A Provenance Management Framework for Knowledge Graph Generation in a Web Portal

Erik Kleinsteuber[1] Samira Babalou[2] Birgitta König-Ries[3]

Abstract: Knowledge Graphs (KGs) are the semantic backbone for a wide variety of applications in different domains. In recent years, different web portals providing relevant functionalities for managing KGs have been proposed. An important functionality of such portals is provenance data management of the KG generation process. Capturing, storing, and accessing provenance data efficiently are complex problems. Solutions to these problems vary widely depending on many factors like the computational environment, computational methods, desired provenance granularity. In this paper, we present one possible solution: a new framework to capture coarse-grained workflow provenance of KGs during creation in a web portal. We capture the necessary information of the KG generation process and store and retrieve the provenance data using standard functionalities of relational databases. Our captured workflow can be rerun over the same or different input source data. With this, the framework can support four different applications of provenance data: (i) reproduce the KG, (ii) create a new KG with an existing workflow, (iii) undo the executed tools and adapt the provenance data accordingly, and (iv) retrieve the provenance data of a KG.

Keywords: Semantic Web; Knowledge Graph; Knowledge Graph Platform; Provenance Tracking; Reproducibility

1 Introduction

Knowledge Graphs (KGs) are graph-structured knowledge bases that store factual information about a particular domain in the form of relationships between entities. They are the semantic backbone for a wide variety of applications. This brings the need to support their management. In consequence, different KG management platforms have been suggested for both scientific and commercial applications [Sy22, SDA20, Ha19, Be20]. Such platforms mostly cover the whole lifecycle of KG application and include relevant services or functionalities for creating, using, and further management of KGs. The KG generation process (see Figure 1, top) can be modeled as an ordered execution of tools to transform source data into a data graph. Ideally, during this creation process, detailed provenance information should be

[1] Heinz-Nixdorf Chair for Distributed Information Systems, Institute for Computer Science, Friedrich Schiller University Jena, Germany erik.kleinsteuber@uni.jena.de
[2] Heinz-Nixdorf Chair for Distributed Information Systems, Institute for Computer Science, Friedrich Schiller University Jena, Germany; German Center for Integrative Biodiversity Research (iDiv), Halle-Jena-Leipzig samira.babalou@uni.jena.de
[3] Heinz-Nixdorf Chair for Distributed Information Systems, Institute for Computer Science, Friedrich Schiller University Jena, Germany; German Center for Integrative Biodiversity Research (iDiv), Halle-Jena-Leipzig; Michael-Stifel-Center for Data-Driven and Simulation Science, Jena, Germany birgitta.koenig-ries@uni-jena.de

doi:10.18420/BTW2023-65

captured and subsequently managed. With this, on the one hand, it is possible to track where information in the KG stems from. This increases trust in the information provided and supports open science principles [SKR22]. On the other hand, provenance information enables rerunning the creation process to check for reproducibility and/or to create an updated version of a KG. Both are important factors in increasing KGs usage. Capturing, storing, and accessing provenance data are complex problems. Solutions to them vary widely depending on many factors like the computational environment, computational methods, desired provenance granularity, and much more. The decision on a provenance solution depends on interests, needs, and expectations of the developers or potential users, and heavily on the domain of applications [PRSA18]. One solution is to integrate a computational task into an environment capable of capturing provenance of the results of these tasks.

In a KG-generation web portal with user interaction, it is necessary to consider dependencies, restrictions, security, and fault tolerance issues of different tools during KG generation process. Third-party tools need to be securely connected, safely executed, and carefully monitored. It is also important to embed these tools in a way that they do not affect each other (e.g., writing files or memory usage). Moreover, requirements on the provenance solution for the web portal might change during development, thus a highly flexible architecture is required. Existing provenance systems (cf. the reviewed systems in [PRSA18]) tend to have constraints on programming languages and on domains. Integrating third-party tools into a computational notebook can be a difficult task depending on the complexity of the tool. Additionally, connecting such a provenance system or computational notebook to the web portal's functionalities adds an extra layer of complexity if possible at all.

