Odlaw
Retroactive GDPR Compliance For Relational Databases

Jearson Alfajardo
Brown University

Connor Luckett
Brown University

Abstract

Odlaw is a command line interface application which allows data controllers with preexisting databases to automatically generate data reports and perform data deletion for the sake of GDPR compliance. As we attempt to move towards full GDPR compliance, some are more prepared than others to retrofit, or even replace, existing systems. Odlaw leverages the referential integrity of a database in order to crawl the schema. This process enables it to automatically find all dependent data related to a given data subject, which can be used for GDPR compliance. We also evaluate Odlaw's ability to produce GDPR-compliant results. The methods provided in this paper have the potential to ease the burden of GDPR compliance on small businesses who are unable to compensate consultants to assist with GDPR compliance.

1 Introduction

Complying with the GDPR can be a frustratingly difficult problem to solve. Some have argued that the best way to comply with the GDPR is to construct a compliant system from the beginning [13]. We call this method *proactive compliance*. Unfortunately, it is often difficult and expensive to replace already-in-place systems.

This dilemma brings us to what we will call *retroactive compliance*, where we design applications to assist data controllers in their pursuit of compliance, while maintaining as much of their current infrastructure as possible. We recognize there is no "one size fits all" solution, and instead we focused our attention to the process of generating data subject reports and deleting all customer data, which we explore further in Section 2.

Many applications are small in scale, and generating data subject reports may seem trivial. Unfortunately, the process of retroactively complying is not always straight-forward. For example, this process quickly becomes less intuitive for larger applications with more tables. There is a need to investigate automated solutions for this problem.

There is a need for algorithmic methods which can be proven to be successful. We turn our attention towards a commonly used tool in many applications: relational databases. Odlaw leverages the fact that many applications are built on relational databases. Odlaw's success is dependent on the database being in at least 3NF (i.e., there exists *referential integrity*). Referential integrity ensures that all table relationships are available in the database's schema.

By querying the schema of the database, we should be able to understand how a database user would create queries to generate reports. We can leverage that knowledge to automatically generate queries to automate the data subject report and data erasure processes. We hope Odlaw will greatly assist many small businesses in their pursuit of GDPR compliance.

Odlaw is a command-line interface tool which can automatically generate data subject reports and completely remove all user-dependent data in order to assist with GDPR compliance. We stress that it is only designed (so far) to accomplish these two goals. However, we envision other tools can be developed in order to retroactively comply. We believe that Odlaw is the first of its kind to be used for retroactive GDPR compliance.

2 Background and Related Work

Since the introduction of the European Union's General Data Protection Regulations in May of 2018, many businesses and individuals have found themselves subject to monetary fines for non-compliance. However, compliance is not always trivial, as many modifications performed to be compliant are all done manually and thus prone to human error. We believe the safest option is to utilize tools to achieve compliance.

Techniques of GDPR compliance have recently been of great interest. Many have taken the stance that new architectures will need to be designed in order to comply [13], and some have proposed examples of such new architectures [10]. Unfortunately, such infrastructure is not widely available nor quite ready for widespread adoption.

Instead, we turn our attention to *retroactive compliance*. We ask: how can we assist data controllers in complying with

the GDPR while maintaining as much of the currently implemented system as possible? As the majority of smaller businesses are "left in the dust" so to speak, by regulations, Odlaw provides a "bolt-on" compliance tool to allow many to meet some of the many requirements of the GDPR. Specifically, this includes requirements for organizations to provide their users specific rights, including the *Right to Access Personal Data*, the *Right to Data Portability*, and the *Right to Erasure*.

Odlaw resolves compliance with the *Right to Access Personal Data* and the *Right to Portability* by generating a **data subject report**, which is a series of tables including all data dependent on the data subject. Compliance with the *Right to Erasure* by a **erasure** request. This is achieved by noting all dependent data for a particular data subject and deleting it.

While others have previously suggested modeling relational databases as a directed graph and even its potentials for deleting all dependent data for some entry [12], we believe that we are the first to apply those methods and techniques into a realizable, open-source application that is also available for widespread use. Further, we recognize its potential to generate reports of user data, which can be rendered or ported to the user. Specifically, we believe that we are the first to consider its potential uses in GDPR and other regulations.

3 Design

The key idea of Odlaw is its representation of a database as a graph in order to perform a graph search. Odlaw queries a database's schema, determines all tables; primary keys; and foreign key constraints in that database. Then, Odlaw constructs a graphical representation of the database. Each table is represented by a node. Each primary key is stored as an attribute for its respective node. Each foreign key constraint is represented as a directional edge, drawn from the table containing the primary key to the table containing the foreign key constraint. An example of this process for the TPC-H database [3] is given in Figure 1.

This example is how Odlaw generates reports. After initialization, Odlaw has a directed, acyclic graph with which it can represent the database. Given a table from which the search should start (a source node) and an identifier for the entry, Odlaw performs a graph search from the source node until it reaches a sink. It builds up a report as it traverses the graph.

