebraker
CSAIL
MIT
stonebraker@csail.mit.edu

Timothy Mattson
Parallel Computing Lab
Intel Corporation
timothy.g.mattson@intel.com

Tim Kraska
CSAIL
MIT
kraska@mit.edu

Vijay Gadepally
Lincoln Laboratory
MIT
vijayg@ll.mit.edu

ABSTRACT

Data privacy within the context of heterogenous data and data management systems continues to be an important issue. At the Poly'19 workshop, held in conjunction with VLDB 2019 in Los Angeles, CA, one of the major themes explored was the implication of data privacy regulations such as GDPR to systems composed of multiple heterogenous databases. This summary outlines some of the major approaches and directions presented by various presenters during the privacy portion of the Poly'19 workshop.

1. INTRODUCTION

In conjunction with VLDB'19, we organized the fourth annual workshop on Polystores (POLY'19 [1]). Half of the workshop focused on traditional topics concerning polystore data systems [1, 2]. The other half of the workshop focused on privacy; in particular, regulations such as the General Data Privacy Regulations (GDPR) [9] and their implications for data systems composed of multiple heterogeneous databases. Tim Kraska joined the Poly'19 organizing committee specifically to help us create a strong program on privacy. In this workshop summary, our focus is only on the GDPR and privacy related portion of the workshop.

2. PRIVACY REGULATIONS

It might be tempting to deemphasize GDPR, treating it as largely a European issue. We note, however, that a new law in California based on GDPR (CCPA [7]) went into effect in January 2020. It is expected that other US states will soon follow California's lead. Hence, the European Union's developments in regulating privacy, as defined by GDPR, will have a global impact.

Workshop keynotes were given by Daniel Weitzner (Founding Director, MIT Internet Policy Research Initiative) and Blaise Aguera y Arcas (Google Incorporated). Two articles [4, 5], from the privacy portion of the workshop, addressed legal and semantic topics of implementing GDPR. In [4], the authors highlight a number of privacy related topics propose a research agenda to bridge the gap between traditional security mechanisms and the needs amplified by polystore and federated databases. The authors of [4] also discuss specific policies such as GDPR, within the context of these research questions. Specifically, the authors propose the following research priorities: (1) matching legal requirements with technology capablities, (2) development of accountable privacy preserving systems, and (3) bringing privacy preserving techniques to private islands. The other policy article, [5], focused on GDPR compliance of large cloud providers and analyzed cases of potential non-compliance. They use these examples to propose best-practices for organizations interested in developing their own GDPR privacy policies. The authors provide a thorough analysis of ten cloud services such as Bloomberg and Uber and highlight particular patterns that may indicate GDPR non-compliance. For readers developing their own privacy policy, the authors also provide practical policy and technology recommendations that can be used.

Four additional papers from the session [10, 8, 6, 3] from the workshop dealt with implementing GDPR privacy guarantees and the majority of the discussion in this short summary will focus on these four technical articles. We will further constrain the scope of this discussion by focusing on just two of

[1] https://sites.google.com/view/poly19/

the guarantees defined by GDPR:

- **The right to be forgotten.** Upon a request, personal data on an individual must be deleted unless other laws take precedent. This requirement applies primarily to social media and other web sites that sell personal profiling information for monetary gain. Users of such services must be able to "opt out". There are numerous anecdotes of enterprises spending millions of dollars to find such information in legacy systems where copying of data is rampant. Supporting GDPR (or the similar California regulation) requires such expenditures, and future systems will have to be cognizant of such requirements.
- **Support for restrictions on data usage, so-called "purposes".** Database Management Systems (DBMSs) have long supported access control services. In SQL DBMSs, such services are standardized and are based on users and roles and apply to tables and views. Essentially all SQL DBMSs support such access control mechanisms using metadata that is checked at run time. However, GDPR expands this notion to the concept of purposes. For example, a user might be allowed to access certain data while helping a customer debug a problem, but not to run a marketing campaign. Presently, SQL access control cannot distinguish between these two cases.

We will discuss these two topics in terms of a simple running example. Consider the following data stored about an individual:

- `Email address (key)`
- `Preferred IP address`
- `Zipcode`
- `Department`
- `Expected salary`
- `Expected age`
- `Expected sex`
- `Expected political persuasion`

Additionally, consider the following three purposes:

1. Retrieve a cell value for some unknown purpose
2. Aggregate a cell value with at least 20 other cell values for statistical purposes
3. Use cell value to help target advertising messages

GDPR allows a consumer to specify, for each cell, which purposes they will allow. That could ultimately be three purposes times seven different attributes, each of which could have a separate "yes" or "no" answer. Much "coarser" granularity is possible, and sites may restrict the number of purposes and group cells into collections to reduce the amount of information required. In time, we expect fine-grained granularity will be the required service, potentially organized in a hierarchy. A typical site might have 1 million customers, each of which may constrain allowed purposes and make requests to be forgotten.

