GraphGuard: Enhancing Data Quality in Knowledge Graph Pipelines

Rene Dorsch[1,*], Michael Freund[1], Justus Fries[1] and Andreas Harth[1]

[1]*Fraunhofer IIS, Fraunhofer Institute for Integrated Circuits IIS Nordostpark 84 Nuremberg Germany 90411*

Abstract

We present GraphGuard, a data validation framework to improve the data quality of pipelines to populate knowledge graphs. The inputs for these pipelines often come from different sources, requiring various approaches for validating the data against different defects. This requirement leads to different formats for validation reports, which reduces contextual, representational, and accessible quality dimensions of data validation. The proposed framework consists of QualityContracts and Guardians. QualityContracts encapsulate the necessary data validation requirements in both human and machine-readable formats. Software agents, called Guardians, use the machine-readable format to execute validation methods. We validate the practicality of our framework on a deployed data processing pipeline at a large European airport over several months of data. A comparative analysis between a basic data processing pipeline and a pipeline using our framework showed improvements in the data quality criteria of believability, interpretability, ease of understanding, consistency of representation, conciseness of representation, and accessibility.

Keywords

Knowledge Graph, Data Validation, Data Quality, Process Optimization

1. Introduction

Knowledge graphs (KG) are powerful tools used across various sectors to incorporate and integrate data from various heterogeneous sources, making them critical for various organizations. Industrial organizations [1, 2, 3], tech giants[4], and worldwide institutions[5] use knowledge graphs as a fundamental component of their data-driven strategies for tasks, such as data aggregation, data analysis, process optimization, and decision-making. Given this extensive usage, ensuring high data quality within these graphs of knowledge is crucial.

To incorporate data into knowledge graphs, data integration pipelines are frequently used [6]. These pipelines consist of multiple loosely coupled components, each executing different tasks to process and transform data. Validation techniques based on the Resource Description Framework (RDF), such as SHACL [7] and SHEX [8], are frequently employed to maintain accuracy in knowledge graphs. However, it's crucial to address data defects, e.g. unavailable data

SemIIM'23: 2nd International Workshop on Semantic Industrial Information Modelling, 7th November 2023, Athens, Greece, co-located with 22nd International Semantic Web Conference (ISWC 2023)

*Corresponding author.

rene.dorsch@iis.fraunhofer.de (R. Dorsch); michael.freund@iis.fraunhofer.de (M. Freund); justus.fries@iis.fraunhofer.de (J. Fries); andreas.harth@iis.fraunhofer.de (A. Harth)

0000-0001-6857-7314 (R. Dorsch); 0000-0001-7116-9338 (M. Freund); 0000-0003-3433-7245 (J. Fries); 0000-0002-9421-8566 (A. Harth)

points or outliers, at their source [9]. Addressing a wide range of data defects requires diverse validation methods, which can differ based on elements such as the type of data they can validate, the rules they employ, the procedures they follow, and the presentation of validation results. As a result, various methods tailored for diverse data types and defect types, along with strategies like functional redundancy [10], are used to validate data across a single processing pipeline. These validation methods produce data in the form of validation reports based on the inspected data. Given the variety of methods used for validation, there is a trade-off between ensuring high accuracy of data and maintaining homogeneity in validation reports. When high accuracy is assured, the validation report becomes more heterogeneous, thereby reducing the contextual, representational, and accessibility criteria of data quality [11]. Lowering these data quality criteria reduces the usability of the validation reports for use cases such as quality assessment [12], assurance [13], control [14], and monitoring [15, 16], data auditing and compliance[1], and data analytics [17, 18]. Previous research is limited by its design. These works tend to focus on reducing heterogeneity by restricting elements of data validation, such as the type of data being validated, the validation procedures employed, and the rules governing data validation [19], and overlook the potential of using diverse validation methods in a compatible manner. For instance, some validation services might require adhering to a specific format [19], may be proprietary [9], or may be limited to a particular data format [7, 8, 20, 21].

Therefore, we investigate the research question, "How can the data quality of heterogeneous data validation methods be improved in the context of data processing pipelines for knowledge graphs?"

Concretely, we introduce a framework that defines validation rules as reusable components. These components are available to a software agent that implements the necessary software to run the validation rules and generate validation reports. This report is available in the processing pipeline and within a knowledge graph, which contains machine and human-readable descriptions of validation rules and the reports in RDF. Our approach is capable of executing the rules established in various validation frameworks while presenting a comprehensive, homogeneous report of the validation procedure.

The main contributions of our work are two-fold. First, we formulate an ontology to represent validation rules, their execution process, and the capabilities of a software agent to execute them. Second, we conduct a comparative evaluation of our framework in an actual real-world scenario.

We present related work and the derived main requirements in section 2. Section 3 introduces our framework for handling heterogeneous data validation reports. We then describe a use case from a European airline in section 4. The use case forms the basis for the evaluation in section 5. Finally in section 6, we provide a summary of our proposal, current limitations, and future work.

2. Preliminaries

Validation is defined by the European Statistical System (ESS) [19], as a "... process which ensures the correspondence of the [...] data with a number of quality characteristics." Hence,

[1]cf. https://cros-legacy.ec.europa.eu/content/overview-data-and-metadata-exchange-ess_en

here we present related research on comprehensive data validation frameworks; frameworks to describe rules for quality characteristics; and the derivation of executable processes from knowledge graphs. Based on the related work, we establish various requirements to design a framework that is capable of handling various validation methods.

