Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Concepts towards an Automated Data Pre-processing and Preparation within Data Lakes

Simone Schmidt

Course of Study:	Informatik
Examiner:	Prof. Dr. Bernhard Mitschang
Supervisor:	Dr. rer. nat. Christoph Stach, Rebecca Kay Eichler, M.Sc.
Commenced:	June 7, 2021
Completed:	December 7, 2021

Abstract

The Internet of Things produces huge amounts of heterogeneous data. Fields like Industry 4.0, smart city development, or the healthcare sector analyse this big data to serve as a basis for many applications. With their central storage, where heterogeneous data is stored in its original format, data lakes allow the analysis of data towards any use case. This schema-on-read approach leaves the transformation of data into an appropriate schema to the user. To achieve this, users need knowledge about the stored data, domain knowledge, and IT knowledge. The people who need the analysis results however are often domain experts and not IT experts.

Possibilities for assisting users in data preparation for novel use cases in data lakes are explored in the scope of this work. Reasons for the difficulty of data pre-processing in data lakes are explored and requirements for a concept for user assistance are derived. Steps are extracted, which a user takes in developing data preparation for a new use case in data lakes. Existing concepts in literature for assisting users in those steps are explored. It is found, that sufficient assistance in data discovery is provided by existing solutions. The support for technical realisation is almost sufficient, but assistance in choosing the right transformations is still lacking. Based on the lessons learned from the analysis of existing solutions a new concept is developed. The concept is based on BARENTS, a data lake concept that enables specification of data preparation in the form of ontologies and automatically performs specified transformations. A new transformation recommender helps users in choosing transformations and creating the ontology to specify data preparation. With a prototypical implementation of the concept, it is demonstrated, how users are assisted in specifying their data preparation needs. The concept is shown to fulfill the stated requirements and enables flexible, user-friendly specification of data pre-processing needs within data lakes.

Kurzfassung

Durch das Internet der Dinge werden große Mengen heterogener Daten generiert. Bereiche wie Industrie 4.0, Smart City oder das Gesundheitswesen analysieren diese als Grundlage für viele Anwendungen. Mit einem zentralen Speicher, in dem Rohdaten in ihrem Ursprungsformat gespeichert werden, ermöglichen Data Lakes die Analyse von Daten für beliebige Anwendungsfälle. Dieser Schema-on-Read Ansatz überlässt die Schematransformation von Daten dem Nutzer. Hierfür brauchen Nutzer Wissen über die gespeicherten Daten, domänenspezifisches Wissen und IT Kentnisse. Personen, die die Analyseergebnisse brauchen sind allerdings meist Domänenexperten und keine IT-Experten.

In dieser Arbeit werden Möglichkeiten zur Nutzerunterstützung bei der Datenvorverarbeitung für neue Anwendungsfälle in Data Lakes untersucht. Gründe, welche die Datenvorverarbeitung in Data Lakes erschweren, werden untersucht und Anforderungen an ein Konzept zur Nutzerunterstützung werden abgeleitet. Schritte, welche ein Nutzer bei der Umsetzung von Datenvorverarbeitung in Data Lakes durchführen muss, werden herausgearbeitet. Literatur zu bestehenden Ansätzen der Nutzerunterstützung bei diesen Schritten wird untersucht. Dabei zeigt sich, dass bestehende Konzepte ausreichende Unterstützung beim Auffinden von Daten bieten und die Unterstützung bei der technischen Umsetzung fast ausreichend ist. Unterstützung bei der Wahl der passenden Datentransformationen ist jedoch ungenügend. Aufgrund der gewonnenen Erkentnisse wird ein neues Konzept entwickelt. Dieses basiert auf BARENTS, welches die Spezifikation von Datenvorverarbeitung in der Form von Ontologien ermöglicht und spezifizierte Transformationen automatisch umsetzt. Ein neuartiges Empfehlungssystem für Transformationen unterstützt Nutzer bei der Auswahl von Transformationen und der Spezifikation der Datenvorverarbeitung als Ontologie. Mithilfe einer prototypischen Implementierung des Konzeptes wird gezeigt, wie Nutzer bei der Spezifikation von Datenvorverarbeitung unterstützt werden können. Es kann gezeigt werden, dass das Konzept die aufgezeigten Anforderungen erfüllt und eine fexible, nutzerfreundliche Spezifikation von Datenvorverarbeitung in Data Lakes ermöglicht.

Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

CSV comma-separated values. 31

EL extract, load. 22

ETL extract, transform, load. 9

JSON JavaScript object notation. 31

RDF resource description framework. 52

XML extensible markup language. 31

1 Introduction

In the context of the Internet of Things [GC20], ubiquitous collection of data via sensors produces huge amounts of data originating from different sources. These massive data sets, which have heterogeneous structures are described by the term big data [SS13]. This huge variety of data can be used to serve many use cases. Novel use cases and analysis tasks arise frequently, as the corresponding data accrues [SBE+21]. In fields like Industry 4.0 [KWXD17], smart city development [ADBG20] or the healthcare sector [LQ20], the analysis of this data can serve as a basis for many applications. Automation of production and machine health control, planning of smart, connected city infrastructures, early disease detection and drug research are only some of the applications enabled by the analysis of big data.

Big data has characteristics, which make storing and analyzing difficult, using traditional means like data warehouses [SS13]. One of these characteristics is variety. The main problem of data warehouses is that they are designed to serve specific, predefined use cases and need data to adhere to a predefined structure, which is defined at design time [KW18]. For data sources providing data in a format which requires severe transformation to fit the schema, this means data is lost during ingestion. Adapting to changes in use cases can require the redesign of the data warehouse, since the data for the use case might have been discarded during ingestion.

A concept for storing, governing and provisioning big data efficiently are data lakes [GGH+20b; Mat17]. The term data lake is used to describe various concepts for central data storage, as there is no agreed-upon definition [GGH+20a]. However most definitions agree, that a central storage is provided, where data from any source is stored in its original format. By storing data in its original form, changes in data do not pose a problem and any use case can be served, as no information is lost through transformation into a unified schema. However, this means that data lakes are designed for schema-on-read instead of schema-on-write, which is the approach applied in data warehouses. Therefore transforming data into a usable schema in preparation for analysis is left to the user. Raw data is rarely usable in its original state and has to be pre-processed to enable analysis. Since pre-processed data only reaches its full potential when utilised in the use case it was envisioned for, users have to adapt pre-processing for each new use case [SBE+21]. Users have to specify, how the data is prepared and

which techniques are used. To achieve this they need knowledge about the stored data as well as domain knowledge and IT knowledge.

There are concepts for storing data, which has been prepared with different degrees of pre-processing, in different partitions, called zones or puddles, and governing the data within one of those zones [GGH+20b; Inm16]. However a concept for transferring data from one stage of pre-processing to the next rarely exists. Users have to perform the pre-processing largely without tool support.

1.1 Goals

In the scope of this work, possibilities for assisting users in creating data pre-processing for new use cases in data lakes are explored. Creating data preparation for new data analyses is neither easy nor intuitive, when working with data lakes. At the same time novel use cases and analysis tasks emerge frequently. Therefore this work explores, how users can be assisted in specifying data pre-processing and preparation for data lakes. The goal is a transfer of data from one data lake zone and processing stage to the next with tool support.

For this purpose, reasons for the difficulty of data processing in data lakes are explored. Requirements for a concept for user assistance are derived based on the difficulties in creating new data preparation. The steps a user takes in developing data processing in data lakes for a new use case are extracted. Literature is explored for existing concepts for assisting users in those steps of preparing data for their use cases. Concepts which are applicable in data lakes are gathered and analysed. Conclusions regarding areas, where users are assisted sufficiently and where the assistance is still lacking are derived. Based on the analysis of existing concepts a new concept is developed, which closes some of the gaps left by existing solutions. The new concept has to be flexible enough to be used in any domain and user friendly enough to be used by data scientists and domain experts with little IT knowledge. A prototypical implementation of the new concept is created. By exemplarily serving a use case, the serviceability of the prototype and the concept is demonstrated. Then for the new concept it is assessed, whether it fulfills the requirements.

This work therefore provides the following contributions towards user-friendly specification of data pre-processing in data lakes:

- Requirements for a concept for user assistance are derived.
- The steps a user has to perform in developing data processing for a new use case are extracted.

- Literature is explored for existing concepts and research gaps are identified.
- A new concept for user-friendly specification of data pre-processing is created, which fulfills the presented requirements.
- The concept is implemented as a prototype.

1.2 Outline

The rest of this work is structured, as outlined below:

Chapter 2 – Basics This chapter introduces basics about data warehouses, big data and data lakes, in order to provide a basis for the discussion about data processing and data preparation in data lakes. It is explored, for which purpose data lakes exist and where the difficulties in realising data analyses lie, when working with data lakes.

Chapter 3 – Requirements Based on the reasons for the difficulty of realising data preparation within data lakes, some requirements are derived in this chapter. The requirements, which a concept towards specification of data preparation and pre-processing in data lakes should fulfill, are defined.

Chapter 4 – Literature Review This chapter examines literature for existing solutions for helping users in realising data pre-processing and preparation for new use cases within data lakes. The steps a user has to perform in order to realise data pre-processing for a new use case are identified. For each step, concepts, which assist a user in preforming the step, are examined. Conclusions regarding the degree of support of the user in each step and overall in realising data pre-processing in data lakes are drawn.

Chapter 5 – Inferences with respect to a new Concept In this chapter conclusions are drawn from the examination of existing solutions regarding research gaps and areas where a demand for improvement of assistance for data preparation in data lakes exists. From these conclusions, insights regarding the focus and some characteristics for a new concept for specification of data pre-processing within data lakes are derived.

Chapter 6 – Discussion of Concepts A newly designed concept for user-friendly specification of data pre-processing within data lakes is introduced in this chapter. Possibilities for realising aspects of the concept are discussed and the final concept is presented.

Chapter 7 – Use Case This chapter introduces an example use case for demonstrating, how a prototypical implementation of the new concept aids the user in specifying her or his data pre-processing needs for new data analysis tasks. For this purpose, a data set is introduced. Some analysis goals for this data along with the pre-processing needed to enable the analysis are presented. In the demonstration, some of these goals serve as the knowledge base of already existing data preparation solutions whereby one serves as the new analysis goal for the user.

Chapter 8 – Prototype A prototypical implementation is created for the new concept. In this chapter, the prototype is described. It is demonstrated, how the prototype can be used to aid the user in specifying data pre-processing for the introduced use case and reach her or his new analysis goal.

Chapter 9 – Assessment In this chapter, the newly designed concept is evaluated against the requirements from Chapter 6. It is assessed, in how far they are fulfilled.

Chapter 10 – Conclusion This chapter concludes the work. It summarises the findings and contributions of this work and shows directions for future research.

2 Basics

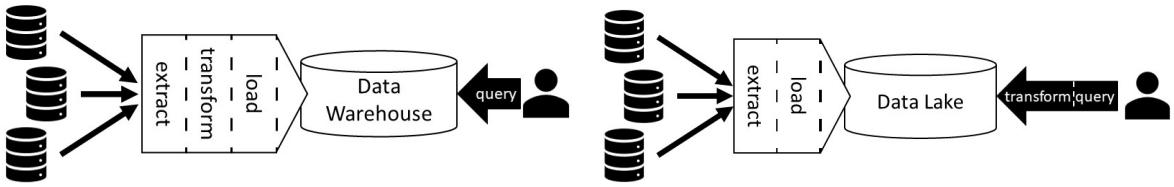
This chapter gives an overview over the basic concepts, which are needed for the discussion of data processing and preparation in data lakes within this work. First, data warehouses are explained in Section 2.1. They are a by now traditional solution for data storage and analysis. Afterwards, the term big data is defined in Section 2.2 and the challenges that come with big data are discussed. Then Section 2.3 describes the concept of data lakes, which were developed, in order to meet the challenges of storing and processing big data. Last, two common data lake architectures are introduced in Section 2.4.

2.1 Data Warehouse

Data warehouses are a well-established solution for data storage and analysis. They were first introduced as "a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions" [Inm05]. A data warehouse accumulates data from a wide range of sources within an enterprise in order to provide strategic information for management decisions [PC03].

Data warehouses use an extract, transform, load (ETL) process, which is visualised in Figure 2.1a. Data for a warehouse needs to first be extracted from, often heterogeneous, operational sources. Then it needs to be cleaned, integrated and aggregated. The data is transformed consistently according to the structure, designated by the data warehouse schema [KW18; PC03]. After being transformed, the data is loaded into the warehouse. The warehouse functions as a central relational database, where all data for analysis is stored [SBE+21]. The integrated data can then be accessed efficiently and in a flexible fashion [PC03]. This gives users a comprehensive analytical view of sparsely distributed operational data.

For a data warehouse, the data schema needs to be known at design time, as only transformed data will be stored in the warehouse [KW18]. Warehouses are designed with specific use cases in mind. The schema is specified to let, so that the warehouse can optimally serve those use cases. This enables domain experts, like business professionals,



(a) In data warehouses, an ETL process results in aggregated data, conforming to a schema, being stored, which is easy to query for the end-user.

(b) In data lakes, a shortened extract, load (EL) process results in raw data being stored, which leaves the transformation to the end-user as part of querying.

Figure 2.1: Comparison of the ETL process of the data warehouse architecture and the data lake architecture.

with little to no IT knowledge to perform data analysis via standardised tools [SBE+21]. Data analysis outside of those predefined use cases however might not be possible.

2.2 Big Data

Many fields which use the Internet of Things [SAB+18; SM18; Sta15], like Industry 4.0 [GS14; SSM18b], smart city development [SB11; SGPM20], or the healthcare sector [SSF17; SSM18a], produce huge amounts of data originating from very different sources. To differentiate this kind of data from more homogeneous, smaller quantities of data, which can be stored and analysed with more traditional means, the term “big data” is used. This huge variety of new data can be used to serve many new use cases, which arise as the corresponding data accrues.

The term big data describes massive data sets, which have very heterogeneous structure [SS13]. Storing, analyzing and visualizing this data can be difficult using traditional means. The challenges of working with big data lie in the three main characteristics of big data: volume, variety and velocity.

The volume of data, or its size, is unusually large, often amounting to terabytes and petabytes [SS13]. The huge scale of big data outstrips traditional store and analysis techniques.

The variety of big data is caused by two reasons [SS13]. Firstly, it comes from a great variety of sources. Secondly, it can come in three types: structured, semi-structured and unstructured. Structured data can be inserted into data warehouses or relational databases as it is. It is already tagged with relevant labels and can easily be sorted. Semi-structured data conforms to some kind of structure and contains tags to separate data elements. However it does not conform to fixed fields. Unstructured data is any

data which does not conform to any specified structure. The structure can be random and complex. Unstructured data is therefore difficult to analyse.

The velocity describes both the rate at which new data is created and the rate at which it needs to be processed [SS13]. Some sources continually produce data at a fast rate, which needs to be stored and analysed, as it arrives. Velocity is therefore relevant in all processes involving big data.

Using big data to its full potential provides new challenges, which data warehouses often can't meet. Warehouses are designed to serve specific, predefined use cases and therefore need data to adhere to a predefined structure, which is defined at design time. Particularly the data variety poses a problem, because semi-structured and unstructured data must either be discarded or undergo heavy transformations in order to adhere to the data warehouse schema, which usually comes with data-loss [RZ19]. Using the data for different purposes usually becomes impossible due to this data-loss. Additionally, schema changes in the original sources or the emergence of new sources are hard or even impossible to handle, without re-designing the data warehouse and the transformations for data integration.

2.3 Data Lake

The concept of data lakes was created in order to store, govern and provision big data [GGH+20b; Mat17]. While there is no agreed-upon definition of data lakes, there are certain core aspects, most definitions agree on. A data lake is a central storage, where data from any source is stored in its original format. Structured data can be stored in a data lake alongside semi-structured and unstructured data. The data can be stored at any rate and by any means. The most important aspect is, that all data is available in its original format at all times [EGG+20]. This enables data lakes to be flexible regarding schema changes in data sources, as well as enabling the analysis of stored data for ever changing use cases [RZ19; SBE+21].

While some definitions see the data lake only as a pure storage solution, many others see the data lake as a data storage, management and analysis platform [AARC19; GGH+20b; RZ19]. Pre-processed data for different types of analysis and analysis results can be stored in addition to the raw data. This makes results reusable, when viewing the data lake as a data analysis platform. In comparison, data warehouses only store pre-processed data for specific predefined analyses.

Unlike a data warehouse, a data lake is always at the risk of becoming a data swamp, where finding, processing and analysing data can become impossible [TR21]. Since a data lake has no single data schema and huge amounts of data from heterogeneous

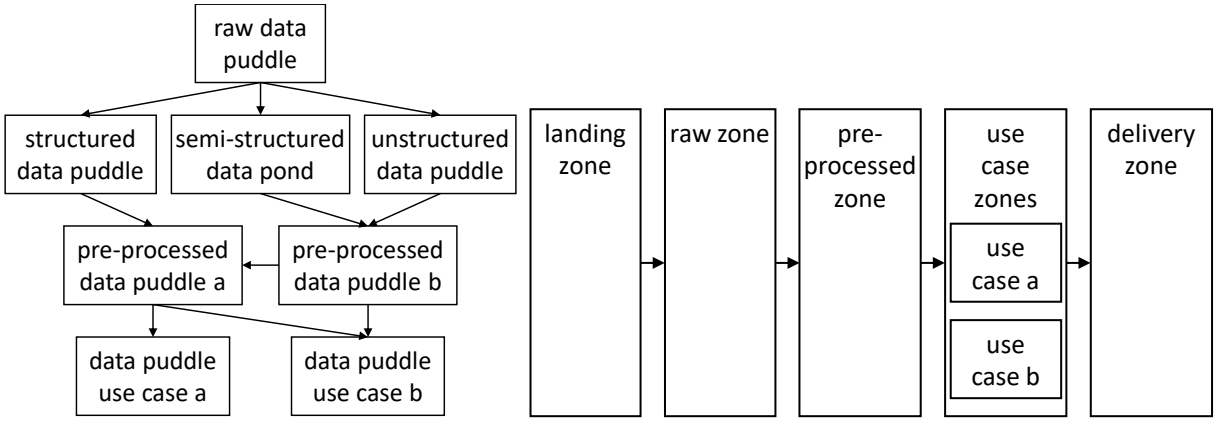
sources are stored there, it has to be designed and managed properly in order to remain usable. The flexibility of allowing any type of raw data and enabling any new use case to be served comes at the risk of the data store becoming chaotic, cluttered and unclear. This problem does not occur in data warehouses, as the adherence to a predefined data schema keeps the data store well organised. Therefore using metadata is important, for a data lake to remain usable in practice [EGG+20]. Various metadata, such as source, schema or semantic information about the data, can be collected and managed in data catalogs [DS21]. These data catalogs provide a well organised overview of data stored in the lake.

When compared to the data warehouse, which uses an ETL process in order to store aggregated data, which conforms to a schema, as can be seen in Figure 2.1a, data lakes use a shortened EL process. In the EL process the data is only extracted from the sources and loaded into the data lake as it is, as can be seen in Figure 2.1b [RZ19; SBE+21]. For the data warehouse this means that the end user can use easy queries to access the data. The process of putting data into the warehouse however is more complex and needs more consideration at the time of the warehouse design. Additionally, some data, which becomes important later on, might be lost due to the transformation. Use cases that were not considered while designing the warehouse might not be viable to fulfill. Data Lakes however allow for flexible use of the data for any use case and no data is lost [GGH+20b; RZ19]. This comes at the cost of end users having to transform data into a usable schema themselves, before they can use it for analysis. For this reason, users of data lakes are usually data scientists [SBE+21]. They have some business knowledge as well as some IT knowledge. While putting data into a data lake is easier, working with this data is not.

2.4 Data Lake Architectures

There are many aspects of data lakes, which can be implemented in very different ways and are part of the architecture of a given data lake [GGH+21]. Data organisation is one of those aspects, which is relevant for the discussion of data discovery in Section 4.2. Below two architectures regarding data organisation in data lakes are introduced, the pond architecture and the zone architecture. An example for each architecture can be seen in Figure 2.2. In both architectures, the goal is to partition the data, to facilitate easier data discovery.

In the *pond architecture* [Inm16], different linked puddles exist, which contain different types of data. For example, as seen in Figure 2.2a, there could be puddles for not yet classified raw data and puddles for structured, semi-structured and unstructured data. As new processing needs and use cases arise, puddles for different types of pre-processed



(a) An example for a data pond architecture. **(b)** An example for a zone architecture.

