Supporting Polystore Queries using Provenance in a Hyperknowledge Graph

Leonardo G. Azevedo[1][0000−0002−2109−1285], Renan Souza[1], Elton Soares[1], Raphael Thiago[1], Anna Oliveira[2], and Marcio Moreno[1]

[1] IBM Research
{lga, rfsouza, raphaelt, mmoreno}@br.ibm.com, eltons@ibm.com,
[2] COPPE/UFRJ
acoliveira@cos.ufjr.br

Abstract. Current modern applications commonly need to manage various types of datasets, usually composed of heterogeneous data and schema manipulated by disparate tools and techniques in an *ad-hoc* way. This demo presents `HKPoly` - a solution that tackles the challenge of mapping and linking heterogeneous data, providing data access encapsulation by employing semantic, provenance, and data linkage.

Keywords: Knowledge Graph · Hyperknowledge · Polystore.

1 Introduction

Modern applications usually manipulate diverse datasets with different models and usages, employing several tools and techniques. As an example, Oil reserves discovery is critical in the O&G industry, and it involves several activities performed by collaborating teams , consuming and generating data of distinct sources, semantics, and format [8]. We demonstrate our proposal in this scenario.

Data management solutions have emerged to handle heterogeneous data access, *e.g.,* distributed file systems, NoSQL, processing frameworks [1]. In general, each solution handles one or a few kinds of data models or formats and does not provide a semantic abstraction for client access. However, modern applications usually require handling heterogeneous data systems. So, a middleware that provides a seamless interface with an independent data model and (perhaps) data schema becomes necessary [9]. Federated data systems are the leading candidates for such middleware [10], but they lack an abstract semantic layer. Semantic mapping and record linkage is still a challenge [5], which states for automatic translation of client utterances to the encapsulated storage systems' dialect and the integration of query results.

This work demonstrates `HKPoly`, a solution to overcome this challenge employing Semantic Web concepts, *i.e.,* Linked Data, Provenance Data, and inference rules to extract meaning and enable reasoning [2][6].

2 Hyperknowledge Polystore (HKPoly)

HKPoly uses abstract global representation and query language to encapsulate heterogeneous remote data stores. It provides semantic mapping and record linkage through a domain ontology, metadata about remote data stores' schemas and the domain ontology, and provenance techniques. Its main components are (Figure 1.a): (i) *HKPoly Core*: controls the access to the remote data stores; (ii) *Provenance Manager*: captures and manages provenance of the processes that manipulate the (remote) data; (iii) *Provenance and Knowledge Graph*: stores provenance data, domain ontology and data stores' metadata, references to remote data objects, and mappings of the domain and remote data store schemas.

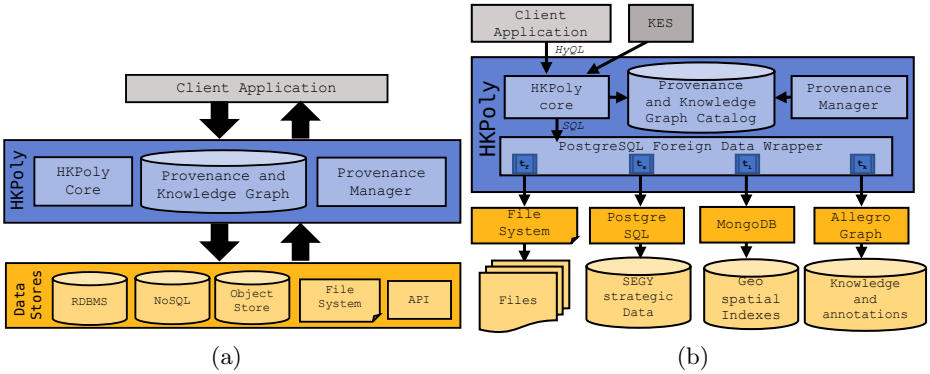

Fig. 1. HKPoly architecture: (a) overview; and, (b) implementation.

Figure 2 presents part of *Provenance and Knowledge Graph* model which extends W3C PROV.The left (yellow) represents metadata about the encapsulated data stores (the DataStores hierarchy and where they run). The right (blue) represents the schemas' metadata, identifiers, attributes, values. This model is represented using Hyperknowledge (HK) - a hybrid conceptual model that handles the multitude of media content and the meaning behind this data [3].

We employed existing tools in HKPoly implementation (Figure 1.b): HKBase, KES, ProvLake [8], and PostgreSQL Foreign Data Wrapper (FDW)[3]. FDW is a PostgreSQL module that allows accessing data stored in external heterogeneous stores. HKBase and KES are components of Hyperknowledge Platform - a set of tools developed specifically for handling Hyperknowledge (HK).

