Towards Data Dependency-Preserving Integration of Relational Databases via the Solid Protocol and Shape Trees

Lukas Kubelka[1], Tobias Käfer[1]

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

Abstract

We present a method to construct write-also Ontology-based Data Access (OBDA) systems for Relational Databases (RDBs) based on the Solid Protocol and Shape Trees. Further, our method automates the process of building user interfaces that offer the functionality to read and write relational data from RDBs in a unified way. Our method could act as a building block towards data dependency-preserving data integration systems.

Keywords

Linked Data, Solid, SQL, Write-also OBDA, SHACL, Shape Trees

1. Introduction

Relational databases (RDBs) are the most prominent form of data repository[1] and thus play a central role in today's data management activities. Although containing large amounts of information, they often remain data silos[2] with the potential value that comes from sharing and combining their data being neglected. One major effort when integrating RDBs is to preserve data dependencies. Meaning, while consolidating data to obtain a coherent view, consuming applications must remain intact and have access to up-to-date data throughout and beyond the data integration endeavor. Two integral properties for data integration systems arise:

1. Data Virtualization — Integration of data happens without transforming or moving it to another data repository.
2. Single Source of Truth (SSOT) — Modification of integrated data equals modification of source data repositories.

Such systems for integrating and sharing data can be built by leveraging the Solid Protocol [1] and further Semantic Web technologies. Initially, by focusing on the interplay between Solid and a single RDB.

A previous demonstration [2] presented how the property of "data virtualization" can be realized through a read-only Ontology-based Data Access (OBDA) approach based on R2RML

The 2nd Solid Symposium Poster Session, co-located with the 3rd Solid Symposium, April 24 – 25, 2025, Leiden, Netherlands

lukas.kubelka@kit.edu (L. Kubelka); tobias.kaefer@kit.edu (T. Käfer)

0009-0008-1828-5207 (L. Kubelka); 0000-0003-0576-7457 (T. Käfer)

[1]https://db-engines.com/en/ranking

[2]https://www.oracle.com/database/data-silos/

mappings [3] that makes RDB data accessible as virtual knowledge graphs (VKGs) [4]. This paper focusses on SSOT and propose an approach for write-also Ontology-based Data Access (OBDA) leveraging Shape Trees [5] and SHACL shapes [6] which allows propagating modifications of such VKGs back to the RDB. Our framework describes relational databases as container-based hierarchies of resources and connects database tables with semantics-preserving SHACL shapes that reflect SQL integrity constraints. Finally, these shapes can be consumed by SHACL-compliant tools, e.g. shacl-form[3], to provide a standards-based way for user interfaces to read and write relational data via the Solid Protocol.

In the following, we present our work's preliminaries (cf. Section 2) and our write-also OBDA framework for imposing the property of SSOT via Solid and Shape Trees (cf. Section 3).

2. Preliminaries

Solid Protocol. The Solid Protocol [1] defines a RESTful [7] API for sharing information based on the Linked Data principles [8] and other Web technologies, e.g. WebID [9], Solid-OIDC [10] and WAC [11]. Central to Solid's data management interface is the concept of *container-based* resource hierarchies which is based on the Linked Data Platform (LDP) [12] recommendation. *Containers* are resources that contain further resources; their state is described using RDF.

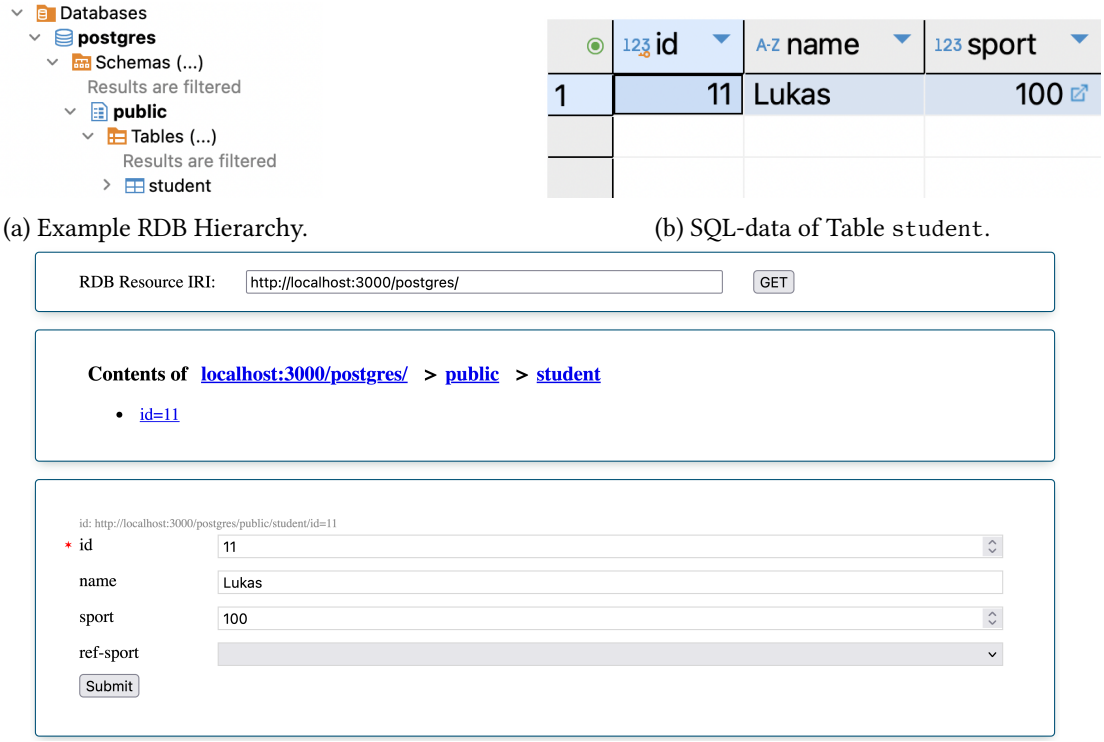
Shapes Constraint Language (SHACL) and Shape Trees. SHACL [6] is a language for defining constraints on the contents of RDF graphs. Such constraints are called *shapes* and make up RDF graphs called *shapes graphs*. Shapes graphs are used to validate target RDF graphs as conforming to the described shapes. Target RDF graphs are called *data graphs*.

