

Karlsruhe Institute of Technology

Institute of Applied Informatics and Formal Description Methods (AIFB)

Web Science

SQL2SOLID - Securely Exposing Legacy SQL Data as Virtual RDF Graphs

Seminar: Linked Data and the Semantic Web Authors: Lukas Kubelka and Benjamin Meyjohann

Introduction

Most medium-sized and large organizations deal with the issue of data silos. These silos, while containing semantically related data, often lead to redundancies and inconsistencies, posing a significant challenge in data integration.

The desirable target state would be an integrated data ecosystem, enabling organizations to make informed decisions based on accurate and up-to-date information. A key aspect of this challenge is the integration of Identity and Access Management (IAM), which in legacy systems is typically performed separately for each data source.

We aim to propose an accessible and adaptable approach for integrating legacy relational data sources into the Social Linked Data (Solid) ecosystem. Solid establishes a holistic data ecosystem for reading

:TablesPrivilegesView rr:sqlQuery """ select *, (SQL query used by the case privilege_type :TablesACLMap to grant read when 'SELECT' then 'Read' access to the WebID when end the RDB role 'soliduser' has) as access_mode, (SELECT rights. Returns a case grantee row for each table. when 'soliduser' then 'https://uzquk.solid.aifb.kit.edu/ public/groups.ttl#soliduser' Mapping that creates end an ACL resource for) as access_subject every table in the from information_schema.table_privileges RDB. The resource where table_schema = 'public'; contains WebIDs and their access rights.

:TablesAclMap a rr:TriplesMap;

and writing linked data in a secure and privacy respecting manner.

For that we utilize the Ontology-based Data Access framework and provide R2RML mappings that enable this transition. Next to the mappings of the actual data, our main contribution lies in the mapping of Rolebased Access Control (RBAC) policies to Web Access Control (WAC) ACL resources.

We leverage the Solid protocol for access control and employ R2RML mappings for data conversion. This method enables parallel operation of both legacy and linked systems, crucial for gradual without migration disrupting current operations.

To address the challenge of converting and managing data, we considered two primary strategies: materialization, where data is preconverted and stored, and virtuali-

on-demand. We chose virtualization due to its real-time Docker Contain synchronization capabilities between lega-«component» cy and linked data, Postgresql DB minimizing storage requirements and avoiding the need for frequent re-materialization in response to data updates.

zation, which converts data

For the practical implementation of our theoretical framework, we selected PostgreSQL as our relational database for its open-source nature and widespread adoption. Ontop was chosen as the virtual knowledge graph system for its ability to provide a SPARQL endpoint for just-in-time data transformation, interfacing seamlessly with both

Approach

Community Server was selected to implement the Solid protocol, offering a comprehensive ecosystem for secure access control over linked data. The resulting architecture of our solution can be seen in Fig. 1.

The Solid server configuration is critical for routing requests to the appropriate data sources. We use RegexRule configurations to direct all RDF graph store queries to the SPARQL endpoint and other

requests to the server's file storage. This ensures that requests for RDF data are processed through the virtualization layer, maintaining real-time data fidelity and applying User Agent the necessary access controls as defined by the R2RML mappings as shown in Fig. 2. Additionally, the R2RML mappings account for Solid's requirement of meta-resources for each resource accessed.

By doing so, we seamlessly translate existing database roles into ACL resources, minimizing user effort in the migration process.

This method integrates PostgreSQL databases with the capabilities of Ontop, providing an accessible

rr:logicalTable :TablesPrivilegesView; Solid rr:subjectMap [Community rr:template "http://localhost:3000/rdf-graph-Server requests store/{table_name}/.acl"; ACL resources rr:class acl:Authorization; using '.acl'. rr:graphMap [rr:template "http://localhost:3000/rdf-graphstore/{table_name}/.acl" This will be dynamically replaced with the rr:predicateObjectMap [WebID from the SQL rr:predicate acl:agentGroup; query for each table. rr:objectMap [rr:column "access_subject"; rr:termType rr:IRI This is dynamically replaced by the 1; access mode from the SQL query rr:predicateObjectMap [for each table. Currently, only 'Read' rr:predicate acl:mode; can be used. rr:objectMap [rr:template "http://www.w3.org/ns/auth/acl#{access_mode}"; '{table_name}' is dynamically 1; replaced by the table name rr:predicateObjectMap [of each query row. rr:predicate acl:accessTo; rr:objectMap [rr:template "http://localhost:3000/rdf-graphstore/{table_name}/" 1; rr:predicateObjectMap [rr:predicate acl:default; SQL query used by rr:objectMap [rr:template "./"] mappings for metadata resources that are required by Solid Community Server. :RDFGraphStoreView rr:sqlQuery SELECT * FROM information_schema.tables WHERE table_schema = 'public'; """ Metadata mapping for the resource itself. :TablesMetadataMap a rr:TriplesMap; rr:logicalTable :RDFGraphStoreView; rr:subjectMap [rr:template "http://localhost:3000/rdf-graphstore/{table_name}/"; rr:graphMap [rr:template "meta:http://localhost:3000/rdf-graphstore/{table_name}/" 1; rr:predicateObjectMap [Solid Community Server rr:predicate rdf:type; always requests metadata



Solution Architecture

PostgreSQL and the Solid protocol. The Solid

Evaluation

Our work integrates Role-based Access Control (RBAC) with Solid WebIDs, requiring minimal additional user knowledge and maintaining pertable access. However, it lacks the more finegrained control of row-level-security and perresource access control of Solid.

By employing virtualization over materialization, significantly reduces storage demands, guarantees real-time synchronization between legacy and linked data, and ensures accessibility by embedding the data in the Solid ecosystem, making it well-suited especially for step-by-step adaptation from legacy relational data to linked data when running both systems in parallel.

solution for exposing legacy relational data as linked data within the Solid ecosystem.

Conclusion

In retrospective, our approach for migration of legacy relational data to linked data has revealed some promising findings for security and enhanced accessibility. It allows for gradual migration and does not require a clear cut because of its capability to keep the data itself in addition to the role-based access control systems of RDBs and Solid's WebIDs in sync.

Nonetheless, there are still issues open for future research. These include writing and updating operations at the linked data level synchronizing to the legacy relational data. Furthermore, we expect a script for generation of the R2RML mappings to be the next big step towards even more accessibility.

- rr:predicateObjectMap [rr:predicate rdf:type; rr:object ldp:BasicTable
- rr:predicateObjectMap [rr:predicate rdf:type; rr:object ldp:Resource] .

rr:object ldp:Table

];

1;

Metadata mapping for the ACL resource.

www.kit.edu

resources using 'meta:'.

:TablesAclMetadataMap a rr:TriplesMap; rr:logicalTable :RDFGraphStoreView; rr:subjectMap [rr:template "http://localhost:3000/rdf-graphstore/{table_name}/.acl"; rr:graphMap [rr:template "meta:http://localhost:3000/rdf-graphstore/{table_name}/.acl" 1;

rr:predicateObjectMap [rr:predicate rdf:type; rr:objectMap [rr:constant rdf:Resource

Fig. 2: The most important R2RML Mappings

KIT – The Research University in the Helmholtz Association