2.1. Related Work

The data validation framework of the ESS [19] validates statistical data from different European countries to generate EU-wide reports. The validation system requires a domain-specific language, called Validation and Transformation Language, that was designed to validate the structure and content of datasets. The reports from the ESS are described in a machine- and human-readable format to improve transparency and reuse of the reports. Ericson [9] presented their validation framework for machine learning-enabled software systems. Similar to the validation system of the ESS, it is able to validate the structure (e.g. size of dataset), and content (e.g missing values, range of values, misformats, outliers). To handle the different required validation methods, they developed a proprietary validation library. In summary, both frameworks are able to validate datasets on their structure and content but require the migration to a specific validation system.

Cross-organizational validation methods primarily rely on open protocols defined by contracts or specifications. Examples include the frictionless-schema [21], which validates tabluar data; the json-schema [20] for JSON-serialized data; and SHACL [7] or SHEX [8], for RDF-described data. However, these methods predominantly focus on schema or pattern validation, necessitating additional frameworks to validate other data characteristics, such as outliers or duplicates.

Zheng et al. [22] employ knowledge graphs to depict and operationalize machine-learning pipelines in RDF format. These pipelines are made executable using a specialized library, which integrates machine-executable semantics into the code. However, a notable limitation of this method is its reliance on a library, hiding the underlying rules and algorithms. Similar to the preceding methods, this approach raises questions about transparency and modifiability. Specifically what the algorithms execute, how the rules can be accessed, and how new methods for processing data can be integrated.

2.2. Requirements

To derive data quality requirements for data validation, we used the framework of data quality introduced by Wang et al. [11] as it is well-established, provides detailed characterizations of data quality dimensions, and can be used for data quality assessment [23] and improvement [24]. Wang et al. identified 16 criteria to characterize data quality and grouped them into four dimensions: intrinsic, contextual, representational, and accessible. Based on the dimension we derived the following requirements for handling heterogeneous data validation. The *intrinsic quality dimension* covers the extent to which data values correspond to reality. To maintain a high level of intrinsic data quality, any validation method must be allowed to achieve the highest accuracy in knowledge graphs [9, 10, 25]. The *accessibility quality dimension* is concerned with the extent to which data is available or obtainable. To enhance the accessibility data quality, it is necessary to report validation rules and the outcomes of such techniques separately from their implementation. This ensures that users can quickly and easily retrieve information about the implemented techniques in the pipeline. In addition, knowledge graphs ease the connection of data with its validation reports. Therefore, the outcomes of the validation methods should be integrable in the knowledge graph. The *representational quality dimension* focuses on the extent to which data can be represented in an understandable and transparent manner. This means that reports should be concise to facilitate the varying needs of different users. Additionally, they should be presented in a human and machine-readable format [26] enabling analysis based on the reports for different use cases [25]. The *contextual quality dimension* focuses on the extent to which data is applicable or relevant to the tasks of a data consumer. To provide higher contextual data quality, validation constraints should be quickly accessible, reusable, and modifiable [9].

3. GraphGuard Framework

Based on the given requirements, we developed GraphGuard a framework to support heterogenous data validation. In this section, we present the two main components, i.e., QualityContracts and Guardians. QualityContracts are formal specifications that define a set of constraints for acceptable data. Guardians are software agents responsible for enforcing these constraints by validating data against QualityContracts. Based on the validation result, Guardians generate a QualityValidationReport.

The general data model for the developed ontology[2] is provided in figure 1. We reused existing ontologies to promote interoperability and to build upon established standards. We used the Data Quality Vocabulary (DQV)[3] to express data quality metadata, Open Digital Rights Language Ontology (ODRL)[4] and the Profiles Vocabulary (PROF)[5] to express the contracts and constraints, Data Catalog Vocabulary (DCAT)[6] and the Software Package Data Exchange Ontology (SPDX)[7] to describe data exchange, and the Provenance Ontology (PROV)[8] for provenance information.

3.1. QualityContracts

A **QualityContract** is an RDF-document, embedding descriptions about the general purpose of the validation procedure and metadata about the contract usage for a specific dataset. To improve readability, a QualityContract always contains a human- and machine-readable description of its content. The human-readable description contains an association with the author and the time when the contract was generated. The human-readable descriptions in QualityContracts enhance the understandability of the contracts for data stewards and domain experts, while machine-readable formats allow for automated processing by software agents like Guardians.

[2]http://www.purl.org/graphguard/ontology
[3]https://www.w3.org/TR/vocab-dqv/
[4]https://www.w3.org/TR/odrl-model/
[5]https://www.w3.org/TR/dx-prof/
[6]https://www.w3.org/TR/vocab-dcat-2/
[7]https://www.spdx.org/rdf/terms/
[8]https://www.w3.org/TR/prov-o/