Figure 2.2: Example architectures for data organisation in data lakes.

and use case data can be created. When data is processed, it is moved from one puddle into the next and then removed from the previous puddle. This way there is less data per puddle. Because of the structure, finding data is easier, when the corresponding puddle is known. However there is no longer one place where the raw data of any data stored in the lake can be found at any time, which contradicts the fundamental idea of a data lake [GGH+19; GGH+20a].

The *zone architecture* [GGH+20b] tries to facilitate the same advantage, without losing the raw form of data. There are different zones containing different processing stages of data. The original raw format of all data can always be found in the raw zone. Additional copies and transformed versions of the data may be in several other zones. As seen in Figure 2.2b, there could, for example, be a landing zone for buffering data which accrues at a too high velocity, a raw zone and a pre-processed zone for cleansed data, as well as any data that was transformed according to some schema. As new use cases arise, several use case zones which hold pre-processed data relevant to these use cases can be added. A delivery zone can be used to provision data to the user [GGH+20b; SBE+21].

3 Requirements

In the concept of data lakes, as discussed in Chapter 2, an EL process is used for data ingestion and data pre-processing is thereby left to the user. Consequently users require assistance in fulfilling their data pre-processing needs. Therefore, the goal of this work is to examine existing approaches for assisting users in data preparation within data lakes and analyse, whether they provide sufficient support. If the existing solutions are insufficient, the goal of this work is to explore concepts towards enabling specification of data pre-processing tasks in a user-friendly and easy way. As established in Chapter 2, data lakes provide the foundation to enable users to analyse data for any given use case. Data lakes are meant to enable the usage of any kind of data in any kind of analysis. However realising the data pre-processing is challenging.

Data stored in a data lake can be of any type, in any state, and from any domain. Flexibility of the concept is therefore important. As new analysis tasks and use cases emerge frequently [SBE+21], the same data lake should be able to serve various use cases over its lifetime. Any concept for creating data pre-processing for analyses should therefore be flexible enough, to be used on any source data and in any domain. Data stored in data lakes can change over time, for example, in case of a schema change in a data source, or because additional sources provide new data for a use case. The concept should be flexible enough to allow for easy adjustment of the specification of the data preparation to the changing data. Use cases can also change over time, as new insights might change the focus of an analysis. Flexible adjustment of existing data preparation to changing use cases should therefore also be possible.

Data lakes use an EL process for ingesting data, as discussed in Chapter 2. The concept of data lakes is designed for schema-on-read, instead of schema-on-write, as applied in a data warehouse, for instance [TR21]. Therefore, the data transformation in preparation for analysis is up to the user. Currently, users of data lakes are mostly data scientists, as, even though they are no IT experts, they possess IT knowledge in addition to domain knowledge. Most domain experts however possess little to no IT knowledge while still having a need for data analysis results, as can be seen in data warehouses. A new concept should provide a user-friendly way to achieve data preparation for a new analysis task, which requires little IT knowledge. Thereby the workload of data scientists would be greatly reduced and the use of data lakes would be feasible for an entirely new user group.

There are already existing solutions, which address some of the challenges of data lakes, in the context of user-friendly data preparation. Many established deployments of data lakes exist as well. A new concept has to be compatible to existing solutions in order to be embeddable in an existing data lake. Any new concept, which is incompatible with existing solutions, cannot be embedded in existing data lakes. Additionally, if the new concept is incompatible with existing solutions for user assistance, the full potential of combining all existing solutions cannot be reached.

As research on data lakes continues, new ways of assisting users in creating data preparation for their use cases will keep appearing. A new concept should be extensible, to allow for future extension through new means of assistance. This way it remains possible to provide optimal support of the user, as new solutions appear.

Therefore the newly developed concept should fulfill the following four requirements:

R1 - Flexibility The concept should provide the required flexibility in order to be applicable in any given domain and in order to be adaptable to changes. It should work on any data from any domain, as data lakes can store various data. The concept should also be applicable for any use case, as data lakes should be able to serve any novel use case, which emerges. In case the data or the use cases change, the concept should allow for easy adjustment of the specification of the data preparation.

R2 - User-friendliness The concept should be user-friendly enough, to enable users with little IT knowledge, to specify their processing needs. This decreases the workload of the current users of data lakes, data scientists. At the same time it opens the use of data lakes to a new user group, domain experts.

R3 - Compatibility with existing Solutions The concept should be compatible with existing data lake concepts. Interoperability with existing solutions, in the context of user-friendly data preparation, is important. Incorporating the concept in existing data lakes should be possible as well. Future development would also be hindered by incompatibility with existing concepts.

R4 - Extensibility The concept should be extensible, to allow for future improvement, as research on data lakes continues. When new ways of user assistance are discovered, it should be possible to extend the concept with those to keep providing optimal support of the user.

4 Literature Review

Novel use cases and analysis tasks for data emerge frequently [SBE+21]. Pre-processed data only reaches its full potential when utilised in the use case it was envisioned for. In order to meet these dynamically changing needs, pre-processing and preparation of data have to be adapted to the respective use case just as frequently. While some data preparation steps are needed in more than one use case, universally applicable data pre-processing does not exist. Therefore, data lake users have to adapt their data preparation solutions for each new use case.

As discussed in Chapter 2 data lakes use an EL process for ingesting data, instead of an ETL process. This means the concept of data lakes is designed for schema-on-read, not schema-on-write. Therefore, in data preparation for analysis, the data transformation is up to the user. Creating the data pre-processing for a new use case is not trivial. Finding all the relevant data can be challenging, and the used data formats and schemata have to be learned on read, which needs knowledge of the stored data and IT knowledge. The data has to be transformed into a usable schema, which needs knowledge about the initial state of data, as well as IT knowledge and domain knowledge. Intended pre-processing results and the transformation steps to reach them need to be identified, which requires domain knowledge. In order to transform these abstract pre-processing steps into a technical realisation, which can be deployed on the underlying platform, IT knowledge is needed.

As explained above, data preparation for new use cases in data lakes currently requires knowledge about the stored data, as well as IT knowledge and domain knowledge. But not all end users of data lakes are IT experts. Currently, they are often data scientists, as they usually have both IT knowledge and domain knowledge [SBE+21]. Gaining knowledge about the stored data however can still be a challenge for them in data lakes. Additionally, while they do have IT and domain knowledge, they aren't IT experts. They would therefore greatly benefit from assistance in gaining knowledge about the data and for tasks which require IT knowledge. Furthermore, as can be seen in the usage of data warehouses, domain experts have a need for data analysis results. It would be advantageous, if they could be enabled to create data pre-processing for new use cases themselves. In the following, if the term user is used, it refers to data scientists or domain experts, who wish to perform data pre-processing for new use on data in a data lake.

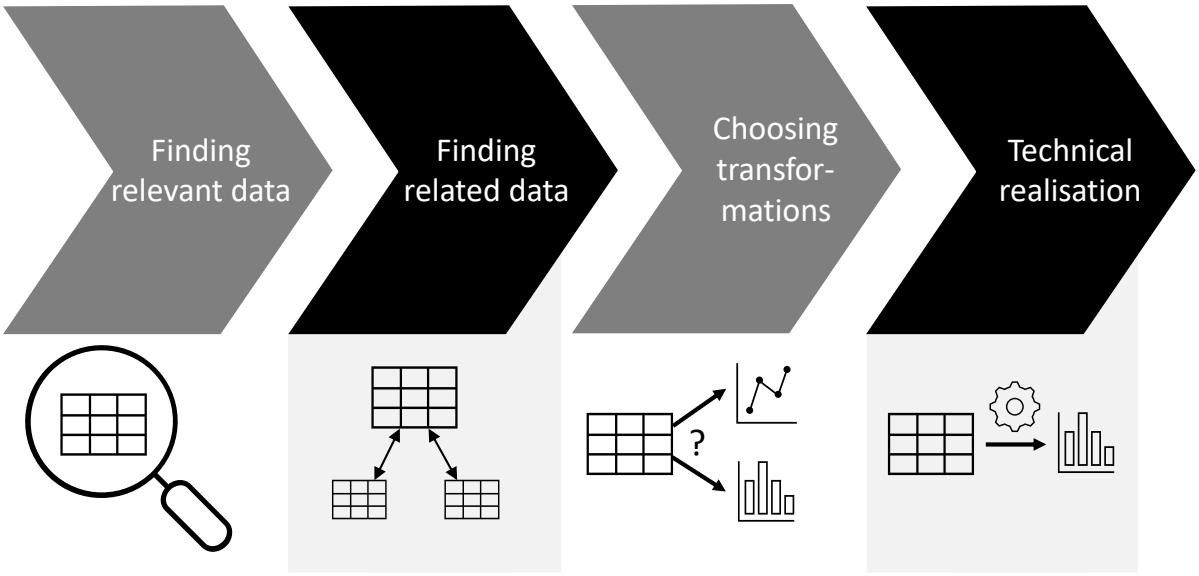


Figure 4.1: Steps for creating data pre-processing for a new use case.

In this chapter, first, in Section 4.1 the steps are identified, which a data lake user has to perform, in order to create data pre-processing for a new use case. Then in Section 4.2, Section 4.3, Section 4.4 and Section 4.5, approaches from literature, for assistance in these steps are discussed. Finally, Section 4.6 summarises the approaches and analyses them for areas, where assistance is still lacking.

4.1 Steps for creating new Data Pre-processing

When data scientists want to reach deployment ready data pre-processing for a new use case, they have to perform several steps. While viewing current literature on data lakes, four steps can be identified, where users can be supported in realising the data preparation for a new use case. These steps are visualized in Figure 4.1.

In order to better illustrate these steps, we will work with the following example: A data scientist is doing research regarding global warming. A lot of weather data has been collected from various sources world wide. She or he wants to analyse how temperatures in different places have been affected recently. Therefore temperature, time and place of measurement are needed in a comparable form.

First relevant data has to be found. Data scientists have to get an overview of the huge amounts of heterogeneous data in the data lake [SBE+21; TR21]. Finding data can be challenging, since huge amounts of data from many different sources can be stored in a data lake as they are. Because the data lake has no central schema, it can

be hard to find, where in the lake which data is stored. Users have to be enabled to get an overview of the data in the data lake and which of it could be relevant to the use case. The goal is to find some data relevant to the use case as a starting point and to identify the format or schema of the data. In the weather data example, some data sources might have provided the weather data from a relational database, while others provided data in JavaScript object notation (JSON), extensible markup language (XML) or comma-separated values (CSV) format. The data scientist now searches for all columns (in the relational and CSV data) or tags (in the JSON and XML data) labeled as "temperature", "temp", "°C" or "°F". She or he additionally searches for badly labeled or unlabeled data with values containing "°C" or "°F". Approaches which can be found in literature, for assistance in this step are discussed in Section 4.2.

The second step is searching for related data. Required data could be scattered across different sources and therefore end up in different parts of the data store of the lake. In case the relevant data points or attributes are scattered across multiple sources, it is easy to miss some of it accidentally [AARC19]. If relevant data is left out of the pre-processing, the end result will be incorrect. The data lake should be searched for data related to the initially chosen data, to minimise the amount of data that is accidentally left out of the pre-processing. In the example, some sources might have provided all temperatures in Kelvin, labeled as "K". The data scientist initially missed those. A relatedness search, which uses domain knowledge about weather data, recommends, that the temperature data in Kelvin is related to the other temperature data. For some of the CSV data, the time and place of measurement are also in different files to the temperature data, connected via ids and foreign key relationships. A relatedness search for possible join candidates shows, where to find the recorded time and place for these temperatures. Section 4.3 discusses current approaches in literature for discovery of related data.

The first and second step are both part of data discovery. There are papers, which discuss a combination of both, as they are connected. However the way they are realised differ and most works will only discuss one or the other. Therefore, they are discussed separately in this chapter.

The third step is choosing transformations. Selected data needs to be converted into a usable schema and afterwards the right pre-processing steps for reaching the desired results need to be performed on it. The initial raw data could be in any state regarding format, schema and quality. Data scientists need to select the right data transformations, which will have the desired effect. Which transformations these are, depends not only on the designated result, but also on the initial state of the data. In the example, there could be sources, where time, place and temperature of measurement are in different files. The data scientist has to realise, that for data from these sources, she or he needs to join the tables first. Afterwards, only time, place and temperature need to be selected for data from all sources. Then all the temperatures need to be converted to the same

temperature scale. This is a multiplication by a simple factor, which depends on the initial temperature scale. Not all sources captured the temperature data at the same rate, some was captured each second, some other was captured each minute and the rest at a hourly rate. In order to get comparable results, the data needs to be aggregated. For each place, the hourly average temperature needs to be calculated. How data scientists could be assisted in selecting the right transformations is explored in Section 4.4.

Lastly, the abstract chosen sequence of data transformations needs to be translated into a technical realisation, which can be deployed on the underlying platform. As mentioned before, data scientists are not primarily IT experts. Enabling business experts with even less IT knowledge to use data lakes is desirable as well. Therefore, it would be advantageous if this step could be performed in a way, which requires little IT knowledge from the user. This means that there should be a solution, where users do not have to produce implementations which run directly on the respective underlying infrastructure. If the transformation can be automated based on examples or the end user only has to formulate the transformation sequence in a more abstract, higher level language, this could be achieved. In the example, if the data scientist now has to write a script which performs the pre-processing, quite a bit of IT knowledge is needed. All the different data, from relational databases, CSV files, XML files and JSON files needs to be read, which works differently for each storage format. Then all transformation steps need to be performed on the data exactly as envisioned before. It would be easier, if the data scientist could specify the pre-processing needs to the platform in a way which requires less IT knowledge. Literature on approaches, that try to enable this, is discussed in Section 4.5.

Section 4.6 concludes the review on the current state of research. Existing approaches for user assistance in each step are summarised. It is analysed, where users are already assisted to a satisfactory degree, and in which areas the assistance is still lacking.

4.2 Finding relevant Data

Two main approaches can be found in literature, which assist in getting a better overview of the data in data lakes and finding relevant data. These are data partitioning and metadata.

Data partitioning uses the data lake architecture to provide data organisation. Data in the lake is divided into smaller more manageable chunks [SBE+21]. Two architectures, which partition data are the data pond architecture [Inm16] and the zone architecture [GGH+20b] introduced in Section 2.4.

In the pond architecture, the way the data is partitioned depends on the implementation. There can be partitions for different raw data types and for pre-processed data for each use case. For each data point only one version is available. After pre-processing, the raw version is removed and only the pre-processed version of the data continues to exist in a different puddle. The zone architecture partitions data depending on processing stage, but each data point can exist in different processing stages. While there is only one partition for all raw data, there can be several zones for different processing stages. After pre-processing of a data point, an additional pre-processed version is put into the corresponding zone, without removing the raw version. Zones can be added for different stages of pre-processing for different use cases.

By partitioning the data according to some features, there is less data to view per partition. The user can get an overview on what kind of data is in each partition. If users are looking for data with certain features and know, that there are partitions, only containing data with those features, they only have to search in those partitions. For example, if a use case with corresponding partition already exists, the user has to search for relevant data in only one partition.

How helpful this partitioning is, depends on the number of partitions. If there are too many partitions, the problem for the user moves from having to find the right data in a huge undivided data lake to finding the right partition among many. Therefore, adding one partition per use case stops being an advantage, if the number of use cases becomes too high. If partitions for different use cases are merged, the number of partitions decreases. However, the amount and diversity of data in a partition increases and it becomes more difficult to find the relevant data in the partition. Overall, while data partitioning is helpful in getting an overview of data and finding data, it has weaknesses, which have to be covered by other approaches.

Metadata stores data about the features of actual data. There are many kinds of metadata, which can be collected. Some types of metadata are used almost universally, like name, size, time of ingestion, the source it was extracted from and the storage format. Other types of metadata are only used in some of the works. Schema data is often collected [DLP+21; DS21; FGG+21; HGQ16; HKN+16; LSSB21]. Often, metadata about the semantic meaning of fields or tags in the schema, or domain specific information about their context is used as well [DLP+21; DS21; GH19; HGQ16; LSSB21]. Statistical data can also be added to the schema data, like value ranges, percentage of missing values or value distributions [GH19]. Metadata can also concern provenance, meaning which transformation was used on what data to generate the described data [DLP+21; FGG+21; GH19; HKN+16; MRZ21]. Access rights and data ownership can also be handled using metadata [DS21; HKN+16].

Often the metadata has to be added to the data manually or delivered together with the data. However there are approaches for automatically extracting metadata [FGG+21;

HGQ16]. For example there are approaches for extracting schema data about relational data as shown in the work of Langenecker et al. [LSSB21]. Approaches for enriching the stored metadata with additional metadata exist as well. Especially semantic information and domain specific information, which is usually not initially available, is of interest. By using existing semantic and domain knowledge and analysing the data Langenecker et al. [LSSB21] and Dibowski and Schmid [DS21] enrich the metadata with semantic and domain information.

Data markets can also provide data for specific use cases together with rich metadata. They are a concept for buying and selling data like physical goods [FSF20]. Data providers offer their data and data consumers can search for data with specific features and obtain it. This enables market basket analyses similar to shopping carts in online stores. Useful data can be recommended based on previously acquired data or the behavior of comparable users. Data providers have to be interested in users finding their data for the concept to work and prepare the data accordingly. In other words, they have to provide detailed metadata along with the data. However, it cannot be assumed that every data provider invests the required effort. Neither can it be assumed, that for every use case an appropriate data offer is available.

Metadata is mostly organised for users in data catalogs [DS21; GH19; HKN+16; LSSB21]. Internally, graph [DS21; FGG+21; MRZ21] and index structures [HGQ16] are in use to store metadata. These structures can be used to find data with specific features based on their metadata. Many approaches use querying languages to let users search for data. Others visualise data based on the stored metadata graphs, sometimes including interfaces for navigation through the visualized structure. Another approach is filtering data by metadata based on keyword searches.

There is work on creating generic metadata models and for improving metadata management [EGG+20; GH19]. The goal is to have universally applicable and extensible models to provide the best tools and strategies for metadata management.

4.3 Finding related Data

Many approaches exist for finding relatedness of data. While some of the approaches for calculating relatedness are strictly rule-based, many use machine learning. The relatedness score can be based on metadata and statistical pre-processing of the content in the data source. Most approaches calculate the relatedness based on a mixture of schema metadata, like attribute names and types, and statistical pre-processing of the corresponding content, like value ranges, statistical distributions and format of the values [AARC19; AARC20; BFPK20; DLP+21; FAK+18]. Other approaches use

either the schema information [ZI19] or content information [HKN+16] in combination with provenance metadata to calculate the similarity. The provenance information gives insight into whether two data sources are used in similar positions in processing chains. In other words it shows, whether it was derived from similar sources by similar transformation chains and whether it was used as input for the same transformations. Domain specific information in the metadata can also be integrated in the relatedness calculation, as shown by Diamantini et al. [DLP+21]. This provides information about which topics within a domain a data source belongs to and adds domain specific similarity measures into the relatedness calculation. The calculation can either be performed on-demand in relation to a given input or run in advance to build relatedness graphs, hierarchical trees and index structures.

The relatedness measure can then be used for two main purposes. One is to recommend candidates, which might provide additional features for the same data points, often in the form of join candidates [AARC19; BFPK20; FAK+18; HKN+16; ZI19]. The other is to provide additional data points for the same features [AARC19; AARC20; BFPK20; DLP+21; ZI19]. Results of the proximity calculation can be provided to the user in different forms. The system can recommend the closest relatives to the data a user is currently viewing [BFPK20; HKN+16; ZI19]. Graphs and index structures can be queried for related data sources [FAK+18; ZI19]. Relatedness graphs or hierarchical trees can be visualised for the user and traversed in graphical interfaces [AARC19; ZI19]. Data can be clustered in order to group data sets, which belong to the same topic [AARC19]. By finding the most common domain descriptions in the metadata of data in the cluster and combining it with domain knowledge, users can search for data belonging to a provided topic [DLP+21].

4.4 Choosing Transformations

Transformations can be sorted into broader categories. A certain category of transformations can only be used on certain categories of data. Megdiche, Ravat, and Zhao [MRZ21] call these categories of transformations coarse-grained operations. The categories are filtering, formatting, aggregation, calculation, consolidation, merging and join. For these coarse-grained operations a few general restrictions regarding input data were identified. While in the general case all transformations can be performed on structured and semi-structured data, some transformations (aggregation, calculation and join) cannot be performed on unstructured data. Operations can also be unary (filtering, formatting, aggregation, calculation, consolidation) transformations which are applied to one single data set, or binary (merging, join) transformations which need to be applied to multiple data sets at once. While for unary operators the structure of the

data is irrelevant, binary operators can only be applied, if all input data is of the same structural type. It can be even be more restrictive for specific transformations and data. For a specific transformation only very specific types of data might be usable as input. Vice versa for specific input data only certain transformations make sense. Dibowski and Schmid [DS21] show for data filtering how only relevant filtering operations and properties to filter by can be displayed to a user, if enough metadata in combination with semantic knowledge is available. Among the viewed literature there is however little research for assisting users in finding the right transformation steps.

