Demand-Driven Data Provisioning in Data Lakes

BARENTS — A Tailorable Data Preparation Zone

Christoph Stach
stachch@ipvs.uni-stuttgart.de
University of Stuttgart, IPVS / AS
Stuttgart, Germany

Julia Bräcker
julia.braecker@lc.uni-stuttgart.de
University of Stuttgart, IBTB / LC
Stuttgart, Germany

Rebecca Eichler
eichlera@ipvs.uni-stuttgart.de
University of Stuttgart, IPVS / AS
Stuttgart, Germany

Corinna Giebler
giebleca@ipvs.uni-stuttgart.de
University of Stuttgart, IPVS / AS
Stuttgart, Germany

Bernhard Mitschang
mitsch@ipvs.uni-stuttgart.de
University of Stuttgart, IPVS / AS
Stuttgart, Germany

ABSTRACT

Data has never been as significant as it is today. It can be acquired virtually at will on any subject. Yet, this poses new challenges towards data management, especially in terms of storage (data is not consumed during processing, i. e., the data volume keeps growing), flexibility (new applications emerge), and operability (analysts are no IT experts). The goal has to be a demand-driven data provisioning, i. e., the right data must be available in the right form at the right time. Therefore, we introduce a tailorable data preparation zone for Data Lakes called BARENTS. It enables users to model in an ontology how to derive information from data and assign the information to use cases. The data is automatically processed based on this model and the refined data is made available to the appropriate use cases. Here, we focus on a resource-efficient data management strategy. BARENTS can be embedded seamlessly into established Big Data infrastructures, e. g., Data Lakes.

CCS CONCEPTS

• **Information systems** → **Extraction, transformation and loading**; **Mediators and data integration**; *Data cleaning*; *Data federation tools*; • **Applied computing** → Bioinformatics.

KEYWORDS

data pre-processing, data transformation, knowledge modeling, ontology, data management, Data Lakes, zone model, food analysis

ACM Reference Format:
Christoph Stach, Julia Bräcker, Rebecca Eichler, Corinna Giebler, and Bernhard Mitschang. 2021. Demand-Driven Data Provisioning in Data Lakes: BARENTS — A Tailorable Data Preparation Zone. In *The 23rd International Conference on Information Integration and Web Intelligence (iiWAS2021), November 29-December 1, 2021, Linz, Austria.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3487664.3487784

iiWAS2021, November 29-December 1, 2021, Linz, Austria

ACM ISBN 978-1-4503-9556-4/21/11…$15.00
https://doi.org/10.1145/3487664.3487784

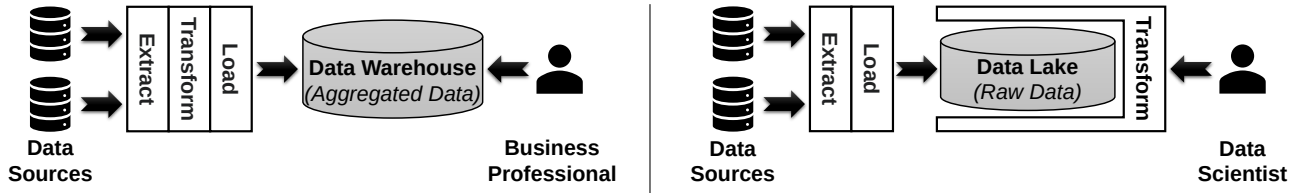
1 INTRODUCTION

"Data is the new oil." This metaphor, coined by Clive Humby (data science entrepreneur and Chief Data Scientist of Starcount) in the year 2006, is still used today to illustrate the importance of data in a digital society. This analogy also appears to be overly apt. Just as oil was a significant driver for the *Technological Revolution*, data is the key resource of the *Industry 4.0*. Similarities can also be identified with regard to the extraction and usage of data — like oil, data has to be *localized* and *extracted* first. Subsequently, data also needs to be cleansed and processed before it can be stored and provided to users for further exploitation [47].