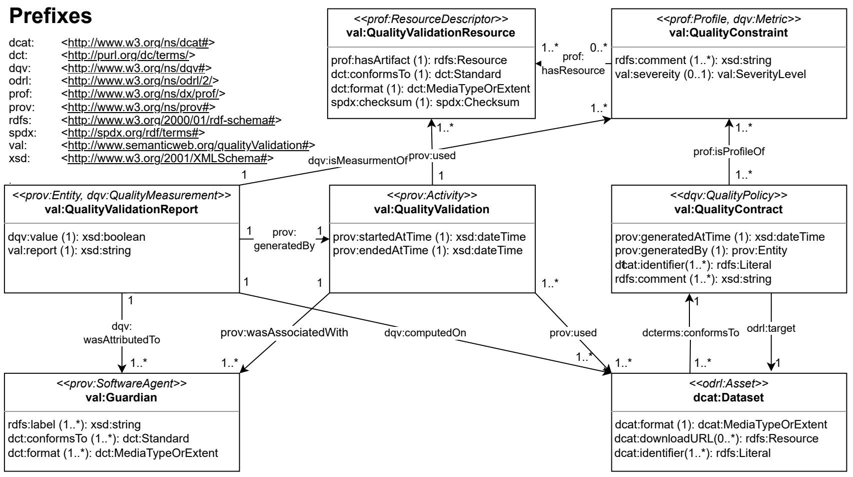

Prefixes
dcat: <http://www.w3.org/ns/dcat#>
dct: <http://purl.org/dc/terms/>
dqv: <http://www.w3.org/ns/dqv#>
odrl: <http://www.w3.org/ns/odrl/2/>
prof: <http://www.w3.org/ns/dx/prof/>
prov: <http://www.w3.org/ns/prov#>
rdfs: <http://www.w3.org/2000/01/rdf-schema#>
spdx: <http://spdx.org/rdf/terms#>
val: <http://www.semanticweb.org/qualityValidation#>
xsd: <http://www.w3.org/2001/XMLSchema#>
<<prof:ResourceDescriptor>>
val:QualityValidationResource
prof:hasArtifact (1): rdfs:Resource
dct:conformsTo (1): dct:Standard
dct:format (1): dct:MediaTypeOrExtent
spdx:checksum (1): spdx:Checksum
<<prof:Profile, dqv:Metric>>
val:QualityConstraint
rdfs:comment (1..*): xsd:string
val:severeity (0..1): val:SeverityLevel
1..*
prof:
hasResource
0..*
1..*
1..*
dqv:isMeasurmentOf
prov:used
1
1
1..*
prof:isProfileOf
1..*
<<prov:Entity, dqv:QualityMeasurement>>
val:QualityValidationReport
dqv:value (1): xsd:boolean
val:report (1): xsd:string
1
prov:
generatedBy
1
<<prov:Activity>>
val:QualityValidation
prov:startedAtTime (1): xsd:dateTime
prov:endedAtTime (1): xsd:dateTime
<<dqv:QualityPolicy>>
val:QualityContract
prov:generatedAtTime (1): xsd:dateTime
prov:generatedBy (1): prov:Entity
dcat:identifier(1..*): rdfs:Literal
rdfs:comment (1..*): xsd:string
1
1
1..*
1
1
dqv:
wasAttributedTo
prov:wasAssociatedWith
dqv:computedOn
prov:used
dcterms:conformsTo
odrl:target
1..*
1..*
1..*
1..*
1..*
1
<<prov:SoftwareAgent>>
val:Guardian
rdfs:label (1..*): xsd:string
dct:conformsTo (1..*): dct:Standard
dct:format (1..*): dct:MediaTypeOrExtent
<<odrl:Asset>>
dcat:Dataset
dcat:format (1): dcat:MediaTypeOrExtent
dcat:downloadURL(0..*): rdfs:Resource
dcat:identifier(1..*): rdfs:Literal

Figure 1: Organization of the ontology of GraphGuard

QualityContracts have at least one QualityConstraint for a single dataset describing validation constraints on the data. **QualityConstraints** are specialized `prof:Profiles` designed to specify constraints of QualityContracts in a human- and machine-readable format. Violations of these constraints can be defined with different levels of severity. For instance, the three levels "info", "warning", and "error" can be used. The "info" level is utilized to describe potential defects not affecting the data validation any further. The "warning" level is used to provide information about potential issues identified by the QualityConstraint. The "error" level signifies a violation of a validation rule that needs to be fulfilled. If data does not fulfill the constraint, the validation process interrupts, a report is sent to a knowledge graph, and the process stops. Not enforced constraints recognize the defect in the data and report the defect to the graph, but do not enforce handling the cause. This reporting procedure enables traceability of defects in the data without interrupting the processing pipeline.

QualityValidationResources are defined for the execution of QualiyConstraints. QualityValidationResources describe machine-readable and executable validation rules. We separated QualityConstraints from specific resources, to provide a modular approach to data validation. This separation ensures that QualityContracts remain flexible and can easily accommodate different types of validation rules without overhauling the entire constraint. For instance, a QualityContract can be defined for a dataset of measurements from different sensors, where different QualityConstraints are defined for different types of defects. A QualityConstraint can be defined to validate the data from the dataset against outliers. Multiple QualityValidationResources implement different methods, e.g., time-based, depth-based, or distance-based methods, that detect different types of outliers and ensure higher accuracy in the data. Hence, our approach enables an easy extension of existing QualityConstraints by further QualityValidationResources.