Shape trees [5] describe the expected layout of a *container-based* resource hierarchy. *Shape tree managers* map resources to shape trees. A resource is called a *managed resource* when a shape tree manager is associated with it. A shape tree has an associated *shape*. Managed resources conform to the shape of their associated shape trees.

Ontology-based Data Access (OBDA). Ontology-based Data Access is a paradigm for creating unified data access layers on top of heterogeneous data sources by utilizing Semantic Web technologies. OBDA systems typically follow a mediator-based approach [13] by mapping source data to RDF using a mapping language, e.g. R2RML [3] as a first step. Afterwards, SPARQL queries are rewritten to a query language understood by the source data repository, e.g. SQL in the case of RDBs. The framework of OBDA can be viewed from a *read-only* and *write-also* perspective [14]. Read-only OBDA systems are only concerned with exposing data from source repositories in terms of coherent domain-specific ontologies. Write-also systems are further capable of propagating modifications to this global view over the data back to the source repositories. The research landscape mainly covers read-only functionality [15].

Relational Databases (RDBs). RDBs store and provision data in terms of the *relational view of modeling information* [16]. The SQL standard [17] is based on this view and provides a

[3] https://github.com/ULB-Darmstadt/shacl-form



(a) Example RDB Hierarchy.

(b) SQL-data of Table `student`.

(c) Custom Solid Client for Reading and Writing SQL-data based on SHACL Forms Generator.

Figure 1: Reading and Writing Relational Data via the Solid Protocol.

language used for access to RDBs. SQL organizes data through *catalogs*, which are collections of SQL-*schemas* that contain several *schema objects*. One central schema object is a *table* which holds SQL-*data* as a collection of *rows* that assign values to the table's *columns*. A table has a *row type*, which is a sequence of columns. A row is a sequence of column values. Tables and columns may be associated with *integrity constrains* that restrict the set of possible column values with respect to all rows in a table.

3. Write-also Ontology-based Data Access via Solid

The SQL standard has the notion of hierarchical relations of catalogs that contain schemas containing tables to organize SQL-data presented as rows. We expose these components in terms of nested `ldp:Container` resources which ultimately contain row `ldp:RDFSource` resources and assign them to Shape Trees that mirror their containment relationships in the database. Figure 1a shows an example of such hierarchy. It is comprised of the catalog `postgres`, the schema `public` and the table `student` containing the row resources shown in Figure 1b. The `student` table's row type is made up of the three columns `id`, `name` and `sport`. We name these resources according to a URI template[4] that is reflected in the breadcrumbs navigation in the

[4] `http://localhost:3000/{catalog_name}/{schema_name}/{table_name}/{row_identifier}`

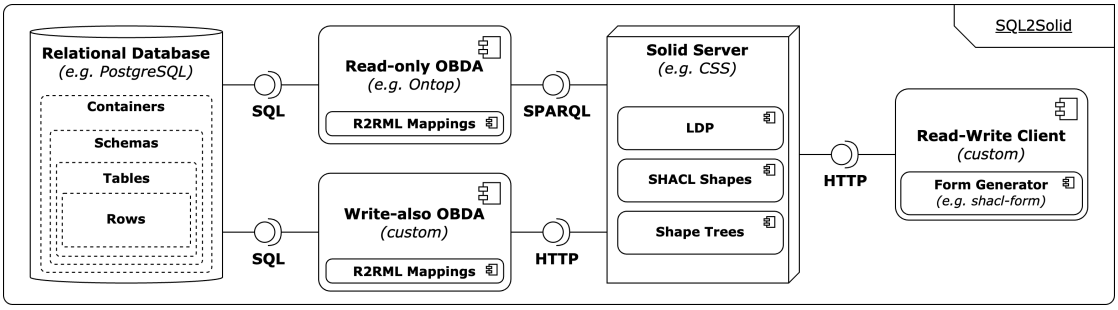


Figure 2: Component Diagram of our Solution Architecture.