Behringer et al. [BHFM20] provide an interface, which allows users to choose applicable transformations from a drop-down menu. This however only works on structured data, which can be displayed in a spreadsheet environment. In data lakes, semi-structured data and unstructured data also occur. Users have to be able to work with those kinds of data and transform them as well.

There is a paper by Megdiche, Ravat, and Zhao [MRZ21], which focuses on metadata regarding data transformations. In addition to metadata on data itself, metadata on previously performed processing steps or transformations is collected. It is automatically captured by the system, whenever a transformation is performed on data in the data lake. This metadata includes what the input data was, the specific transformation that was performed, which broader category it belongs to, where the implementation of the transformation can be found and what the output data was.

In this approach, the transformation metadata is collected in a graph database, so that connections between transformation steps are easy to capture. This graph data base can be queried in order to find specific information. For a data set, all data created from this data can be found, as well as all transformations, which were previously performed on a data set. The graph can be searched, to find out, if a specific transformation or series of transformations has already been implemented before and the corresponding implementation can be found. This increases the reusability of existing implementations. These capabilities can also be used, when creating new analyses. Users can look at existing metadata with similar input to the input data they identified or output data similar to the desired output. The existing transformations which were identified as relevant by the user can serve as a basis or inspiration for the new pre-processing.

While this encompasses the scope of the work, the concepts can function as a basis for more sophisticated user assistance in the future. By analysing the metadata on transformations in combination with metadata on the input and output data new insights could be gained. For example: What kinds of input can be used for an existing transformation or group of transformations? Which transformations can be used on a specific input or category of inputs? What kind of output data does a transformation produce? Are there transformations or transformation types, which are usually used together?

4.5 Reaching a technical Realisation

Enabling users with little IT knowledge to use data lakes is desirable. In literature, approaches can be found, which circumvent the need for end users to produce implementations of their transformation sequence, which run directly on the underlying infrastructure themselves. Two of these are automation and abstraction of transformations. Fully automated transformations however are impossible, humans have to be involved in the process [BHM18]. The end goal of pre-processing has to be provided by humans with domain knowledge. Humans with domain knowledge should be incorporated into the process of specifying the transformations, in order to prevent a mismatch between desired result and actual result [BHFM20].

Jin et al. [JACJ17] aim to provide fully automatic data transformations with their system FOOFAH. Users specify for a small amount of stored example input data, how the output data is supposed to look. FOOFAH then analyses the example and automatically produces a program, which performs the transformation. For example it could be used for transforming semi-structured data into a relational schema or data from an unusable relational schema into a usable one. However this only works on simple transformations so far. Furthermore it is only meant for schema transformations. Therefore, while it could be used to help with schema transformations, it does not provide any help for different types of transformations. It provides no general solution, which enables automatic data transformation for any kind of data pre-processing.

A slightly different approach is presented by Behringer et al. [BHFM20]. Here users see part of the data in a spreadsheet-like environment and can specify their transformation steps via drop-down menus of possible operations and parameters for the operations. The transformations are automatically executed and results are shown in the spreadsheet. Possible conflicts based on mismatch between sample data and the entire data set are shown, so that the data scientist can adapt the transformations accordingly. However this approach only works on structured data, which can be displayed in a spreadsheet. It is therefore not suitable for use in data lakes. In data lakes semi-structured data and unstructured data also occur. Users have to be able to transform those kinds of data as well to prepare them for their use case.

Constance, introduced by Hai, Geisler, and Quix [HGQ16], provides abstraction to a higher level querying language. They introduced an abstract query language, which queries for data based on metadata and allows to perform simple transformations like filtering or join operations. The user has to provide the specification of the transformation as a query. Internally, Constance searches for data, which has the metadata features specified in the query. It then replaces the description of the data in the queries with the actual data sources. For each data source, the new queries are translated into lower level queries, which work directly on the used data bases and infrastructure, of

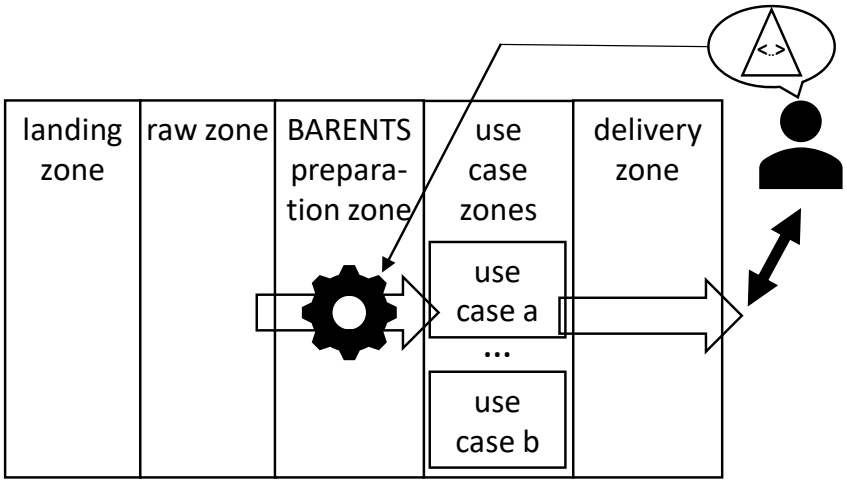


Figure 4.2: BARENTS architecture.

the data source. Constance automatically runs these low-level queries directly on the infrastructure. The results are collected, combined and returned to the user. However in this query language so far only generic queries are possible. The user is very restricted in which transformations can be accomplished. Not all new use cases are answerable using this query language.

BARENTS, a very recent approach introduced by Stach et al. [SBE+21], provides abstraction of transformation steps by expressing them as ontologies. Ontologies are a concept to provide semantic descriptions of objects, relations and rules in a domain [BHRZ11]. Data scientists provide metadata on what features the data, produced by the pre-processing, is supposed to have in the form of ontologies. For each transformation step, they provide which data should be used as input, which transformation should be executed on the data and the zone in which the output data should be stored. The collection of these informations for all transformation steps are the ontology. BARENTS parses the ontology and pre-processing steps are derived from it. The architecture of BARENTS, illustrated in Figure 4.2, is based on a zone architecture. In the preparation zone, the pre-processing steps derived from the ontology serve as configuration. Affected data is requested from the raw zone by the preparation zone. There the required transformations are applied sequentially to the raw data. Then the results are forwarded to the specified use case zone. The user only has to provide the ontology and BARENTS handles how to perform the transformations. This enables users to specify transformations in a flexible and abstract way. So far, BARENTS only supports four types of transformations. The types are filter, map, reduce and procedure. Filter allows to remove unwanted data points and attributes from the data. In the categories of transformations by Megdiche, Ravat, and Zhao [MRZ21] it is equal to filtering. Map is used, if each data point should be modified according to the same rule. It is equal

to the categories formatting and calculation. Reduce aggregates all data points into a single value. It is equal to the category aggregation. Procedure is used in all cases, where the transformation does not fit filter, map or reduce. In this case a user-defined function has to be provided, which carries out the transformation. There are however additional common transformation categories, which should be supported natively and not have to be provided as user-defined function. In the categories by Megdiche, Ravat, and Zhao [MRZ21] these are consolidation, merging and joining. Consolidation is a broader category. Some of its capabilities are covered by the BARENTS types filter, map and reduce. One important functionality however is not covered: splitting data. In some use cases, splitting a data set and then aggregating the data per split is necessary, for example when calculating monthly averages. Merging data for the same attributes from different data sets and joining data sets to combine different attributes are also important capabilities. BARENTS is still lacking splitting, merging and joining as natively supported transformation types. By adding those, BARENTS could be greatly improved.

4.6 Lessons Learned regarding existing Solutions

While there is a lot of research, providing ways to assist users in creating data preprocessing for new data analyses, not all steps of the process are equally supported. The results of this literature review are summarised in Table 4.1.

As can be seen, there is a lot of research on how to find relevant data and additional related data. Research on this topic is very mature and support for these steps is sufficiently covered by literature. The user can be aided in finding relevant data through easier data discovery. This is enabled through search and filtering capabilities of data and visualisation of the data available in a data lake in traverseable structures, based on their metadata. All aspects of data discovery are sufficiently covered. These are partitioning the data, gathering and enriching metadata, presenting the data in the lake and the corresponding metadata to the user in an accessible manner and advanced querying of data based on various metadata. Finding related data is also sufficiently supported by the current state of research. Support for the user is provided by relatedness graphs, along which data sets can be explored. Easy discovery of data for joint transformation is enabled. Both finding data with additional attributes, as well as finding additional data for the same attributes, is supported by recommender systems. Relatedness of data can be calculated based on different features of the data for different purposes. The relationships can also be presented to the user in different manners, depending on the purpose, to enable user friendly interaction.

In contrast, the other two steps, choosing the right transformations and creating a technical realisation, which performs the designated transformations on the data, are

Steps	**User support through**	**Discussed in**	**Sufficiently covered by literature**
Finding relevant data	- easier data discovery - filtering of data based on metadata - visualisation of available data based on metadata	Giebler et al. [GGH+20b], Inmon [Inm16], Langenecker et al. [LSSB21], Fernandez, Subramaniam, and Franklin [FSF20], Francia et al. [FGG+21], Megdiche, Ravat, and Zhao [MRZ21], Eichler et al. [EGG+20], Hai, Geisler, and Quix [HGQ16], Diamantini et al. [DLP+21], Dibowski and Schmid [DS21], Gröger and Hoos [GH19], Halevy et al. [HKN+16]	yes
Finding related data	- exploration of data sets along relatedness graphs - easier discovery of data for joint transformation - recommender systems for additional interesting data	Bogatu et al. [BFPK20], Fernandez et al. [FAK+18], Diamantini et al. [DLP+21], Alserafi et al. [AARC20], Zhang and Ives [ZI19], Halevy et al. [HKN+16], Alserafi et al. [AARC19]	yes
Choosing transformations	- metadata on processing steps - findability and reusability of existing implementations and processing workflows	Megdiche, Ravat, and Zhao [MRZ21], Behringer et al. [BHFM20]	no
Technical realisation	- automation of easy transformations - abstraction of processing steps	Hai, Geisler, and Quix [HGQ16], Jin et al. [JACJ17], Behringer et al. [BHFM20], Stach et al. [SBE+21]	almost

Table 4.1: Current state of research on user support for creating new data analyses in data lakes.

only insufficiently covered by literature. Approaches from literature can currently only provide rudimentary assistance in choosing the right transformations. The support is either limited to structured data and is therefore insufficient for data lakes, or only lies in collecting metadata on processing steps. The metadata increases the findability and reusability of existing implementations of transformations and processing workflows. There is a lot of future potential in analysing this data. Prefiltering of possible transformations, based on the type of input and recommending transformations based on previous usage patterns would be achievable further steps, which are not yet covered by literature. Extending the research in that direction would be an interesting axis for future works. While the research on how to help the user to reach a technical realisation is a bit more advanced, it still leaves room for improvement. Users can currently be supported through automating translation into executable implementations and execution thereof. However, this only works for easy schema transformations or for exclusively structured data. Abstraction of processing steps into a higher level representation and automatic execution thereof provides much more flexible pre-processing options. The respective approaches however have not reached their full potential yet. Constance for example allows for only limited pre-processing options. BARENTS, while being extensible, is still lacking a few types of common transformations as natively supported transformation types. Hence, this step is only almost sufficiently covered.

In accordance with other research [GGH+20b; HQJ21; RZ19], the conclusion is reached, that the latest state of research is still far from providing an automated process for creating data pre-processing for new use cases. There are however first approaches for automating some of the steps. While fully automated data processing might never be possible, providing a lot of tool support and automation of parts of the procedure are possible and should be the goal.

5 Inferences with respect to a new Concept

Insights regarding research gaps and areas where a demand for improvement of data preparation in data lakes exists, can be derived from Chapter 4. These insights will be discussed in the following chapter and conclusions regarding the focus and characteristics a new concept for data preparation in data lakes should have are drawn.

As explained in Chapter 3, a new concept for data preparation should be compatible with existing solutions (R3). There are already existing solutions, which address some of the challenges of data lakes, in the context of user-friendly data preparation, as discussed in Chapter 4. If users are already supported to a satisfactory degree in a specific area, improving upon those solutions, should not be a high priority. The concept should provide novel means of assistance, where current concepts leave a gap. Improving upon an area, where user assistance is still lacking should be a priority over improving upon areas with sufficient, well established means of user assistance.

A user performs four steps, which were identified in Section 4.1, when creating the data preparation for a new analysis task. Namely, 1) finding data relevant to the use case, 2) searching for related data in order not to accidentally miss some relevant data, 3) choosing the right transformations, which have to be performed on the data and finally 4) reaching a technical realisation of those transformations. The degree of support for each of the steps is different, as visualised in Figure 5.1. For some of the steps there are existing, well established, efficient solutions. These steps are colored green in the figure. For other steps the support is still insufficient, colored red, or the existing solutions haven't reached their full potential, colored yellow.

For the first step, finding relevant data, the support is sufficient. Solutions for thorough support in data discovery are available to the user. Data partitioning and metadata management provide sophisticated solutions for getting an overview over data and finding relevant data. Since this step is sufficiently supported, a new concept does not have to focus on providing assistance for this step. The concept should however be compatible with the existing solutions, in order to be embeddable in an existing data lake. Architectures for data partitioning help the user in getting an overview over data. Examples for those architectures are the pond architecture and the zone architecture. As

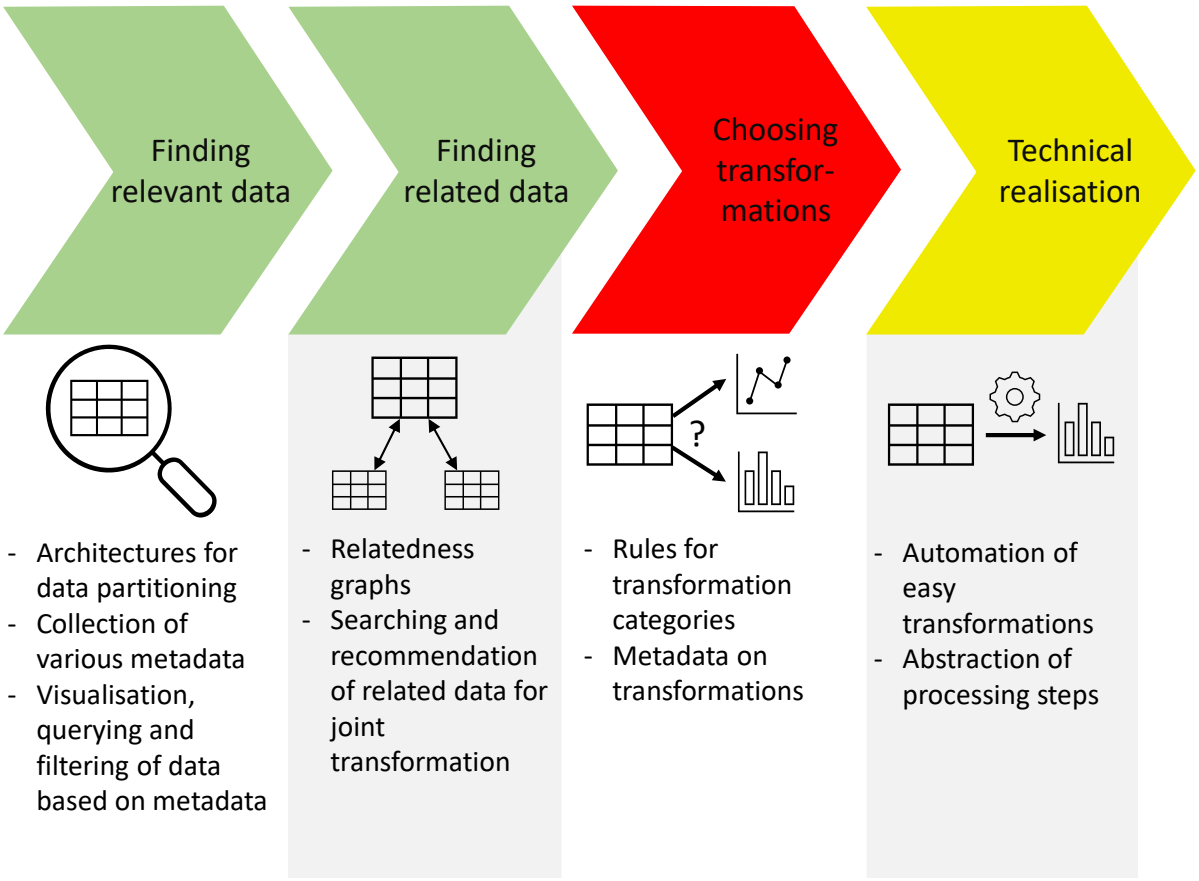


Figure 5.1: Steps for creating data pre-processing for a new use case and current state of research regarding each step.

the zone architecture retains all raw data in its original form but the pond architecture does not, the zone architecture is better suited to general use. Therefore the new concept has to be compatible with zone architectures. Metadata management provides sophisticated ways to find relevant data. Collection of and enrichment with various metadata allows to retain information about many features of data. Especially metadata regarding schema, semantic meaning, provenance and previous usage in transformations can provide helpful insights when working on data pre-processing. Metadata can be collected in data catalogs for the user to allow for filtering or querying, or stored in graph structures for a traverseable graphical representation. As collection, enrichment, storage, querying and visualisation of metadata are not affected by approaches which do not directly concern metadata management, a new concept will automatically be compatible with these solutions. The new concept could however serve as a new source from which metadata can be collected, if it stores information about the data.

The second step is finding related data. Relatedness measures of data are used to prevent overlooking some of the relevant data, which should have been included in an analysis. Relatedness of data can be calculated in different ways and presented to the user through visual graph representations, querying capabilities and recommendations for additional interesting data. This step is therefore sufficiently supported and the new concept should not focus on providing additional assistance for this step. Relatedness of data can be calculated based on different features extracted from metadata about the data and by analysis of the values or content of data. Relatedness graphs can give a visual representation of relations and querying for and recommendation of close relatives are available to the user. As the calculations are based on metadata or data itself and either calculated on-demand or stored in separate structures, a new concept will not interfere with these solutions. A new concept will automatically be compatible.

Choosing the right transformations is the third step. Existing support consists of general rules on which categories of data are compatible with which categories of transformations and collecting metadata on previously performed transformations. The general rules are however too broad and do not narrow the transformation options down significantly in most cases. Additionally, data and transformations can be categorised differently. A user has to match the data and transformation categories of the rules to the ones she or he is working with, without support. The metadata on processing steps is only collected. Users have to analyse it themselves, if they want insights into which transformations can be used on which data. For the second step, users can simply filter or query directly for most related data or get automatic recommendations. In comparison users cannot simply query for recommended transformations and have to filter the possible transformations themselves. User support in choosing the right transformations to use on the data is insufficient. There are many ways in which to improve upon the existing solutions. Using the existing general rules as a basis, possible transformations can be automatically filtered based on which data the user selected as input. By analysing collected metadata on the previous usage of transformations, patterns can be detected. In combination with metadata about input data, more precise recommendations for transformations can be provided for a selected input. Patterns on which transformations are often used in combination can be used to recommend additional transformations in a transformation sequence. When additional data about the users who performed certain transformation sequences is available, transformations can be recommended based on usage patterns of similar users. Because the support for this step is still lacking, a new concept should provide support for this step. It could be based on any of the ideas for improvement introduced above.

Lastly, the fourth step is creating a technical realisation of the planned data preparation. Users can be supported through automation of transformations based on examples and abstraction of processing steps. The approaches however are still lacking to differing degrees. Automation based on examples only works for simple schema transformations.