Peter Sondergaard (Senior Vice President of Gartner) emphasizes the fact that data in general only becomes valuable when it is refined in his 2011 statement that *"information is the oil of the 21st century"*. For instance, datasets may contain inconsistent or inaccurate values, and relevant attributes might not be included in a dataset. Such impurity has to be cleansed in the run-up to an actual analysis. Furthermore, data is typically not available in the form needed for the intended analyses. Besides cleansing, data must therefore also be transformed, pre-processed, and enriched to gain meaningful information from it [30].

Although data shares many characteristics with oil, there are some crucial differences. This becomes particularly evident with regard to the following aspects: While oil is a finite resource, data can be generated at will. Additionally, data is not consumed during processing, i. e., it does not disappear — in terms of storage, concepts for managing *Big Data* are required. This is further aggravated as data is heterogeneous. Novel tasks emerge frequently, which is why pre-processing needs to be adapted dynamically to the ever-changing challenges. At the same time, refined data is highly specific, i. e., it only reaches its full potential in the use case envisaged through pre-processing. A ready-made one-size-fits-all solution is therefore not an option. It is rather necessary to involve the analyst during data preparation and to adapt it to the respective use case [11].

Moreover, data is not only a driver in industry, but also for, e. g., research and the private sector. A key issue for any data-driven project is rarely a lack of data, but rather the unavailability of the right data in the right form at the right time. John Naisbitt (author in the domain of futures studies) emphasizes that this is a key issue for a digital society with his revised allegory: *"We are drowning in data but starving for information."*



(a) Data Warehouse Architecture

(b) Data Lake Architecture

Figure 1: Comparison of the Data Warehouse Architecture and the Data Lake Architecture.

To this end, we introduce *BARENTS*[1], a tailorable data preparation zone for Data Lakes. It enables analysts to model how datasets have to be pre-processed to become meaningful information. This information is assigned to use cases to enable a demand-driven data provisioning. We make the following contributions:

1.) We introduce an ontology-based method that enables even non-IT experts to **specify the data requirements** of their use cases.

2.) We **manage the data** of the use cases in a resource-efficient manner that addresses the requirements inherent in a Big Data context.

3.) We demonstrate how BARENTS can be **embedded seamlessly** into an established Big Data infrastructure.

The remainder of this paper is structured as follows: In Section 2, we examine the state of the art with regards to data management. Section 3 identifies requirements towards demand-driven data provisioning. Based on this, we discuss related work in Section 4. Subsequently, we introduce BARENTS and provide details on its implementation in Section 5. BARENTS is evaluated in Section 6, before Section 7 concludes this paper.

2 DATA PROVISIONING ARCHITECTURES

When it comes to managing and providing large amounts of data in an efficient manner, *Data Warehouses* have long been considered the system architectures of choice. The basic idea is that there is a central relational database, which is considered as the single point of truth and serves as a foundation for any analysis. It has a predefined schema. Thereby business professionals even with little IT know-how are enabled to analyze the contained data with standardized tools (e. g., online analytical processing or reporting tools). The Data Warehouse is populated by external data sources. This entails that data extracted from these sources needs to be transformed first, before it is loaded into the Data Warehouse, in order to match the given schema. In this process, it is inevitable that the extracted raw data is altered and aggregated. This pre-processing and the resulting loss of information limits the analysis potential to a few predefined tasks [21].

Yet, due to the Internet of Things, more flexible solutions without such rigid schemes are needed. Moreover, the inherent transformations cause a significant delay before data is available for analysis. Lastly, the sheer processing of *who*, *what*, *when*, and *where* questions as intended in Data Warehouses is no longer adequate for most use cases [45].

Data Lakes represent such a dynamic system architecture. Similar to Data Warehouses, they provide a central gateway to all data extracted from sources. However, data is stored in an untransformed manner, i. e., in its original format and without a uniform schema. For this, heterogeneous Big Data technologies such as Hadoop are applied. To analyze the raw data, it has to be transformed beforehand, i. e., in addition to the actual analysis, users must also take care of the transformation. This requires extensive IT skills, which is why the target users are data scientists. Yet, they also face the problem of finding all required data in the Data Lake and recognizing the respective data formats on read [17].

