On Constructing Enterprise Knowledge Graphs Under Quality and Availability Constraints

Matthew Kujawinski[1(✉)], Christophe Guéret[1], Chandan Kumar[3], Brennan Woods[3], Pavel Klinov[2], and Evren Sirin[2]

[1] Accenture Labs, San Jose, USA
{matthew.kujawinski,christophe.gueret}@accenture.com
[2] Stardog, Arlington, USA
{pavel,evren}@stardog.com
[3] Accenture, Dublin, Ireland
{chandan.w.kumar,brennan.woods}@accenture.com

Abstract. Knowledge graph technologies have proven their applicability and usefulness to integrate data silos and answer questions spanning over the different sources. However the integration of data can pose some risks and challenges (security, audit needs, quality control, ...). In this paper we abstract from two client use-cases, one in the banking domain and one in the pharmaceutical domain, to highlight those risks/challenges and propose a generic approach to address them. This approach leverages Semantic web technologies and is implemented using Stardog.

Keywords: Data integration · SHACL · Knowledge graphs

1 Introduction

Combining different data silos into an Enterprise Knowledge Graph (EKG) delivering a 360 degrees view of the data held across those silos is becoming a key asset for several major industries. Be it to better track information about customer across different departments or integrate data from a factory line [7], knowledge graphs provide a streamlined access to data and enable advanced querying capabilities [6].

Beyond the common need of creating a knowledge graph from data found in different silos, Stardog and Accenture found some specific requirements when working on client projects. We hereby report in particular about the needs emerging from two use-cases, one applied to a banking client and the other applied to a pharmaceutical company.

This paper aims at presenting the approach to tackle the requirements and give some example information from the use cases, under the limits of what the clients agreements make it possible for us to disclose. To be more specific, we consider the contribution of this paper as follows:

A. Hotho et al. (Eds.): ISWC 2021, LNCS 12922, pp. 699–713, 2021.
https://doi.org/10.1007/978-3-030-88361-4_41

Paper Main Contributions

- The synthesis of requirements expressed from different industrial deployments of knowledge graphs and the proposal for a generic architecture for addressing them;
- The implementation report of the architecture and its usage for two different client case-studies currently in production stage;
- Some lessons learned and suggestions for future work on Semantic Web technologies and standards which could contribute to better addressing the highlighted challenges.

The remainder of this paper first overviews related work in Sect. 3 to discuss how current best practices and known approaches address our requirements detailed in Sect. 2. The implementation we have of the approach we recommend is then described in Sect. 4 and our example use cases in Sect. 5. We finally conclude on our lessons learned and offer suggestions for further work in Sect. 6.

2 Platform Requirements

The general requirement expressed is the integration of data from different sources in order to solve business intelligence needs spanning over several of those sources. These specific requirements emerging, at least partially, from the two concrete EKG deployments presented in this paper and some others highlight some specific business needs sometimes overlooked by the Semantic Web research community. More specifically we have identified and faced additional constraints around security and entitlements (Requirement 1 - R1), graph preservation (R2), data availability (R3) and quality (R4):

R1. Security and entitlements: Creating an integrated knowledge representation can represent a significant risk for cyber attacks as all the information that would otherwise be found in different sources, each having to be attacked one after the other, is now readily available from one source only and already semantically integrated. This key feature of an EKG could turn into a major flaw under the wrong usage context. Besides the cyber-threat risk, combining data can also lead to a risk of infraction against regulations such as the European GDPR by creating Personally Identifiable Information (PII) data when connecting information otherwise acceptable when consumed separately. In order to mitigate both risks we need to ensure that entitlements to the original data sources are not superseded by the graph: someone not having access to a data found in a particular silo should not have access to this data once integrated into the EKG.

R2. Graph preservation: Businesses using an EKG for activities relevant to regulatory frameworks (for instance, GDPR) may be asked to produce a copy of the graph at a particular point in time. Those time-stamped archives of the production data need to be archived and preserved for a couple of years in order to enable audit enquiries. To a lesser extent, the software preservation of the tools used to query the data should also be considered.