Abstraction of transformations works slightly better. One of the solutions provides an environment, where transformations are specified via drop-down menu, but which only works for structured data. A different approach provides an abstract language, which only allows for very limited pre-processing options. Another one, BARENTS, allows abstract specification of data preparation where any type of transformation is supported. However, while BARENTS supports some transformation types natively, other common transformations are only supported trough extensibility by user-defined functions. None of the solutions natively support all types of data and all types of transformations. Therefore the support for this step is not yet sufficient. Providing a user-friendly way of specifying the transformations which should be performed is not sufficient, if this does not include some transformations, which users frequently need. BARENTS, e.g. does not support some very common steps in data pre-processing through its native transformation types, namely merging, joining and splitting of data sets. The concept could be easily improved by adding additional native transformation types in order to cover more of the common transformations. Then it would provide very sophisticated, user-friendly means for a user to provide an abstract description of the pre-processing needs and leave the technical realisation to the platform. Sufficient support for this step could therefore easily be reached. As the support in this step is almost sufficient, while a new concept could improve upon this step, it should not be prioritised over the third step, where support is still insufficient. A new concept should be compatible with BARENTS, as this is currently the best solution for support in this step. Because BARENTS is based on a zone architecture, a new concept which is compatible with BARENTS would be compatible with zone architectures, too.

In conclusion, support for the first and second step, finding relevant and related data, is sufficient and support for the third step, technical realisation of data preparation, is almost sufficient. The third step, choosing the right transformations however is not sufficiently supported. Hence, when creating the new concept, providing help in choosing the right transformations should be a focus. Accordingly, the new concept will help the user in choosing the right transformation sequence to specify the data preparation needs for her or his use case. As explained in Chapter 3, a new concept for data preparation should be user-friendly enough, to be usable with little IT knowledge (R2). By providing the abstract description of the transformation sequence in the form of ontologies, the new concept will be compatible with BARENTS. Since BARENTS provides an abstract way of describing the data preparation and the user can leave the technical realisation to the platform, this would provide user-friendly means of specifying the data preparation. By ensuring compatibility with BARENTS for the technical realisation, compatibility with zone architectures for data partitioning is automatically reached as well. The new concept will therefore help the user in creating ontologies, that describe the data pre-processing, which can be used as input for the configuration of data pre-processing in BARENTS. While a new concept should be compatible with existing

solutions (R3), most solutions work very independently of the rest of the architecture anyway. For the rest of the solutions, while a new concept can easily use them, no special consideration has to be taken to ensure compatibility.

In summary the new concept should therefore:

L1 Provide assistance for the user in choosing the right data transformations.

L2 Help the user in formulating the transformation sequences in the form of ontologies.

L3 Provide input for the configuration of data pre-processing in BARENTS.

6 Discussion of Concepts

As discussed in Chapter 5 existing solutions do not provide sufficient support in all steps a user has to perform while specifying pre-processing for new use cases. The goal for the rest of this work is therefore to come up with a new concept towards a user-friendly and easy way to specify data pre-processing tasks.

The concept provides assistance to a user for choosing the right transformations to perform on the data, as discussed in Chapter 5 (L1), since this is the step, where users are currently insufficiently supported. This can be achieved by different means like automatic pre-filtering of transformations based on which data a user is viewing or by providing a pre-computed list of suggested transformations for all data in the data lake. A flexible solution, regarding the way the user interacts with it and the way the suggested transformations are presented, is a separate interactive recommender. Interaction with the recommender could happen either directly or through an additional user interface. Possibilities on how to realise this recommender are discussed in Section 6.1

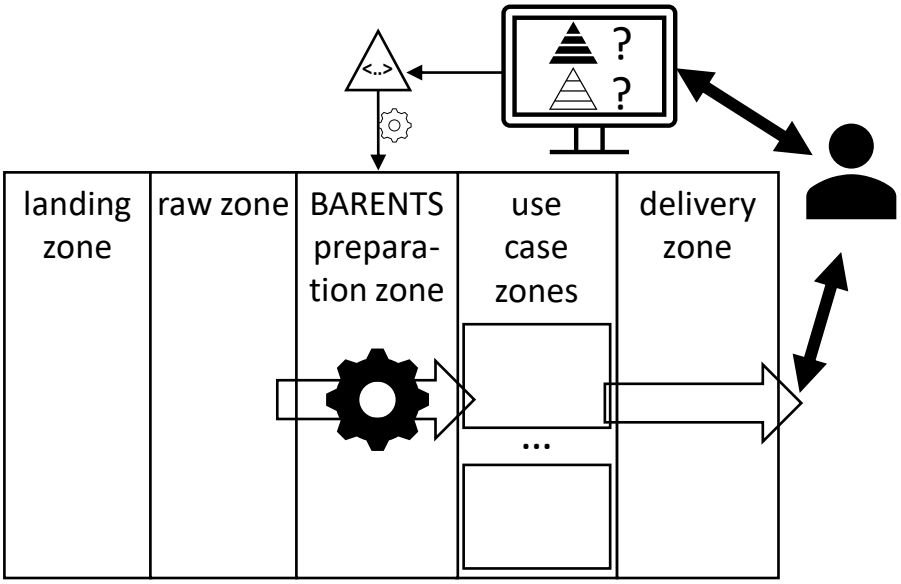


Figure 6.1: The new concept, based on the BARENTS architecture. A recommender system aids the user in creating the ontology, which serves as configuration for the data processing in BARENTS.

The new concept should be compatible with BARENTS, as discussed in Chapter 5, since BARENTS is a good solution for technical realisation of data pre-processing. As stated in Chapter 3 (R1), a concept for specifying data preparation should be flexible enough to be applicable in any domain and for any use case. Compared to other concepts for technical realisation of pre-processing, like FOOFAH and Constance, BARENTS is more universally applicable and flexible. As BARENTS can be used for any transformation and on any data, it can enable any use cases. The used representation of data preparation as ontologies also works within any domain. A concept for specifying data preparation should also be user-friendly and require little IT knowledge to use, as specified in Chapter 3 (R2). BARENTS provides a user-friendly way for specifying pre-processing needs and applies the specified transformations on the data for the user. It abstracts the implementation of pre-processing steps away from users through configuration of data pre-processing in the form of ontologies. Users do not need a lot of IT knowledge, when working with BARENTS in data preparation, as they do not have to provide code themselves. A new concept should also be compatible with existing solutions towards user assistance in data preparation within data lakes, as specified in Chapter 3 (R3). As discussed in Chapter 5, by being compatible with BARENTS, a new concept is also compatible with zone architectures and is automatically compatible with the rest of the solutions. The new concept uses BARENTS as a basis for performing specified transformations, which works towards fulfilling requirements (R1), (R2) and (R3). However with BARENTS, users still have to write the ontology, which specifies their data preparation, themselves to provide the configuration for the data preparation zone in BARENTS. The new concept assists the user in providing input for the configuration of data pre-processing in BARENTS (L3). This way the new concept is even more user-friendly, than using BARENTS directly. The concept has an interactive component, where the user can specify the pre-processing in cooperation with the recommender. A finished specification is then used as configuration for BARENTS and BARENTS carries out the specified data preparation. This concept is visualized in Figure 6.1.

The specified data preparation which BARENTS uses as configuration must be formulated in the form of ontologies. The recommendation component therefore provides help to the user in formulating the transformation sequences in the form of ontologies (L2). A graphical representation of one transformation in the ontology of BARENTS can look like in Figure 6.2. BARENTS works with different layers. Attributes of the source data are found in the data layer. Temporary data generated as output of a transformation step is found in the information layer. Result data, which is stored in one of the use case zones, is found in the knowledge layer. For data in the data layer, the source of the data in the raw data zone needs to be specified. In the example, *indoor temperature* and *outdoor temperature* from a source containing weather data in the raw zone, are present in the data layer. It needs to be specified as well, as part of which results in the knowledge layer or part of which transformation, that produces an interim result in the information layer,

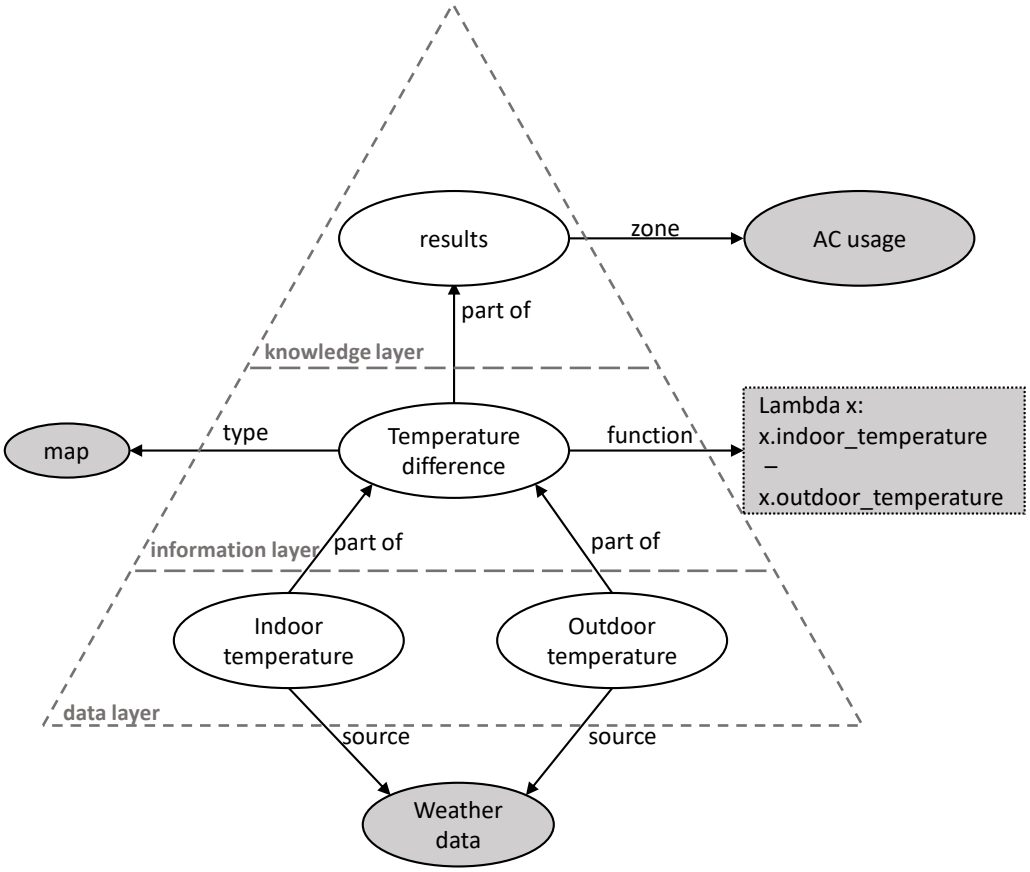


Figure 6.2: Example for a graphical representation of part of a BARENTS ontology.

the data is used. Indoor and outdoor temperature are used for calculating the *temperature difference* in the example. For the interim results in the knowledge layer, the type of transformation needs to be specified, as well as the function, which is evaluated as part of the transformation. In the example, a mapping transformation is performed, which calculates the function $x.indoor\_temperature - x.outdoor\_temperature$ for each data point x in the source. The results of this mapping are available as *temperature difference* in the information layer. It is also specified, as part of which further transformations within the information layer or results in the knowledge layer the interim result is used. Temperature difference is a part of *results* in the knowledge layer, in the given example. For the results in the knowledge layer, the zone where the result is stored is specified. In the example, the result named *result* is saved in the use case zone *AC usage*.