The QualityValidationResources are essentially the functional components of a QualityContract. The resources implement technical or domain-specific validation rules to evaluate the results of a pipeline component. Technical validation rules describe assumptions about data types and meta-properties of a dataset, such as the existence or uniqueness of values. Domain-specific validation rules focus on rules related to a specific domain of interest. For instance, expert knowledge about the processed data can be expressed with that.

```
[] a val:QualityContract ;
odrl:target [
    a dcat:DataSet ;
    dcat:identifier "my_sensor_data.csv";
    dcat:format "text/csv"] ;
prov:generatedAtTime "2023-08-14T07:51:28.607114" ;
prov:wasGeneratedBy :me ;
prof:isProfileOf <my_constraint>.
<my_constraint> a val:QualityConstraint ;
    rdfs:comment "The data must always have a MAC-Address, a datatype,
    ↪  and a single datum in each row.";
    prof:hasResource [
        a val:QualityConstraintResource;
        prof:hasArtifact <my_sensor_constraints.yml>;
        dcat:format "application/yaml";
        dcat:conformsTo <https://specs.frictionlessdata.io/data-package/>
        ↪  ].
```

```
fields:
    - name: MAC
      type: string
      format: default
      pattern: "^[0-9a-f]{2}(:[0-9a-f]{2}){5}$"
    - name: value
      type: number
      constraints:
        required: true
    - name: type
      type: string
      constraints:
        enum: ["Temperature", "Pressure",
↪   "Humidity"]
```

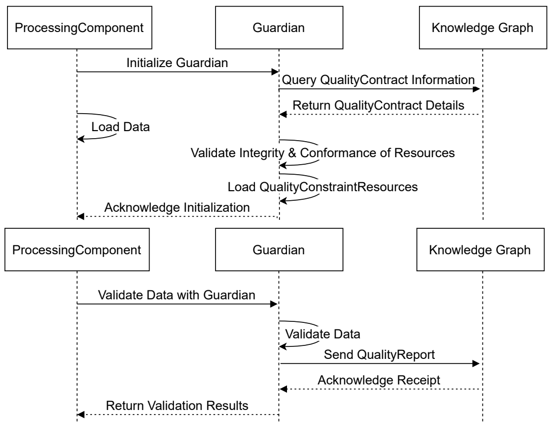
Figure 2: An instantiation of a quality contract with the QualityConstraint and QualityValidationResource (left) and the Artifact of the QualityValidationResource (right)

To allow software agents the interpretation and execution of QualityValidationResources each resource requires an **Artifact** (`prof:hasArtifact`) that refers to machine-executable code. An Artifact can be any validation algorithm ensuring the QualityValidationResource. To ensure the integrity of the Artifact the QualityValidationResources provides a Checksum. The Guardian interprets the Artifact based on the specified format (`dct:format`) and the required environment (`dct:conformsTo`) that is defined as a standard. Figure 2 illustrates an example of a QualityContract for a dataset that contains tabular data. It shows that a dataset called "*my_sensor_data.csv*" with the media type "text/csv" needs to fulfill a QualityConstraint. The QualityConstraint is a constraint restricting the format of the MAC address and the type of sensor information provided. To validate the data from "*my_sensor_data.csv*", the artifact "*my_sensor_constraints.yml*" will be used by a Guardian that can interpret frictionless-schema specifications (section 3.2) that are defined in the "application/yaml" format.

3.2. Guardians

Guardians are `prov:SoftwareAgents`, which enforce QualityContracts, by validating data in data processing pipelines. Their primary role in the pipeline is to act as gatekeepers, ensuring that only data conforming to the QualityContracts, gets processed. Figure 3 illustrates the execution sequence of a Guardian for a dataset that will be validated and the resulting QualityValidationReport.

At the start of data processing, the data is loaded and the Guardian is initialized. The initialization process involves querying contracts from a designated knowledge graph that contains the QualityContracts for the particular dataset. QualityConstraints and QualityValidationResources are obtained from the QualityContracts. The integrity of the QualityValidationResources is verified using the supplied Checksum and compared against the standards that the Guardian can conform to (`dct:conformsTo`). In the presented example, the Guardian can conform to QualityConstraintResources developed as frictionless-schemas or Python code that does not require any specific libraries. In the final phase of the initialization, the Guardian loads the Artifacts of QualityValidationResources to enable validation.


ProcessingComponent
Guardian
Knowledge Graph
Initialize Guardian
Query QualityContract Information
Return QualityContract Details
Load Data
Validate Integrity & Conformance of Resources
Load QualityConstraintResources
Acknowledge Initialization
ProcessingComponent
Guardian
Knowledge Graph
Validate Data with Guardian
Validate Data
Send QualityReport
Acknowledge Receipt
Return Validation Results

```
<my_Guardian> rdfs:label "My Guardian";
    dct:accessURL <http://purl.org/graphguard/guardian>;
    dcat:conformsTo <http://python.org>,
    ↪  <https://specs.frictionlessdata.io/data-package/>.
[] a val:QualityValidationReport;
    dqv:computedOn <my_data_set>;
    dqv:isMeasurementOf <my_constraint>;
    dqv:result True;
    val:report """
    Validation for my_sensor_data.csv was successful. No
    ↪  Validation errors detected.
    """;
    prov:generatedAtTime "2023-08-14T07:51:28.607114" ;
    prov:generatedBy [
        a val:QualityValidation;
        prov:startedAtTime "2023-08-14T07:51:27.43123" ;
        prov:endedAtTime "2023-08-14T07:51:28.605312";
        prov:used <my_validation_resource>, <my_data_set>;
        prov:wasAssociatedWith <my_Guardian>
        ].
```