R3. Data availability: The two case studies reported on in this paper depend on the availability of data close to being the live equivalent of the data sources. We also found out cases where the data from the source was deemed too sensitive to be exported to disk via traditional ETL processes and had to be queried live from the source only, and then be consumed only in-memory.

R4. Knowledge Graph Data Quality: The data acquired from the different sources is expected to come as inconsistent because of known issues, and should otherwise be treated with caution in order to avoid mistakes during the construction of the EKG. As the aim for constructing this graph is to enable answering business questions spanning over several sources it is critical that the information surfaced is as accurate as possible. This last requirement comes in a possible opposition to the previous one, R3, as the data needs to be as fresh as possible and yet undergo a verification step to check that no erroneous information might be consumed by end-users. For example, each clinical study should have a single stage information such as Phase I or Phase II indicating the current phase. If the information about clinical study is duplicated in multiple data sources there can be discrepancies about the current stage. We would like to detect and correct these errors before the information is shown to end users.

The Semantic Web research community, the W3C and the industry ecosystem all together proposed solutions addressing some of those requirements. Our work and contribution consisted in identifying the standards best fit for each task and connect things together in a sound pipeline. In the following section we review the relevant technologies and related work.

3 Related Work and Technologies

The approach described in this paper relates to different interests of the Semantic Web research community. We hereafter name relevant work in the domain of secure data access, preservation, knowledge graph construction, and data quality control.

Security and Entitlements (R1)

The data access for our platform had to be easy and secured. In terms of ease of access, consuming the data from a knowledge graph can be done with languages such as SPARQL, GraphQL or Gremlin but all those require getting familiar with the specific aspects. The specifications for Linked Data Platform[1] and Linked Data API[2] offers a level of simplifications by enriching the options for de-referencability. GRLC [8], for instance, implements the latter to wrap SPARQL queries into more intuitive RESTful API calls. But neither of these tackle the aspect of security and entitlements. In fact, in terms of security, and

[1] https://www.w3.org/TR/ldp/.
[2] https://github.com/UKGovLD/linked-data-api.

as noted in the requirement R1, we need to ensure that the level of access granted at the data source level is reflected at the EKG level. So the result of the queries is expected to differ based on the credentials of the client emitting those queries.

As explained in details by Kirrane *et al.* in [5], there is now a large variety of access control models and standards which have been proposed and applied to Linked Data. Our work can be seen as fitting in the Role Based Access Control (RBAC) model by controlling the access to different named graphs based on the roles associated to the credentials of the user.

Graph Preservation (R2)

Because of its Web-based and dynamic nature, the preservation of Linked Data represents a challenge [1]. We can highlight here the role of the Memento protocol [10] to enable the access to historical descriptions of resource descriptions, however the focus of our approach is closer to saving static dumps of entire graphs. The objective for our use-case is to freeze the knowledge graph in time to eventually later on re-play what consuming application would have had access to. In this respect, HDT [2] is a more relevant work enabling storing large graphs in a compressed and queriable way.

Data Availability (R3)

The Semantic Web community has been active proposing architectures to transform silos into Knowledge graphs. Acknowledging the strong presence of RDBMS systems R2RML came in early and is featured in Sequeda *et al.* "pay as you go" methodology [9] for building EKGs using an Ontology-based Data Access approach. R2RML is a technology part of the virtualization of Knowledge graphs [11] which contrary to rigid one-way mass ETL process introduces flexibility and dynamism in data access. Beyond R2RML, platforms such as Metaphactory [4] and Stardog[3] show how many more types of data sources can be virtualized with the same benefits.

KG Data Quality (R4)

The integration of data from different silos is likely to cause inconsistencies as with the example given earlier for the requirement R4. Assessing data quality has several dimensions [12] including, for instance, looking at the overall shape of the graph to find inconsistencies in it [3] but what is of interest to us here is not to check for quality in terms of potential value. Our focus is on detecting constraint violation with respect to how the data is expected to be shaped. One of the latest addition to the family of Semantic Web standards, SHACL, comes in handy there.