3. IMPLEMENTATION APPROACHES

The four implementation papers in the Poly'19 workshops can be grouped into three technical approaches:

3.1 Approach #1: [8, 10, 6]

Store each user's data in a separate physical "segment", which we term a "shard". Hence, raw data is partitioned into shards, one per user. In our running example, there will be 1 million shards. Then, a pipeline of operations is proposed to generate materialized views (MVs) of interest. These MVs aggregate information from the 1 million source shards. A user query simply goes to the correct MV and runs his/her particular query. In this approach, the pipeline will contain normalization operations (into common units for example) as well as SQL operations. However, to be mindful of purposes, each MV must have data allowed for only one purpose. Consider an MV which aggregates salary by department. There will have to be three versions of this MV, one for each purpose, and a user will have to query the one that corresponds to their purpose. In such a system, significant redundancy is inevitable.

In read-almost-always decision support applications, this approach will probably work. However, on every update some-to-most of the MVs will have to be recreated. If one is lucky, this can be done incrementally. For example, if the department of "Joe" is changed from "candy" to "shoe" and a particular MV is storing the average salary by department, then the two departments can be readily adjusted without examining other data. On the other hand, if the MV contains a join, then incremental update becomes much more difficult. In fact, some vendors make no effort to update MVs, choosing to let them "decay" over time, recreating them periodically. Whenever, a user shard is deleted,

the same propagation issues arise. For applications where there are substantial updates, approach 1 is not likely to work well.

Lastly, there is a substantial data discovery problem to decide which MVs must be interrogated. If a consumer wants information on both age and salary, they will likely have to interrogate two MVs, one for age and one for salary. This will be required whenever a consumer allows different access by cell for the various purposes.

Wang et. al. [10] have a similar pipeline approach. However, the "shard" for each user is associated with a collection of constraints on access. Think of these as a mix of SQL access control and GDPR purposes managed through a rule-oriented constraint language. The paper demonstrates that a rule language is rich enough to define GDPR specifications. That is not surprising since GDPR deals mostly with constrained access where rule systems are an obvious mechanism. If an analysis program wishes to access multiple shards to create an output shard (think of this as constructing a materialized view), then the system proposed in [10] can automatically and quickly figure out which constraints apply to the output shard. Hence, [10] has a mechanism to propagate access constraints through a pipeline of materialized views. However, for GDPR compliance, this amounts to simply taking the intersection of all the allowed purposes for the various cells. Lastly, this approach has all of the issues noted above concerning the solution in [8].

On the other hand, Pasquier et al [6] deal with GDPR at the operating system level and do not assume all data is stored in a DBMS. Given a pipeline of operations, this paper allows them to be a "mix and match" of operations in various subsystems (web servers, Hadoop, etc.). They assume provenance is stored about each operation. In effect, this is "who did what to whom". At the OS level all semantics is lost, and it is not clear how to support purposes. In addition, it is not clear what purposes even means in [6]. Lastly, it is not clear how the approach in [6] will continue to work as systems scale to large sizes.

In summary, Approach #1 is delete optimized. It is straightforward to support the right to be forgotten; one merely deletes the indicated user shard and recreates or updates all downstream MVs. It seems plausible to include a purpose system into this pipeline as proposed in [2]

3.2 Approach #2: [3]

Here, the focus is on a "clean" entity-relationship (E-R) DBMS schema for data that stores each fact exactly once. Materialized views in this approach must be disallowed, since they introduce replication, which make it difficult to find and delete personal data. Therefore, this approach advocates merely adding a table of user identifiers which are connected to the "clean" schema with the relationship "is relevant to". As such, deleting a user's information is easy, as one need only follow the E-R diagram from a user's entry point deleting information along the way. In fact [3] shows an implementation where this deletion can be performed in a lazy fashion. In effect, this is a different way to optimize for deletions.

Physically clustering E-R data to optimize retrievals is an interesting future exercise. As such, this schema will have no redundancy and consume minimal space. In contrast to Approach 1, this scheme will optimize space consumed (redundancy) to achieve better deletion performance. There is no recomputation time on deletes, but worse (perhaps significantly worse) query performance.

In addition, [3] proposed storing the purposes that apply to each cell as a bit vector for each cell. This will optimize purpose-oriented processing but, of course, require (perhaps significant) additional space.