In this paper, we present our solution to providing users with all necessary information (so-called provenance data) about the KG generation process in a web portal. We present the provenance data of KG generation process as a workflow. It holds information about all executed tools and necessary inputs and outputs during KG generation. We propose a framework that can capture the coarse-grained workflow provenance of generated KGs in a web portal. This framework is customized in our studied platform, but it can be adopted for any other platform. We capture the necessary information about the user, source data, intermediate results, and executed tools during the KG generation process. We ensure that the data about the computational tasks needed to create the KG are stored in a reusable workflow associated to the KG. We retrieve the captured provenance data accordingly to provide user convenience and better insight into the KG generation process. Our captured workflow can be rerun over the same or different source data. We show different applications of the provenance data. Moreover, we show how the workflow in our framework can be mapped to the W3C PROV ontology [Le13].

The rest of the paper is organized as follows. Section 2 presents preliminaries along with literature review. Section 3 shows our proposed framework, followed by implementation detail in Section 4. We conclude the paper in Section 6.

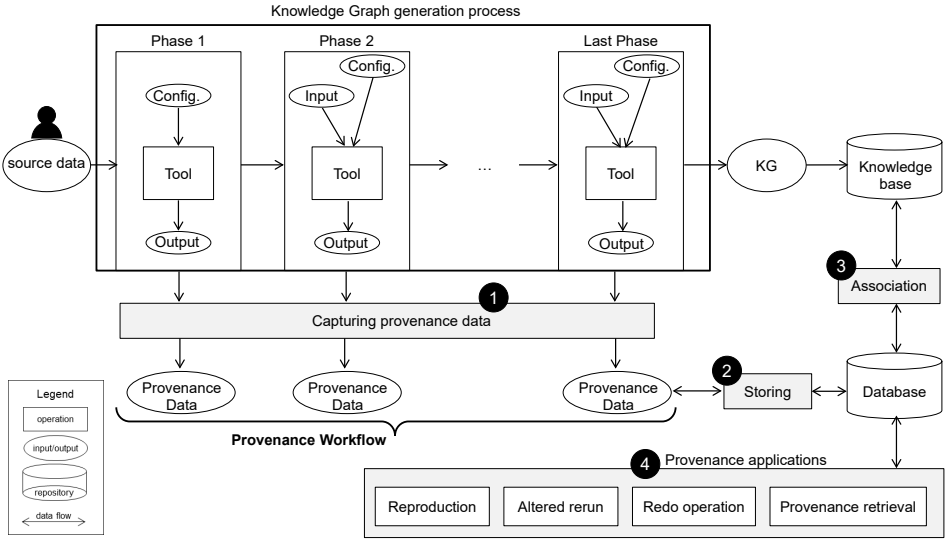


Fig. 1: Overview of our management framework for managing the provenance of the Knowledge Graph generation in a web portal.

2 Literature Review & Preliminaries

The provenance of an object is the history of its origin and derivation [MMW13]. Provenance tracking records the provenance of an object. In the literature, there have been different surveys (cf. [PRSA18, HDBL17]) on provenance characteristics and provenance models. The importance of provenance on large-scale KGs and the Web of Data has been highlighted in [Ho20]. As a solution to manage the provenance, computational Notebooks (cf. Jupyter Notebook [Kl16]) have gained widespread adoption in recent years, cf. ProvBook [SKR18]. However, implementing large, complex projects in a notebook, especially when multiple programming languages are used, is not straightforward. Another issue is the automation of a notebook. We faced different problems to connect a piece of software implemented in Jupyter Notebook to the backend of our studied platform, and executing cells when we receive specific user requests. On the other hand, web-based interactive development environments such as JupyterLab that can be hosted and accessed by multiple people would introduce security issues. Existing tools such as Open Refine (`https://openrefine.org/`) among others, can also track applied operations on the data and thus can be used as a provenance solution. However, we did not use such tools as a provenance solution, as they are not a fully-fledged development environment and it is not always possible to extend those tools with arbitrary code and still make use of its provenance features. To the best of our knowledge, a few KG platforms [Sy22, SDA20, Ha19, Be20] apply a provenance solution in some capacity. Of those, Blue Brain Nexus [Sy22] is the only one explicitly mentioning the importance of provenance data and their usage of the W3C PROV ontology [Le13].

The other platforms did not explain their approach on provenance management in their publications.

Preliminaries. To generate a KG, different phases need to be carried out (see Figure 1, top), where in each phase a (different) tool will be executed. Definition 1 shows our definition of the KG generation process.