Currently, the main interaction between Odlaw and an Odlaw user is a Python command line interface (CLI). A database administrator can supply their credentials to validate a data subject report or data subject erasure request. The database administrator can choose to have the results exported to CSV files, which can be distributed to the user who requested the report. Data subject erasure is done similarly with a different flag.

4 Implementation

The codebase for Odlaw consists of roughly 800 lines of code. This small size is due to our utilization of powerful libraries which do most of the "heavy lifting". Odlaw was written in Python and makes use of the popular Pandas [11], NetworkX [6], and SQLalchemy [5] packages for its core functionality.

We faced several challenges during the development process. Our original intent was to start with PostgreSQL. Instead, we began with SQLite for developmental purposes. In addition, this was our first interaction with the Python virtual environment. We recognize that the storage footprint of the application is not minimal, but the use of the virtual environment will allow others to utilize the application until we are able to package and distribute.

Of course, Odlaw's power comes with a caveats. Odlaw requires administrator's privilege to access a database in order to function properly. We note that we intend Odlaw only to be used server-side. We envision that an Odlaw operation would possibly be called by a remote procedure call.

The exportation of data poses some unique challenges: (1) unique identifiers can be sensitive in nature, and (2) there is a risk of an adversarial user "reverse-engineering" the database.

Unique identifiers are often used for API calls, and displaying them to a user may be a poor privacy decision on behalf of the data controller. Further, the identifier is typically a long integer, and it should have little to no value to a user. The first challenge is easily overcome. Odlaw offers a `-censor` flag to prevent all private key entries from being displayed to the user. In addition, Odlaw also offers a `-block` flag to censor individual columns from the user if absolutely necessary. Other useful features include a `-joined` flag to display the components of the database graph and a `-show` flag to present the graphical form to the Odlaw user.

The output of an Odlaw report is no more than a subset of the original database. Table and column names are left unchanged. An adversarial user having an understanding of the schema may be of security concern; however, after consulting data subject reports from large companies such as Spotify [7], we feel comfortable in the amount of information being released to the user.

5 Evaluation

We tested Odlaw's general usefulness by submitting a graph generation request for several sample databases. Generating TPC-H databases was, as usual, a struggle. Unfortunately, due to the fact that the schema for **lobste.rs** [9] was written in Rust, we were unable to create a MySQL instance in time.

Success varied since Odlaw is dependent on a database being in at least 3NF in order to be successful. We noted that **HotCRP** [8] was particularly poor example for Odlaw, as shown in Figure 3, where the there exist absolutely no foreign

key references whatsoever. Although Odlaw could successfully generate a graph, that graph is completely useless to Odlaw. Odlaw is unable to provide GDPR-compliant results.

Another interesting example of failure is the sample database of employees provided by Oracle [4], as shown in Figure 2, which we call **Employees**. We consider an *Employee* to be the data subject in this example. The resulting graph is composed of three separated sub-graphs. If no data in the smaller sub-graphs are dependent on *Employee* data, then this is not a cause for concern. However, we have no way of knowing this for certain. Still, Odlaw is unable to provide GDPR-compliant results.

We also tested a SQLite database representing a record store [2] and a MySQL database representing a car dealership [1]. We consider them to be accurate reflections of a small business which would be a typical Odlaw user. They contained properly specified foreign key relationships and are ready for use with Odlaw.

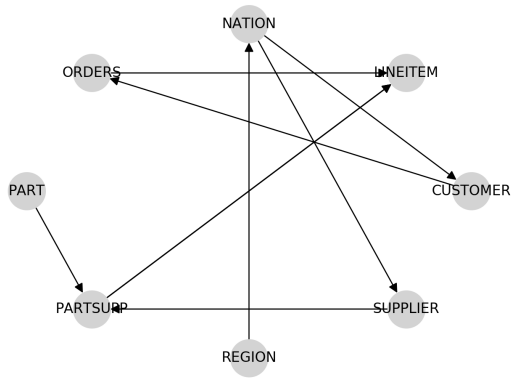

Figure 1: TPC-H in Odlaw

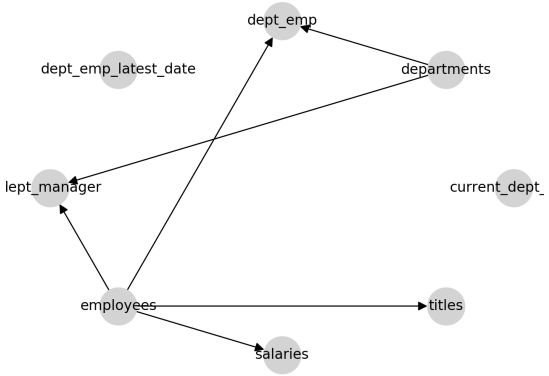

Figure 2: Employees in Odlaw

We tested Odlaw's performance by conducting test jobs on TPC-H databases. First, we generated a 1 GB SQLite instance of a TPC-H database. We submitted and timed the