Close to our needs, Metaphactory [4] uses SHACL to control the quality of the data as it goes though the federated query layer "Ephedra" of the platform.

[3] https://www.stardog.com/categories/virtualization/.

This approach does not however tackle R4 as we expressed it because we need to identify errors before they get a chance to be consumed by downstream clients applications.

The architecture presented here in this paper builds upon and takes inspiration from this state of the art to stitch together a pipeline addressing the four requirements in scope. We hereafter report on the generic pipeline architecture and our implementation leveraging the capabilities of the knowledge integration platform Stardog.

4 Architecture and Implementation

The solution architecture is comprised of three main parts: data ingestion, reasoning, and API enabled querying. Our clients choose Stardog over other vendors largely because of its virtualization capabilities. We then moved on to implementing the other requirements with Stardog and external tools the platform natively integrates with. We would however like to highlight that those requirements expressed in the introduction and the pipeline we report on in this paper go beyond our specific use-cases and could be re-implemented by other vendors.

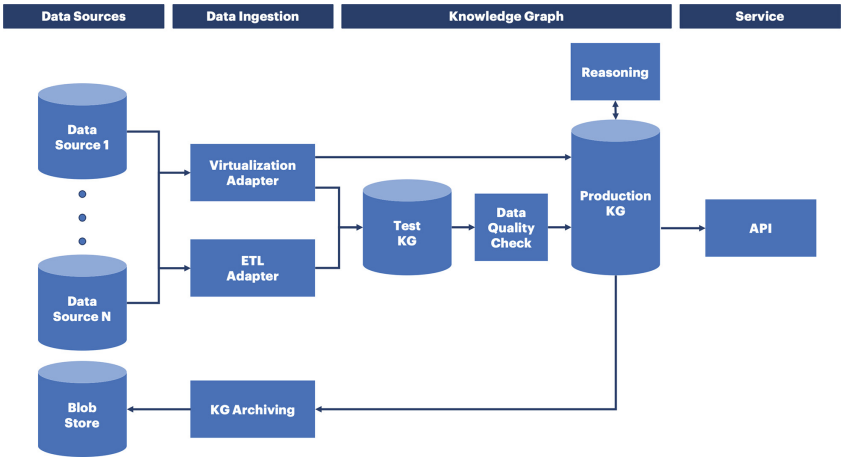

Fig. 1. The overall architecture and data flow

Figure 1 shows this architecture in term of functional blocks. One particularity to note is the materialization of some of the virtualized sources via a staging/production DB whilst some others are directly consumed live from the production DB. We hereafter go into the details of the implementation of each of those functional blocks and motivate our design choices.

4.1 Data Ingestion

The first set of functional parts of the pipeline concern the virtualization of the data sources. We use two parallel approaches to be able to tackle possible varied complexities in using the data sources: one is based on using native adapters from Stardog and the other leverages Apache Nifi as a tightly coupled ETL approach.

Virtualization adapter is one of the virtualization adapter supported by Stardog. We use the language SMS[4] in order to be able to cover for the full range of possible data sources, beyond the limitation to relational databases from R2RML. An SMS file is created to define a mapping from the data source to the knowledge graph.

ETL Adapter is there for cases where virtualization is not available or SMS is not expressive enough. We use Apache NiFi as a fall back option to turn the source data into triples.

Archiving of a named graph inside the production DB is done before a new version of this named graph is preserved along with metadata indicating when it had been used. This allows for auditing of previous data when looking for historical information. Highly sensitive data sources are by default excluded from this process to comply with not persisting the data they serve.

Data from all sources is sent first to a designated testing database under a specific named graph for each data source. We materialize it to enable a quality check and bypass some possible performance issues. Data sources with highly sensitive data are directly pushed to the production graph without being materialized, they are made available as-is via the virtualization adapter and do not get persisted.

