SQL2Solid: Exposing Data and Permissions from Relational Databases via the Solid Protocol

Lukas Kubelka[1], Benjamin Meyjohann[1], Christoph H.-J. Braun[1] and Tobias Käfer[1]

[1]Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Abstract

We address the challenge of integrating relational databases with the Solid Protocol. The goal: Make legacy tabular data accessible as virtual Web resource representations in RDF graphs, while propagating and re-using the database's existing roles and permissions for access management. We present our solution comprised of an SQL database, an RDF graph and Web resource virtualisation layer using the mapping language R2RML, and the Solid Protocol for authentication, authorization and data access.

Keywords

Linked Data, Solid, SQL, OBDA, Web Access Control

1. Introduction

The Solid Protocol [1] aims to break open data silos by decoupling users' identity, provided data and consuming applications. But especially in medium-sized and large organizations, data silos are still commonplace[1]. As for data management, relational databases (RDBs) are the most prevalent type of data management system in practice[2], having been around for decades. Supporting RDBs as data sources is therefore a must for any data integration endeavor [2], for example when adopting the Solid Protocol. When doing so, two questions arise:

1. Data — How can we expose the RDB's data representation in *records* and *relations* using Solid's data representation in RDF-based *Web resource* and *collection/Linked Data Platform container resource* representations?
2. Identity — How can we expose the RDB's Identity and Access Management (IAM) that gives permissions to the RDB's *users* and *roles* using Solid's access control mechanism that gives permissions to WebIDs for *agents* and *groups*?

Answering these questions would then allow Solid Apps to consume data from existing RDBs.

To bridge this gap between legacy RDBs and the Solid Protocol, we follow the Ontology-Based Data Access (OBDA) approach [3]: We use the mapping language R2RML [4] to map between the data that follows a relational schema into an RDF dataset with RDF terms, thereby creating

The 1st Solid Symposium Poster Session, co-located with the 2nd Solid Symposium, May 02 – 03, 2024, Leuven, Belgium

lukas.kubelka@student.kit.edu (L. Kubelka); benjamin.meyjohann@student.kit.edu (B. Meyjohann); braun@kit.edu (C. H.-J. Braun); tobias.kaefer@kit.edu (T. Käfer)

0009-0008-1828-5207 (L. Kubelka); 0009-0000-1623-212X (B. Meyjohann); 0000-0002-5843-0316 (C. H.-J. Braun); 0000-0003-0576-7457 (T. Käfer)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1]https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-data-driven-enterprise-of-2025

[2]https://db-engines.com/en/ranking_trend

CEUR-WS.org/Vol-3947/short4.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

a virtual (i. e. not materialised) layer on top of the actual data. This RDF dataset is then exposed according to the Solid protocol. Internally, HTTP requests [5] according to the Solid Protocol are turned into SPARQL queries over the virtual layer (cf. the SPARQL Graph Store Protocol [6]), which in turn are translated into SQL queries over the actual data layer using the defined R2RML mappings [7].

We apply OBDA to both raw data and to roles and permissions as stored in the RDB. The RDB's raw data and corresponding tabular information are transformed to be accessible via the Solid Protocol. Roles are mapped to groups of WebIDs [8] and corresponding permissions are transformed to Web Access Control rules [9] for checking authorization in the Solid Protocol.

We present our work's foundations, our architecture, and how we lift data and identities.

2. Preliminaries

Relational Databases (RDBs). RDBs organise data in `tables` whose structure and properties are specified in a table's `schema`. An SQL `view` is a virtual table that is based on the result of a SQL query. A `role` in an RDB is a collection of privileges or permissions to perform specific actions or operations on database objects.

Ontology-based Data Access (OBDA). OBDA aims to bridge the gap between an organization's data layer containing heterogeneous data sources [3]. The idea is to build a mediator access layer [10] using a shared conceptualization of a common domain of interest, i.e. an ontology [11], for users to interact with [12]. Technical details about the underlying data sources are abstracted away [7] and data access across data sources is homogenised: One way realizing this approach is Ontology-Mediated Query Rewriting [7], where SPARQL queries to the conceptual mediator layer, e.g. a SPARQL endpoint, are re-formulated into database specific queries, on the data access layer, e.g. SQL queries to a legacy RDB. This re-writing of queries is enabled by declarative mappings between RDF and relational database schemas: The RDB2RDF Mapping Language (R2RML) [4] is the W3C Recommendation for expressing such mappings. An R2RML mapping consists of one or more triples maps that are applied to every row of a logical database table. They express the rules by which zero or more RDF triples are generated from each such row. A logical table is either an existing database table or view, or it defines a virtual view (R2RML view) through a valid SQL query.