middle panel of our Solid client (cf. Figure 1c). This panel is used to request resources and traverse the RDB's container hierarchy. An important role in our framework play the associated SHACL shapes of table Shape Trees. They drive the main interaction between Solid and the RDB as they restrict the modification possibilities of table row resources in a straight-forward way and form the basis for the automated generation of read-write user interfaces in our demo. An example for this can be seen in the bottom panel of our client. Based on the associated SHACL shape of the viewed row resource, an HTML form is generated on-demand that lets users modify the row referent in the source database. In the form it can be seen that the shapes must not only reflect the table's row type, but also a table's column and integrity constraints. For example, the column `id` reflects the table's primary key and is thus marked as a required field in the form. We generate such shapes through constraint rewriting [18].

Figure 2 illustrates the architecture of our approach. Our Solid client consumes and modifies database resources that get exposed by a Solid server. The server propagates read-write requests to the respective middleware, which consumes (or updates) SQL data in the RDB. To this end, the middleware either sends a message to the read-only OBDA middleware from [2], which translates SPARQL queries to SQL SELECT statements according to [19], or it POSTs to the write-also middleware that constructs an SQL INSERT, UPDATE or DELETE statement.

In our demo, we use PostgreSQL[5] as the RDB, the Community Solid Server (CSS)[6] as the Web server implementing the Solid-based Linked Data API and shacl-form[7] as the main interaction component in our Solid app for reading and writing relational data.

4. Conclusion

We presented an approach for write-also Ontology-based Data Access based on the Solid Protocol and further Semantic Web technologies. Our approach offers the generation of read-write UIs to existing RDBs by leveraging Shape Trees and SHACL shapes and could be used as a fundamental building block for data dependency-preserving data integration systems, i.e. system offering data virtualization and a single source of truth strategy for data repositories.

[5] https://www.postgresql.org/
[6] https://github.com/CommunitySolidServer/CommunitySolidServer
[7] https://github.com/ULB-Darmstadt/shacl-form

References

[1] S. Capadisli, T. Berners-Lee, K. Kjernsmo, Solid Protocol, Version 0.11.0, W3C Solid Community Group, 2024. URL: https://solidproject.org/TR/protocol.

[2] L. Kubelka, B. Meyjohann, C. H.-J. Braun, T. Käfer, Sql2solid: Exposing data and permissions from relational databases via the solid protocol, in: Proceedings of the 2nd Solid Symposium, Leuven, Belgium, 2024.

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

[4] G. Xiao, L. Ding, B. Cogrel, D. Calvanese, Virtual knowledge graphs: An overview of systems and use cases, Data Intelligence 1 (2019) 201–223.

[5] E. Prud'hommeaux, J. Bingham, Shape Trees Specification, Editor's Draft, W3C, 2021. URL: https://shapetrees.org/TR/specification/.

[6] H. Knublauch, D. Kontokostas, Shapes Constraint Language (SHACL), W3C Recommendation, W3C, 2017. URL: https://www.w3.org/TR/shacl/.

[7] R. T. Fielding, Architectural styles and the design of network-based software architectures, Ph.D. thesis, University of California, Irvine, CA, USA, 2000.

[8] T. Berners-Lee, Linked data, 2006. URL: https://www.w3.org/DesignIssues/LinkedData.

[9] 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/.

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

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

[12] S. Speicher, J. Arwe, A. Malhotra, Linked Data Platform 1.0, W3C Recommendation, W3C, 2015. URL: https://www.w3.org/TR/ldp/.

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

[14] G. De Giacomo, D. Lembo, X. Oriol, D. F. Savo, E. Teniente, Practical update management in ontology-based data access, in: C. d'Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudré-Mauroux, J. Sequeda, C. Lange, J. Heflin (Eds.), The Semantic Web – ISWC 2017, Springer International Publishing, Cham, 2017, pp. 225–242.

[15] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, M. Zakharyaschev, Ontology-based data access: A survey, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 5511–5519.

[16] E. F. Codd, A relational model of data for large shared data banks, Communications of the ACM 13 (1970) 377–387.

[17] J. Melton, Iso/iec 9075-1 information technology-database languages-sql-part 1: Framework (sql/framework), ISO/IEC 2016 (2016) 9075–1.

[18] R. B. Thapa, M. Giese, A source-to-target constraint rewriting for direct mapping, in: Proceedings of the 20th International Semantic Web Conference, ISWC 2021, Springer

Nature Switzerland, Virtual Event, 2021, pp. 21–38.
[19] C. Ogbuji, SPARQL 1.1 Graph Store HTTP Protocol, W3C Recommendation, W3C, 2013. URL: https://www.w3.org/TR/sparql11-graph-store-protocol/.