A URI definition document and global ontologies are in place within the organization to ensure URIs are minted in a consistent way and that a common terminology is used to document the data.

Cron jobs and scheduled ETL pipelines in Apache NiFi are used to refresh data regularly, anywhere from daily to monthly based on the frequency of updates in the data sources.

4.2 Quality Control and Reasoning

It is important to validate the data and also allow for expressing inferred connections between data brought together in the knowledge graph. We implemented a balanced approach that capitalized on the benefits of Semantic Web technology without sacrificing data quality.

Quality check is a module looking at the data in the testing database and validating it against a set of shapes expressed in SHACL. Based on the report, the data is pushed into production or an alarm is raised. Checking SHACL

[4] https://docs.stardog.com/virtual-graphs/mapping-data-sources#sms2-stardog-mapping-syntax-2.

constraints is implemented via SPARQL queries and thus can handle both materialized and virtualized data. The former is more efficient since more optimizations are available, e.g. validating many RDF nodes at once with more complex queries. For the latter, Stardog uses a two-step procedure internally which first generates SPARQL queries for SHACL constraints and then rewrites them to SQL using the mappings. That process is transparent to the user. It has performance overhead which can be mitigated by using Stardog cache nodes[5].

Reasoning is done by Stardog's built-in reasoner which performs inferences on the data based on the ontology. Notably, reasoning in Stardog is based on query rewriting, not materialization, and thus is agnostic to whether the data is virtualized or ETL'ed. Reasoning also allows creating a level of abstraction between client applications and data sources so that queries do not break when data source schemas change.

Ontologies are edited using Protégé and maintained as dedicated assets ingested by the platform. Data is validated using the SHACL shapes and against the ontology terms, meaning all data is associated with a validated ontology in the production graph.

4.3 Data Usage

The last functional part of our architecture is around the data access. Here we need to cater to different kinds of access via APIs and end-user tools but most importantly do so whilst keeping an eye on R1 about data access. As highlighted in the introduction, missing on controlling for entitlements to the data sources could lead to security and privacy risks.

The architecture shows only API under service layer as a representative catch-all for accessing data in Stardog. However, the architecture supports all components listed below for real-world implementation:

Data API we wrap SPARQL queries, and other Stardog-specific query tooling such as path finding, into an API for easier user consumption without requiring end-users to write and test their own SPARQL, or GraphQL, queries;

BI integration following a need expressed to have a view over the content of the graph via business intelligence tools (for example, Tableau[6]), we leverage Stardog's ability to expose the graph as a set of relational tables which can be queried with SQL;[7]

Graph browsing is the third piece offering a browsing interface for the graph for stake-holders not able to use the APIs, the relational view, or SPARQL.

[5] Cache nodes are nodes in the Stardog cluster which transparently cache (and periodically refresh) virtualized data: https://docs.stardog.com/cluster/operating-the-cluster/cache-management.

[6] https://www.tableau.com/.

[7] https://docs.stardog.com/query-stardog/bi-tools-and-sql-queries.

We implement this functionality by deploying Stardog Explorer[8], the latest addition to the family of Stardog products.

In order to ensure entitlements are checked via *all* the data access interfaces we implemented a restriction based on named graphs. Each of the data sources in the pipeline is identified with an IRI in the EKG. The access rules for each named graph are defined to align with those of the data source. Stardog's Named Graph Security mechanism[9] ensures that every query gets to see only named graphs which the current user has access to. Internally, all access methods: API calls, GraphQL queries, SQL queries, etc., are compiled into SPARQL so that check cannot be bypassed.

The remaining part is user authentication for which we use Kerberos and LDAP. The full approach is depicted in Fig. 2.

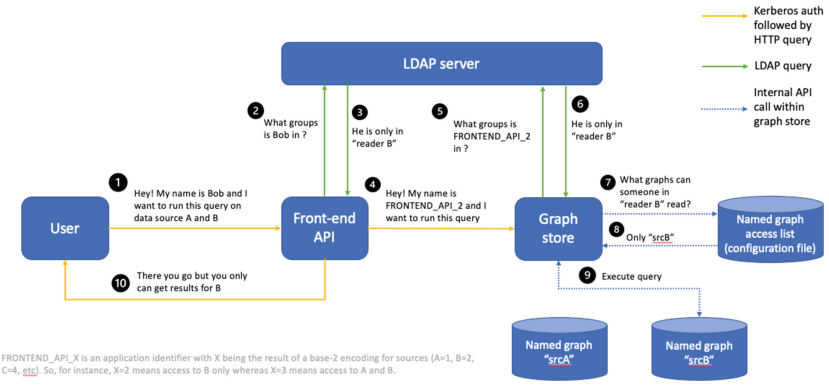

Fig. 2. Approach to handle entitlements by combining named-graph access rules, Kerberos for the authentication and an LDAP directory for listing groups

It is important to note here that API calls are identified the same way as human users in the systems and need to use their own credentials. In order to pass along the entitlements of a user into the API call, we associate each user with a group name in the LDAP directory for which the graph store then handles authorization. Each API call proceeds as follows:

- The user identifies itself to the front-end API ❶;
- The front-end queries the LDAP directory for the list of groups the requester is in ❷ and gets a response ❸;
- Based on the response the API call is authenticated as a user matching a list of groups ❹. For this to work, a set of users called "FRONTEND_API_X" is created for each possible group combination X where X is a base 10 number (Group A = 1, Group B = 2, Group C = 4, ...);

[8] https://www.stardog.com/blog/stardog-explorer-early-access-release/.
[9] https://docs.stardog.com/operating-stardog/security/named-graph-security.

- The graph store executes a query ❺ similar to ❶ and gets a response in ❻ (also aligning with ❷);
- The server then refers internally to the named graph security configuration to check which named graphs can be consumed by the user ❼, and gets a response in ❽;
- The query is finally executed against the allowed graphs ❾ and the results are returned back to the user in ❿.

5 Use-Cases

Accenture and Stardog deployed the above architecture in production in two different places. The first one, a major financial services company in need for tracking the flow of PII data across its systems. The second, a major pharmaceutical company that wants to provide a unified view of its disconnected data sources to its employees. We found those two relevant to report on as they both share the need for the four above stated requirements yet with different focus points. For example, the auditing need is more important for the financial context than it is for the internal R&D of the medical deployment.

5.1 Financial Services Use-Case

A Data Center of Excellence (CoE) sits within a large financial services company providing centralized data services to the various lines of business throughout the company. Consumers of the CoE's data services have asked for more transparency into the impacts and dependencies of changes that affect infrastructure components, applications, and sensitive PII data elements. However, providing that information to the data consumers presents two sets of challenges: ensuring seamless data source integration and resolving unique data consumer requirements.

The types of data required by the data consumers consists of infrastructure components, data elements, and business processes, etc. and each type of data resides in a separate data source. Additionally, each data consumer has their own unique set of data requirements that the CoE's solution must meet. Each data source has its own security controls, access permissions, data models, and data quality standards. For example, the data source containing infrastructure data grants end-user permissions only for the specific columns within a specific table that the end-user requested access. Additionally, while the infrastructure data source contained information about PII data, that data is not consistently formatted or held to the same data quality standards as the data source containing information about all data elements. So, those two sources can't be simply joined together. Regarding the data consumers, each one provides the CoE with their own set of requirements for the solution. This means there are many consumers with different questions and requirements about the same data. So, the data model for the knowledge graph solution needs to be general enough to support

all consumers while not over generalizing at risk of misrepresenting, or losing the intended meaning, of the underlying data sources.

If we summarise based on the requirements stated in Sect. 2, we have:

- **R1**: the data consumers can only view data in the graph that comes from sources they have already been granted access;
- **R2**: each version of the graph must be persisted for future audit needs;
- **R3**: some data consumers will use the graph for activities which are time dependent (*e.g.* track changes);
- **R4**: quality checks are crucial for mitigating errors when integrating data sources which may provide conflicting data about the same entity.

Our knowledge graph solution models the data from all sources via a unified graph schema based on the schemas of the underlying data sources. Data from each source is then brought into the knowledge graph based on the described architecture, either through virtualization or standard ETL pipelines, in accordance with the unified graph schema. The schema is based on an ontology designed by the CoE for this use case and ensures support for the unique use cases of each consumer, while maintaining the intended meaning of the underlying data sources. As depicted in Fig. 1, the solution is comprised of four distinct layers: data source layer, data ingestion layer, knowledge graph layer, and API/presentation layer.

In the data source layer we currently connect to two data sources (containing information on infrastructure and data elements respectively), with plans to expand to connecting to at least four additional sources (adding information about the company's business processes, change records, risk assessments, and a taxonomy repository).

The data ingestion layer includes two distinct methods of bringing data into the knowledge graph: data virtualization and ingestion by ETL pipelines. Our solution utilized data virtualization for connecting to data sources containing PII data, where the data was not permitted to leave its originating data source. For all other sources, Apache NiFi is used for creating and executing ETL pipelines on a scheduled basis. These pipelines would get data required by the consumers of the knowledge graph, apply transformations, and load the data into the knowledge graph based on the unified schema. Data is loaded into the knowledge graph as triples within a named graph that indicates from which data sources the data originates.

There are currently about 6.8 million triples loaded into the knowledge graph and the plan is to scale that to 15+ million triples over the coming months by connecting to the previously mentioned data sources. The triples in the knowledge graph are made available to data consumer via API endpoints which trigger SPARQL queries on the knowledge graph. Since all triples in the knowledge graph are assigned to a specific named graph based on the underlying data source, SPARQL queries are only executed over named graphs which the user has permissions to view. This feature provides an additional layer of security in the knowledge graph layer, and helps prevent exposing PII data that must be contained solely to its named graph.

The CoE's requirements for the knowledge graph solution are driven by the requirements of the data consumers. There are currently three main data consumers who's use cases and requirements drove the current solution and the long-term plan is to make the solution available to all lines of business in the company. The three data consumers use cases dealt with change management and how changes affect sensitive data elements which reside on infrastructure that the consumers owned. In the following examples, a change is defined as an alteration to a piece of infrastructure, software/code, data store, or business process.

- When planned or unplanned changes occur, what are the infrastructure components and data elements that are impacted and who is responsible for those components and data elements.
- For a given line of business, who is responsible for each PII data element, where does that data exist, where does that data originate from, where is that data being distributed to, and what security controls exist for that data (such as encryption).

The three main data consumers are actively using our solution in favor of the previous solution. Our solution has been perceived as an improvement by providing more clarity to users as they interpret the information presented by the solution about their infrastructure components and data elements. For example, there are many different types of infrastructure components such as networks, SANs, virtual servers, etc. with many instances of each type of component, and our solution more clearly shows how these components are dependent on one another. Additionally, our solution more clearly explains where PII data originates from outside of the user's consumer group, whereas the previous solution did not reliably identify where PII data existed.

Our solution meets the requirements of the CoE and the data consumers by implementing the following features:

- Provides complete data lineage across all target infrastructure
- Identifies ownership (both individual personnel and lines of business) of all target infrastructure
- Enforces security controls in all layers of solution architecture:
  - Data Source Layer: Each data source has its own security controls, so users with access to the knowledge graph solution also go through the data approval process for each data source that is used by their consumer group.
  - Ingestion Layer: Only approved members of the CoE can execute ETL pipelines
  - Knowledge Graph Layer: Named graphs categorize triples by source system and Role-Based Access Controls (RBAC) are implemented for granting permissions to users based on the source system requirements of their line of business.
  - API/Presentation Layer: Users only have the ability to execute queries that their role has permissions to execute query. User and role information

is passed in tandem with the query, so users will receive different results for the same query depending on what data they have permissions to view.

5.2 Pharmaceutical Use Case

In one of the world's largest pharmaceutical companies each Research and Development lab has their own data systems and processes that are specialized to the lab's needs which is typical for companies of this size. This results in proliferation of copies of data which quickly gets out of date. For example, it is difficult for someone in oncology to see if there is any related data within the vaccine lab.

We have used the architecture described above to provide a unified view over multiple data sources so researchers, clinicians, analysts and data scientists within the organization can access the information they need in a single location. The data sources range from structured relational databases to semi-structured data sources such as MongoDb and Elastic Search.

There are currently 5 data sources that has been mapped to RDF using SMS mappings. Each data source is materialized via a NiFi workflow to a distinct named graph (so that the graph represents the triplified view of the relational source). The central entity in the knowledge graph is the concept of a project. Projects are linked to information on genes, assays, targets and activities in several internal and external databases. The materialization process unifies the different naming and identifier schemes used across data sources to provide a unified view. There current contents of the knowledge graph contain many different entity types such as projects (8.4K), people (2.9K), clinical studies (47.8K), genes (5.6K) and chemical compounds (14.1M) for a total of about 100M triples. The plan is to increase the number of data sources integrated to 11 later this year that will increase the size of the knowledge graph to 500 million triples.

Data validation is an important requirement especially because some of the data sources are created from unstructured documents using automated NLP techniques. Technical publications, internal documents and various other kinds of unstructured documents are processed by an NLP pipeline that extracts structure content and saves the results in the knowledge graph. The automated extraction process might introduce inaccurate information. In addition, there might be inconsistencies between structured data sources when they have overlapping information. At the beginning of the project simple SPARQL queries have been used for quality checks and over time these queries have been migrated to shape constraints in SHACL.

The main use case for the knowledge graph is to power a Google-like environment for the employees in the R&D division to search the data and traverse the graph and entities based on context and knowledge. Access to detailed information about past projects makes it possible to see connections that would have been invisible before. R&D decisions can be based on all the relevant data rather than relying on tribal knowledge and personal experience.

To recall our list of requirements stated in Sect. 2, here is how those map to this specific use-case:

- **R1**: is less complex than in the financial case, but controls need to be introduced nonetheless to restrict the access to the graph;
- **R2**: the graph should be persisted in order to trace back the thought process leading to a particular trial being conducted;
- **R3**: in order to support the R&D process the data needs to be aggregated from a variety of sources and be up to date all the time;
- **R4**: quality checks are important to mitigate the errors the NLP pipeline can be reasonably expected to make.

The next steps is to go beyond search and explore use case and provide an API end point for data scientists and researchers to the data from the knowledge graph within their own projects and not have to rely on the search interface. One potential data science project is using the characteristics of particular gene targets to predict if a trial will be successful.

6 Conclusion

In this paper we described the implementation of a knowledge graph architecture and discussed how it is applied for two separate use cases: tracking the flow of PII data across many infrastructure components within a financial services company and providing pharmaceutical company employees with a unified view of disconnected data sources. The role of Semantic Web technologies is critical to the success of both use cases. We modeled disconnected data in unified schemas, used data virtualization and materialization to create knowledge graph instances, made the data available to many end-users via authenticated APIs, and leveraged shape-checking standards (SHACL) to keep an eye on the quality of the data.

In our experience data virtualization is a successful approach to integrating siloed data. It does, however, impose certain restrictions on data processing tasks, such as SHACL or querying. Not all of SPARQL can be translated into a query language supported by some upstream data source in the organization. That is true, for example, for arbitrary property paths or SPARQL extensions, like full-text search or path finding queries. In practice, such issues can usually be avoided by using less expressive queries (like fixed-length paths expressed as SPARQL BGPs) or by doing less computation in the upstream system and thus bringing more intermediate query results to Stardog (i.e. at performance cost). When none of these compromises is acceptable, it is always possible to use Stardog cache nodes to transparently cache and refresh virtualized data inside the Stardog cluster[10].

The implemented architecture is not limited to the two described industries and can be applied to use cases in other domains that require connecting siloed data sources containing sensitive data such as manufacturing, energy production,

[10] https://docs.stardog.com/cluster/operating-the-cluster/cache-management.

or healthcare. In future work, we would like to see updates on R2RML for non-relational data, as well as authentication support for LDP. Those two aspects limit the capabilities we can offer using standards, or require more custom code to be implemented, and their development could be leveraged by both our implementation and the broader Semantic Web community. Within our architecture we will seek to improve SPARQL query performance over a partially virtualized knowledge graph.

References

1. Batsakis, S., et al.: PRELIDA D3.1 State of the art assessment on Linked Data and Digital Preservation. European Commission (2014)
2. Fernández, J.D., Martínez-Prieto, M.A., Gutiérrez, C., Polleres, A., Arias, M.: Binary RDF representation for publication and exchange (HDT). J. Web Semant. **19**, 22–41 (2013). http://dblp.uni-trier.de/db/journals/ws/ws19.html#FernandezMGPA13
3. Guéret, C., Groth, P., Stadler, C., Lehmann, J.: Linked data quality assessment through network analysis. In: Proceedings of the 10th International Semantic Web Conference (ISWC 2011) (2011)
4. Haase, P., Herzig, D.M., Kozlov, A., Nikolov, A., Trame, J.: Metaphactory: a platform for knowledge graph management. Semantic Web **10**(6), 1109–1125 (2019). https://doi.org/10.3233/SW-190360
5. Kirrane, S., Mileo, A., Decker, S.: Access control and the resource description framework: a survey. Semantic Web **8**(2), 311–352 (2017). https://doi.org/10.3233/SW-160236
6. Li, X., Lyu, M., Wang, Z., Chen, C.H., Zheng, P.: Exploiting knowledge graphs in industrial products and services: a survey of key aspects, challenges, and future perspectives. Comput. Ind. **129**(March) (2021). https://doi.org/10.1016/j.compind.2021.103449
7. Mehdi, A., Kharlamov, E., Stepanova, D., Loesch, F., Grangel-González, I.: Towards semantic integration of bosch manufacturing data. In: Suárez-Figueroa, M.C., Cheng, G., Gentile, A.L., Guéret, C., Keet, C.M., Bernstein, A. (eds.) ISWC Satellites. CEUR Workshop Proceedings, vol. 2456, pp. 303–304. CEUR-WS.org (2019). http://dblp.uni-trier.de/db/conf/semweb/iswc2019p.html#MehdiK0LG19
8. Meroño-Peñuela, A., Hoekstra, R.: grlc makes GitHub taste like linked data APIs. In: Sack, H., Rizzo, G., Steinmetz, N., Mladenić, D., Auer, S., Lange, C. (eds.) ESWC 2016. LNCS, vol. 9989, pp. 342–353. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47602-5_48
9. Sequeda, J.F., Briggs, W.J., Miranker, D.P., Heideman, W.P.: A pay-as-you-go methodology to design and build enterprise knowledge graphs from relational databases. In: Ghidini, C., et al. (eds.) ISWC 2019. LNCS, vol. 11779, pp. 526–545. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30796-7_32
10. Van de Sompel, H., Nelson, M.L., Sanderson, R., Balakireva, L.L., Ainsworth, S., Shankar, H.: Memento: Time Travel for the Web (2009). arXiv:0911.1112

11. Xiao, G., Ding, L., Cogrel, B., Calvanese, D.: Virtual knowledge graphs: an overview of systems and use cases. Data Intell. **1**(3), 201–223 (2019). https://doi.org/10.1162/dint_a_00011
12. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S.: Quality assessment methodologies for linked open data: a systematic literature review and conceptual framework. Semantic Web **1**, 33 (2012). http://www.semantic-web-journal.net/sites/default/files/DQ_Survey.pdf