The Solid Protocol. The Solid Protocol [1] is a bundle of specifications covering agent identification, authentication, authorization and data interaction. For agent identification, Solid relies on WebIDs [8], a HTTP URI that identifies an agent. For agent authentication, Solid relies on Solid-OIDC [13], a modified version of OpenID Connect [14]. For agent authorization, Solid relies on specifying access control rules following the Web Access Control (WAC) specification [9] or the recently introduced but not widely adopted Access Control Policies (ACP) [15]. Using these access control rules, once an agent is authenticated at a Web server or service, the server or service will determine if the agent is allowed to proceed with a certain action on data. Finally, for data interaction, Solid borrows from the Linked Data Platform (LDP) [16], effectively extending LDP with access control mechanisms. LDP specifies a RESTful [17] resource interface

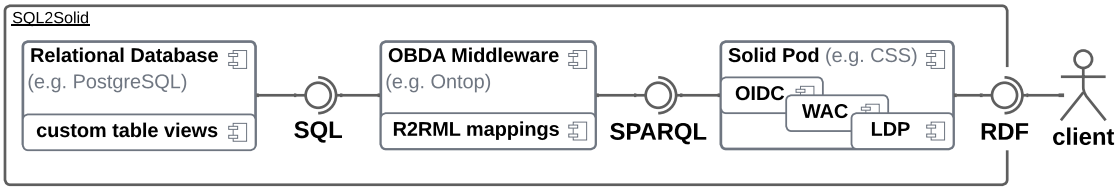


Figure 1: UML component diagram of the solution architecture.

and defines behaviour and RDF representation of collection resources as LDP Containers. That is, Solid adopts the document-centric style of data management from LDP, where (information) resources, i.e. documents or containers, are contained in containers. Solid does not, however, specify how the actual data is to be stored, e.g. in a triple store or a relational database. A Web server that implements the Solid Protocol is called a Solid Pod (Personal Online Datastore).

3. System Architecture

To leverage data from a legacy RDB, we combine the components presented in Section 2. The RDB provides access to legacy data, roles and permissions via an SQL interface, which is consumed by an OBDA middleware. The OBDA middleware uses R2RML mappings to translate incoming SPARQL queries into SQL queries and to lift the SQL results to SPARQL results, e.g. RDF graphs. The SPARQL interface provided by the middleware is consumed by a Solid Pod. Following the Solid Protocol, the Pod provides an LDP-based data interaction interface under access control to clients. Figure 1 illustrates this system architecture.

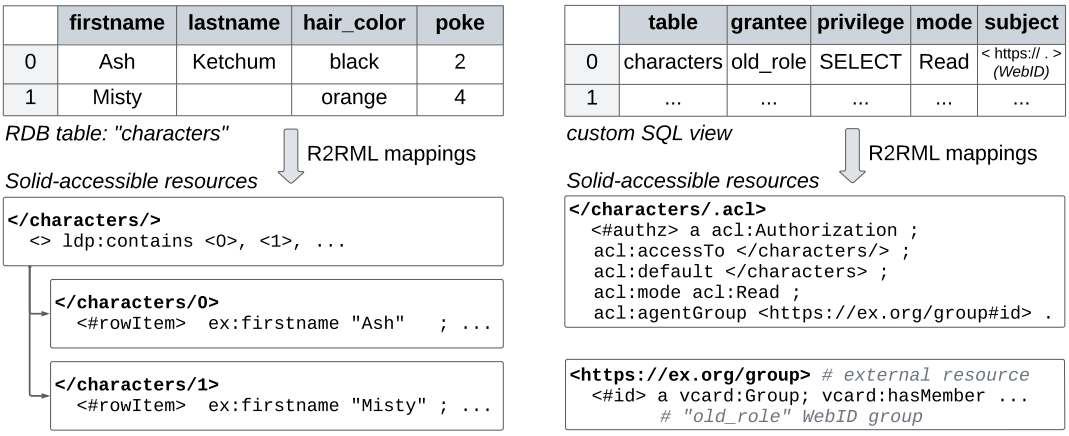
Consider an HTTP request being handled by this system architecture: Upon receiving an HTTP request, the Solid Pod authenticates the client according to Solid-OIDC. After successful authentication, the Pod generates SPARQL queries to (a) check the Web Access Control rules on the requested resource and then (b) to retrieve the requested data. The ODBA middleware receives the SPARQL requests, translates them into SQL queries using the R2RML mappings, and issues the SQL queries to the RDB. Lifting the SQL query results using the mappings again, the middleware provides the results of the Pod's SPARQL queries. The Pod checks if the client is authorized according to the retrieved access control rules, and grants or denies data access.

In our demo, see Figure 1, we use PostgreSQL[3] as the relational database, Ontop[4] with R2RML mappings [4] for OBDA, and the Community Solid Server (CSS)[5] for the Solid-based interaction interface that is publicly exposed to clients via the Web.

4. From Legacy Data, Roles and Permissions to the Solid Protocol

In this section, we present how the mapping works to use RDB legacy data in the Solid Protocol.