Definition 1 ***KG generation*** *is an ordered execution of tools in different phases, where source data d_s is the input and a Knowledge Graph (KG) is the output. Formally, this yields $KG \leftarrow generation(d_s)$.*

The tools are executed to perform different computational tasks e.g., cleaning datasets, linking the entities to external resources, and generating specific output. In this paper, we define a tool as:

Definition 2 *A* ***tool (TO)*** *is an executable piece of software that performs some computational task. The input of a tool is a file along with a configuration. The output of a tool is a new processed file.*

A configuration is a set of input parameters for executing the tool. Every configuration is a set of key-value pairs, containing the name of a parameter and its value. For simplicity, we assume that tools are run sequentially to generate a KG. This execution order needs to be preserved. We assume the input of a tool execution is the output of the prior tool execution. Every tool execution has a file as an input and a new file as an output. We define any such file as a data object.

3 Our Proposed Provenance Management Framework

Figure 1 shows our provenance management framework in the KG generation of a web portal. The users go through different phases to generate the KG based on their source data. The resulting KG is added to the knowledge base. Thus, with each generation of a new (sub-)KG, the overall KG (in the remainder of the paper referred to as main KG) is extended. The provenance data of each phase is captured (❶), stored (❷), and associated with the generated KG (❸). The provenance data can be used for various applications (❹).

3.1 Capturing provenance data

The provenance capture mechanisms collect information related to the KG generation process. In the web portal, each KG generation starts by uploading source data d_s by a user.

After that, the user selects a tool with a specific configuration and then the tool gets executed. This happens sequentially multiple times until the KG is generated. In our framework, we capture (see ❶ in Figure 1) the provenance of each data object, that got processed by a tool execution during the KG generation process. All executed tools, configurations, and input and output of each phase of the KG generation are saved separately. For each, we save a set of information such as the version of the tool or the storage location of a file. We capture all provenance data of an executed tool at every phase. We call the information about all executed tools in the sequential phases of the KG generation a workflow (see Definition 3). Note that, the provenance data of a data object includes all stored data of prior phases until that phase. In this view, the provenance data of the generated KG is the complete workflow W.

Definition 3 *A **workflow** W is an ordered collection of provenance data of executed tools in all phases of the KG generation process.*

3.2 Provenance storage and association

During KG generation, we store (see ❷ in Figure 1) the provenance of each produced data object. Note that, each generated KG is a sub-KG of the main KG in the web portal. We consider two approaches for provenance storage and associating provenance data to a generated sub-KG (see ❸ in Figure 1).

Approach 1: Provenance data is saved in the knowledge base. This would require the provenance data to be represented as triples. These triples could be grouped by another graph name. The provenance data can then be coupled by a triple featuring both the provenance data of graph name and the KG.

Approach 2: The generated KGs are stored in a knowledge base, while their provenance data are stored in a relational database. To handle the provenance storage in the web portal, we store provenance data of KG generation process (i.e., workflow) as a JSON-string inside the provenance record. Listing 1 shows a layout of a workflow W holding for $Phase_i$. For every new phase, a new key $Phase_{i+1}$ gets created together with a new dictionary and new values.

```
{"Phase_i":
{
        "input": "referenceTo(data object)",
        "tool": "referenceTo(TO)",
        "configuration":
        {
                "argname1": "value1",
                "argname2": "value2",
                ...
        },
        "output": "referenceTo(data object)"
}
}
```

List. 1: Example of a workflow holding one phase in JSON-string.

For each request of KG generation in the web portal, we create a new entry in the relational database (provenance information of KG generation's phases). Table 1 shows a simplified version of this database (the database, including provenance data of tools, datasets, and users, are not shown here). Each KG generation process has a primary key and belongs to a specific project. All provenance data of a KG generation's phases (i.e., the workflow) is saved as a provenance record. We support here both graph- and triple-levels.

At the graph-level, we store the reference (URI of the sub-KG in the knowledge base) to this sub-KG in our relational database (Column 3 of Table 1). In this case, for each saved sub-KG in the knowledge base, we save the reference (id) of that sub-KG next to its provenance data. In the triple-level, subjects, predicates, and objects of all triples are annotated (e.g., via `rdfs.seeAlso`, `rdfs.comment`, or other defined annotations) with their respective provenance ids. With the provenance id we mean the first column of Table 1. If a user wants to retrieve the provenance of a specific sub-KG or a term (subject, predicate, or object), we can lookup for its provenance data via the reference of the sub-KG. The association is of type "no-coupling" according to [PRSA18]. The associated provenance data can be stored in different formats such as JSON, XML, or turtle (an RDF KG relying on the PROV-O ontology (see next section)). In this approach, provenance data is stored in the relational database and can be exported in different formats.