Figure 3: Sequence Diagram of the validation with Guardians (left) and QualityValidationReport (right)

The data is validated against rules (e.g., according to figure 2) induced by the QualityValidationResources, which produces a QualityValidationReport (e.g., according to 3). The QualityValidationReport presents the outcome in a format that is understandable to both machines (`dqv:result`) and humans (`val:report`). Furthermore, it provides insight on the QualityValidation process that was conducted. The machine-readable result can either be true or false, representing a successful or failed validation of the data. To enable tracking of validation activities and to document the results, the QualityValidationReport is sent to the knowledge graph. With this information, an assessment can be made regarding the effectiveness and compliance of validation methods. If the results indicate the need, a notification system may be activated to inform users about the current status of the validation step. After successful validation, the Guardian reports its results back to the pipeline. If the validation fails, the result is passed to the processing pipeline, allowing the handling of the exception.

As a proof of concept, we implemented the Guardian in Python as a reusable library[9]. The provided library enables Guardians to automatically query a knowledge graph for QualityContracts for a given dataset; interpret QualityValidationResources for an implemented standard (currently pure Python code, data provided from the frictionless schema, and SHACL rules); validate data based on the provided validation rules and methods; and send validation reports to a specified knowledge graph.

4. Case Study

We implemented the framework at the Munich Airport that handles an average of 80.000 passengers daily. The airport employs the knowledge graph for various applications, including generating insights about the performance of the baggage handling system, analyzing baggage traces with process mining, and optimizing the luggage handling process with data-driven value stream analysis. The deployment of the knowledge graph pipeline at the airport is shown in figure 4. Various cleaning (step 1), processing (step 2), transformation (step 3), and mapping

[9]https://github.com/wintechis/guardian

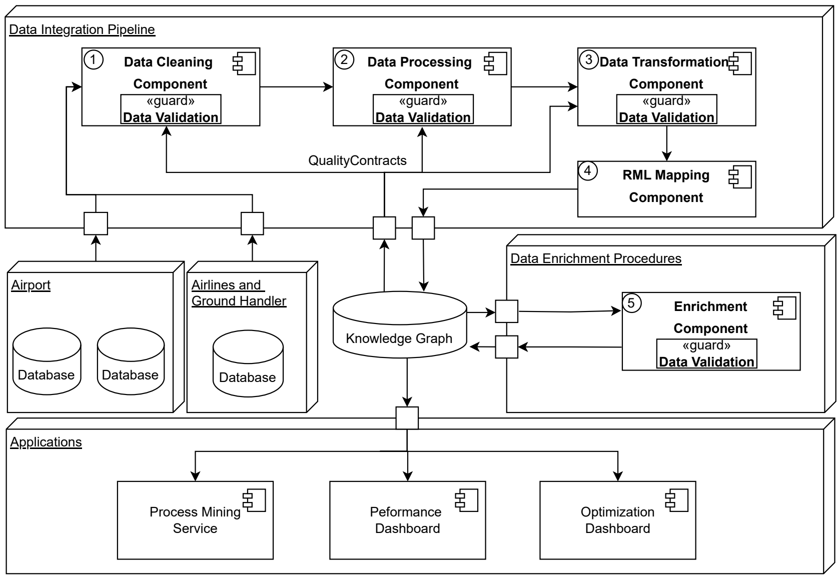

Data Integration Pipeline
1 Data Cleaning Component «guard» Data Validation
2 Data Processing Component «guard» Data Validation
3 Data Transformation Component «guard» Data Validation
4 RML Mapping Component
QualityContracts
Airport
Database
Database
Airlines and Ground Handler
Database
Knowledge Graph
Data Enrichment Procedures
5 Enrichment Component «guard» Data Validation
Applications
Process Mining Service
Peformance Dashboard
Optimization Dashboard

Figure 4: Deployed data transformation pipeline at the airport

rules (step 4) are applied to the source data on a daily basis. Data is automatically extracted from the graph for further analysis and enriched with additional information (step 5). The data is validated at each step, and the validation reports are incorporated into the knowledge graph to improve transparency. Implemented validation methods are used to validate data, for instance against the following defects:

- **Duplicates**: The baggage identifiers used by each airline are repeating over a specific range of numbers. This requires (in step 1) to recognize and handle different baggages with the same identifier.
- **Corrupted data**: Data may also be compromised by defects invoked during check-in. During check-in, employees of the airline can add information in a free text field. This can cause corruption during serialization (step 1), which affects the processing of the data.
- **Unavailable data attributes**: The baggage handling process relies heavily on messages from airlines. These are standardized messages [27], that include many optional information in different formats. The data needs to be validated before processing (in step 3 and step 5) to prevent the generation of URIs for non-existent information during mapping.
- **Contextualized data**: The standardized messages include information, that can only be understood within the context (e.g. last 5 messages) of multiple messages. To prevent defects, data needs to be validated (in step 2) such that some information can not exist twice in the same context.

We have implemented our framework on top of the deployed processing pipeline for the knowledge graph. The framework replaces previous validation procedures by introducing

Guardians that perform the same validation procedures. The validation results are stored in the same knowledge graph as the integrated data, which allows faster access to the validation reports. This implementation has improved the transparency of the processing pipeline. It allows data analysts to familiarise themselves with the data and its processing, build greater confidence in the data, and provide a clear and traceable perspective on the results of validation procedures for different data sources.