HKBase provides a RESTful API to manipulate structured data (represented in HK), and unstructured data (*e.g.,* images or videos). HKPoly is provided as an HKBase service. HyQL[4] is the HKBase's Hyperknowledge Query Language [7].

[3] https://www.postgresql.org/docs/9.5/postgres-fdw.html

[4] The HyQL grammar is presented in https://ibm.ent.box.com/v/iswc2021-hyql-grammar

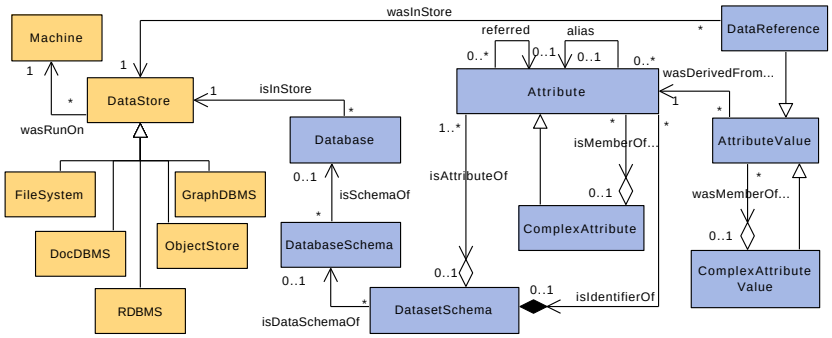

Fig. 2. `HKPoly` *Provenance and Knowledge Graph* data model.

`KES` (Knowledge Explorer System) [4] provides a user interface (UI) for management of `HK` bases (*i.e.,* knowledge bases with `HK` specifications). `KES` promotes creating knowledge representations, validating and curating knowledge through an interactive visual approach. We used `KES` to navigate, manipulate, and visualize the *Provenance and Knowledge Graph*.

Provenance Manager was inherited from ProvLake implementation [8] . It creates a provenance graph from data captured during workflow execution, but, in `HKPoly`, these graphs are augmented with polystore semantics.

`HKPoly` steps are: (i) **Create FDW tables**. (ii) **Create metadata** about: domain ontology; data store configuration (*i.e.,* access and schema metadata); FDW schemas; mappings among domain ontology and FDW schemas, and data store schemas and FDW schemas. (iii) **Run the workflows** that manipulates the remote data, **and capture provenance**. (iv) **Process queries**.

In step (iii), the workflows' implementations are instrumented to capture the workflow's time, input, and output data and its data transformations, including references to the remote data. This data is sent to *Provenance Manager* which stores it in the *Provenance and Knowledge Graph*.

In step (iv), a *Client Application* sends a `HyQL` query to `HKPoly`. `HKPoly` parses the query to identify the queried elements and the related data sources. It discovers the FDW foreign tables and creates a SQL query. Then, it sends the query to PostgreSQL, which access the remote data using FDW wrappers[5].

Although several works have tackled the problem of database federation from different perspectives [1] [10]. Our solution encapsulates the remote data and the complexity of its underlying model. The *Client Application* does not have to specify the paths to navigate to remote data. It specifies a query considering the domain ontology, and `HKPoly` uses the captured data references and schema metadata and mappings to get the remote data using FDW.

[5] We used Multicorn (`https://multicorn.org/`) and file_fdw (`https://www.postgresql.org/docs/9.5/file-fdw.html`) PosgreSQL FDW extensions for the wrappers implementation.

3 HKPoly in use

We employed HKPoly in an Oil reserves discovery scenario, which is critical in the O&G industry. It involves several activities, including seismic image interpretation. We considered the scenario's heterogeneous data aspect (Figure 3 [8]).

Each activity uses and generates data from/to data stores with heterogeneous data models. The first activity processes geological raw data files (residing on Parallel File System) to extract necessary metadata and assess their data quality (stored in PostgreSQL - R-DBMS). The second activity uses the high-quality data files and generates geospatial indexes (stored in MongoDB - Doc DBMS) to accelerate geospatial queries over the geological data. The third activity uses the high-quality data files and augments the raw geodata files with extra knowledge informed by geoscience experts (stored in AllegroGraph - T-DBMS). The last activity prepares the learning datasets to be used by the DL algorithms.

