Towards Query-Driven Data Minimization

Peter K. Schwab, Julian O. Matschinske, Andreas M. Wahl, and Klaus Meyer-Wegener

FAU Erlangen-Nürnberg, Informatik 6 (Datenmanagement)

Abstract. Due to the data privacy laws inured recently, data minimization is a highly topical issue. So far, many companies have collected and processed a vast amount of personal data while treating data privacy negligently. They now need to narrow down collection and processing, but do not know, which data is essential for transacting their businesses. To overcome these problems, we propose query-driven data minimization. Our approach grants user-specific access to database schemas based on SQL reports and supports deletion of unneeded schema elements.
In this paper, we outline the schema of a query repository for managing query meta-information. We introduce a user story emphasizing the benefits of our approach for query-driven data minimization and we sketch the basic architecture of a framework implementing our approach.

1 Introduction

The EU General Data Protection Regulation[1] (GDPR) recently took effect and requires substantial changes for almost every institution storing personal data. For example, Art 5 GDPR postulates the principle of data minimization, i. e. to collect, process, and use as little personal data as possible. However, it is a tremendous effort to evaluate which data must really be collected and which users are allowed to use them.

On one side, there are data scientists with access to large parts of the companies' datasets. They are often unaware of data privacy and will not restrict data access by themselves. On the other side, there are privacy officers who are well aware of data privacy. They usually cannot prohibit fine-grained, technical data access, e. g. based on SQL queries, but only step in when data collection or processing has already taken place.

The data-privacy community advocates privacy by design [7], which incorporates privacy protection into the overall design of technical systems. In this paper, we take up privacy by design and propose a minimally-intrusive approach regarding query-driven data minimization in existing relational IT landscapes. We enable privacy officers to grant customized access to relational databases based on the queries that users run on these systems. Furthermore, our approach shows which data are actually not used in any data processing and lists the queries that have inserted the data into the database. Privacy officers can trace and delete unneeded data as well as prevent their future collection. To reach this, we have extended an initial design for a query repository proposed in [8] in order to log queries and enrich them with meta-information.

[1] `http://eur-lex.europa.eu/eli/reg/2016/679/2016-05-04`

2 Query Repository

The query repository records SQL-based queries and their characteristics. Fig. 1 shows an Entity-Relationship diagram with its conceptual schema.

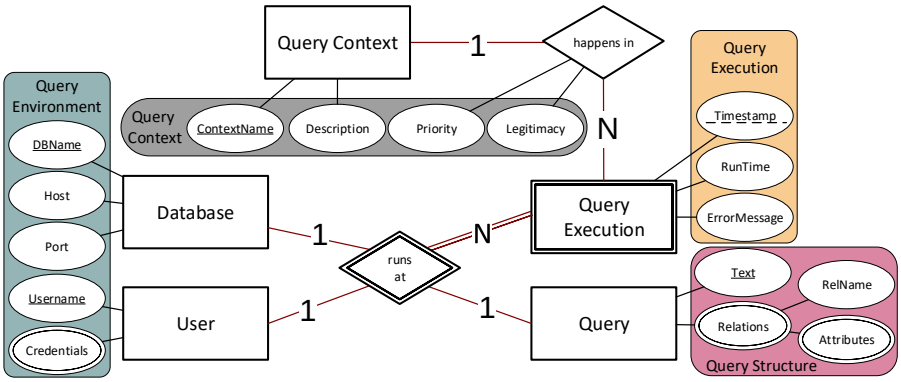

Fig. 1. The conceptual schema of the query repository

While most entities, relationships, and attributes are self-explanatory, Query Context needs some remarks. It has an identifying ContextName and an optional Description. These attributes point out a user's intention or the situation in which she runs a query via the relationship happens in. If a Query Execution happens in a certain Query Context, the user can additionally specify the query's priority and its legitimacy in this context. A query execution can only be done in one context, but a context may contain many query executions.

We organize query characteristics in four groups highlighted in color in Fig. 1. The first one is the Query Structure that we derive automatically from the query itself. The other groups hold meta-information on the queries. The Query Environment is derived from the user and her initial connection to the target database. The Query Execution stems from the query, the executing user, and the database. Finally, the Query Context cannot be derived automatically; it must be entered manually. It typically contains tacit user knowledge, which we externalize here.

3 Benefits of a Query Repository

To explain the benefits of a query repository, we describe a user story from a clinical research scenario: The databases of a clinical information system (CIS) store data required for the clinical operation. The hospital's employees access these databases directly in sessions using SQL-based tooling and generate reports for their purposes. For example, Bob is responsible for billing of services provided by the hospital. He reads the patients' address data as well as ICD (International Classification of Diseases) codes stored in case files. Doctor Carol generates reports for a clinical study based on case files ("secondary use" of data). She reads medication data as well as data regarding the patients' health status. Our query repository (QRep) logs all queries submitted to the CIS and automatically derives related query characteristics.

In the past, Bob's and Carol's access to the databases was unrestricted, i. e. Bob had access to medication data, although not relevant for billing purposes, and Carol was able to see address data not needed for her clinical study. We are in June 2018 and Alice, the hospital's privacy officer, wants to apply Art 5 GDPR in terms of data minimization. Therefore, she displays Bob's report results together with its related queries in the QRep WebApp. She marks all queries as 'legal' if the reports are compliant with Art 5 GDPR. Otherwise, she identifies unauthorized queries and marks them as 'illegal'. The QRep server stores her markers as Query Context information. Finally, Alice repeats the procedure for Carol's reports.

Next, Alice clicks a button and the QRep server determines all schema attributes and relations used in the queries marked as 'legal', grouped by database user. Our tool then retrieves the entire schema information from the databases, requests Alice to decide each user's access rights to unused schema elements and finally enforces these rights to the databases. In the example, Carol must no longer access the entire relation with the patient's address data, and Bob has access only to the ICD attribute in relation to the case-related diagnostic data of the patients. This ensures data minimization regarding data processing.

Alice clicks another button and our tool determines the schema elements to which no user is granted read access anymore. It also lists the queries that have inserted the data. Alice can see now that there is, e. g. an attribute holding a patient's nationality that is not needed for any data processing and can delete this attribute by a single click. She also sees that employee Dave ran the related queries, contacts him and ensures data minimization regarding data collection.

4 Reference Architecture

To maintain compliance with data-protection regulation, responsible officers can use our WebApp. It provides an easy-to-use interface and does not require any knowledge of SQL. The WebApp interacts with the QRep server via a JSON-based REST API. The overall system architecture is shown in Fig. 2. The QueryManager passes incoming SQL queries to the Analyzer module, which enriches them with Query Structure characteristics by using Apache Calcite as parser. The DB Analyzer connects with target databases to determine Query Execution characteristics. Users can add Query Context characteristics via WebApp. The QRep DB persists this information. The SearchModule allows for browsing queries and their characteristics and displaying report results. The DB Analyzer also provides full target schema information for detecting unneeded relations and attributes.

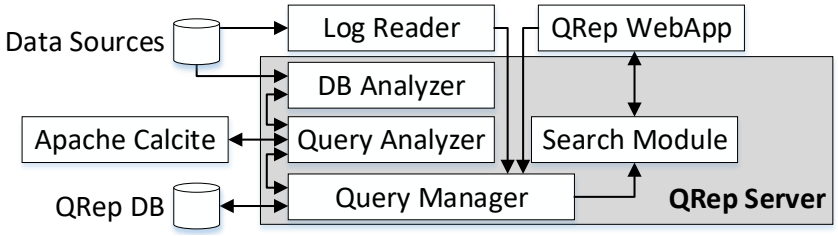

Fig. 2. The overall architecture of our framework

5 Related Work

Khoussainova et al. propose dedicated systems for query management [3]. However, they focus on query completion [4]. Srivastava et al. provide a relational framework for managing queries [9]. We follow their approach and manage queries and their meta-information together in a relational model. A tool for session-based query-log browsing is suggested in [2]. However, queries are not enriched with meta-information and browsing is limited to keyword-based search. Van den Bussche et al. provide user-defined functions to query ASTs stored as XML files using a relational database [1]. Query context and other meta-information are not considered. This is also the case for several graph-based approaches for representing SQL queries, e. g. [5, 6]. In [10], we already analyzed query logs to facilitate data integration. However, we did not yet provide a way to report user-specific meta-information about queries. Manta and SQLDep[2] visualize schema lineage of SQL queries without considering query context.

6 Summary and Future Work

Our framework supports data minimization regarding the collection and processing of data. The query-driven approach allows to customize user access to the data and to identify unneeded schema elements. We plan to evaluate our approach based on publicly accessible query logs. Regarding data minimization, we also aim to consider unneeded tuples.

Acknowledgement: The authors would like to thank the anonymous reviewers for their valuable remarks.

References

1. Van den Bussche, J., Vansummeren, S., Vossen, G.: Towards practical meta-querying. Information Systems **30**(4), 317–332 (2005)
2. Khoussainova, N., et al.: Session-based browsing for more effective query reuse. In: Proc. SSDBM. pp. 583–585. Springer (2011)
3. Khoussainova, N., Balazinska, M., Gatterbauer, W., Kwon, Y., Suciu, D.: A case for a collaborative query management system. In: Proc. CIDR (2009)
4. Khoussainova, N., Kwon, Y., Balazinska, M., Suciu, D.: Snipsuggest: Context-aware autocompletion for SQL. Proc. VLDB **4**(1), 22–33 (2010)
5. Koutrika, G., Simitsis, A., Ioannidis, Y.E.: Explaining structured queries in natural language. In: Proc. ICDE. pp. 333–344. IEEE (2010)
6. Papastefanatos, G., et al.: Hecataeus: A framework for representing SQL constructs as graphs. In: Proc. EMMSAD. vol. 5 (2005)
7. Schaar, P.: Privacy by design. Identity in the Information Society **3**(2), 267–274 (Aug 2010). https://doi.org/10.1007/s12394-010-0055-x
8. Schwab, P., et al.: Query-driven data integration (short paper). In: Proc. LWDA. CEUR Workshop Proc., vol. 1670, pp. 206–211. CEUR-WS (2016)
9. Srivastava, D., Velegrakis, Y.: Intensional associations between data and metadata. In: Proc. SIGMOD. pp. 401–412. ACM (2007)
10. Wahl, A.M., et al.: Query-driven knowledge-sharing for data integration and collaborative data science. In: Proc. ADBIS. pp. 63–72 (2017)

[2] `https://getmanta.com/` and `https://sqldep.com/`