5. Evaluation

In this section, we evaluate the framework according to the quality dimensions outlined in section 2. First, we present the evaluation framework and results. Second, we provide a detailed comparison of the different dimensions of data quality.

5.1. Comparative Analysis

We evaluated our framework based on the Data Quality Dimensions [11] - accessibility, intrinsic, contextual, and representational. Wang et al. provided for each of the dimensions further criteria to discuss the data quality of systems. Data in the context of our framework is a triple based on the used Dataset, the QualityContracts, and the QualityValidationReports. Based on these criteria, table 1 illustrates the results of comparing two versions of the pipeline presented in section 4; one without the data validation framework (baseline) and one where the framework has been applied. Evaluation results are expressed through arrows, which represent changes in comparison to the baseline. Results are excluded for quality criteria when the framework does not directly affect the metric (represented as $\varnothing$).

Table 1
Quality comparison between a pipeline implementing the GraphGuard framework and a baseline.

Quality Dimension	Quality Criteria	Comparison
Intrinsic	Believability	$\nearrow$
	Accuracy	$\rightarrow$
	Objectivity	$\varnothing$
	Reputation	$\varnothing$
Contextual	Value-added	$\varnothing$
	Relevance	$\varnothing$
	Completeness	$\varnothing$
	Timeliness	$\rightarrow$
	Appropriate amount of data	$\nearrow$
Representational	Interpretability	$\nearrow$
	Ease of understanding	$\nearrow$
	Representational consistency	$\nearrow$
	Concise representation	$\nearrow$
Accessibility	Accessibility	$\nearrow$
	Access secure	$\varnothing$

5.2. Intrinsic Data Quality

Intrinsic Data Quality focuses on the extent to which data values are in conformance with reality. The baseline implementation, evaluated the validation rules as an embedded part of the pipeline, and its results were only used for the processing. In comparison to the baseline, our framework improves **Believability** - the extent to which data is accepted or regarded as true or credible - by enabling users to query and retrieve QualityContracts that have been used for validation of the datasets. Furthermore, it allows the users to retrieve the concrete QualityConstraints with their implementation, thus improving the credibility of the used data. **Accuracy** is the extent to which data is correct and certified error-free. Unlike the baseline approach, our framework separates the application logic of the pipeline from the validation rule. The validation rule is validated by Guardians using QualityContractResources for the computation. The Guardians use the same libraries and tools to understand the validation rules as the baseline implementation. Therefore, no change in the accuracy can be observed. **Objectivity** is the extent to which the data is unbiased and impartial. Our framework does not directly affect the objectivity of the dataset that needs to be validated but the objectivity of the validation rules. The validation rules are separated from the implementation in the pipeline, reducing the potential of developing ad-hoc rules. Therefore, the framework provides means to develop the processing pipeline and the validation rules separately, which may improve the overall objectivity. Similar to Objectivity, **Reputation** - the extent to which data is trusted or highly regarded in terms of their source or content - is not affected directly. However, the validation rules can be easier accessed through the knowledge graph providing them, and Quality reports are provided, building trust in the content of the data. With this information, data can be more transparently assessed and thereby enhance its overall reputation.

5.3. Contextual Data Quality

Contextual Data Quality relates to the extent to which data applies to a data user's tasks. The **Value-added** criteria refers to the extent to which users gain an advantage from the provided data. **Relevancy** measures the extent to which data is applicable and helpful for the application. **Completeness** defines the extent to which breadth, depth, and scope of the data is necessary. Despite their importance, these criteria are outside the scope of our proposal. The three quality criteria are mainly focused on the provided use case. Nevertheless, we believe that applying our framework can aid in evaluating the quality of the knowledge graph pipeline for adopting new use cases. The framework offers easy-to-access information about existing data sources, characterizes the data, and offers reusable validation methods. This approach not only makes evaluations and assessments of the pipeline's data relevancy and completeness easier but also supports the creation of new knowledge graph population pipelines.

Timeliness measures the availability of new data points to a user. Our approach demands additional time for the validation process, given that the QualityValidationReport is published on the knowledge graph before the result is delivered to the processing element. As a result, an improvement in the timeliness can be reported to the end user, while a decrease in the timeliness can be observed for the processing component. **Appropriate amount of data** is defined as the extent to which the quantity of available data is appropriate. Our framework generates for each dataset and each QualityConstraint a QualityValidationReport. Similarly, the baseline produces a result for each validation rule. However, with our framework, more data is available about the validation, describing not only the result but also the validation time, the involved QualityConstraint, and the involved software agent.

5.4. Representational Data Quality

Representational Data Quality focuses on the extent to which data is presented in a way that is understandable and clear for the data user‘s task. It incorporates **Interpretability** - the extent to which data is in an appropriate language and data definitions are clear - that is enhanced by our framework as it utilizes RDF to describe QualityContracts, QualityValidation for the validation process, QualityConstraints for the validation rule, and QualityValidationResources for the explicit implementation. Additionally, our framework thoroughly incorporates existing ontologies to enhance both ease of understanding and reusability (**Ease of understanding** - the extent to which data is unambiguous and easy to understand). The baseline utilizes different validation frameworks for different validation tasks. This produces different types of outcomes, impacting its interpretability and ease of understanding. Compared to the baseline pipeline, consistent reporting of validation information from the data is provided, which can be used in applications to indicate the health of the pipeline and databases. The results are generated in two formats - a machine-readable version that contains information solely about the validation outcome and a longer, human-readable format that presents details generated by the different validation methods. Therefore, **Representational consistency** - the extent to which data is always presented in the same format and is compatible with previous data - and **Concise presentation** - the extent to which data is presented compactly without being overwhelming - are improved over the baseline pipeline.