Note that both Approaches #1 and #2 make onerous restrictions on the schemas/applications supported. It is widely known that the "clean" schemas required by Approach #2 may have poor run-time performance, since most enterprises diverge from clean schemas for performance reasons, and materialized views are introduced for the same reason. Arguably, Approach #2 also requires substantial changes to existing applications. Approach #1, on the other hand, will fail badly when there are significant updates. Therefore, we now turn to a third approach, that can deal effectively with legacy environments.

3.3 Approach #3:

Our final approach assumes the schema is arbitrary. We see this approach in [3] which includes a proposal for situations where the schema is unconstrained. This case, of course, applies to legacy information systems where the schema has evolved without regard for GDPR. Unlike in Approach #1 where there is a stylized way for information to flow among MVs, Approach #3 must cope with arbitrary data movement. There seems no other way to support this, other than maintaining detailed lineage to keep track of such information redundancy. The overhead of such a scheme is expected to be very onerous unless GDPR is supported at a coarse-grained level, which might have limitations of its

own.

In summary, the three approaches differ in:

- Generality of schemas supported
- Space consumed
- Cost of supporting the right to be forgotten
- Cost of supporting purposes

Hopefully, our discussions in this summary will generate additional proposals and a more detailed analysis of the options for addressing GDPR. Additional work is certainly warranted. Also, one is always reminded that the devil is in the details. There are many unaddressed issues:

- What happens when data is not associated with a single user? I.e. what about a marriage between two persons?
- As is often pointed out in these papers, a log fundamentally violates the right to be forgotten. I.e. one can "forget" information on a human in the database, but that information is still in the database log. Selectively purging the log will make certain kinds of recovery impossible. In other words, one is well advised to trust the DBAs, who have access to the log. Multiple "corner cases" exist of this sort.
- Record-level provenance, whether supported by the OS [6] or the DBMS [3] should be merged into a DBMS log to avoid duplication of effort and data. How to structure such a log is an interesting question.
- Compilation. To achieve high performance, purposes and provenance will have to be aggressively compiled. This will, of course, interfere with flexibility and changeability.
- How to support external applications or data movement across companies. For example, what happens if a user creates a Python notebook and copies some data into it to build an ML model. If a user requests to be forgotten, the whole Python notebook has to be deleted as an update to the notebook is not always possible.
- How can an entire company be efficiently transitioned to be GDPR compliant?

4. CONCLUSIONS

Hopefully, this summary and the privacy papers from Poly'19 will spur others to consider implementation issues raised by GDPR, especially ideas that make it relatively easy to support the right to be forgotten. The Poly'20 workshop[2] is scheduled to be held in conjunction with VLDB 2020 and we look forward to additional research and proposals in this area.

5. ACKNOWLEDGEMENTS

The authors wish to thank the program committee of the Poly'19 workshop.

6. REFERENCES

[1] Gadepally, V., Mattson, T., Stonebraker, M., Wang, F., Luo, G., Laing, Y., and Dubovitskaya, A. *Heterogeneous Data Management, Polystores, and Analytics for Healthcare: VLDB 2019 Workshops, Poly and DMAH, Los Angeles, CA, USA, August 30, 2019, Revised Selected Papers*, vol. 11721. Springer Nature, 2019.

[2] Gadepally, V., Mattson, T., Stonebraker, M., Wang, F., Luo, G., and Teodoro, G. *Heterogeneous Data Management, Polystores, and Analytics for Healthcare: VLDB 2018 Workshops, Poly and DMAH, Rio De Janeiro, Brazil, August 31, 2018, Revised Selected Papers*, vol. 11470. Springer Nature, 2019.

[3] Kraska, T., Stonebraker, M., Brodie, M., Servan-Schreiber, S., and Weitzner, D. Schengendb: A data protection database proposal. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 24–38.

[4] Kroll, J. A., Kohli, N., and Laskowski, P. Privacy and policy in polystores: A data management research agenda. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 68–81.

[5] Mohan, J., Wasserman, M., and Chidambaram, V. Analyzing gdpr compliance through the lens of privacy policy. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 82–95.

[6] Pasquier, T., Eyers, D., and Seltzer, M. From here to provtopia. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 54–67.

[7] Rothstein, M. A., and Tovino, S. A. California takes the lead on data privacy law. *Hastings Center Report 49*, 5 (2019), 4–5.

[8] Schwarzkopf, M., Kohler, E., Kaashoek, M. F., and Morris, R. Position: Gdpr compliance by construction. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 39–53.

[9] Tankard, C. What the gdpr means for businesses. *Network Security 2016*, 6 (2016), 5–8.

[10] Wang, L., Near, J. P., Somani, N., Gao, P., Low, A., Dao, D., and Song, D. Data capsule: A new paradigm for automatic compliance with data privacy regulations. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. Springer, 2019, pp. 3–23.

[2] https://sites.google.com/view/poly20/home