4 Implementation

We have partially tested our provenance management framework in the iKNOW project (`https://planthub.idiv.de/iknow/`), which is distributed under an open-source license in `https://github.com/fusion-jena/iKNOW`. iKNOW [Ba21] is a planned semantic-based toolbox for Knowledge Graph creation and evolution in the biodiversity domain. For the

Tab. 1: Provenance data about the KG generation process in the web portal.

Key	Project Name	RefTo Sub-KG	Provenance Record
001	NameA	$ex : G_1$	[[(phase,“1”), (input,“file12”), (tool,“ToolA”), (output,“file33”),(config,“arg1”)] [(phase,“2”), (input,“file33”), (tool,“ToolD”), (output,“file53”),(config,“arg3”)] [(phase,“3”), (input,“file53”), (tool,“ToolB”), (output,“file22”),(config,“arg6”)]]
002	NameB	$ex : G_2$	[[(phase,“1”), (input,“file10”), (tool,“ToolC”), (output,“file44”),(config,“arg1”)] [(phase,“2”), (input,“file44”), (tool,“ToolE”), (output,“file42”),(config,“arg4”)] [(phase,“3”), (input,“file42”), (tool,“ToolG”), (output,“file12”),(config,“arg5”)] [(phase,“4”), (input,“file12”), (tool,“ToolF”), (output,“file68”),(config,“arg8”)]]

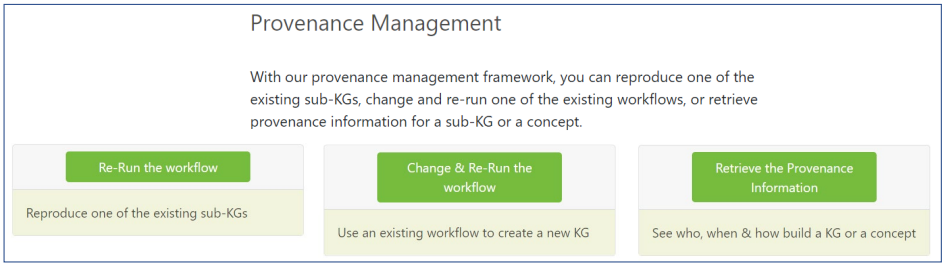


Fig. 2: Provenance Management GUI.

backend, we used the Python web framework Django (`www.djangoproject.com/`). For building user interfaces on the frontend, we used Svelte (`https://svelte.dev/`). We used Docker (`www.docker.com/`) to encapsulate the different pieces of software that are - or will be - implemented on our server. A Docker container packages up code and all associated dependencies. This prevents dependency issues and provides an isolated runtime environment that can be used to serve all kinds of tasks. We used Postgresql (`www.postgresql.org/`) for saving data of the portal functionalities along with provenance data, and Blazegraph DB (`www.blazegraph.com/`) for storing and accessing KGs. We currently implemented the second approach of provenance association.

5 Provenance Applications

Figure 2 shows the GUI of provenance management. Our provenance framework can support four different applications (see ❹ in Figure 1):

1. Reproducibility. A reproducible KG can increase trust in the information provided and support open science. Reproducibility of a KG is the capability of getting the same KG by recreating or reproducing the KG. Reproducing the KG in our framework can be done by automatically running a pre-existing workflow of a sub-KG with the goal of reproducing the same KG. Through our GUI, the user can select one of the existing workflows and run the whole process. After executing the workflow, the resulting KG can be downloaded

separately. We also show the original version of the KG, so that the user can compare their triples and metadata (such as the number of triples).

2. Altered rerun. Let us consider this scenario, where a user wants to generate a new KG based on one of the existing workflows W of a pre-generated KG while possibly changing tools, configurations, or even the source data. To achieve this, we let the user select one of the existing workflows (see Figure 3), make the desired changes on that (such as selecting other tools or changing the tool's configuration), and then run the workflow over the same or another source data. The main advantage here is the possibility of generating a new KG automatically with an already known workflow W. This provides user convenience. In the end, we save this altered workflow as a new workflow in our portal.