5.5. Accessibility Data Quality

Accessibility Data Quality focuses on the extent to which data is available or obtainable. **Access security** - the extent to which access to data can be restricted and therefore kept secure - is beyond the scope of our work, as it relates more to the design of the data transformation pipeline, rather than the quality of the data or the validation procedure. Nevertheless, we retrieve validation rules from Artifacts provided by QualityValidationResources. These Artifacts can come from multiple sources, each potentially secured by different methods. Within the pipeline, the integrity of the rules is validated by a provided checksum and then executed in the Guardian. This ensures that access security is maintained. In comparison to the baseline pipeline, our approach improves **Accessibility** - the extent to which data is available or easily and quickly retrievable. By employing RDF to represent QualityContracts and QualityValidations, our framework simplifies the querying of Contracts and validations results.

6. Conclusion

In this paper, we addressed the data quality of validation methods. More specifically we investigated the tradeoff between heterogeneity of validation reports and accuracy of validation results, and presented an answer to the research question: "How can the data quality of heterogeneous data validation methods be improved in the context of data processing pipeline for knowledge graphs?" We propose a two-part framework consisting of QualityContracts and Guardians. QualityContracts describes sets of validation rules and includes the required information for data validation in both human and machine-readable formats. Guardians, interpret the machine-readable information, execute the validation rules, and produce comprehensive reports of both the results and the validation process.

We evaluated our approach through a comparative analysis with a processing pipeline for knowledge graph population implemented at a European airport. By comparing the initial version of the pipeline with its version after incorporating our framework, we observed improvements in the intrinsic (believability), representational (e.g. clarity and conciseness), and accessibility (ease of access) criteria of data quality, all while maintaining a high level of result accuracy created by different validation methods.

Our framework improves criteria of data quality, by relying on a software agent (Guardian), which implements different validation methods. As a result, Guardians are inherently equipped with specific libraries and features tailored to evaluate the specified QualityContracts. However, this built-in setup means that Guardians posses limited flexibility to add new validation constraints from new validation methods. This limitation necessitates regular updates of the Guardian. Ideally, Guardians would be able to autonomously update their libraries based on predefined criteria or settings. Additionally, while our framework has been designed with scalability in mind, we have not yet conducted empirical scalability tests. Therefore, although we anticipate that the system to scale effectively, it remains unverified. Furthermore, our framework does not yet provide tools to streamline the creation of QualityContracts, QualityConstraints, and QualityValidationResources. Consequently, significant manual input is required to define these resources.

7. Acknowledgements

This work was funded by the Bayerisches Verbundforschungsprogramm (BayVFP) des Freistaates Bayern through the KIWI project (grant no. DIK0318/03).

8. Author Contributions

Conceptualization, R.D., M.F., and J.F; methodology, R.D.; software, R.D.; validation, R.D.; writing - original draft preparation, R.D.; writing - review and editing R.D., M.F., and J.F.; supervision A.H.; funding acquisition, A.H. All authors have read and agreed to the published version of the manuscript.

References

[1] E. G. Kalaycı, I. Grangel González, F. Lösch, G. Xiao, A. ul-Mehdi, E. Kharlamov, D. Calvanese, Semantic Integration of Bosch Manufacturing Data Using Virtual Knowledge Graphs, in: J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne,

L. Kagal (Eds.), The Semantic Web – ISWC 2020, volume 12507, Springer International Publishing, Cham, 2020, pp. 464–481. doi:10.1007/978-3-030-62466-8_29.

[2] T. Hubauer, S. Lamparter, P. Haase, D. Herzig, Use Cases of the Industrial Knowledge Graph at Siemens, in: ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks, Monterey, USA, 2018.

[3] T. Liebig, A. Maisenbacher, M. Opitz, J. R. Seyler, G. Sudra, J. Wissmann, Building a Knowledge Graph for Products and Solutions in the Automation Industry (2019). URL: https://ceur-ws.org/Vol-2489/paper2.pdf.

[4] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, J. Taylor, Industry-scale knowledge graphs: Lessons and challenges, Communications of the ACM 62 (2019) 36–43. doi:10.1145/3331166.

[5] D. Diefenbach, M. D. Wilde, S. Alipio, Wikibase as an Infrastructure for Knowledge Graphs: The EU Knowledge Graph, in: A. Hotho, E. Blomqvist, S. Dietze, A. Fokoue, Y. Ding, P. Barnaghi, A. Haller, M. Dragoni, H. Alani (Eds.), The Semantic Web – ISWC 2021, volume 12922, Springer International Publishing, Cham, 2021, pp. 631–647. doi:10.1007/978-3-030-88361-4_37.

[6] G. Tamašauskaitė, P. Groth, Defining a Knowledge Graph Development Process Through a Systematic Review, ACM Transactions on Software Engineering and Methodology 32 (2023) 27:1–27:40. doi:10.1145/3522586.