Listing 6.1 Example for the RDF/XML representation of the ontology in Figure 6.2.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dl="http://example.com/">
   <rdf:Description rdf:about="http://example.com/indoor_temperature">
      <dl:layer>Data Layer</dl:layer>
      <dl:source>weather_data</dl:source>
      <dl:partOf rdf:resource="http://example.com/temperature_difference"/>
   </rdf:Description>
   <rdf:Description rdf:about="http://example.com/outdoor_temperature">
      <dl:layer>Data Layer</dl:layer>
      <dl:source>weather_data</dl:source>
      <dl:partOf rdf:resource="http://example.com/temperature_difference"/>
   </rdf:Description>
   <rdf:Description rdf:about="http://example.com/temperature_difference">
      <dl:layer>Information Layer</dl:layer>
      <dl:function>lambda x : x.indoor_temperature - x.outdoor_temperature</dl:function>
      <dl:type>map</dl:type>
      <dl:partOf rdf:resource="http://example.com/results"/>
   </rdf:Description>
   <rdf:Description rdf:about="http://example.com/results">
      <dl:layer>Knowledge Layer</dl:layer>
      <dl:zone>AC usage</dl:zone>
   </rdf:Description>
</rdf:RDF>
```

While the ontology can be created in any way, for example with graphical tools, as input for BARENTS it needs to be specified in RDF/XML[1] representation. The resource description framework (RDF) [SR14] is a formal language for modelling of ontologies. RDF describes the relationships between objects in an ontology, in the form of triples, containing a subject, a predicate which describes the relationship and an object. A possible RDF/XML representation of the previous example can be seen in Listing 6.1. For example the subject indoor temperature is connected via the predicate *partOf* to the object temperature difference and the subject temperature difference is connected via the predicate *function* to an object, which contains the representation of the function.

When use cases change the ontology needs to be adapted. The user however does not necessarily need to provide new rules for every transformation step. Some of the already specified transformation rules from other use cases might be partly or wholly reusable for the new use case. To reuse the defined rules however, the user needs to be able to find their specification. To prevent users from having to define the same transformation rules over and over again, a recommender can help the user in finding the ontology specification of an applicable transformation. A recommender can traverse the

[1] RDF/XML syntax definition: http://www.w3.org/TR/rdf-syntax-grammar/

existing RDF graph and search for transformations with specific type, input or following transformations. By providing the transformations in their RDF/XML representation, a recommender helps the user both in finding the right transformations and writing the specification. Writing the RDF triples, which specify the needed transformations for the ontology is easier for the user when part of an RDF graph is provided, which only needs to be adapted. The usage of BARENTS in combination with the recommender is very user-friendly and requires little IT knowledge, which works towards achieving (R2). There are many possible ways to realise this recommender, which are discussed in Section 6.1.

6.1 Recommender Concepts

The recommender always receives some input from the user and provides recommended transformations in the form of corresponding RDF triples. This way the user gets a transformation recommendation and help in writing the RDF triples in one step. The user can then adapt the provided RDF representation to her or his needs. A finished RDF graph is then used as configuration in BARENTS. How the recommendation for the user is reached can be based on different approaches and which input the user has to provide depends on the approach.

The whole ontology, which BARENTS uses as configuration, is available in a single RDF graph. This graph can serve as a source of information on how transformations are being used so far. A separate version of the RDF graph can also be stored for the recommender. RDF graphs can easily be merged [Tea09]. Each time the configuration of BARENTS is changed, the new ontology can be merged with the stored graph. The separate store can therefore be used to retain information about previously used transformations or transformation sequences, which were removed from the current configuration for analysis purposes.

Additional resources like metadata catalogs for data in the data lake or user profiles with metadata about the user could also be used as basis of analysis in order to provide nuanced recommendations. Below, four approaches for how the recommender could work are introduced and discussed:

Recommendation Approach 1: In this approach, represented in Figure 6.3, the user provides data that should be transformed as input. Recommendations in this approach are based on the general rules by Megdiche, Ravat, and Zhao [MRZ21]. For example, if two sources are selected, the transformation must be a merge or a join, while for one data source the transformation must be a filtering, formatting, aggregation, calculation or data splitting operation. If the data is unstructured, aggregation, calculation and

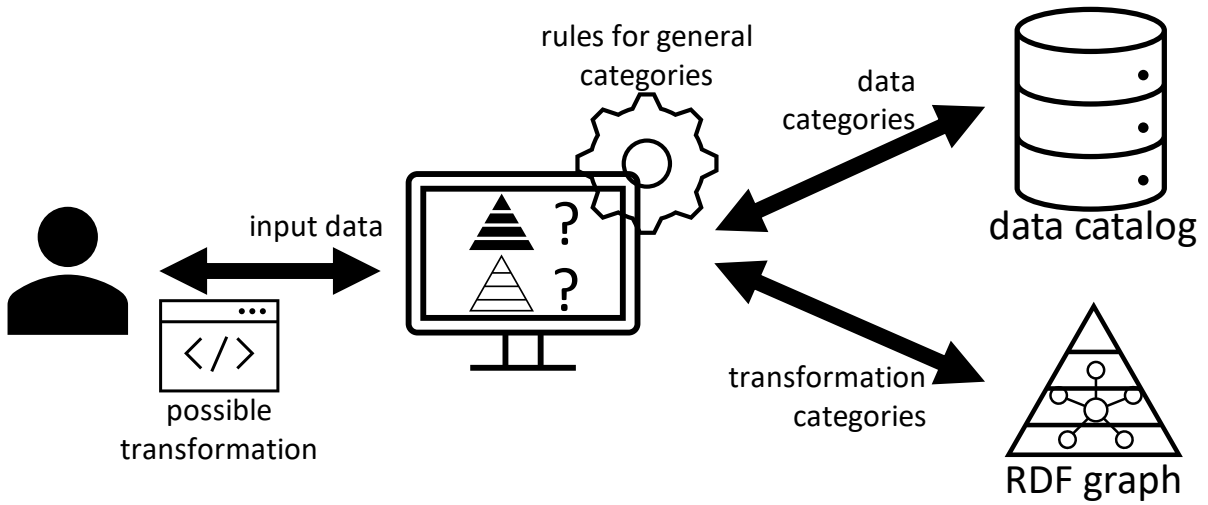


Figure 6.3: Recommendation Approach 1.

join cannot be applied. Additional rules can be used as well. Depending on the kind of data values, certain operations can or cannot be performed. For example for text, mathematical operations like + or - and some statistical operations like mean and median cannot be used, but text equality, searching for sub-strings or calculating a levenshtein distance work. At the same time, while all the mathematical and statistical operations can be used on numerical values, searching for sub-strings or calculating a levenshtein distance makes no sense.

Information on past data preparation from the RDF graph and metadata from a data catalog are used in this approach. The recommender uses metadata on the data to infer, which category of data this belongs to and what kinds of values are present. A JSON file for example is semi-structured, while a text file is unstructured. Existing transformations in the RDF graph are sorted into transformation categories based on their type. For instance, a BARENTS filter is a filter, a reduce is an aggregation, a map can be formatting or calculation and splitting, merging and joining can be made available via a procedure with corresponding user-defined function. The definition of the function can be used in addition, to look for occurrences of operations, which only work on certain types of values. Based on the rules, for each of the transformation categories, which can be performed on the input data, one example from the existing transformations in the RDF graph is selected. If the user wants to transform input from a text file, the recommender might therefore recommend a filter transformation, which looks for occurrences of keywords within the text. For relational data with numerical values however, a reduce, which calculates mean values for each column might be recommended. The RDF/XML representation of each transformation is presented to the user. This way the user can

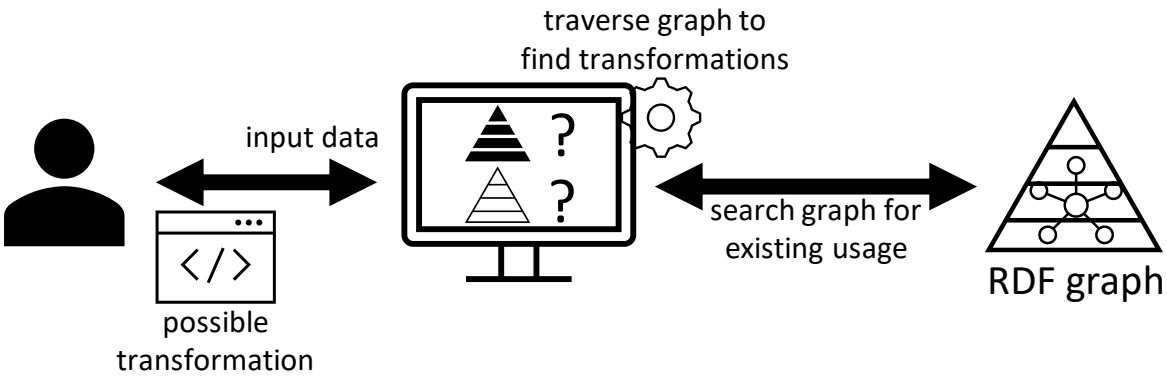


Figure 6.4: Recommendation Approach 2.

explore the ontology for existing transformations, which could generally be applied to the input data. In case no transformation for a category exists in the ontology, a basic example transformation for the category is provided by the recommender. This way the user at least has an example of how a corresponding transformation might be expressed in the ontology. The user can then choose one of the provided transformation examples and adapt it to her or his needs.

This approach works, even if no ontology exists, as it can at least provide examples for possible transformations. However recommendations based on general rules will always be rather generic and might not reduce the number of options enough, if many implementations are already available. It is also highly likely, that this approach will present many false positives to the user, because not every constraint can be easily expressed in a general rule. Additionally it depends on the availability of accurate metadata.

Recommendation Approach 2: Similar to the first approach, the user provides data she or he wants to transform, as can be seen in the visualisation in Figure 6.4. In this approach recommendations are based on prior usage of this data, by analysing the existing ontologies. The approach can provide the recommendation either only for one attribute of the data, which exists as an object in the data layer of the RDF graph or for an entire data source. If the recommendations should be provided for a data source, the recommendations are provided for each attribute from this source, which exist as objects in the data layer. Only information on past data preparation from the RDF graph is used. The recommender searches the stored RDF graph for occurrences of this input data. By traversing the graph, transformations for which the data previously served as input can be extracted. This is done by following the *partOf* relations within the graph. All triples with either the transformation or the attribute as subject can be collected, to provide the whole definition of the relationship to the user. A list of transformations in RDF/XML

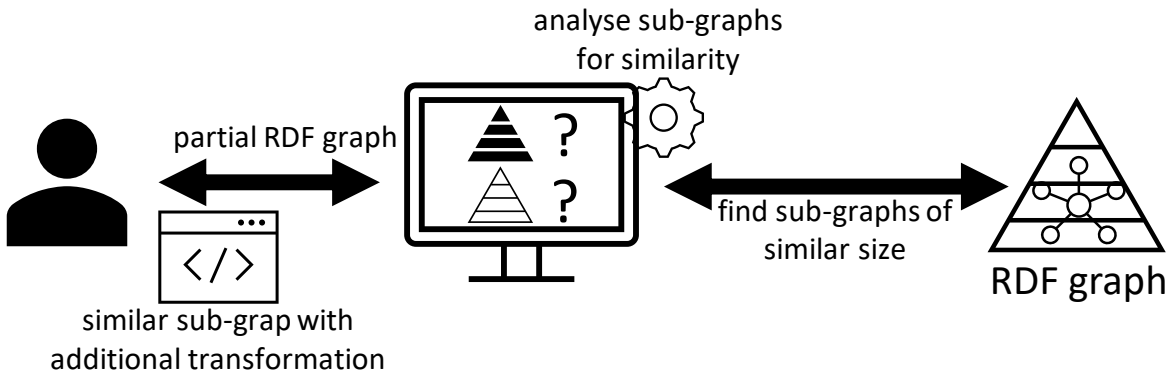


Figure 6.5: Recommendation Approach 3.

representation together with the RDF/XML representation of the attribute is provided to the user. This way, if a user knows, that certain data is needed in analysis, but does not know how to reach the right pre-processing, the provided recommendations can serve as a starting point. A user finds transformations, created by other users of the data lake, which might directly solve the data preparation need of this user. It might also show new applications of the data to the user, she or he would not have thought of. If a user knows, that certain data will be needed in an analysis, but does not know the state of the data, the transformations used by others might give important insights. In case the number of found transformations is too big, it can be reduced by choosing only one example per BARENTS transformation type. Then the user can choose one of the provided examples and adapt it according to the preprocessing needs.

This approach has the advantage, that it only needs sources, which are available in BARENTS anyway, namely the ontology. It does not rely on the presence and quality of information outside the scope of influence of the concept itself. However, if the ontology has not been created yet, no recommendations can be made. Even if the ontology already exists, if it is small, it is likely, that for a lot of use cases none of the found transformations is applicable. In contrast, every transformation which is found is definitely applicable on the data exactly as it is. There can be no false positives. This approach is also able to improve over time. As the number of specified transformations increases, more recommendations can be made to the user. The likelihood of a usable transformation already existing increases, as more transformation rules are added.

Recommendation Approach 3: In this approach, shown in Figure 6.5, the user provides a partial RDF graph instead of input data. This way a user can not only find a single first transformation for data, but continually extend a graph with several pre-processing steps via the recommendations. For some data preparation use cases, long chains of transformations need to be applied to the data. With this approach users can be assisted

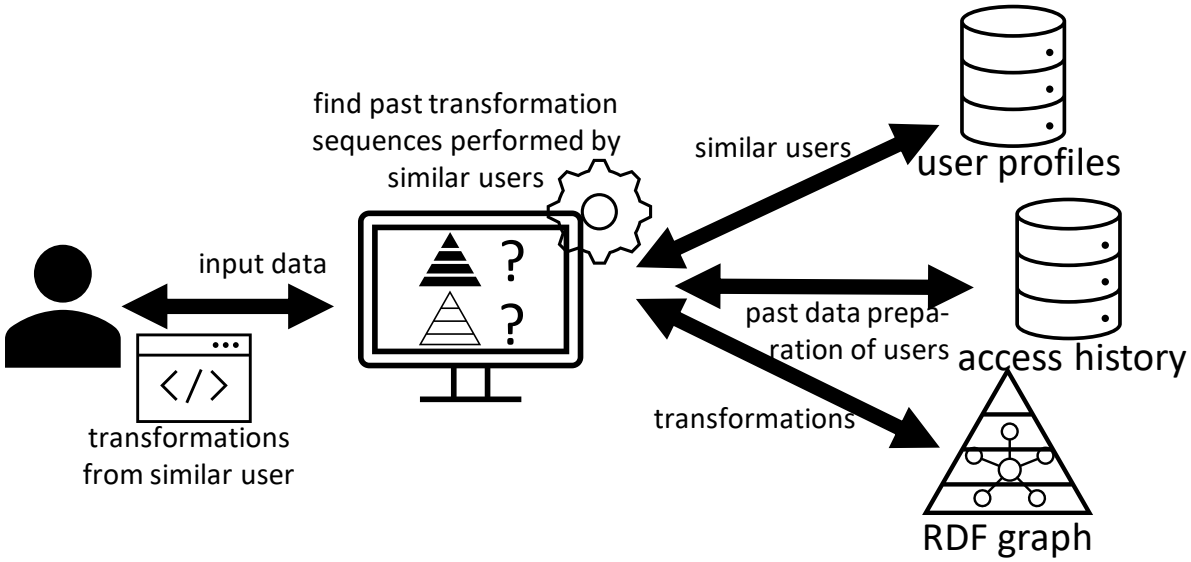


Figure 6.6: Recommendation Approach 4.

in the creation of such transformation chains. The recommendations in this approach are based on similar transformation sequences. Only information from the RDF graph on past data preparation is used. Based on the provided RDF graph, the stored graph is searched for sub-graphs of similar size. Those sub-graphs are analysed regarding similarity to the user provided graph. Sub-graphs, which have a few transformation steps in common with the provided graph, but contain one or two additional transformations, are selected. Those sub-graphs are presented to the user in RDF/XML representation to provide the additional transformations. A user can find additional transformation steps to continue the data preparation. She or he can also find additional intermediate steps, which might have been left out on accident. The user can choose one of the graphs containing additional transformations and adapt it to her or his needs.

Like Approach 2, this approach has the advantage, that it only requires information from the RDF graph for its recommendations. It does not depend on other sources, which might not be available or only provide low quality information. This approach has also an advantage over Approach 1 and 2 as it is the only one, which allows for attractively extending the graph. The others only allow for the recommendation of one initial transformation step.

Recommendation Approach 4: This approach, which is visualised in Figure 6.6, bases the recommendations on usage patterns of similar users. Information on past data preparation from the RDF graph is used in combination with user profiles and access history on which user performed which transformations in the past. The user provides the data he or she is interested in transforming to the recommender. Profiles of users

can be compared. Users with similar analysis interests, past performed data analyses, user groups or usage patterns of transformations can be identified. If a user requests a transformation recommendation for some input data, the recommender first identifies the users most similar to the current user. Access history can then provide information on whether these users have preformed transformations on the data the current user is viewing. For the users, which have performed transformations on the input data, the corresponding transformations are determined by referencing the access history and the RDF graph. From the access history, even information on what sequences of transformations similar users performed in the past, can be gathered. Those transformation sequences are extracted from the RDF graph. The chosen transformation sequences are presented in RDF/XML representation to the user, who can select one of them and adapt it accordingly. If there are several transformation sequences, they can even be ranked by how similar the user who performed them is to the current user.

This approach has the advantage, that it can find both single transformations as well as transformation sequences to the user. Compared to the other approaches it has the added benefit of providing an easy way of ranking the results. However this approach needs both a detailed access history, as well as user profiles containing extensive information about a user. It cannot be guaranteed, that this information is always available.

Additional considerations: A recommender could later be adapted to also work the other way around. The user chooses a transformation or provides a partial RDF graph and the recommender provides input data it could be applied to. For example, the recommender could analyse the existing ontologies as metadata on past transformations. It could find, which data transformations were previously used on and recommend the data to the user. However, as mentioned before, support in finding data is already sufficient, while support for choosing the right transformations is still lacking. This work therefore does not focus on this possibility any further.

6.2 Final Concept

Each of the four approaches presented above can serve as a transformation recommender, which helps in specifying RDF graphs as input for BARENTS. Table 6.1 gives an overview of the characteristics of each approach. Several approaches could also be combined, e.g. by providing a front end, where the user can choose, which of the recommenders should be used in the background. Combining approaches could also work by forwarding the request of the user to all recommenders in the background, gathering the recommendations from all approaches and providing some from each recommender to the user. By combining approaches, it is possible to first support the user in creating an initial partial RDF graph with one transformation of the input data,

	recommendation based on	**input data**	**output data**	**resources needed for the knowledge base**
Recommendation Approach 1	general rules for data categories and transformation categories	a data source	one example RDF/XML representation of a transformation per applicable category	RDF graph, metadata
Recommendation Approach 2	past usage of the inpt data in transformations	a data source	example RDF/XML representations of a few transformations previously preformed on this input	RDF graph
Recommendation Approach 3	past usage of transformations in transformation chains	a partial RDF graph	RDF/XML representation of a graph containing one or two additional transformations which have previously been performed directly before or after one in the input transformations	RDF graph
Recommendation Approach 4	pre-processing behaviour of similar users	a data source	RDF/XML representation of transformations which similar users previously performed on the input data	RDF graph, user profiles, access history

Table 6.1: Overview over the recommendation approaches.

provided by the user. Afterwards the user can iteratively extend the RDF graph, by requesting additional transformations for the new partial RDF graph. As most data preparations need more than one transformation step, iteratively extending the RDF graph should be possible. Recommendation Approach 3 is the only one, which allows for requesting additional transformation steps for an existing RDF graph. It is therefore used as part of the final concept. To provide a good starting point to the user, one of the other approaches should be used in addition to Approach 3, to provide an initial partial RDF graph.

