# Proposal: GDPR Compliance by Construction in Noria

Wensi You
*Brown University*

Zeling Feng
*Brown University*

Zhoutao Lu
*Brown University*

## 1 Introduction

EU's General Data Protection Regulation(GDPR) grants individuals significant control of and powers over their own data. It challenges the design of applications that store/process personal data. GDPR compliance by construction [4] proposes using the partially-stateful dataflow model to grant users the "right to access", the right to erasure "without undue delay", the right to data portability and the right to object "anytime to processing". Besides, many web applications need to serve users' requests at low latency. Noria [3] proposes a novel partially-stateful data-flow model for building long-live, low-latency applications. This project aims to investigate how GDPR compliance by construction design principles could be applied to the Noria data-flow system to achieve both GDPR compliance and high performance.

## 2 Design and Challenges

### 2.1 User shards

If we have a way to split the user data into different universes, then it is by construction easy for each user to access and erase his/her data which is compliant to Regulation. The question becomes whether we should shard the user data logically, or physically. This makes up the main design question. Currently, Noria [3] has sharding support but it is based on Key ranges rather than Users. It is not feasible to predict how many users before the app is launched. So instead of doing a physical sharding, we propose to do it logically, i.e., by using an index on the column that represents a data subject. Now we need to consider providing use a mechanism for developers to specify which column is representing the real end user. A candidate solution is `SET USER-COLUMN column name`.

### 2.2 Description of materialized views

GDPR requires that users be provided with information regarding "the purposes of processing". To achieve that, we may attach a specification that describes the purpose of each materialized view. To get all purposes related to the user's data, we could 1) accumulate that information on base tables and upon request, we could find all tables that contain the user's information and return the union of all specifications, or 2) perform an ad-hoc traverse on the graph to find all reachable materialized views and return a set of specifications. The trade-off here is when queries are changed we may need to do additional updates in the first approach but we could save the time to traverse the graph in this way.

### 2.3 Erasure policy

To enable the right to erasure, it requires our system to delete all direct and derived user. For this case, Noria [3] already has the novel mechanism to track the derived information and make updates with eventual consistency in the dataflow when base tables change. However, as [4] points out, it is also a common case that user's withdrawal on their data might require application-specific anonymization rather than outright data removal. To cope with that, we assume at least 2 different cases when requesting erasure, i.e. directly removal and objection or possibly also a third one – anonymization. For simplicity, we would also need to attach a specification on each base table to indicate the respective response policy upon erasure request. To anonymize the data, we also need to update the corresponding user shards and reassign the data to a separate shard for the anonymized user.

### 2.4 Trusted transfer for user shards

We already have the ability to export all data related to a specific user, now we have to resolve the problem with cryptography schemes. OpenPGP [2] is a promising message format to ensure the confidentiality and integrity of the data transfer. GnuPG [1] is an open-source implementation of OpenPGP. We can encrypt the whole data export using the new data controller's public key so that only the new data controller can be the recipient and decrypt data. Also, we require the old data controller to sign on the message so that the new data controller can verify that the data is unaltered and be trusted.

## 2.5 Guard data for an objecting user's shards

GDPR also requires that users have the right to object at any time to the processing of their personal data. And borrowing the idea from [4], we could augment the data-flow in Noria [3] with a new operator "guard", which will check whether the related user has objected to the specific processing. Whether a user allows certain processing can be referred to as "objection policy". And these policies should be stored within users' own shards. First, we need to find a way to relate those policies to the specification of all processing or materialized views. Second, when data flow from base tables in Noria, the system should be aware of the policies related to the owner of those data. We may need to attach the policies along with the data flow. Third, we need to add the guard operators somewhere in the subgraph for that query(or materialized view) with respect to the processing. ① The most straightforward way might be to add those guard operators just under base tables. But this requires disabling operator sharing in Noria, as a shared operator would come from two materialized views that users could have conflicting policies. ② Alternatively, in order to keep operator sharing, we could find the most strict boundary of the subgraph for each materialized view and guard just on the boundary. This approach has several challenges: 1) it requires designing an algorithm to detect those boundaries when subgraph could overlap with each other; 2) when the queries are changed dynamically, we also need to add the mechanism to move those operators around. ③ To simplify adding guard operators in the dynamically changing dataflow graph, we could always add the guard operator just on the top of the materialized view. The trade-off here is that we may spend lots of unnecessary computations on the corresponding subgraphs. Besides, it also puts a challenge in dealing with aggregation operators. Users' data will be aggregated in some sharing operators, such as `TopK` or `Count` and when passing through the guard operator, we need to be able to update the aggregated value when removing objecting users' data. (*This is not that trivial to implement and we may need more research in that direction.*)

## 3 Summary

Reconstructing Noria to be GDPR compliant-by-construction has important implications. It may indicate the possibility of a web application backend framework that not only achieves high performance but also conforms to high standards of data privacy. Previous designs normally have to sacrifice one for the other, but a compliant Noria may constitute a promising option for any future read-heavy web applications.

## References

[1] The gnu privacy guard. `https://www.gnupg.org/index.html`. Online; accessed 10 October 2019.

[2] IKS GmbH H. Finney D. Shaw R. Thayer J. Callas, L. Donnerhacke. Rfc 4880. 2007.

[3] Jonathan Behrens Lara Timbo Araujo Martin Ek Eddie Kohler M. Frans Kaashoek Robert Morris Jon Gjengset, Malte Schwarzkopf. Noria: dynamic, partially-stateful data-flow for high-performance web applications. In *USENIX Symposium on Operating System Design and Implementation (OSDI)*, 2018.

[4] M. Frans Kaashoek Robert Morris Malte Schwarzkopf, Eddie Kohler. Position: GDPR Compliance by Construction. In *Poly 2019 workshop at VLDB*, 2019.