[7] H. Knublauch, D. Kontokostas, Shapes Constraint Language (SHACL), W3C Recommendation, W3C, 2017. URL: https://www.w3.org/TR/2017/REC-shacl-20170720/.

[8] E. Prud'hommeaux, J. Labra Gayo, H. Solbrig, Shape expressions: An rdf validation and transformation language, ACM International Conference Proceeding Series 2014 (2014). doi:10.1145/2660517.2660523.

[9] L. E. Lwakatare, E. Rånge, I. Crnkovic, J. Bosch, On the experiences of adopting automated data validation in an industrial machine learning project, 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (2021) 248–257. doi:10.1109/ICSE-SEIP52600.2021.00034.

[10] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, N. Tang, Detecting data errors: Where are we and what needs to be done?, Proceedings of the VLDB Endowment 9 (2016) 993–1004. doi:10.14778/2994509.2994518.

[11] R. Y. Wang, D. M. Strong, Beyond Accuracy: What Data Quality Means to Data Consumers, Journal of Management Information Systems 12 (1996) 5–33. doi:10.1080/07421222.1996.11518099.

[12] L. Cai, Y. Zhu, The Challenges of Data Quality and Data Quality Assessment in the Big Data Era 14 (2015) 2. doi:10.5334/dsj-2015-002.

[13] H. Sándor, B. Genge, Z. Szántó, Sensor data validation and abnormal behavior detection in the Internet of Things, in: 2017 16th RoEduNet Conference: Networking in Education and Research (RoEduNet), 2017, pp. 1–5. doi:10.1109/ROEDUNET.2017.8123740.

[14] D. N. Bonter, C. B. Cooper, Data validation in citizen science: A case study from Project FeederWatch, Frontiers in Ecology and the Environment 10 (2012) 305–307. doi:10.1890/110273.

[15] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Biessmann, A. Grafberger, Automating large-scale data quality verification, Proceedings of the VLDB Endowment 11 (2018)

1781–1794. doi:10.14778/3229863.3229867.

[16] L. Ehrlinger, V. Haunschmid, D. Palazzini, C. Lettner, A DaQL to Monitor Data Quality in Machine Learning Applications, in: S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A. M. Tjoa, I. Khalil (Eds.), Database and Expert Systems Applications, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2019, pp. 227–237. doi:10.1007/978-3-030-27615-7_17.

[17] E. Breck, M. Zinkevich, N. Polyzotis, S. Whang, S. Roy, Data validation for machine learning, in: Proceedings of SysML, 2019. URL: https://mlsys.org/Conferences/2019/doc/2019/167.pdf.

[18] R. L. Akeson, X. Chen, D. Ciardi, M. Crane, J. Good, M. Harbut, E. Jackson, S. R. Kane, A. C. Laity, S. Leifer, M. Lynn, D. L. McElroy, M. Papin, P. Plavchan, S. V. Ramírez, R. Rey, K. von Braun, M. Wittman, M. Abajian, B. Ali, C. Beichman, A. Beekley, G. B. Berriman, S. Berukoff, G. Bryden, B. Chan, S. Groom, C. Lau, A. N. Payne, M. Regelson, M. Saucedo, M. Schmitz, J. Stauffer, P. Wyatt, A. Zhang, The NASA Exoplanet Archive: Data and Tools for Exoplanet Research, Publications of the Astronomical Society of the Pacific 125 (2013) 989–999. doi:10.1086/672273.

[19] N. Fursova, Methodology for Data Validation 1.1, Technical Report, 2018. URL: https://cros-legacy.ec.europa.eu/system/files/ess_handbook_-_methodology_for_data_validation_v1.1_-_rev2018_0.pdf, last accessed on 2023-08-25.

[20] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, D. Vrgoč, Foundations of json schema, in: Proceedings of the 25th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2016, pp. 263–273.

[21] O. K. Foundation, Frictionless specifcations, 2023. URL: https://github.com/frictionlessdata/specs, last accessed on 2023-08-25.

[22] Z. Zheng, B. Zhou, D. Zhou, A. Soylu, E. Kharlamov, Executable Knowledge Graph for Transparent Machine Learning in Welding Monitoring at Bosch, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, ACM, Atlanta GA USA, 2022, pp. 5102–5103. doi:10.1145/3511808.3557512.

[23] L. L. Pipino, Y. W. Lee, R. Y. Wang, Data quality assessment, Commun. ACM 45 (2002) 211–218. doi:10.1145/505248.506010.

[24] C. Cappiello, C. Francalanci, B. Pernici, Data quality assessment from the user's perspective, in: Proceedings of the 2004 International Workshop on Information Quality in Information Systems, IQIS '04, Association for Computing Machinery, New York, NY, USA, 2004, p. 68–73. URL: https://doi.org/10.1145/1012453.1012465. doi:10.1145/1012453.1012465.

[25] S. Shankar, L. Fawaz, K. Gyllstrom, A. G. Parameswaran, Moving fast with broken data, 2023. arXiv:2303.06094.

[26] O. ten Bosch, M. van der Loo, Standard Report Structure Essnet ValiDat Integration, 2018. URL: https://cros-legacy.ec.europa.eu/system/files/wp2-genericvalidationreport.pdf, last accessed on 2023-08-25.

[27] I. A. T. Association, Recommended Practice 1745 - Baggage Information Messages, 2016.