Recommendation Approach 1 and Recommendation Approach 4 both depend on the existence of additional information sources with high quality metadata or access and user data. The existence of those sources, while likely, cannot be guaranteed. Gaining access to those resources might also be a hurdle, when trying to embed the new concept into a data lake. Like Recommendation Approach 3, Recommendation Approach 2 only depends on information from the stored RDF graph. Hence, using these approaches in combination would be a good combination.

The final version of the concept therefore uses a combination of Approach 2 and Approach 3 for the recommender, as visualised in Figure 6.7. First the user provides input data and receives recommendations for the first transformation in the form of a partial RDF graph. The user iteratively creates the ontology with the help of the recommender. When the user indicates to the recommender, that the ontology is finished, the recommender provides the RDF/XML file to BARENTS as new configuration for the preparation zone. The recommender then merges the new RDF graph into the saved graph. BARENTS applies the specified transformations to the data and provides the results of the pre-processing to the user via the delivery zone.

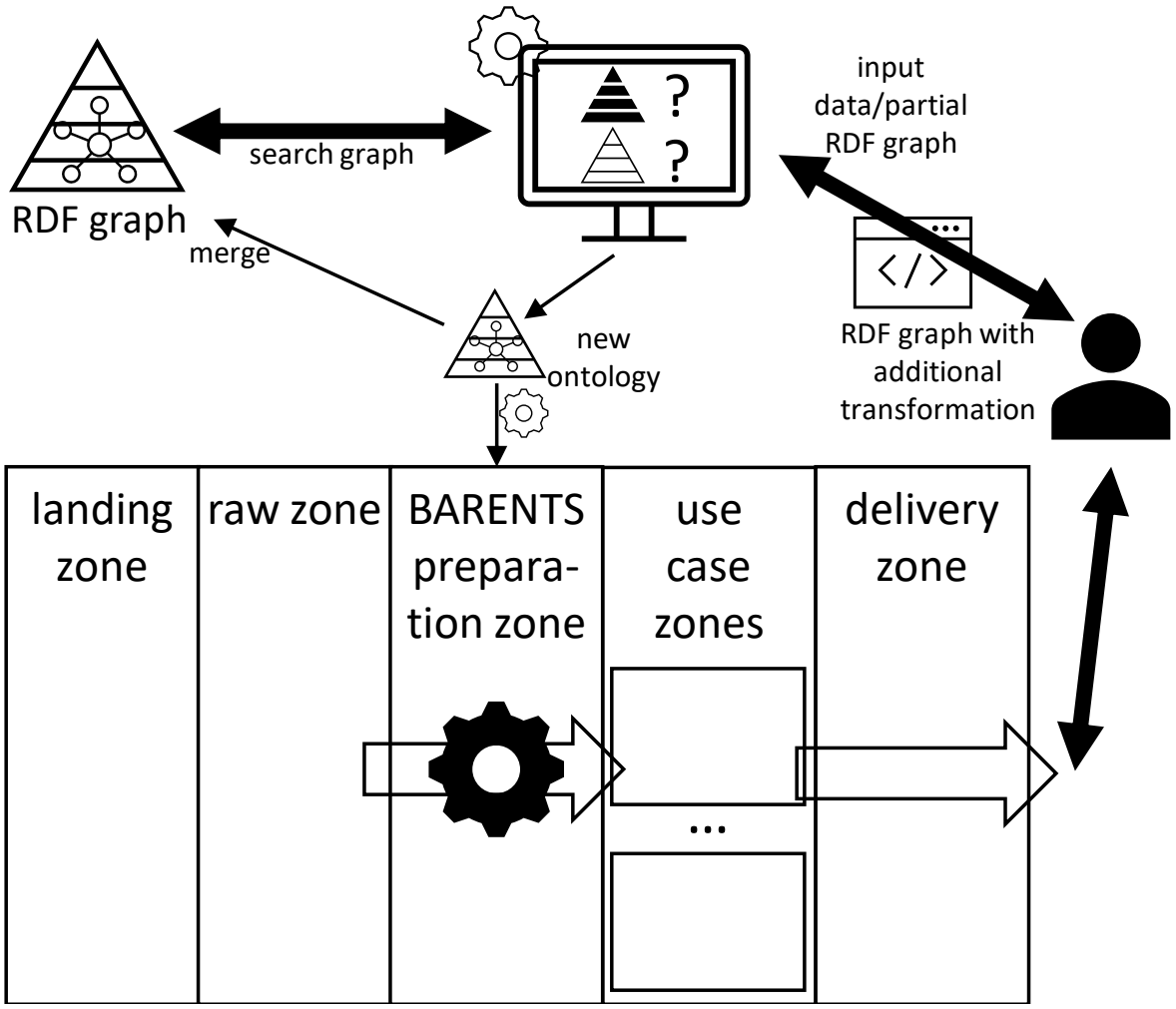


Figure 6.7: Final version of the concept, which uses a combination of Recommendation Approach 2 and Recommendation Approach 3 for the recommender.

7 Use Case

This chapter introduces a use case, which is used to exemplarily demonstrate, how the prototypical implementation of the concept assists the user in the specification of pre-processing. For this purpose a data set is introduced, on which analysis should be performed. Two pre-processing chains for different analyses are introduced, which will serve as status quo in the demonstration. A different analysis goal is introduced, which will serve as the new analysis goal of the user.

Gao et al. [GMB+21] collected indoor and outdoor weather data and the physiological and mental state of participants in a school. The weather data was collected over six months, only partly overlapping with four weeks where the data on the participants was collected. In the longitudinal indoor and outdoor weather study many measurements were collected inside several classrooms and directly outside of the school in five minute intervals. Measurements include indoor and outdoor temperature, indoor and outdoor humidity and state of the air conditioner. In the user study the physical state of users was tracked via wristbands which collected measurements such as acceleration of the wrist, skin temperature, heart rate and electrodermal activity. Via surveys, which participants submitted three times a day, additional data about the physical comfort and emotional and mental state of users was collected. Participants gave answers about their thermal perception and preference, current clothing level, engagement during the lectures and mental arousal. The data from the weather study alone can be used to analyse air conditioning usage patterns, as Gao et al. [GMB+21] suggested. In combination with the occupant study it could provide insights into relationships between indoor and outdoor climates and the physiological, mental and emotional state of occupants. The data is stored in several hierarchical directories of CSV data.

For the use case the following three data analysis scenarios will be considered. Scenario one and two will serve as previous analysis goals, for which the data preparation has already been specified. They will serve as the knowledge base in the demonstration. Scenario three will serve as the new data preparation need a user wants to specify.

Scenario one: The goal of the analysis is to see, if the difference between indoor and outdoor temperature has an effect on how occupants perceive the indoor temperature. In Figure 7.1 the resulting data pre-processed for this scenario is visualised.

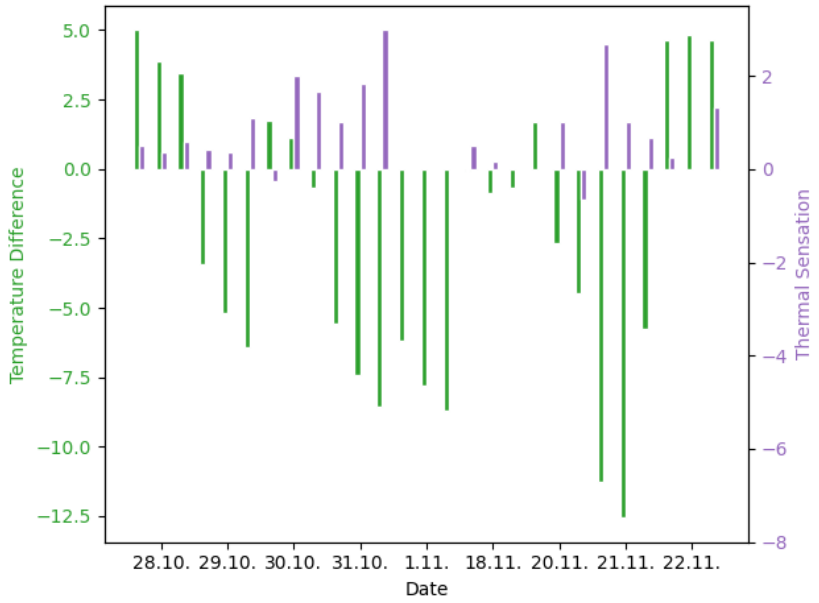


Figure 7.1: Visualisation of the output data of pre-processing scenario one.

As both weather data as well as participant data is needed but the studies only partly overlap, for both sources data outside that time frame has to be filtered out. This can be achieved through a filtering operation, which discards data for which the timestamp does not lie in this period of time. For the weather data, the temperature difference between inside and outside has to be calculated for each time of measurement. An example for how this transformation could be expressed inside an ontology can be seen in Figure 6.2 and Listing 6.1. A mapping transformation is used to calculate *indoor_temperature - outdoor_temperature* for each data point. Since the surveys, where participants rated their thermal perception were collected thrice a day in regards to the past time period, having precise times makes no sense. The times when the surveys were submitted and when the weather data was collected need to be mapped to one of three surveys per day: morning, noon or afternoon. Therefore a mapping function is used. Timestamps before 12:00 pm are mapped to morning, timestamps after 14:30 pm are mapped to afternoon and the rest is mapped to noon. Participants rated their thermal perception on a scale of -3 for cold to 3 for hot. An average thermal perception can therefore be calculated for each survey. A procedure for grouped aggregation can be provided by a user-defined function. It can be used to group by day and time of day and calculate the mean thermal perception for each group. The average temperature difference needs to be calculated as well and the results need to be joined based on the date and time of day. For the average temperature difference the same kind of grouped aggregation can be used. Data is again grouped by date and time of day and the average temperature difference between inside and outside is calculated per grouping. For the join, a different user-defined function

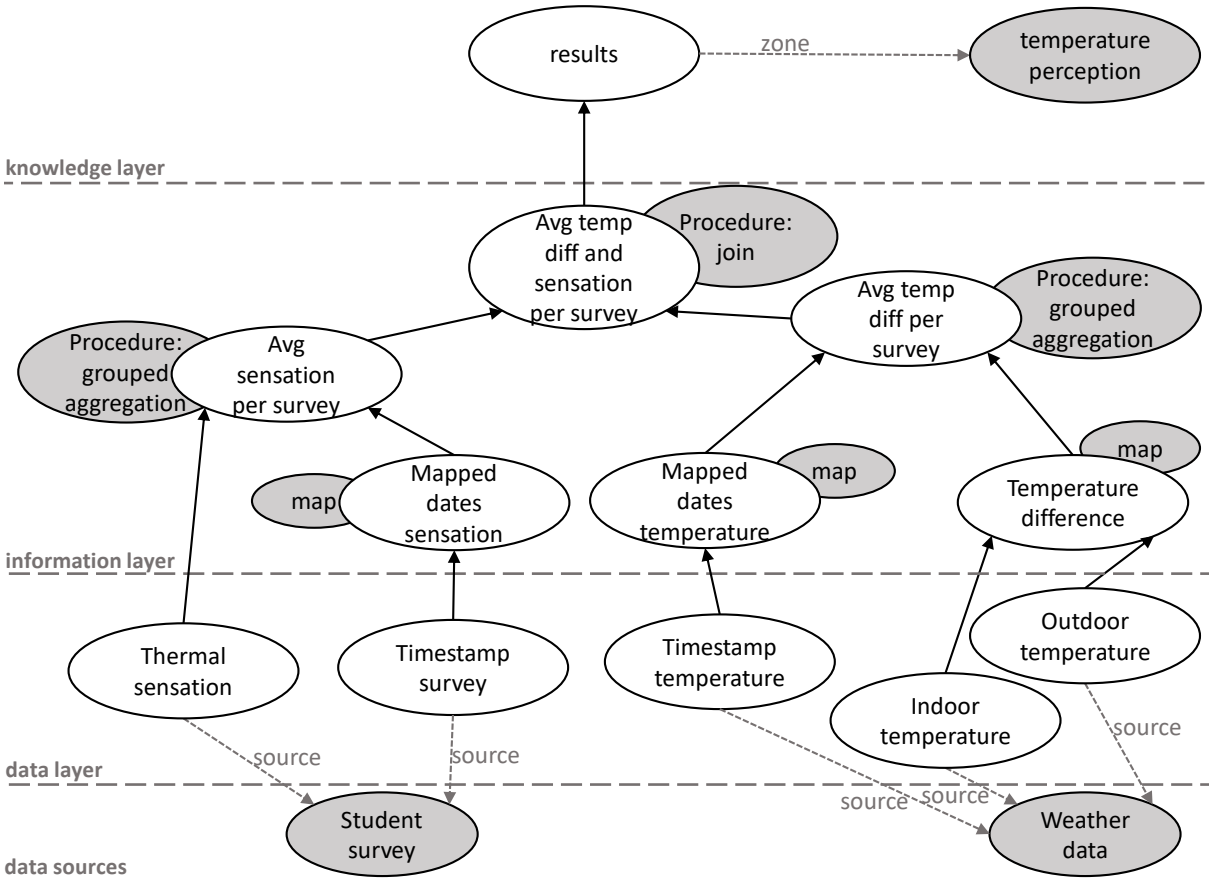


Figure 7.2: Visualisation of a possible ontology for pre-processing scenario one.

is used. The procedure joins the inputs based on the day and time of day, so that for each conducted survey the average temperature difference and the average thermal perception is available.

A possible representation of these steps as ontology can be seen in Figure 7.2.

Scenario two: The goal of this analysis is to see if there is a difference in how occupants perceive temperature between genders. In Figure 7.3 the pre-processed data of this scenario is visualised.

As the participant data, which contains gender information lies in a CSV file in a different directory from the survey data, both sources need to be joined based on the participant IDs. A join procedure similar to the last step of the previous scenario is used to combine the data from both sources. Then all male students are filtered out, so that only female students remain in one set and all female students are filtered out, so that only male students remain in another set. Filter transformations can be used for this purpose. One filters by *gender = 'Female'* the other filters by *gender = 'Male'*. In a first step the IDs

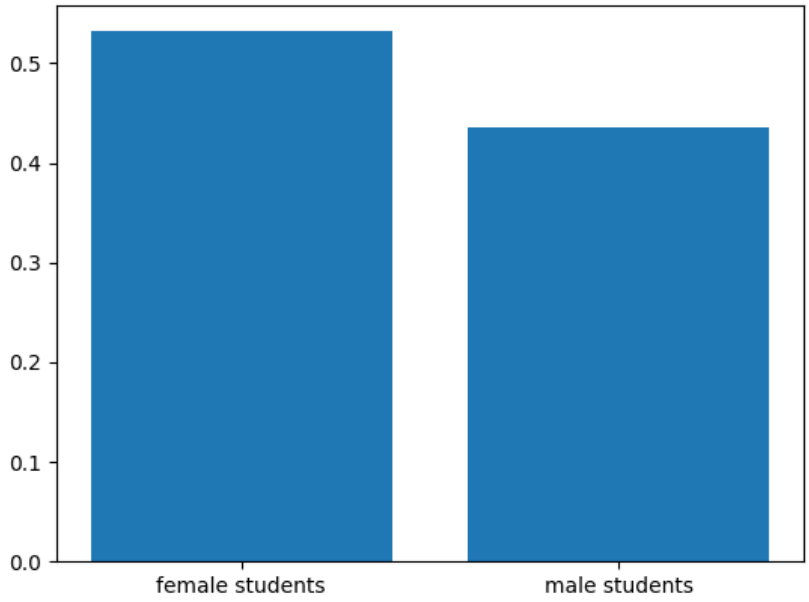


Figure 7.3: Visualisation of the output data of pre-processing scenario two.

are filtered so that only the IDs of students of one gender are left. A second step filters the thermal perception so that only the thermal perceptions for data points with IDs in that set are left. For each gender the thermal sensations are gathered and the average is calculated. A simple reduce transformation can be used, to calculate the mean over the thermal sensations in the corresponding set. Figure 7.4 shows a possible representation of these steps as ontology.

Scenario three: As could be seen in the results of scenario two, the temperature perception can be different based on gender. The user now wants to see, which influence it has on the analysis from scenario one. Instead of having one pre-processing result, where for each survey the average temperature difference and overall average thermal sensation are available, now two results are wanted. One where the average thermal sensation of female participants is available together with the survey time and average temperature difference and another one for male participants. The data resulting from the desired pre-processing is visualised in Figure 7.5.

In order to achieve the desired result, the user needs to join the survey data with the student information in order to filter the thermal sensations by gender like in scenario two. The concept introduced in Section 6.2 can help the user in finding the specification for these transformations in the existing RDF graph. For example, when the user requests transformations which can be applied on *gender* as input data, the recommender from

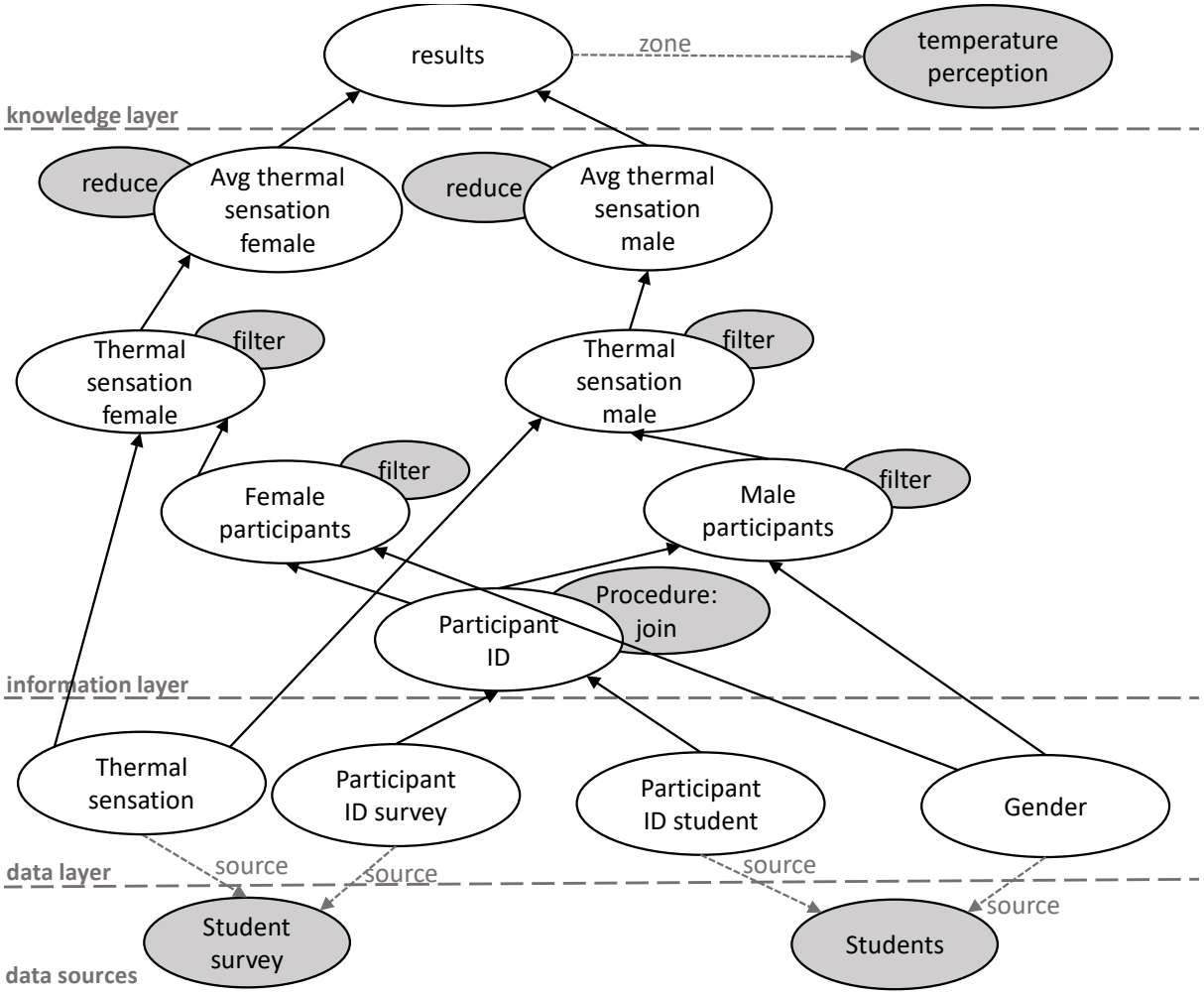
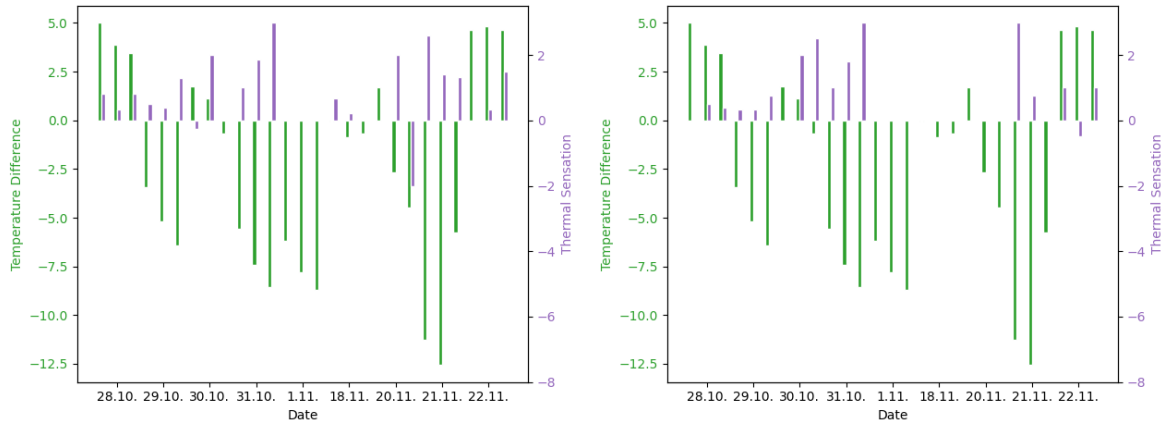


Figure 7.4: Visualisation of a possible ontology for pre-processing scenario two.

the concept would provide two transformations. It would provide the transformation for filtering out all female participants and the transformation for filtering out all male participants. By using the new graph containing these transformations as input for the recommender, the user would receive the transformations for filtering the thermal sensations by gender as the next recommendations. This way the user can gradually find the RDF/XML specification of all existing transformations, which can be reused. The mapping of timestamps and calculation of temperature difference and averaging the temperature difference per survey can be done like in scenario one. Their specification can be reused as well. However for the thermal sensation the averaging per survey needs to be done for the sets of filtered thermal sensation separately. Here the user will have to adapt the existing specification. The averages for each gender can then again be combined with the averages for the temperature difference, like in scenario one. This specification needs to be adapted, because the user now needs to perform



(a) Data for female participants.

(b) Data for male participants.

Figure 7.5: Visualisation of the output data of pre-processing scenario three.

this step twice for the different genders. A possible representation of the transformation steps as ontology can be seen in Figure 7.6.

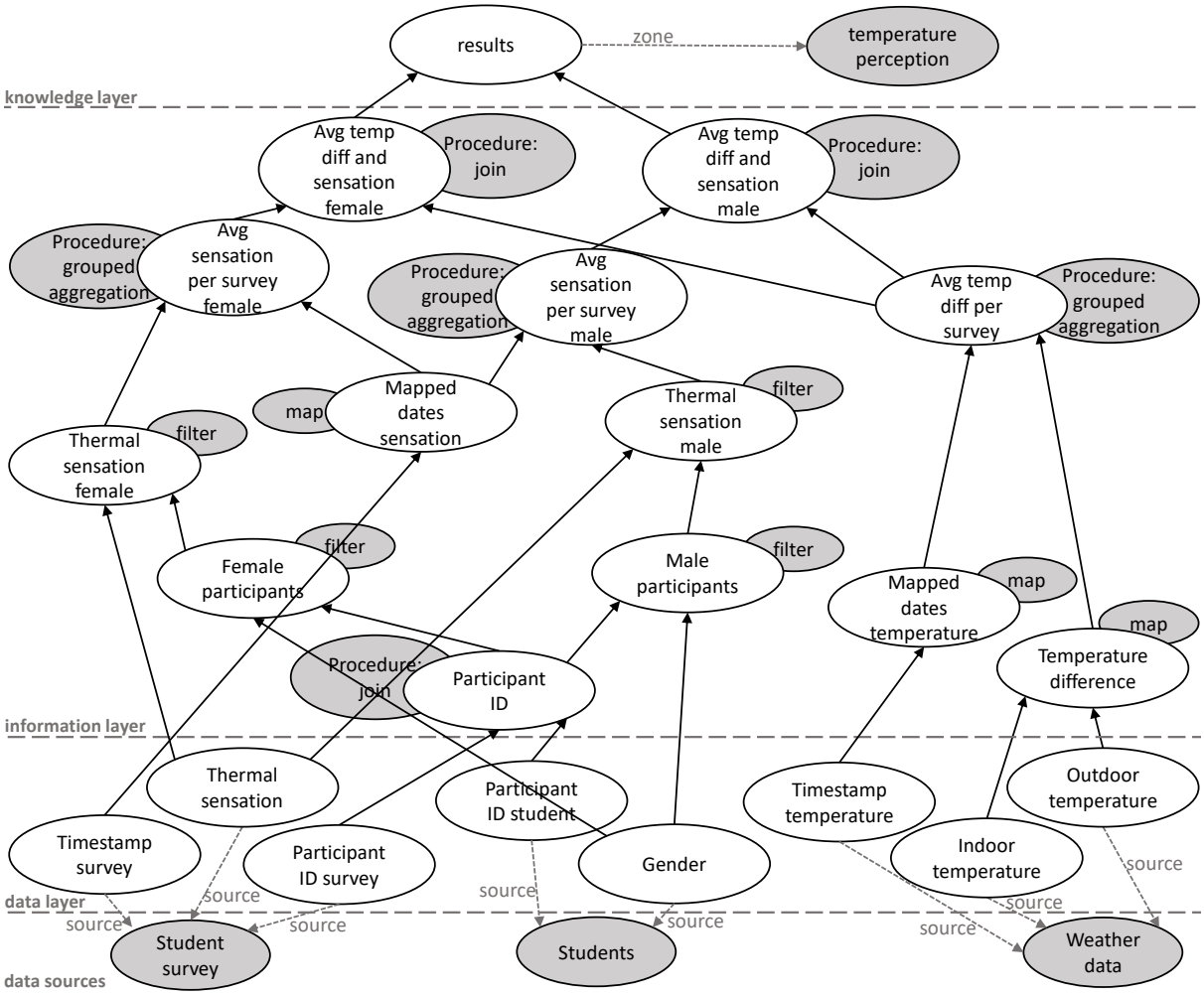


Figure 7.6: Visualisation of a possible ontology for pre-processing scenario three.

8 Prototype

As proof of concept, a prototypical implementation of the concept has been created. The prototype is introduced in this chapter. In Section 8.1 the architecture of the prototype is introduced. The components of the prototype are described and details about the implementation of the recommender are provided. Afterwards, in Section 8.2, it is shown, how a user can adapt the specification of pre-processing to new analysis needs, based on the use case introduced in Chapter 7. The prototype gives users access to transformation recommendations, based on different recommendation approaches, or lets them submit a finished pre-processing specification to BARENTS. A user interface lets users choose, which of these functionalities should be accessed. One recommendation approach provides specific transformation recommendations to the user, which can be applied directly to the input data specified by the user. The other recommendation approach receives a partial specification of pre-processing from the user. It provides additional transformation options for different points in the specified transformation sequence.

8.1 Architecture

The concept uses three components: the recommender, BARENTS and the stored specification of the RDF graph. These components also exist in the prototypical implementation, as can be seen in the visualisation of the architecture in Figure 8.1. The code was written in Python, because Python is a powerful programming language for working with graphs [TS17]. For parsing, serialising and traversing the RDF graph, RDFLib[1] is used.

BARENTS is only simulated in the prototype, as the concept does not change, how BARENTS works internally. The concept only provides the finished ontology as input for the configuration of the BARENTS preparation zone. In the simulated preparation zone in the prototype, scripts for a few predefined data pre-processing pipelines exist. If the simulated preparation zone receives the ontology, which describes one of these pre-processing use-cases, the prepared script is executed. The script collects the relevant

[1] RDFLib website: https://rdflib.dev/

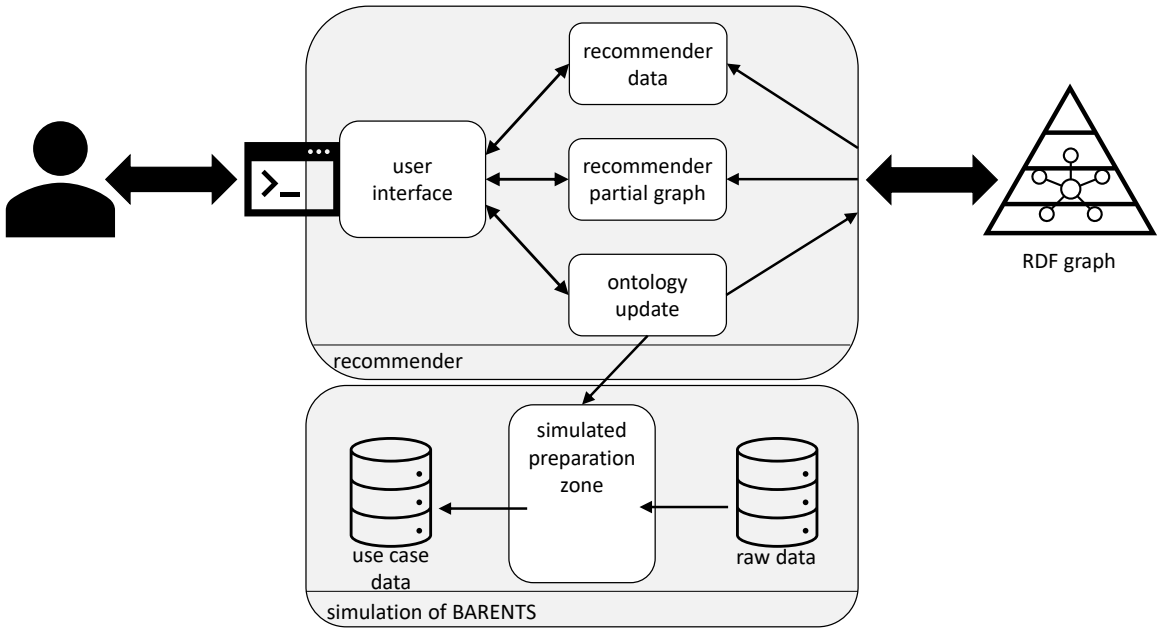


Figure 8.1: Architecture of the prototypical implementation.

input data from the stored raw data, transforms it and saves the pre-processed data in a different location: the simulated use case zone. Join operations and aggregations on grouped data are used in the scripts and the ontology. This simulation of BARENTS works under the assumption, that user-defined functions were provided, which perform these operations. They are marked with the type “procedure: join” or “procedure: grouped aggregation” in the ontology. The raw data is the data from the use case described in Chapter 7 and therefore consists of several hierarchical directories with CSV data. With a corresponding adapter for data access, BARENTS can use input and produce output of any format. The used data formats of the stored data therefore have no impact on the recommendation component and the prototype overall. For the sake of simplicity, the results are also stored in CSV format. For each use case zone, a different directory is used.

The RDF/XML specification of the **RDF graph** is stored separately. When the recommender needs information for predictions from the existing ontology, it parses the RDF graph from the stored RDF/XML representation. The recommender can also update the RDF graph by merging it with the newly constructed graph.

The **recommender** is the main part of the prototype. It consists of four components: a user interface, two transformation recommenders and one component for updating the ontology. Of the transformation recommenders, one implements Recommendation Approach 2 from Section 6.1 and one implements Recommendation Approach 3. Approach 2 recommends transformations for input data and Approach 3 recommends additional

transformations for partial RDF trees. In both cases the recommendations are based on existing transformation rules in the ontology. The users can either reuse the existing rules as they are, if they provide the required pre-processing or use them as a basis and adapt them for their needs.

The **user interface** lets the user interact with the recommendation component. Users can choose one of the recommendation approaches or to submit a finished specification. While a more fleshed out implementation of the concept would use a graphical user interface, in this prototypical implementation the user interface is realised via command line interface. Depending on the command the user puts in, the next input of the user is forwarded to either one of the recommenders or to the ontology update component. If the command was "data", the user interface expects the next input to be the name of one or several (separated by ",") data sources or data attributes in the data layer of the RDF graph. The one line of input is read from the console and used as input for the recommender based on Approach 2 (*recommender data*). If the command was "rdf", the user interface expects several lines of input, with the RDF/XML description of a partial ontology. Several lines (until an empty line appears) are read from the console and used as input for the recommender based on Approach 3 (*recommender partial graph*). If the command was "update", the user interface expects several lines of input, with the RDF/XML description of a finished ontology. Several lines (until an empty line appears) are read from the console and used as input for the ontology update component.

From the **recommender data**, users can receive transformation recommendations for a single source of data or for the joint transformation of data from two sources. Recommendations for a single source of data can be either for all attributes of the source, or for a specific attribute only. For example, based on the use case from Chapter 7, a user might want to perform a detailed analysis of weather patterns. She or he can request transformation recommendations for the data source *Weather Data*. Any transformation for deriving additional attributes from the data, like *temperature_difference* between outdoors and indoors, will be provided and can now be included in the analysis. A user might also want to find transformations for specific attributes only. For example, the user might want to see, if there are differences between the experiences of different genders, but is unsure on how to separate the data by gender. The user can request transformations for the attribute *gender* and will receive the transformations, which filter participant IDs by gender. For the same scenario, the user, might also have known, that *gender* is found in student data, while the data on the experiences of participants is found in survey data. By requesting transformations for the sources *Students* and *Student Survey*, the join transformation, which joins both sources based on participant IDs is provided.

The **recommender data** receives input from the user interface and parses the RDF graph from the stored RDF/XML representation. If the user submitted a single input,

Listing 8.1 An excerpt from the code of the *recommender data*.