3. Undo operation. Let us consider a user is in the process of generating a new sub-KG wanting to undo one or multiple tool executions. In this scenario, we let the user roll back one or several executed tool(s). Upon this action, the provenance data will be updated accordingly. Some additional implementation details (e.g., deleting files, or making provenance data and files consistent in the database) have to be considered to ensure the safety of the operation. This application is implemented in the Knowledge Graph generation scenario (see Figure 5).

4. Provenance retrieval. Retrieving provenance data is important for a user to view the data and understand how tools were executed during KG generation. Through our GUI (see Figure 4, top) users can select which sub-KG they want to retrieve provenance data. They can also retrieve the provenance data of a specific term. The system first search on which sub-KG the term exists. It then shows the list of sub-KGs to the user. Then, the user can select one of the sub-KG to see its provenance data. Figure 4, down, shows the result of provenance retrieval. The user can observe who, when and how the sub-KG is built. We currently offer the possibility to download provenance data of a sub-KG as a JSON. Moreover, our provenance data is mapped to the popular, standardized PROV-Ontology and can be downloaded in any RDF Syntax. We plan to provide downloading provenance data as a Turtle file, mapped to PROV-Ontology. A workflow can be mapped to the PROV-Ontology, considering the following rules:

- Every provenance data of each phase of KG generation gets a URI and becomes a `prov:Activity`.
- Every data object and configuration of a tool gets a URI according to its reference and becomes a `prov:Entity`.
- Every tool gets a URI according to its reference and becomes a `prov:Agent`.
- For every phase (provenance data of each phase), there is a triple with the URI of the phase as the subject, `prov:wasAssociatedWith` as the predicate and the URI of the tool as the object
- For every phase, there is a triple with the URI of the phase as the subject, `prov:used` as the predicate and the URI of the input data object as the object

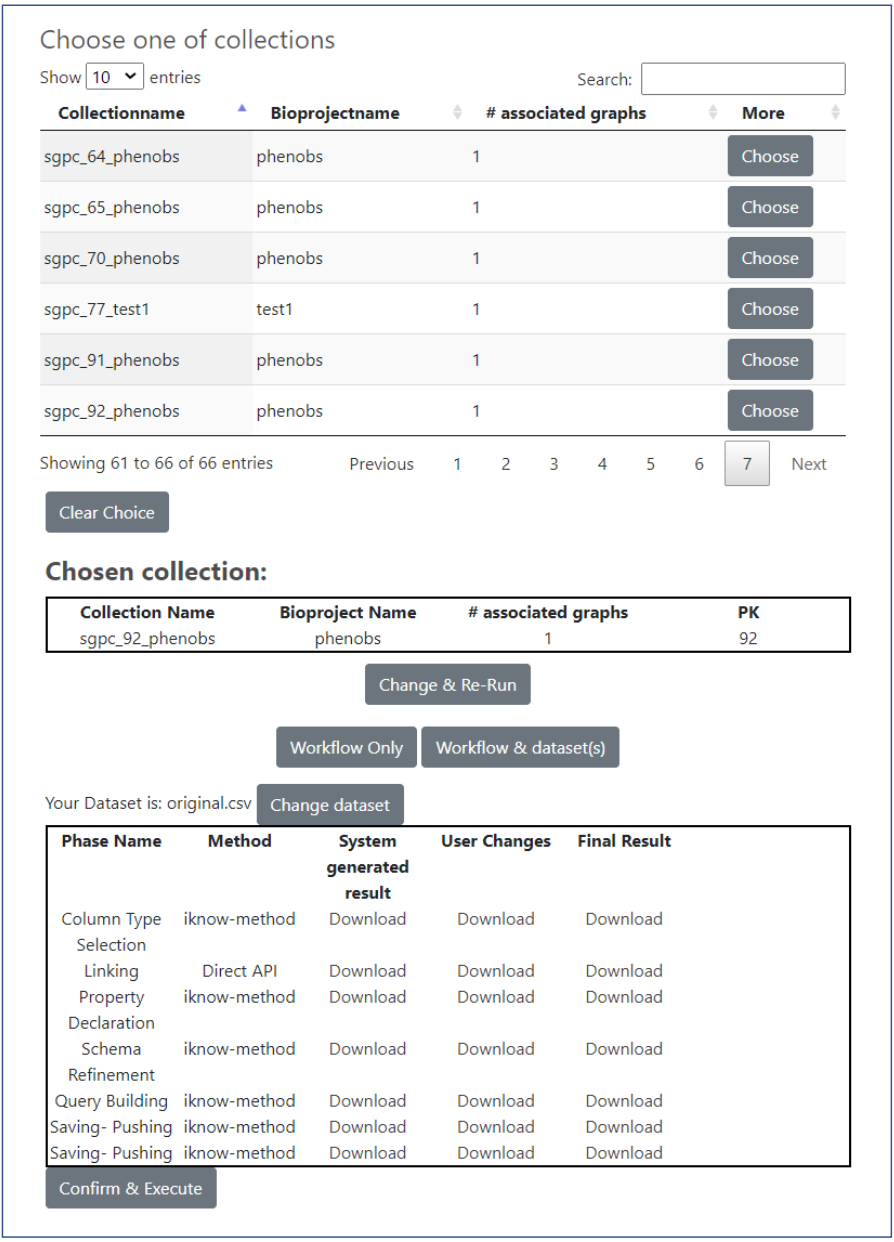
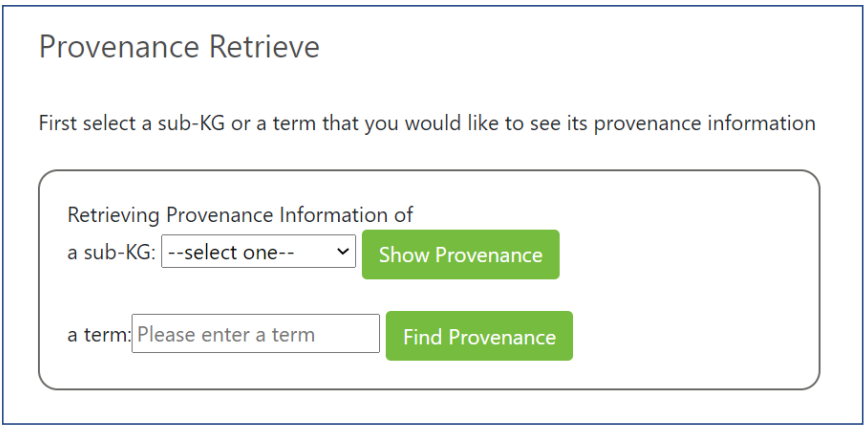


Fig. 3: The GUI of altered rerun scenario.



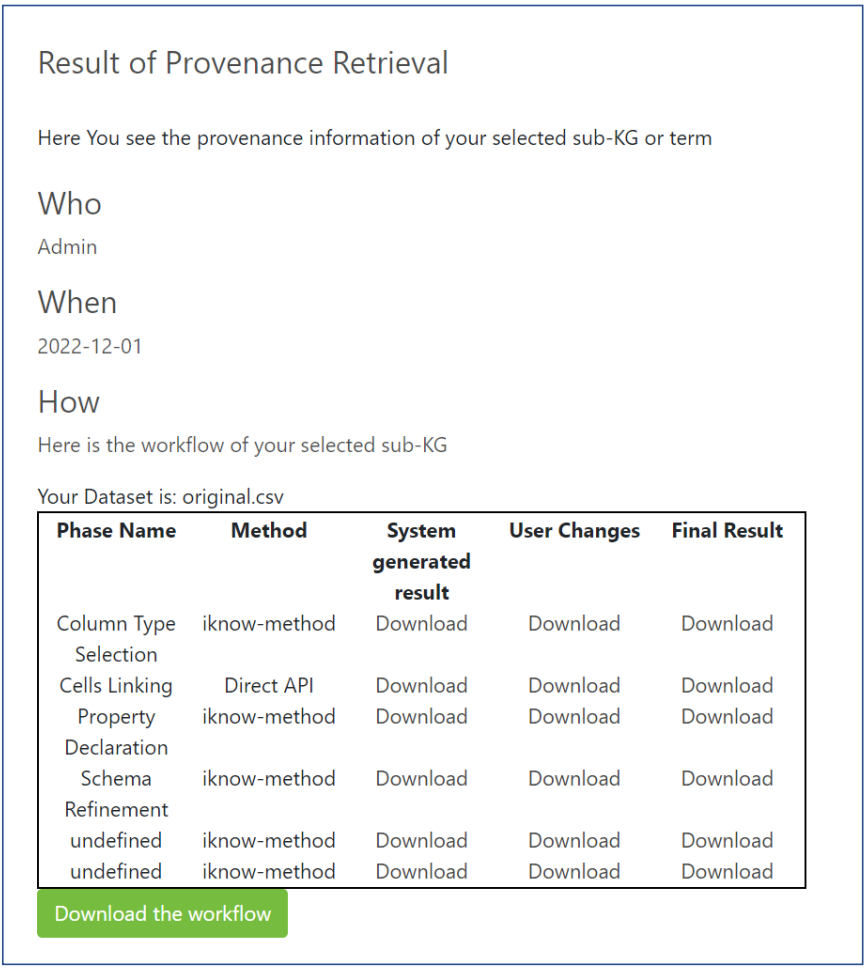


Fig. 4: Top: Users select which term or sub-KG they want to retrieve the provenance; Down: It shows the result of provenance retrieval.

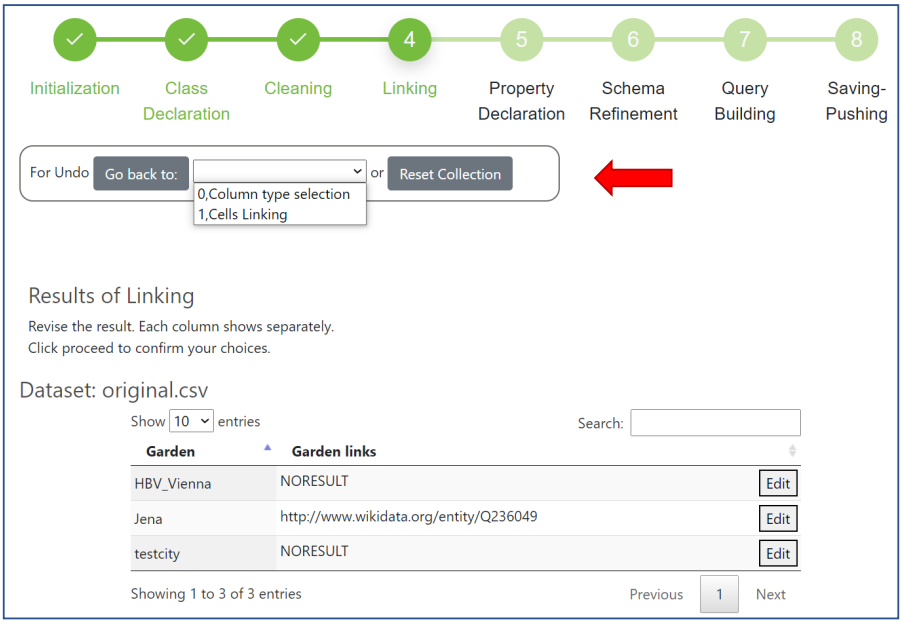


Fig. 5: Undo operation in the Knowledge Graph generation process.

- For every output data object there is a triple with the URI of the output data object as the subject, `prov:wasGeneratedBy` as the predicate and the URI of the according phase as the object
- The configuration gets a URI according to the index of the phase and becomes a `prov:Entity`. For every key-value pair in the configuration, a new triple can be created with the URI of the configuration as the subject.
- To show that the configuration is used in the phase, a triple `<ex:phase_i prov:used ex:config_i>` is generated.
- Additionally, the triple `<ex:output prov:wasDerivedFrom ex:input>` can be generated to show that the output data object was derived from the input data object.

Listing 2 shows an example of a phase in PROV format in the turtle syntax. The URIs of phase, tool, configuration, input, and output data objects are simplified to `ex:phase_i`, `ex:tool`, `ex:config_i`, `ex:input_i`, and `ex:output_i`, respectively. The URI of a tool `ex:tool` does not need an index, because we always have a finite set of known tools and no new tools get generated during tool execution. For every other subject, the URIs can be individualized e.g., by appending the index of phase **i**. For every following phase, the URI of the last output data object `ex:output_i` can be used without generating a new URI e.g.,

ex:input_i+1. In general, the URIs of data objects do not necessarily have to contain the terms input or output. This is just for explanatory purposes. It is only important that these URIs are unique.

```
ex:tool a prov:Agent .
ex:config_i a prov:Entity .
ex:phase_i a prov:Activity ;
           prov:used ex:input ;
           prov:wasAssociatedWith ex:tool ;
           prov:used ex:config_i .
ex:input_i a prov:Entity .
ex:output_i a prov:Entity ;
            prov:wasGeneratedBy ex:phase_i ;
            prov:wasDerivedFrom ex:input .
```

List. 2: Example of a phase in PROV format and the turtle syntax.

6 Conclusion & Future Work

In this paper, we provided the core concept and design of a framework to capture, store and retrieve provenance data of KG generation in a web portal and show an environment capable of provenance management. We presented four different applications to show the benefit of our proposed framework. However, the experimental test over this framework via a user study stays for our future work. Another future plan is extending the provenance capture and storage for tools with special requirements outside of our definition. This can involve e.g., multiple inputs and outputs of a tool. A possible solution to retrieve provenance data more efficiently and to enable retrieving triples based on the provenance data of the KG they belong to, can be achieved by saving provenance data in the same location as the KG is. In this way, a single query on the knowledge base can be issued, that filters results accordingly. Thus, we will handle this issue in our future work, too.

Bibliography

[Ba21] Babalou, Samira; Schellenberger Costa, David; Kattge, Jens; Römermann, Christine; König-Ries, Birgitta: Towards a Semantic Toolbox for Reproducible Knowledge Graph Generation in the Biodiversity Domain - How to Make the Most out of Biodiversity Data. In: INFORMATIK 2021. Gesellschaft für Informatik, Bonn, pp. 581–590, 2021.

[Be20] Berven, Arne; Christensen, Ole A; Moldeklev, Sindre; Opdahl, Andreas L; Villanger, Kjetil J: A knowledge-graph platform for newsrooms. Computers in Industry, 123:103321, 2020.

[Ha19] Haase, Peter; Herzig, Daniel M; Kozlov, Artem; Nikolov, Andriy; Trame, Johannes: metaphactory: A platform for knowledge graph management. Semantic Web, 10(6):1109–1125, 2019.

[HDBL17] Herschel, Melanie; Diestelkämper, Ralf; Ben Lahmar, Houssem: A survey on provenance: What for? What form? What from? The VLDB Journal, 26(6):881–906, 2017.

[Ho20] Hogan, Aidan: Web of data. In: The Web of Data, pp. 15–57. Springer, 2020.

[Kl16] Kluyver, Thomas; Ragan-Kelley, Benjamin et al.: Jupyter Notebooks-a publishing format for reproducible computational workflows. In: In Positioning and Power in Academic Publishing: Players,Agents and Agendas, F. Loizides and B. Schmidt (Eds.). volume 2016. IOS Press, pp. 87–90, 2016.

[Le13] Lebo, Timothy; Sahoo, Satya; McGuinness, Deborah; Belhajjame, Khalid; Cheney, James; Corsar, David; Garijo, Daniel; Soiland-Reyes, Stian; Zednik, Stephan; Zhao, Jun: Prov-o: The prov ontology. 2013.

[MMW13] Majumdar, Rupak; Meyer, Roland; Wang, Zilong: Provenance verification. In: International Workshop on Reachability Problems. Springer, pp. 21–22, 2013.

[PRSA18] Pérez, Beatriz; Rubio, Julio; Sáenz-Adán, Carlos: A systematic review of provenance systems. Knowledge and Information Systems, 57(3):495–543, 2018.

[SDA20] Staar, Peter W. J.; Dolfi, Michele; Auer, Christoph: Corpus processing service: A Knowledge Graph platform to perform deep data exploration on corpora. Applied AI Letters, 1(2):e20, 2020.

[SKR18] Samuel, Sheeba; König-Ries, Birgitta: ProvBook: Provenance-based Semantic Enrichment of Interactive Notebooks for Reproducibility. In: ISWC (P&D/Industry/BlueSky). 2018.

[SKR22] Samuel, Sheeba; König-Ries, Birgitta: End-to-End provenance representation for the understandability and reproducibility of scientific experiments using a semantic approach. Journal of Biomedical Semantics, 13(1):1, December 2022.

[Sy22] Sy, Mohameth François; Roman, Bogdan; Kerrien, Samuel; Mendez, Didac Montero; Genet, Henry; Wajerowicz, Wojciech; Dupont, Michaël; Lavriushev, Ian; Machon, Julien; Pirman, Kenneth et al.: Blue Brain Nexus: An open, secure, scalable system for knowledge graph management and data-driven science. Semantic Web, (Preprint):1–31, 2022.