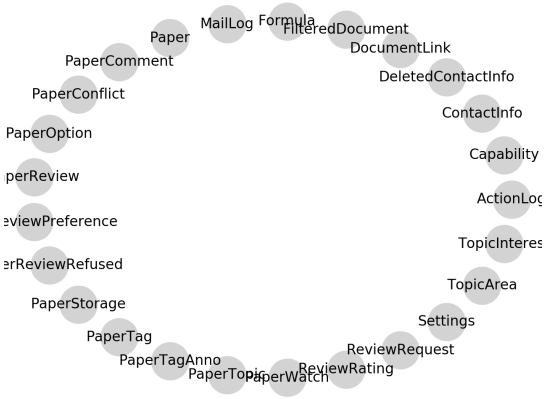

Figure 3: HotCRP in Odlaw

Dialect	Action	Mean (ms)	Median (ms)
SQLite	Access	208	218
SQLite	Erasure	190	190
MySQL	Access	134	146
MySQL	Erasure	98	99

Table 1: Benchmarking Results on TPC-H

generation of 1,000 data subject reports. Then, we submitted and timed the "GDPR compliant" deletion per Odlaw of 1,000 users within the database. Next, we generated a 1 GB MySQL instance of a TPC-H database and repeated the testing process. The submission of an Odlaw job was performed locally on the database's host machine. Our results, as shown in Table 1. We note that it is a bit odd that MySQL actions took less time than an in-memory database such as SQLite. All testing was conducted on a 2019 MacBook Pro with an Intel Core i5 processor.

We note that reports generated were not saved. Rather, they were printed to the console. This was done to monitor the application. Considering that file operations are faster than console printing, we consider our recorded values to be a pessimistic evaluation of query throughput. Thus, Odlaw should be able to consistently generate reports for a user in under a second. Because we consider the number of users who actually request their data to be ported or deleted to be only a small subset of all users, we believe this query throughput is reasonable. Of course, we note that the time to complete an Odlaw job is proportional to size of the database (i.e., more data takes longer to report or delete).

We are substantially more concerned with the protections in place to manage a call to Odlaw (i.e. an adversarial user requesting a data report for another user), which is outside the scope of our control. Considering that Odlaw returns *all* user-dependent data, an adversarial call to the application would reveal everything known about the victimized user.

6 Future Work

In the future, we would like to add PostgreSQL functionality to attract more potential users.

Currently, Odlaw ignores any cycles in the graphical representation of the database. Modifications to the current graph search algorithm will be needed to overcome this.

Odlaw's data search algorithm is wholly dependent on the searched database being in at least 3NF. We recognize that not all relational databases meet this requirement. We also recognize that not all users of Odlaw are "tech-savy". We hope to implement tools which will assist administrators in developing referential integrity. We hope to expand Odlaw's usability by providing functionality within the graphical layer to assist data controllers in specifying foreign key constraints. Specifying foreign key constraints is indispensable to the success of Odlaw, so encouraging data controllers to do so is of great interest.

Finally, we neglected that a report truly consists of data from data-subject-dependent data For example, in TPC-H the data controller has data regarding where the Customer is (Nation), but that data is not displayed in the report because Nation is not dependent on Customer data. Rather, Customer data is dependent on Nation data. This complicates the use of Odlaw. However, this problem is quickly fixed with a reversal of the graph and a second search. We intend to add this functionality as soon as possible.

7 Conclusion

Odlaw provides an open-source option for many businesses and individuals who desire a "quick fix" to meet a few specific requirements of GDPR compliance. Although it is not by any means an "all in one" solution, we hope it sparks further discussion on techniques for retroactive GDPR compliance. Our source code for the project is available at https://www.github.com/cobelu/Odlaw.

References

[1] Mysql sample database. http://www.mysqltutorial.org/mysql-sample-database.aspx.

[2] Sqlite sample database and its diagram (in pdf format). https://www.sqlitetutorial.net/sqlite-sample-database/.

[3] Tpc-h. http://www.tpc.org/tpch/.

[4] Employees sample database. https://dev.mysql.com/doc/employee/en/, Dec 2019.

[5] Michael Bayer. Sqlalchemy. In Amy Brown and Greg Wilson, editors, *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks*. aosabook.org, 2012.

[6] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[7] Spotify USA Inc. *Understanding my data*. https://support.spotify.com/us/account_payment_help/privacy/understanding-my-data/.

[8] Eddie Kohler. Hotcrp. https://hotcrp.com/.

[9] Lobsters. lobsters. https://github.com/lobsters/lobsters, Nov 2019.

[10] Alana Marzoev, Lara Timbó Araújo, Malte Schwarzkopf, Samyukta Yagati, Eddie Kohler, Robert Tappan Morris, M. Frans Kaashoek, and Sam Madden. Towards multiverse databases. In *Proceedings of the Workshop on Hot Topics in Operating Systems, HotOS 2019, Bertinoro, Italy, May 13-15, 2019*, pages 88–95, 2019.

[11] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[12] Radoslav Radev. Representing a relational database as a directed graph and some applications. In *BCI*, 2013.

[13] Malte Schwarzkopf, Eddie Kohler, M. Frans Kaashoek, and Robert Tappan Morris. Position: GDPR compliance by construction. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare - VLDB 2019 Workshops, Poly and DMAH, Los Angeles, CA, USA, August 30, 2019, Revised Selected Papers*, pages 39–53, 2019.