```
...

def print_transformations(graph, output_graph, attribute, ex):
   for transformation in graph.objects(attribute, ex.partOf):
      layer = graph.value(transformation, ex.layer, None)
      if layer == Literal('Information Layer'):
         output_graph.add((attribute, ex.partOf, transformation))
         output_graph.add((transformation, ex.type, (graph.value(transformation, ex.type,
             None))))
         output_graph.add((transformation, ex.function, (graph.value(transformation,
             ex.function, None))))
         output_graph.add((transformation, ex.layer, layer))
         print(str(output_graph.serialize(format='xml').encode("utf-8")).replace('\\n',
             '\n'))

         output_graph.remove((attribute, ex.partOf, transformation))
         output_graph.remove((transformation, ex.type, (graph.value(transformation,
             ex.type, None))))
         output_graph.remove((transformation, ex.function, (graph.value(transformation,
             ex.function, None))))
         output_graph.remove((transformation, ex.layer, layer))
         print()

...
```

the RDF graph is searched for occurrences of the input as a source or as an object in the data layer. For a source, the recommender identifies each attribute of this source, which occurs in the data layer and recommends transformations for each of them. Transformations in which the object in the data layer was used as input are found. For each transformation, the full description of the transformation is returned to the user together with the description of the attribute in RDF/XML format. Listing 8.1 shows, how the transformations for an attribute are identified, added to the output and printed on the console. If the user submitted two inputs, the RDF graph is searched for occurrences of each input as a source. Transformations in which attributes from both sources are used as input are identified. Full descriptions of the transformations are returned to the user together with the description of the attributes in RDF/XML format.

Users can iteratively add additional transformations to their pre-processing specification with the help of the **recommender partial graph**. Using the join example from before, a user might now have a partial RDF graph, which joins student information and survey results. The user now wants to filter the relevant data by gender. Using the RDF/XML description of the partial pre-processing as input, the recommender can in a first step provide the existing transformations, which filter the students by gender. Requesting

additional transformations again, with the student filters added into the graph, the recommender can provide the transformations for filtering out the thermal sensations of those students.

The **recommender partial graph** receives the RDF/XML specification of a RDF graph as input from the user interface and parses the stored RDF/XML specification of the RDF graph for the ontology. First the stored RDF graph is searched for occurrences of the input graph as a part of the overall graph. If it can be found, the recommender searches the stored RDF graph for *partOf* relationships. Relationships starting at one of the transformations also contained in the input and ending at another transformation not contained in the input are selected. In case such a transformation is found, it is added to the input and the resulting graph containing the additional transformation is returned to the user in RDF/XML syntax. A second approach for finding additional transformations exists, in case the first approach does not find an additional transformation. This approach uses networkx[2], a Python package for representing and studying graphs and networks. The *partOf* relations of the input RDF graph are used to create a networkx graph. For small input RDF graphs, small edits like adding or removing a triple from the ontology RDF graph are performed on a copy of the networkx graph. The graph edit distance is calculated. If it lies between one and two, an additional transformation is found. An excerpt from the implementation of this approach can be seen in Listing 8.2. For bigger inputs, the calculation of graph edit distance is too slow. The recommender tries to find a triple from the ontology RDF with the *partOf* predicate, where only either the subject or the object is already a node in the graph. A RDF graph which contains the additional transformation is retuned to the user in RDF/XML format.

The **ontology update** component acts as an interface between the recommender component and the BARENTS component. A user can change the pre-processing configuration of BARENTS, after checking the validity of a new BARENTS ontology via this component. It is not a required component of the concept, as users can also provide the ontology for configuring data pre-processing to BARENTS without this additional interface. The component receives the RDF/XML representation of a RDF graph as input from the user interface. If the input has the proper format, the component checks, whether the RDF graph is a proper BARENTS ontology. It needs to contain attributes in the data layer, which have a source. All attributes need to be used in either a transformation in the information layer or a result in the knowledge layer. The transformations in the information layer need a type and a function and also need to be part of either another transformation or a result. Results in the knowledge layer need to have a zone, the result should be stored in. If the RDF graph fulfills all those requirements, it is forwarded to the simulated preparation zone in the simulated BARENTS component. There it acts as

[2]networkx website: https://networkx.org/

Listing 8.2 An excerpt from the code of the *recommender partial graph*.

```
...

def recommend_new_transformation_slow(input_ex, input_graph):
   ex, graph = load_knowledge_graph()
   G, G2 = prepare_graphs(input_ex, input_graph)

   for (s, p, o) in graph:
      print('.')
      if p == ex.partOf:
         s_in_g = True
         o_in_g = True
         if s not in G:
            G.add_node(s)
            s_in_g = False
         if o not in G:
            G.add_node(o)
            o_in_g = False
         if not G.has_edge(s,o):
            G.add_edge(s, o)
            distance = nx.graph_edit_distance(G, G2)
            layer = graph.value(s, ex.layer, None)
            if(distance > 2):
               G.remove_edge(s, o)
               if not s_in_g:
                  G.remove_node(s)
               if not o_in_g:
                  G.remove_node(o)
            elif distance > 0 and distance <= 2 and layer == Literal('Information Layer'):
               print('Result found!')
               print_result(input_graph, graph, input_ex, ex)
               return

...
```

configuration for the data pre-processing. Then the ontology update component parses the stored RDF/XML representation of the RDF graph, both graphs are merged and the new graph is stored.

8.2 Demonstration

In the use case introduced in Chapter 7 data pre-processing for two different analyses was already specified as part of the knowledge base. In one analysis, the temperature difference between outside and inside was compared to the thermal sensations of the

participants for each conducted survey. The other analysis compared the average thermal perception per gender.

We now assume the user was involved in the specification of data preparation for the first analysis. She or he now sees the results of the second analysis and wants to know, if the influence of the temperature difference between inside and outside on thermal perception is different per gender.

The user was not involved in the specification for the second analysis and gender information did not occur in data preparation for the first analysis. As a first step the user therefore requests recommended transformations for the data attribute *gender*. An excerpt from this interaction can be seen in Listing 8.3. The recommender finds two transformations in which gender was used as input. One for filtering out all male participants and one for filtering out all female participants.

The user sees, that a *participant_id* was used in the filtering transformation in addition to the gender. A data request with *participant_id* returns that it is neither a source nor an attribute, it must therefore be the result of a transformation. Next the user queries for the source *Student Survey* since that was one of the sources in the first analysis scenario. For the attribute *participant_id_survey* from this source, the join transformation *participant_id* is returned. It uses a *participant_id_student* in the transformation, which must be an attribute from the source *Students*, which also contains *gender*. Now that the user has all the transformations for filtering the students by gender, she or he wants to know how to continue.

As the next step, the user wants additional transformations for the partial RDF graph. She or he enters the graph, which contains the transformation which results in filtering out only female students. The recommender provides an RDF graph containing the additional transformation for filtering out the thermal perception of female students only. An excerpt containing this interaction can be found in Listing 8.4. The user submits the same request with the transformation for filtering out male students and receives the filter transformation for the thermal sensation of male students.

Now the user has all additional transformations she or he needs. The user adds those transformations to the RDF/XML specification of the first analysis. She or he adapts the graph by making two versions of the aggregation transformation for calculating the average thermal sensation per survey. One uses the filtered sensations of female students as input instead of all sensations, the other uses the sensations of male students. Two versions of the join with the aggregated temperature differences are also created, which use aggregated thermal sensations of one gender each.

The RDF graph for the data pre-processing is now finished and the user submits it. Both the stored RDF graph, as well as the configuration of the simulated BARENTS preparation

Listing 8.3 Excerpt from an interaction between the user and the recommender for finding transformations for input data. “»” indicates user input.

```
...
Please enter your query:

>> data

Please enter your input data. If you wish to transform data from more than one source,
    please separate by ",".

>> gender

Found as data attribute in the data layer in the ontology.
Usage in transformations:
b'<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:ex="http://example.org/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
  <rdf:Description rdf:about="http://example.org/gender">
    <ex:source>Students</ex:source>
    <ex:layer>Data Layer</ex:layer>
    <ex:partOf rdf:resource="http://example.org/female_participants"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/female_participants">
    <ex:type>filter</ex:type>
    <ex:function>lambda x: x.gender = \'Female\' and x.participant_id </ex:function>
    <ex:layer>Information Layer</ex:layer>
  </rdf:Description>
</rdf:RDF>
'

b'<?xml version="1.0" encoding="UTF-8"?>

...

  <rdf:Description rdf:about="http://example.org/male_participants">
    <ex:type>filter</ex:type>
    <ex:function>lambda x: x.gender = \'Male\' and x.participant_id </ex:function>
    <ex:layer>Information Layer</ex:layer>
  </rdf:Description>
</rdf:RDF>
'
...
```

Listing 8.4 Interaction between the user and the recommender for finding additional transformations. “»” indicates user input.

```
...
Please enter your query:

>> rdf

Please enter your partial rdf graph in rdf/xml format. Enter an empty line when the input
    is finished.

>> <?xml version="1.0" encoding="UTF-8"?>
>> <rdf:RDF
>>  xmlns:ex="http://example.org/"
>>  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>> >
...
>>  <rdf:Description rdf:about="http://example.org/female_participants">
>>   <ex:type>filter</ex:type>
>>   <ex:function>lambda x: x.gender = \'Female\' and x.participant_id </ex:function>
>>   <ex:layer>Information Layer</ex:layer>
>>  </rdf:Description>
>> </rdf:RDF>
>>