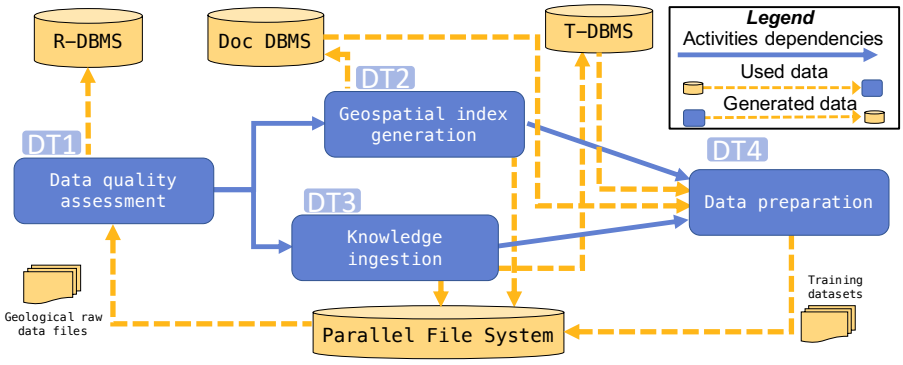

Fig. 3. Workflows and data stores. Figure adapted from [8].

The user is an ML expert with deep knowledge in the domain, and, after running the workflow, s/he has to report the ML model results. It requires querying the processed domain data which resides in the remote data stores.

The user application interacts with HKPoly through its endpoints, *e.g.,* sending queries in HyQL (Listing 1.1). This query is received by HKPoly, and parsed, resulting in the SQL query to be performed in FDW (Listing 1.2). The HyQL query is in the domain abstraction level, *i.e.,* the user does not have to be aware of the complexities underlying the heterogeneous remote data systems. Hence, HKPoly is easier to use than the user writing the FDW query or implementing scripts to get the data from each remote data store independently.

Listing 1.1. HyQL to get data of seismic Netherlands.

```
1  select Seismic.inline, Seismic.crossline, Seismic.hasWell, Seismic.
       hasHorizon, Seismic.epsg
2  where  Seismic from geological_data_ingestion_workflow
3  and Seismic.name  = "Netherlands"
```

Listing 1.2. SQL to get remote data of seismic Netherlands.

```
select distinct ag."hasHorizon", mg.uri,
                pg.crossline, ag."hasWell", pg.inline
from segy fl, kb_seismic ag, mongo_seismic mg, seismic_header pg,
( VALUES ( 'netherlands.sgy',
    'http://br.ibm.com/hkpoly/seismicData_ABox#Netherland_3D',
    'http://br.ibm.com/hkpoly/seismicData_ABox#Netherland_3D', 1 ))
    as p(FileSystem1_prov_id, Allegro1_prov_id,
        Mongo1_prov_id, Postgres1_prov_id)
where ag.uri=p.Allegro1_prov_id AND mg.uri=p.Mongo1_prov_id
      AND pg.id=p.Postgres1_prov_id
```

Demo video 1: `HKPoly` architecture; ProvLake and Data Store metamodel; load domain knowledge, load data stores' configuration; load domain knowledge schema; load FDW mappings.
`https://ibm.box.com/v/iswc2021-hkpoly-demo-video1`
Demo video 2: The scenario used in the demo; input HyQL; FDW generated SQL for remote data access; an example of user code provenance instrumentation; data visualization in `KES`; `HKPoly` steps; and, `HKPoly` service running.
`https://ibm.box.com/v/iswc2021-hkpoly-demo-video2`

References

1. Azevedo, L.G., Soares, E.F.d.S., Souza, R., Moreno, M.F.: Modern federated database systems: An overview. In: 22nd International Conference in Enterprise Information Systems (ICEIS). pp. 276–283 (2020)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web : a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American **284**(5), 34–43 (2001)
3. Moreno, M.F., Brandao, R., Cerqueira, R.: Extending hypermedia conceptual models to support hyperknowledge specifications. International Journal of Semantic Computing **11**(01), 43–64 (2017)
4. Moreno, M.F., Santos, R., Santos, W., Brandão, R., Carrion, P., Cerqueira, R.: Handling hyperknowledge representations through an interactive visual approach. In: IEEE Intl. Conf. on Information Reuse and Integration. pp. 139–146 (2018)
5. Özsu, M.T., Valduriez, P.: Principles of distributed database systems. Springer, 4th edn. (2020)
6. Patel, A., Jain, S.: Present and future of Semantic Web Technologies: a Research Statement. Intl. Journal of Computers and Applications pp. 1–10 (01 2019)
7. Prud, E., Seaborne, A.: Sparql query language for rdf (2008), `https://www.w3.org/TR/rdf-sparql-query/`, accessed in April 12st, 2021
8. Souza, R., Azevedo, L., Thiago, R., Soares, E., et al.: Efficient runtime capture of multiworkflow data using provenance. In: 2019 15th International Conference on eScience (eScience). pp. 359–368 (2019)
9. Stonebraker, M.: The case for polystore. `https://wp.sigmod.org/?p=1629` (2015)
10. Tan, R., Chirkova, R., Gadepally, V., Mattson, T.G.: Enabling query processing across heterogeneous data models: A survey. In: IEEE Intl. Conf. on Big Data (Big Data). pp. 3211–3220. IEEE (2017)