[3]https://www.postgresql.org/
[4]https://ontop-vkg.org/
[5]https://github.com/CommunitySolidServer/CommunitySolidServer



(a) From RDB table to LDP Container and Resources. (b) From RDB roles to Web Access Control.

Figure 2: Lifting RDB data to the Solid Protocol.

Data. To lift the relational legacy data to RDF, usual R2RML mappings are defined using domain-specific ontologies. To also make them accessible using the Solid Protocol, we also map the structural information of the table itself to LDP Containers and Resources.

Consider the following example: The RDB contains a table named `characters`, where its rows represent data on characters of Pokémon trainers. We choose to define the R2RML mappings such that the table is interpreted as an LDP Container and the single table rows as resources contained in this container, see Figure 2a.

The container is made accessible through the Solid Pod using the table's name as the path of the URI, e.g. `http://example.org/characters/`. The resources corresponding to table rows are accessible using their row ID, e.g. `http://example.org/characters/{rowID}`. Using different mapping approaches is possible, of course, depending on the use case at hand.

Roles and Permissions. To also re-use legacy permissions with the Solid Protocol, the RDB's existing roles are interpreted as groups of users having the same set of access privileges: RDB roles are mapped to URIs that identify groups of WebIDs. Managing members of these WebID groups is then independent from the RDB. Assigned permissions of that group are still specified by the RDB. In this way, any WebID may be assigned a legacy RDB role. This approach decouples identity of users from the data source they access.

Under the hood, the Role-WebID mapping is defined in a custom SQL view using a simple SQL query. This custom SQL view is then used in R2RML mappings to lift the legacy roles and permissions to be made accessible as *.acl* files via the Solid-based interface, e.g. `http://example.org/characters/.acl`. Using the R2RML mappings, the middleware transforms the RDB's tabular data about a role's permitted operations on a specific RDB table into an RDF graph representing a resource's access control list, see Figure 2b.

5. Conclusion

We presented an approach to integrate legacy RDBs with the Solid Protocol: We set up OBDA to the raw data as well as to roles and permissions for usage according to the Solid Protocol. While our solution currently only supports reading legacy data under pre-existing roles and permission; supporting writing new data including access rules is ongoing work.

References

[1] S. Capadisli, T. Berners-Lee, R. Verborgh, K. Kjernsmo, Solid Protocol, Version 0.9.0, W3C Solid Community Group, 2021. URL: https://solidproject.org/TR/protocol.

[2] M. Rodriguez-Muro, M. Rezk, Efficient SPARQL-to-SQL with R2RML mappings, Journal of Web Semantics 33 (2015) 141–169.

[3] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, in: Journal on data semantics X, Springer, 2008, pp. 133–173.

[4] S. Das, S. Sundara, R. Cyganiak, R2RML: RDB to RDF Mapping Language, W3C Recommendation, W3C, 2012. URL: https://www.w3.org/TR/r2rml/.

[5] R. Fielding, J. Reschke, Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, Internet Standards Track document, IETF, 2014. URL: https://www.ietf.org/rfc/rfc7230.txt.

[6] C. Ogbuji, SPARQL 1.1 Graph Store HTTP Protocol, W3C Recommendation, W3C, 2013. URL: https://www.w3.org/TR/sparql11-graph-store-protocol/.

[7] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: A survey, International Joint Conferences on Artificial Intelligence, 2018.

[8] A. Sambra, H. Story, T. Berners-Lee, WebID 1.0 - Web Identity and Discovery, W3C Editor's Draft, W3C, 2014. URL: https://www.w3.org/2005/Incubator/webid/spec/identity/.

[9] S. Capadisli, Web Access Control, Editor's Draft, W3C Solid Community Group, 2022. URL: https://solid.github.io/web-access-control-spec/.

[10] G. Wiederhold, Mediators in the architecture of future information systems, Computer 25 (1992) 38–49.

[11] N. Guarino, D. Oberle, S. Staab, What is an ontology?, Handbook on ontologies (2009) 1–17.

[12] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, Ontop: Answering SPARQL queries over relational databases, Semantic Web 8 (2017) 471–487.

[13] A. Coburn, elf Pavlik, D. Zagidulin, Solid-OIDC, W3C Editor's Draft, W3C Solid Community Group, 2022. URL: https://solidproject.org/TR/oidc.

[14] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, C. Mortimore, OpenID Connect Core 1.0, Final Specification, 2014. URL: https://openid.net/specs/openid-connect-core-1_0.html.

[15] M. Bosquet, Access Control Policy (ACP), Editor's Draft, W3C Solid Community Group, 2022. URL: https://solid.github.io/authorization-panel/acp-specification/.

[16] S. Speicher, J. Arwe, A. Malhotra, Linked Data Platform 1.0, W3C Recommendation, W3C, 2015. URL: https://www.w3.org/TR/ldp/.
[17] R. T. Fielding, Architectural styles and the design of network-based software architectures, Ph.D. thesis, University of California, Irvine, CA, USA, 2000.