b'<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:ex="http://example.org/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
 <rdf:Description rdf:about="http://example.org/gender">
  <ex:layer>Data Layer</ex:layer>
  <ex:partOf rdf:resource="http://example.org/female_participants"/>
  <ex:source>Students</ex:source>
 </rdf:Description>
 <rdf:Description rdf:about="http://example.org/female_participants">
  <ex:function>lambda x: x.gender = \'Female\' and x.participant_id </ex:function>
  <ex:type>filter</ex:type>
  <ex:layer>Information Layer</ex:layer>
 </rdf:Description>
 <rdf:Description rdf:about="http://example.org/thermal_sensation_female">
  <ex:function>lambda x: x.female_participants and x.thermal_sensation </ex:function>
  <ex:type>filter</ex:type>
  <ex:layer>Information Layer</ex:layer>
  <ex:partOf>lambda x: x.female_participants and x.thermal_sensation </ex:partOf>
 </rdf:Description>
</rdf:RDF>
'
...
```

zone are updated. The simulated preparation zone applies the transformations to the raw data and stores the new results in the use case zone.

Conclusions regarding the Prototype

In conclusion, it can be seen, that the prototype implements all relevant functionalities of the concept. The prototype implements Recommendation Approach 2 and can therefore help the user in finding initial transformations for a data source. Recommendation Approach 3 is implemented in the prototype as well and can help the user find additional transformations for a partial RDF graph. The user can iteratively build the pre-processing specification for an analysis use-case, with the help of both recommendation approaches. Using the finished specification as the configuration for the data preparation zone, BARENTS can then apply the specified transformations and provide the pre-processed data to the user. Through the demonstration of how the prototype can help a user in specifying the pre-processing needs for a specific use case as a BARENTS ontology, the applicability of the prototype and the concept is shown. By the prototypical implementation and the demonstration, a proof of concept is presented. The concept however still needs to be assessed against the stated requirements for a concept for assisting users in data preparation within data lakes.

9 Assessment

In the following chapter, the new concept, conceptualised in Chapter 6, is evaluated against the requirements introduced in Chapter 3. It is discussed, in how far each requirement for concepts towards enabling specification of data pre-processing tasks is fulfilled. The results of the evaluation are visualised in Table 9.1.

R1 - Flexibility

The concept should provide the required flexibility in order to be applicable in any given domain and in order to be adaptable to changes. It should work on any data from any domain and the concept should also be applicable for any use case. In case the data or the use cases change, the concept should allow for easy adjustment of the specification of the data preparation.

The new concept is based on BARENTS, which supports the work with all types of data and supports any transformation, even if some transformations are only supported through user-defined functions. Data preparation needs are specified in ontologies with the help of a recommender. Ontologies can be used to model relationships in any domain. Specification of pre-processing for any use case with data from any domain is therefore possible. Adapting to changes in the underlying data or the use cases is also easy. The ontology only has to be adapted accordingly by changing some of the RDF triples. Overall, the concept therefore provides flexibility regarding applicability and adaption to change. Requirement *R1 - Flexibility* is fulfilled.

R2 - User-friendliness

The concept should be user-friendly enough, to enable users with little IT knowledge, to specify their processing needs.

BARENTS performs the specified transformations for the users and they do not have to provide a technical realisation themselves. Users only need to provide a specification of the pre-processing needs in the form of an ontology comprised of an RDF graph. The

concept aids users in creating these ontologies by providing an interactive recommender, which recommends initial or additional transformations to the user. Those recommendations are provided in the form of partial RDF graphs. Users therefore do not need IT knowledge, they only need to be able to specify the processing needs as an RDF graph with help from the recommender. Since the recommender provides examples of how transformations look when expressed in the RDF/XML notation, users do not even require in-depth knowledge of RDF. Therefore the concept provides a user-friendly way of specifying data preparation needs, which can be used by users with little IT knowledge. Requirement *R2 - User-friendliness* is fulfilled.

R3 - Compatibility with existing Solutions

The concept should be compatible with existing data lake concepts. Interoperability with existing solutions, in the context of user-friendly data preparation, is important and incorporating the concept in existing data lakes should be possible.

Since the new concept uses BARENTS as a basis, it is compatible with BARENTS, which is currently the best solution in technical realisation of data preparation. For the same reason, the concept can also be embedded in any data lake, where BARENTS can be incorporated. BARENTS itself is based on the widely used zone architecture, therefore the concept is also compatible with zone architectures, a solution for data partitioning, as discussed in Chapter 5. As argued before, the concept is automatically compatible with the rest of the solutions. Metadata, the other solution for finding data besides data partitioning, is only affected by approaches which directly concern metadata management, which the new concept does not. Users can use solutions for finding data based on metadata and use the new concept for specifying their processing needs at the same time. Relatedness measures of data are based on metadata or data itself and stored in separate structures, if at all. The new concept does not affect those solutions. Finding related data for joint transformation and then using the new concept to choose and specify the transformations is possible. So far assistance in choosing transformations was only provided through general rules and collection of metadata on previously performed transformations. While the rules were not chosen as a basis for the recommender, users can still apply them on top of the recommendations. The solution for metadata collection could also be embedded in a data lake, where the new concept is used, they do not interfere with each other. Metadata about transformations could still be used to gain additional insights and might also provide the basis of a future extension of the new concept. In summary, the concept is compatible with existing solutions and can be incorporated in any data lake, where BARENTS already is in use or could be incorporated. Requirement *R3 - Compatibility with existing Solutions* is fulfilled.

requirement	contributions towards fulfilling the requirement	fulfilled
R1 - Flexibility	- all data types and transformations supported - specification of pre-processing for any use case - ontologies can model relationships in any domain - ontologies can be adapted to data or use case changes	yes
R2 - User-friendliness	- specification of pre-processing in the form of ontologies - support in choosing transformations - examples provided to aid in writing ontologies - BARENTS performs the specified transformations	yes
R3 - Compatibility with existing Solutions	- compatible with BARENTS - compatible with zone architectures - automatically compatible with metadata and data relatedness solutions	yes
R4 - Extensibility	- extensible through interface for selecting data - extensible through visual representation of RDF graph - extensible through graphic interface - extensible through using additional recommendation approaches	yes

Table 9.1: Evaluation of the concept against the requirements from Chapter 3.

R4 - Extensibility

The concept should be extensible, to allow for future improvement. When new ways of user assistance are discovered, it should be possible to extend the concept with those.

The concept can easily be extended, by putting an interface, which provides additional features in front of the concept and only interacting with the recommender in the background. For example, currently the user has to start already knowing what data should be used as input. By providing an interface, which lets the user select data first and then move on to the recommender with that data, the concept could be extended. One of the data querying solutions introduced in Section 4.2 or one of the related data recommendation solutions from Section 4.3 can easily be used for selecting data. A front end could also visualise the RDF graph and extend the concept by providing a graphical interface, while still using the recommender in the background. The recommender itself can also easily be extended by combining it with one of the ideas form Section 6.1, which were not chosen for the final concept. For example, Recommendation Approach 4 finds which transformations a similar user performed on the data and recommends them. An extended recommender could have two different recommenders run in the background and have a user interface, which passes user input through and presents one recommendation per background recommender to the user. Additionally, the concept uses ontologies for specifying pre-processing. There are already many existing ontologies with domain knowledge [DS21], which describe relations of concepts within that domain. Those could be used to enrich, the BARENTS ontologies with semantic descriptions. Overall the concept is easily extensible in different ways. Requirement *R4 - Extensibility* is fulfilled.

Conclusion

As can be seen the chosen concept fulfills all requirements specified in Chapter 3. It provides flexible, user-friendly means for users with little IT knowledge to specify data pre-processing for any use case. The concept is compatible with existing solutions, which aid users in data preparation and is extensible through new means of assistance.

However the concept still has its weaknesses. As recommendations for transformations are solely based on a knowledge base of existing pre-processing specifications, the recommender does not work, if the knowledge base is too small or does not yet exist. If new data sources are added to the data lake, transformation rules for this source do not yet exist and therefore no recommendations can be provided. For these cases adoption of Recommendation Approach 1 from Section 6.1 would slightly alleviate the problem, as at least the generally applicable transformation categories can be provided based on the metadata for a source. AI-based approaches could also be used to analyse existing transformation rules and metadata of the sources, to find patterns and predict likely usable transformations. The usage of recommenders might result in sub-optimal transformation chains, as the user might simply apply one recommended transformation after the other, without checking if there would be a more efficient solution. If users rely on the recommender too much, the additional issue arises that the knowledge base stops growing because no users provide new transformation rules. This problem might be mitigated by providing a graphical tools for specifying the ontologies, which alleviates the complexity of creating a RDF graph. Therefore, while the concept has its weaknesses, solutions can be found to mitigate the issues.

10 Conclusion

Data lakes are a concept for storing, governing and provisioning big data efficiently [GGH+20b; Mat17]. They provide a central storage, where data from any source is stored in its original format. Any use case can be served using this data, as no information is lost through transformation into a unified schema. However this schema-on-read approach leaves transforming data into a usable schema in preparation for analysis to the user. Pre-processed data only reaches its full potential when utilised in the use case it was envisioned for [SBE+21]. Therefore users have to adapt pre-processing for each new use case. Users have to specify, how the data is prepared and which techniques are used. To achieve this they need knowledge about the stored data as well as domain knowledge and IT knowledge. The people who need the analysis results however are often domain experts and not IT experts. Realising data preparation for new use cases is neither easy nor intuitive, when working with data lakes. At the same time novel use cases and analysis tasks emerge frequently [SBE+21].

In the scope of this work, possibilities for assisting users in data preparation for novel use cases in data lakes, were explored. For this purpose, reasons for the difficulty of data pre-processing in data lakes were explored and requirements for a concept for user assistance were derived. A concept for data preparation has to be flexible enough to be used in any domain and on any use case and user friendly enough to be used by data scientists and domain experts with little IT knowledge. It also has to be compatible with existing solutions for assisting users in data preparation and extensible to allow for future improvement. Four steps were extracted, which a user takes in developing data preparation for a new use case in data lakes. They are finding relevant data, finding related data, choosing transformations and technical realisation. Literature was explored for existing concepts for assisting users in those steps of preparing data for their use cases. It was found, that sufficient assistance in finding relevant data and exploring related data is provided by existing solutions. The support for technical realisation is almost sufficient, but assistance in choosing the right transformations is still lacking.

Based the lessons learned from this analysis of existing concepts a new concept was developed. It is based on BARENTS, which enables specification of data preparation in the form of ontologies and automatically performs specified transformations. A recommender was added, which helps users in choosing transformations and assists in creating

the ontology for the specification of data preparation. A prototypical implementation of the new concept was created and it was shown, how it can assist users in specifying their data preparation needs. The concept was assessed against the requirements and shown to fulfill all of them. Therefore it enables user-friendly specification of data pre-processing needs within data lakes.

Future Works

While the concept allows flexible data preparation for any use case, it still has its weaknesses. The recommender needs a knowledge base of existing transformation rules to be able to provide transformation recommendations. It could be extended through additional recommendation approaches. A rule based approach or AI based approach could provide recommendations if not enough transformation rules are available in the knowledge base. In case too many possible transformations are found, having a rating system for how good a recommended transformation is would also be an advantage. This way only those transformations, which were ranked highest, are presented to the user. With a recommender based on usage patterns of similar users, the found transformations could be ranked by how similar the users, who provided them, are to the current user. Then only the most promising options can be presented to the user. Another aspect, which could be addressed by future works is assessing the quality of transformation recommendations. While the provided concept can recommend transformations to the user, the quality of these recommendations is unclear. Future works could evaluate the different recommendation approaches by consulting domain experts towards rating the quality of provided recommendations.

Another direction for future works could be providing a graphical interface, where users can both build the RDF graph for their use case in a graphical environment and request recommendations for additional transformations. By providing the graphical environment, users could select a specific part of the RDF graph, where they want to attach a new transformation.

Future works could also improve upon the transformation types of BARENTS. Currently only map, reduce and filter are available without user-defined functions. However joining and merging of data from different sources and splitting a data set into smaller groupings are commonly needed in data preparation as well. By having natively supported transformation types, which provide these capabilities, the need for user-defined functions and therefore also for IT assistance could be reduced.

A different direction for future research could also be finding best practises for specifying the ontologies. This could help users avoid common pitfalls and prevent the specification of sub-optimal data preparation solutions.

Bibliography

[AARC19] A. Alserafi, A. Abelló, O. Romero, T. Calders. “Keeping the Data Lake in Form: DS-kNN Datasets Categorization Using Proximity Mining.” In: *Model and Data Engineering*. Springer International Publishing, 2019, pp. 35–49 (cit. on pp. 23, 31, 34, 35, 40).

[AARC20] A. Alserafi, A. Abelló, O. Romero, T. Calders. “Keeping the Data Lake in Form: Proximity Mining for Pre-Filtering Schema Matching.” In: *ACM Trans. Inf. Syst.* 38 (2020). DOI: 10.1145/3388870. URL: https://doi.org/10.1145/3388870 (cit. on pp. 34, 35, 40).

[ADBG20] S. B. Atitallah, M. Driss, W. Boulila, H. B. Ghézala. “Leveraging Deep Learning and IoT big data analytics to support the smart cities development: Review and future directions.” In: *Computer Science Review* 38 (2020), p. 100303 (cit. on p. 17).

[BFPK20] A. Bogatu, A. A. Fernandes, N. W. Paton, N. Konstantinou. “Dataset discovery in data lakes.” In: *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE. 2020, pp. 709–720 (cit. on pp. 34, 35, 40).

[BHFM20] M. Behringer, P. Hirmer, M. Fritz, B. Mitschang. “Empowering Domain Experts to Preprocess Massive Distributed Datasets.” In: *Business Information Systems*. Springer International Publishing, 2020, pp. 61–75 (cit. on pp. 36, 37, 40).

[BHM18] M. Behringer, P. Hirmer, B. Mitschang. “A Human-Centered Approach for Interactive Data Processing and Analytics.” In: *Enterprise Information Systems*. Springer International Publishing, 2018, pp. 498–514 (cit. on p. 37).

[BHRZ11] W. Bartussek, B. Humm, A. Reibold, T. Zeh. “Zur Definition von ‚Ontologie’in den Informationswissenschaften.” In: *Hochschule Darmstadt (h_da)* (May 2011) (cit. on p. 38).

[DLP+21] C. Diamantini, P. Lo Giudice, D. Potena, E. Storti, D. Ursino. “An approach to extracting topic-guided views from the sources of a data lake.” In: *Information Systems Frontiers* 23 (2021), pp. 243–262 (cit. on pp. 33–35, 40).

[DS21] H. Dibowski, S. Schmid. “Using Knowledge Graphs to Manage a Data Lake.” In: *INFORMATIK 2020*. Gesellschaft für Informatik, Bonn, 2021, pp. 41–50. DOI: 10.18420/inf2020_02 (cit. on pp. 24, 33, 34, 36, 40, 81).

[EGG+20] R. Eichler, C. Giebler, C. Gröger, H. Schwarz, B. Mitschang. “HANDLE-A Generic Metadata Model for Data Lakes.” In: *International Conference on Big Data Analytics and Knowledge Discovery*. 2020, pp. 73–88. DOI: 10.1007/978-3-030-59065-97 (cit. on pp. 23, 24, 34, 40).

[FAK+18] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, M. Stonebraker. “Aurum: A data discovery system.” In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. 2018, pp. 1001–1012 (cit. on pp. 34, 35, 40).

[FGG+21] M. Francia, E. Gallinucci, M. Golfarelli, A. G. Leoni, S. Rizzi, N. Santolini. “Making data platforms smarter with MOSES.” In: *Future Generation Computer Systems* 125 (2021), pp. 299–313. DOI: https://doi.org/10.1016/j.future.2021.06.031. URL: https://www.sciencedirect.com/science/article/pii/S0167739X21002260 (cit. on pp. 33, 34, 40).

[FSF20] R. C. Fernandez, P. Subramaniam, M. J. Franklin. “Data Market Platforms: Trading Data Assets to Solve Data Problems [Vision Paper].” In: *CoRR* abs/2002.01047 (2020) (cit. on pp. 34, 40).

[GC20] P. Gulia, A. Chahal. “Big Data Analytics for IoT.” In: *International Journal of Advanced Research in Engineering and Technology (IJARET)* 11 (2020) (cit. on p. 17).

[GGH+19] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, B. Mitschang. “Leveraging the Data Lake: Current State and Challenges.” In: *Big Data Analytics and Knowledge Discovery*. Springer International Publishing, 2019, pp. 179–188 (cit. on p. 25).

[GGH+20a] C. Giebler, C. Gröger, E. Hoos, R. Eichler, H. Schwarz, B. Mitschang. “Data Lakes auf den Grund gegangen.” In: *Datenbank-Spektrum* 20 (2020), pp. 57–69 (cit. on pp. 17, 25).

[GGH+20b] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, B. Mitschang. “A Zone Reference Model for Enterprise-Grade Data Lake Management.” In: *Proceedings of the 24th IEEE Enterprise Computing Conference (EDOC 2020)*. 2020. DOI: https://doi.org/10.1109/EDOC49727.2020.00017 (cit. on pp. 17, 18, 23–25, 32, 40, 41, 83).

[GGH+21] C. Giebler, C. Gröger, E. Hoos, R. Eichler, H. Schwarz, B. Mitschang. "The Data Lake Architecture Framework: A Foundation for Building a Comprehensive Data Lake Architecture." In: *Proceedings der 19. Fachtagung für Datenbanksysteme für Business, Technologie und Web (BTW 2021)*. 2021, pp. 351–370. DOI: 10.18420/btw2021-19 (cit. on p. 24).

[GH19] C. Gröger, E. Hoos. "Ganzheitliches Metadatenmanagement im Data Lake: Anforderungen, IT-Werkzeuge und Herausforderungen in der Praxis." In: *BTW 2019*. Gesellschaft für Informatik, Bonn, 2019, pp. 435–452. DOI: 10.18420/btw2019-26 (cit. on pp. 33, 34, 40).

[GMB+21] N. Gao, M. Marschall, J. Burry, S. Watkins, F. D. Salim. "Understanding occupants' behaviour, engagement, emotion, and comfort indoors with heterogeneous sensors and wearables." In: *CoRR* abs/2105.06637 (2021) (cit. on p. 61).

[GS14] C. Gröger, C. Stach. "The Mobile Manufacturing Dashboard." In: *Proceedings of the 2014 IEEE International Conference on Pervasive Computing and Communications Workshops*. Ed. by G. Záruba, K. Farkas, M. Mühlhäuser, J. Payton. PerCom '14. Budapest: IEEE, Mar. 2014, pp. 138–140. ISBN: 978-1-4799-2736-4. DOI: 10.1109/PerComW.2014.6815180 (cit. on p. 22).

[HGQ16] R. Hai, S. Geisler, C. Quix. "Constance: An Intelligent Data Lake System." In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD '16. Association for Computing Machinery, 2016, pp. 2097–2100. DOI: 10.1145/2882903.2899389. URL: https://doi.org/10.1145/2882903.2899389 (cit. on pp. 33, 34, 37, 40).

[HKN+16] A. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, S. E. Whang. "Goods: Organizing Google's Datasets." In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD '16. Association for Computing Machinery, 2016, pp. 795–806. DOI: 10.1145/2882903.2903730. URL: https://doi.org/10.1145/2882903.2903730 (cit. on pp. 33–35, 40).

[HQJ21] R. Hai, C. Quix, M. Jarke. "Data lake concept and systems: a survey." In: *CoRR* abs/2106.09592 (2021) (cit. on p. 41).

[Inm05] W. H. Inmon. *Building the data warehouse*. John wiley & sons, 2005 (cit. on p. 21).

[Inm16] B. Inmon. *Data Lake Architecture: Designing the Data Lake and avoiding the garbage dump*. Technics publications, 2016 (cit. on pp. 18, 24, 32, 40).

[JACJ17] Z. Jin, M. R. Anderson, M. Cafarella, H. V. Jagadish. "Foofah: Transforming Data By Example." In: *Proceedings of the 2017 ACM International Conference on Management of Data*. Association for Computing Machinery, 2017, pp. 683–698. DOI: 10.1145/3035918.3064034. URL: https://doi.org/10.1145/3035918.3064034 (cit. on pp. 37, 40).

[KW18] P. P. Khine, Z. S. Wang. "Data lake: a new ideology in big data era." In: *ITM web of conferences*. Vol. 17. 2018, p. 03025 (cit. on pp. 17, 21).

[KWXD17] M. Khan, X. Wu, X. Xu, W. Dou. "Big data challenges and opportunities in the hype of Industry 4.0." In: *2017 IEEE International Conference on Communications (ICC)*. 2017, pp. 1–6. DOI: 10.1109/ICC.2017.7996801 (cit. on p. 17).

[LQ20] Z. Lv, L. Qiao. "Analysis of healthcare big data." In: *Future Generation Computer Systems* 109 (2020), pp. 103–110 (cit. on p. 17).

[LSSB21] S. Langenecker, C. Sturm, C. Schalles, C. Binnig. "Towards learned metadata extraction for data lakes." In: *BTW 2021* (2021) (cit. on pp. 33, 34, 40).

[Mat17] C. Mathis. "Data lakes." In: *Datenbank-Spektrum* 17 (2017), pp. 289–293 (cit. on pp. 17, 23, 83).

[MRZ21] I. Megdiche, F. Ravat, Y. Zhao. "Metadata Management on Data Processing in Data Lakes." In: *SOFSEM 2021: Theory and Practice of Computer Science*. Springer International Publishing, 2021, pp. 553–562 (cit. on pp. 33–36, 38–40, 53).

[PC03] F. Paim, J. de Castro. "DWARF: an approach for requirements definition and management of data warehouse systems." In: *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.* 2003, pp. 75–84. DOI: 10.1109/ICRE.2003.1232739 (cit. on p. 21).

[RZ19] F. Ravat, Y. Zhao. "Data Lakes: Trends and Perspectives." In: *Database and Expert Systems Applications*. Springer International Publishing, 2019, pp. 304–313 (cit. on pp. 23, 24, 41).

[SAB+18] C. Stach, S. Alpers, S. Betz, F. Dürr, A. Fritsch, K. Mindermann, S. M. Palanisamy, G. Schiefer, M. Wagner, B. Mitschang, A. Oberweis, S. Wagner. "The AVARE PATRON - A Holistic Privacy Approach for the Internet of Things." In: *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications*. Ed. by P. Samarati, M. S. Obaidat. Vol. 2. SECRYPT '18. Porto: SciTePress, July 2018, pp. 372–379. ISBN: 978-989-758-319-3. DOI: 10.5220/0006850305380545 (cit. on p. 22).

[SB11] C. Stach, A. Brodt. “vHike – A Dynamic Ride-sharing Service for Smartphones.” In: *Proceedings of the 2011 IEEE 12th International Conference on Mobile Data Management*. Ed. by A. Zaslavsky, P. K. Chrysanthis, D. L. Lee, H. Lei, C. Roncancio. MDM ’11. Luleå: IEEE, June 2011, pp. 333–336. ISBN: 978-1-4577-0581-6. DOI: 10.1109/MDM.2011.33 (cit. on p. 22).

[SBE+21] C. Stach, J. Bräcker, R. Eichler, C. Giebler, B. Mitschang. “Demand-Driven Data Provisioning in Data Lakes: BARENTS - A Tailorable Data Preparation Zone.” In: *Proceedings of the 23rd International Conference on Information Integration and Web-based Applications & Services (iiWAS2021); Linz, Austria, November 29-December 1, 2021*. Association for Computing Machinery (ACM), 2021, pp. 191–202. DOI: 10.1145/3487664.3487784. URL: http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2021-11&engl=0 (cit. on pp. 17, 21–25, 27, 29, 30, 32, 38, 40, 83).

[SGPM20] C. Stach, C. Gritti, D. Przytarski, B. Mitschang. “Trustworthy, Secure, and Privacy-aware Food Monitoring Enabled by Blockchains and the IoT.” In: *Proceedings of the 2020 IEEE International Conference on Pervasive Computing and Communications Workshops*. Ed. by C. Julien, P. Nurmi, J. X. Zheng. PerCom ’20. Austin: IEEE, Mar. 2020, 50:1–50:4. ISBN: 978-1-7281-4717-8. DOI: 10.1109/PerComWorkshops48775.2020.9156150 (cit. on p. 22).

[SM18] C. Stach, B. Mitschang. “CURATOR—A Secure Shared Object Store: Design, Implementation, and Evaluation of a Manageable, Secure, and Performant Data Exchange Mechanism for Smart Devices.” In: *Proceedings of the 33rd ACM/SIGAPP Symposium On Applied Computing*. Ed. by H. M. Haddad, R. L. Wainwright, R. Chbeir, R. A. Haraty, A. N. Papadopoulos, J. Sun. DTTA ’18. Pau: ACM, Apr. 2018, pp. 533–540. ISBN: 978-1-4503-5191-1. DOI: 10.1145/3167132.3167190 (cit. on p. 22).

[SR14] G. Schreiber, Y. Raimond. *RDF 1.1 Primer*. 2014. URL: https://www.w3.org/TR/rdf11-primer/ (cit. on p. 52).

[SS13] S. Sagiroglu, D. Sinanc. “Big data: A review.” In: *2013 International Conference on Collaboration Technologies and Systems (CTS)*. 2013, pp. 42–47. DOI: 10.1109/CTS.2013.6567202 (cit. on pp. 17, 22, 23).

[SSF17] C. Stach, F. Steimle, A. C. Franco da Silva. “TIROL: The Extensible Interconnectivity Layer for mHealth Applications.” In: *Proceedings of the 23rd International Conference on Information and Software Technologies (ICIST 2017)*. Ed. by R. Damaševičius, V. Mikašytė. Vol. 756. CCIS. Druskininkai: Springer, Oct. 2017, pp. 190–202. ISBN: 978-3-31967-642-5. DOI: 10.1007/978-3-319-67642-5_16 (cit. on p. 22).

[SSM18a] C. Stach, F. Steimle, B. Mitschang. “The Privacy Management Platform: An Enabler for Device Interoperability and Information Security in mHealth Applications.” In: *Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies*. Ed. by R. Zwiggelaar, H. Gamboa, A. Fred, S. Bermúdez i Badia. Vol. 5. HEALTHINF ’18. Funchal: SciTePress, Jan. 2018, pp. 27–38. ISBN: 978-989-758-281-3. DOI: 10.5220/0006537300270038 (cit. on p. 22).

[SSM18b] C. Stach, F. Steimle, B. Mitschang. “THOR — Ein Datenschutzkonzept für die Industrie 4.0: Datenschutzsysteme für die Smart Factory zur Realisierung der DSGVO.” In: *Informatik 2018: Zukunft der Arbeit - Zukunft der Informatik, Tagungsband der 48. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 26.09. - 27.09.2018, Berlin*. Ed. by C. Czarnecki, C. Brockmann, E. Sultanow, A. Koschmider, A. Selzer. Vol. 285. LNI. Berlin: GI, Sept. 2018, pp. 71–83. ISBN: 978-3-88579-679-4. URL: https://dl.gi.de/handle/20.500.12116/17235 (cit. on p. 22).

[Sta15] C. Stach. “How to Deal with Third Party Apps in a Privacy System — The PMP Gatekeeper.” In: *Proceedings of the 2015 IEEE 16th International Conference on Mobile Data Management*. Ed. by C. S. Jensen, X. Xie, V. Zadorozhny. MDM ’15. Pittsburgh: IEEE, June 2015, pp. 167–172. ISBN: 978-1-4799-9972-9. DOI: 10.1109/MDM.2015.17 (cit. on p. 22).

[Tea09] R. Team. *Merging graphs - rdflib 6.0.1 documentation*. 2009. URL: https://rdflib.readthedocs.io/en/stable/merging.html (cit. on p. 53).

[TR21] D. Taniar, W. Rahayu. “Data Lake Architecture.” In: *Advances in Internet, Data and Web Technologies*. Springer International Publishing, 2021, pp. 344–357 (cit. on pp. 23, 27, 30).

[TS17] A. Tosyali, S. Sarkar. *Which programming language would be efficient to use for graph mining?* 2017. URL: https://www.researchgate.net/post/Which-programming-language-would-be-efficient-to-use-for-graph-mining (cit. on p. 69).

[ZI19] Y. Zhang, Z. G. Ives. “Juneau: data lake management for Jupyter.” In: *Proceedings of the VLDB Endowment* 12 (2019) (cit. on pp. 35, 40).

All links were last followed on December 05, 2021.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature