Building Massive Knowledge Graphs using an Automated ETL Pipeline

Aaron Eberhart
metaphacts GmbH
Germany
ae@metaphacts.com

Peter Haase
metaphacts GmbH
Germany
ph@metaphacts.com

Wolfgang Schell
metaphacts GmbH
Germany
ws@metaphacts.com

ABSTRACT

Knowledge graphs are extremely versatile semantic tools, but there are current bottlenecks with expanding them to a massive scale. This concern is a focus of the Graph-Massivizer project, where solutions for scalable massive graph processing are investigated. In this paper we'll describe how to build a massive knowledge graph from existing information or external sources in a repeatable and scalable manner. We go through the process step-by-step, and discuss how the Graph-Massivizer project supports the development of large knowledge graphs and the considerations necessary for replication.

CCS CONCEPTS

• **Information systems** → **Data exchange**; **Mediators and data integration**.

KEYWORDS

Graph-Massivizer; metaphactory; ETL; RDF

ACM Reference Format:
Aaron Eberhart, Peter Haase, and Wolfgang Schell. 2024. Building Massive Knowledge Graphs using an Automated ETL Pipeline. In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE Companion '24), May 7–11, 2024, London, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3629527.3652900

1 INTRODUCTION

A knowledge graph is a flexible semantic tool that can serve as the foundation for a wide variety of information representation purposes and use cases, such as fast-tracking drug discovery and reducing research costs, smart manufacturing solutions to support human manufacturing planners, and global fraud detection and risk management. It also unlocks AI initiatives by enriching existing black-box solutions with machine-interpretable semantics and adding a layer of trust and transparency. While knowledge graphs are extremely versatile, there are current bottlenecks with expanding them to a massive scale. This concern is a focus of the Graph-Massivizer [3] project, where solutions for scalable massive graph processing are investigated. In this paper we'll describe how to build a massive knowledge graph from existing information or external sources in a repeatable and scalable manner. We go through the process step-by-step, and discuss how the Graph-Massivizer project supports the development of multiple large knowledge graphs and the considerations necessary for replication.

ICPE Companion '24 , May 7–11, 2024, London, United Kingdom

ACM ISBN 979-8-4007-0445-1/24/05…$15.00
https://doi.org/10.1145/3629527.3652900

1.1 Knowledge Graphs

Knowledge graphs are large networks of entities representing real-world objects, like people and organizations, and abstract concepts, like professions and topics, and their semantic relations and attributes. Knowledge graphs help organizations centralize, organize and understand internal data, often stored away in disparate sources. Depending on the volume of data a knowledge graph varies in size, ranging from a simple knowledge graph of a few to one with an extensive repository with millions of entities and interlinked relations.

There are multiple approaches to creating a knowledge graph this large in size, including using an ETL (extract-transform-load) or ELT (extract-load-transform) pipeline, which we'll explore in this paper. The ETL pipeline was developed as part of Graph-Massivizer, an EU-funded research project dedicated to researching and developing a scalable, sustainable and high-performing platform based on the massive graph representation of extreme data.

1.2 The Graph-Massivizer Project

The Graph-Massivizer project is developing a suite of five open-source software tools encompassing the sustainable life cycle of processing extreme data as massive graphs. These massive graphs support use cases such as green AI for a sustainable automotive industry, a data center digital twin for sustainable exascale computing and more.

The tools focus on holistic usability (from extreme data ingestion and massive graph creation), automated intelligence (through analytics and reasoning), performance modeling, and environmental sustainability tradeoffs, supported by credible data-driven evidence across the computing continuum. For example, the Graph-Massivizer's Graph-Inceptor tool is designed for creating knowledge graphs and storing graph data, offering two primary services:

(1) An extract, transform and load (ETL) pipeline requiring deployment to an IT cloud infrastructure consisting of servers, storage systems and databases
(2) A graph processing framework delivered as a Java library and made available in HPC clusters

Furthermore, the project consortium aims to create an integrated platform that is user-friendly and easy to deploy in enterprise environments The platform will tightly integrate the tools developed by Graph-Massivizer to provide a comprehensive offering.

2 KNOWLEDGE GRAPH CREATION PROCESS

Once a user has decided on their approach, they can start creating their knowledge graph, which involves many different tasks and phases. We'll explore in-depth some of the aspects to consider for the knowledge graph creation process.

2.1 FAIR Data Principles

A knowledge graph is only one system within an enterprise environment consisting of interconnected systems and data sources. One key aspect in this world of interconnected systems and data is following the FAIR [5] data principles, which ensures the reusability and interoperability of a knowledge graph, allowing users to enrich and extend their knowledge graph for various use cases and applications.

A knowledge graph supports the FAIR principles with the use of unique and persistent identifiers for entities, providing linked metadata describing the origin and modalities for accessing and using datasets, the use of semantic data models, and building on open standards for storing, accessing, and querying data. The following sections provide more information on how this works in detail.

2.2 Graph data model

Graphs are represented using the Resource Description Framework (RDF) [6], a semantic web standard for data interchange on the web. By using RDF, a graph can be expressed as a set of statements (or triples), each of which describes a single fact It allows for easy merging, linking and sharing of structured and semi-structured data across various systems and applications.

In this paper we only consider RDF-based graphs. There are other graph models, e.g.Labeled Property Graphs (LPG). Using RDF-star[1], any graph can be expressed, so RDF-star can also be used as a bridge to and from Labeled Property Graphs. RDF-star is an extension of RDF and also supports expressing statements on statements, which allows one to model edges with attributes.

2.3 Iterative approach

When creating a knowledge graph from scratch, it is useful to apply an iterative approach, involving:

(1) identifying source datasets and making them accessible
(2) defining a semantic data model using ontologies and vocabularies
(3) defining RDF mappings to convert from structured source data to RDF
(4) pre-processing source data (per file), e.g., to clean up data
(5) performing RDF conversion using the provided mappings
(6) post-processing intermediate results (per file), e.g., to create additional relations or aggregate data
(7) loading RDF data into the knowledge graph to persist the data in a graph database
(8) post-process intermediate results (whole graph), e.g., to create additional relations or aggregate data
(9) performing data validation to ensure the graph conforms to the defined data model.

[1]https://www.w3.org/groups/wg/rdf-star/publications/

Figure 1: Iterative KG Creation Approach

When violations are observed during data validation, the results can be used as a starting point to improve the pipeline. For example, source data can be fixed by performing data cleansing, adjusting the ontology or RDF mappings, or performing another iteration of the data integration process or ETL pipeline.

2.4 Providing dataset metadata

Data catalogs are a core building block for any FAIR data implementation, as they connect the available data assets with the knowledge graph. They support both interoperability as well as accessibility, as defined in the FAIR data principles.

In this approach, the data catalog is represented as a knowledge graph itself. It is semantically described with descriptive metadata and access metadata and is interlinked with other parts of the knowledge graph—such as ontologies and vocabularies—and it is embedded into and connected with data assets. Dataset descriptions (or data catalogs) are based on open and extensible W3C standards (e.g., DCAT) to make the data discoverable, accessible and traceable. With dataset descriptions, humans and machines (i.e., AI/ML algorithms) can consume data in context since the data is directly linked to the models and dataset descriptions, which themselves are based on open standards, are shareable and can even be queried all at once through a single, semantic query language.

2.5 Semantic Data Model

The next step in creating the knowledge graph is defining the data model. A knowledge graph typically follows one or multiple well-defined schemas which are specified using ontologies and vocabularies.

2.5.1 Ontologies. Ontologies are semantic data models that define the types of entities that exist in a domain and the properties that can be used to describe them. An ontology combines a representation, formal naming and definition of the elements (such as classes, attributes and relations) that define the domain of discourse. One may think of it as the logical graph model that defines what types (sets) of entities exist, their shared attributes and logical relations.

Ontologies can be specified using open standards like Web Ontology Language (OWL) [1] and Shapes Constraint Language (SHACL) [2].

2.5.2 Vocabularies. Vocabularies are controlled term collections organized in concept schemes that support knowledge graph experts, domain experts and business users in capturing business-relevant terminology. A term could include preferred and alternative labels (synonyms) in multiple languages and carries natural language definitions. Terms can be related to each other or defined as loosely related. The most common examples of different types of vocabularies are thesauri, taxonomies, terminologies, glossaries, classification schemes and subject headings, which can be managed using SKOS as an open standard.

2.6 RDF Mappings

The mapping process enables simple conversion, from a huge amount of source data to RDF, in an automated fashion. Converting structure data to RDF can be done by mapping certain elements and attributes from the source files to RDF data using a set of mapping rules.

As an example, all values of a column in a CSV file or a table in a relational database are mapped to RDF statements with the row's unique key being mapped to a subject IRI, the column to a predicate and the row value to the object position of a triple. Mapping rules can be provided either in a declarative way or programmatically.

2.6.1 Declarative mappings. Declarative mappings follow the no-code approach, meaning they can be defined using a simple text editor or visual tools, without requiring special programming skills.

The mappings are defined using the standardized Relational Mapping Language (RML). RML itself is also based on RDF, so both data model (ontology), mappings (RML maps) and instance data all use the same format. RML supports both tabular/relational and hierarchical data structures in formats like CSV, JSON or XML. Support for other formats can be provided as well.

RML defines just the mapping language. A wide range of implementations in the form of mapping engines (most of them open-source) are available. They can be used either as stand-alone tools or embedded into custom applications as a library.

2.6.2 Programmatic mappings. Implementing the mapping process using a custom program is the most flexible way to convert data to RDF. All means provided by the programming language and its ecosystem— such as frameworks and libraries—can be used (e.g., accessing data in various formats). Also, language-specific connectors, such as JDBC to access relational databases in the Java programming language, or web service connectors provide great flexibility. The biggest advantage is full control over the mapping process, as any kind of algorithm, data generation, use of caches and memory, navigating data structure or control flow is possible.

2.6.3 Choosing between declarative or programmatic approach. Using declarative mappings based on RML is the quickest and easiest way to implement mappings from structured data to RDF, as it follows a pre-defined approach that covers many use cases and formats and does not require special programming skills.

Only when declarative mappings do not suffice for the mapping at hand, should mappings be implemented as a custom program. While programmatic mappings allow for greater flexibility, this approach also requires more effort and programmatic skills, which are not necessarily available to people implementing a data pipeline.

In some cases where declarative mappings support most data structures to be mapped to RDF and only a few more complicated cases cannot be covered, a hybrid approach may be suitable. In that case, most mappings would be implemented declaratively in RDF and only a few special cases be handled by custom coding.

2.7 Performing pre- and post-processing

Besides converting source data as-is to RDF, sometimes additional steps are required to conform to the graph data model. This may be performed as pre- or post-processing steps, either on the original source before the RDF conversion or after.

Pre-processing steps typically work on the unit of a single source file. Typical examples are data cleansing, filtering of invalid data, splitting out units from numerical values, and datatype conversions to conform with certain numeric or date-time formats.

Post-processing steps may either be performed on the intermediate RDF files or the whole graph. Typical examples are tasks such as: specify the named graph for a set of statements, update graph metadata, such as the timestamp of last update of a dataset based on source data, and others.

2.8 Data Ingestion

The result of the previous steps is composed of a set of files in RDF format. This set of files may already be used to distribute the data in RDF format, e.g. as a data product.

As a next step, ingesting this file-based dataset into a graph database provides a base for easy querying and graph analytics supported by the database engine.

In addition to loading the data into the database for querying using the SPARQL query language, creating a full-text search index enables additional capabilities when searching for textual data in the graph. This is typically handed off from the database to specialized and tightly integrated full-text search engines like Lucene[2], Solr[3], or Elasticsearch[4].

2.9 Performing data validation

Once all data has been converted to RDF and is ingested in the database, it can be submitted to a data validation to ensure good data quality.

When defining the ontology using OWL and SHACL, the model description can be used to automatically validate the database and ensure that data follows the defined model. This can be done using a so-called SHACL engine, which verifies that the data in the database adheres to the shapes defined in the ontology. SHACL engines are provided by (commercial) RDF databases as well as open-source projects or commercial tools such as metaphactory.

[2]https://lucene.apache.org/
[3]https://solr.apache.org/
[4]https://www.elastic.co/

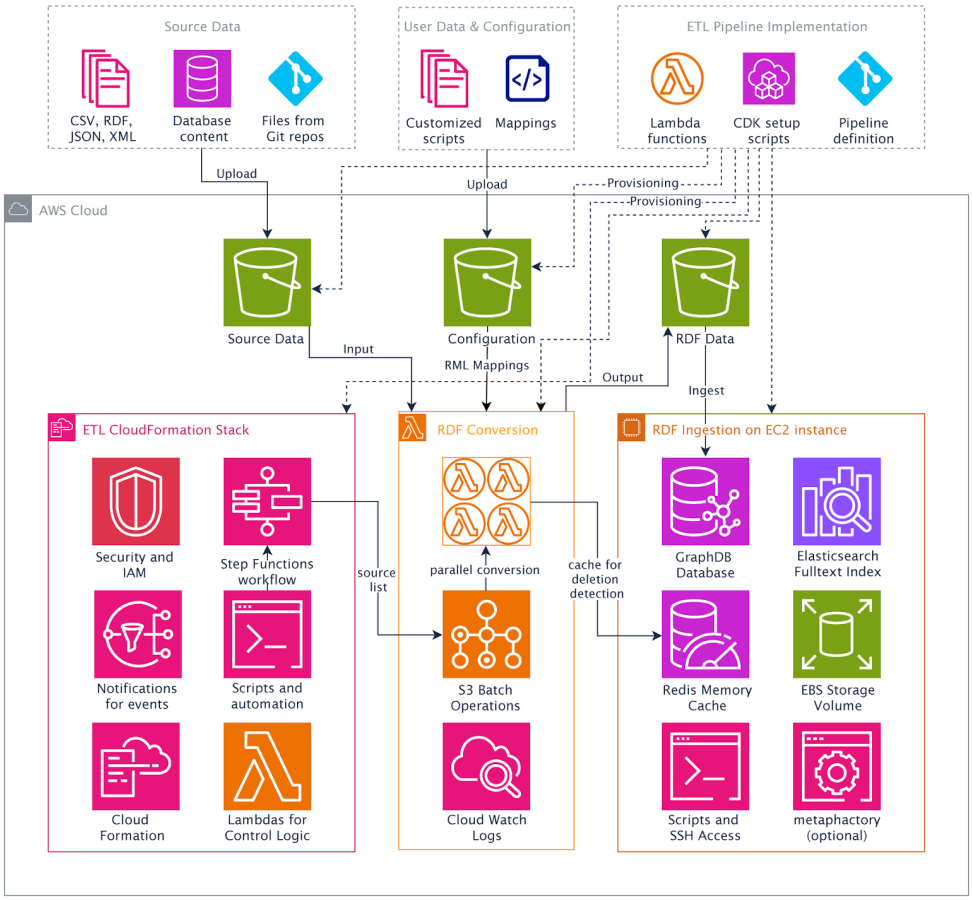

Figure 2: ETL Architecture

3 ARCHITECTURE

The pipeline uses a multitude of AWS services to implement the RDF conversion and ingestion process with a cloud-native approach resulting in high parallelization and efficient use of resources. Details on this architecture can be found in Figure 2.

4 EXAMPLE

The Graph Massivizer use cases from the data center, industrial, and financial domains are based on either commercial, internal or sensitive datasets, so they cannot be used for a public demonstration. Instead, we'll use the Dimensions Covid Dataset[5] [4] as an example of a large, publicly available dataset to create a scientific knowledge graph using the ETL pipeline.

The dataset provides information on global publications, academic papers, authors, research organizations, funders, grants, datasets and clinical trials. The zipped dataset (1.09GB) is available for download on Figshare. The data files are in CSV format, the fields are described in the documentation of the main Dimensions dataset (although not all documented fields are available in this publicly available subset).

The semantic data model and dataset description as well as the corresponding RML mappings are provided as an example in the ETL pipeline Git repository[6].

5 FUTURE WORK

The ETL pipeline will be extended and integrated to work with the Graph-Massivizer toolkit. This process will continue through the duration of the project, adapting to evolving project needs and use cases.

In parallel with Graph-Massivizer developments, work on the ETL pipeline will also provide and extend capabilities with the metaphactory platform.

Acknowledgement This project has received funding from the European Union's Horizon Research and Innovation Actions under Grant Agreement N° 101093202.[7]

[5] https://www.dimensions.ai/covid19/

[6] https://github.com/metaphacts/metaphacts-etl-pipeline

[7] More information available at: https://graph-massivizer.eu/

REFERENCES

[1] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah McGuinness, Peter Patel-Schneijder, and Lynn Andrea Stein. 2004. *OWL Web Ontology Language Reference*. Recommendation. World Wide Web Consortium (W3C). See http://www.w3.org/TR/owl-ref/.

[2] Dimitris Kontokostas and Holger Knublauch. 2017. *Shapes Constraint Language (SHACL)*. W3C Recommendation. W3C. https://www.w3.org/TR/2017/REC-shacl-20170720/.

[3] Radu Prodan, Dragi Kimovski, Andrea Bartolini, Michael Cochez, Alexandru Iosup, Evgeny Kharlamov, Jože Rožanec, Laurenţiu Vasiliu, and Ana Lucia Vărbănescu. 2022. Towards Extreme and Sustainable Graph Processing for Urgent Societal Challenges in Europe. In *2022 IEEE Cloud Summit*. 23–30. https://doi.org/10.1109/CloudSummit54781.2022.00010

[4] Dimensions Resources. 2021. Dimensions COVID-19 publications, datasets and clinical trials. (9 2021). https://doi.org/10.6084/m9.figshare.11961063.v42

[5] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* 3 (2016).

[6] David Wood, Markus Lanthaler, and Richard Cyganiak. 2014. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. W3C. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.