Both architectural models are shown schematically in Figure 1 and their key characteristics are summarized in Table 1. It is evident that demand-driven data provisioning requires a compromise between both approaches combining the simple operability of Data Warehouses with the flexibility of Data Lakes. In the following, we examine the requirements towards such a compromise.

3 REQUIREMENT SPECIFICATION

Based on the work by Grossmann and Rinderle-Ma [16], we derive requirements towards a data management infrastructure, so that demand-driven data provisioning is feasible. We only consider requirements related to data provisioning. In a digital society, these requirements are relevant for any kind of data-intensive task.

R_1 – Adaptable: While in Data Warehouses analyses are known in advance, Data Lakes are supposed to be flexible. Since raw data

Table 1: Comparison of the Key Characteristics of a Data Warehouse and a Data Lake.

	Data Warehouse	Data Lake
Data	• Structured • Aggregated	• Structured & Unstructured • Raw
Storage	• Homogeneous • Schema-on-Write	• Heterogeneous • Schema-on-Read
Usage	• Predefined Tasks • Business Professional	• Support for any Use Case • Data Scientist
	➢ *Simple Data Retrieval*	➢ *Flexible Data Management*

[1]Our approach is named after the navigator and explorer Willem Barents — the historic Barents discovered new land in the seas, while our approach explores new insights within a Data Lake.

has to be transformed to be suitable for the intended purpose, i. e., the respective analysis, transformations have to be tailorable. Yet, a simple parameterization of pre-defined transformations is not sufficient; joins between data sources or the application of user-defined functions must be possible as well.

R_2 – Versatile: Similar to the ever-increasing number of use cases, i. e., analyses, new data types and data formats are also being added regularly to a Data Lake. Therefore, an approach towards a demand-driven data provisioning also has to be extensible in terms of new data sources as well as new data types and data formats to reflect this dynamic.

R_3 – Big-Data-Ready: Storing all raw data generates a high data volume. Due to the use-case-dependent transformation required for a demand-driven data provisioning, this data is additionally stored several times in different aggregation states. Therefore, the required disk space should also be restrained to a certain level, e. g., by not keeping all pre-processed data in a materialized manner.

R_4 – Easy-to-Use: Domain experts who have the expertise to specify the requirements for their analyses are usually no IT experts. Data Lakes solve this by requiring data scientists to prepare the data for every analysis. Yet, if no data scientist is available, non-IT experts must be able to tailor the transformations according to their use cases.

R_5 – Resource-Efficient: An approach has to be resource-efficient. On the one hand, the specification of the requirements for the analyses has to be accomplished in a time-saving manner. In particular, the user should not have to reinvent the wheel over and over again for each new use case. On the other hand, the overhead caused by the realization of these specifications, i. e., the actual data transformation and provision, has to be reasonable.

R_6 – Compatible: An approach should be compatible to pre-existing infrastructures, i. e., it has to be embeddable into today's Big Data architectures.

Having identified the requirements towards enabling demand-driven data provisioning, we analyze how related work addresses these challenges in the following.

4 RELATED WORK

Related work in terms of approaches to improve data discovery in flexible data provisioning architectures can be divided into four categories, namely *data partitioning*, *data catalogs*, *index structures*, and *navigation assistance* [32]. In addition, there is a recent trend towards the use of *AI-based pre-processing approaches* which prepare raw data for specific use cases [28]. We assess these five concepts in terms of how they facilitate demand-driven data provisioning. Due to the large number of similar approaches, we only discuss representatives for each category.

Data Partitioning. The most intuitive approach to facilitate data discovery is partitioning the data and keeping the amount of data in each partition small. A *Data Puddle* therefore only contains data for a single use case. If the contained data is also transformed, the boundaries to a Data Warehouse become blurred. To support multiple use cases, several Data Puddles can be linked to create a *Data Pond* [14]. Yet, this kind of partitioning causes some problems, e. g., there no longer is a single point of truth and data relevant for multiple use cases must be kept redundant in several Data Puddles. Moreover, similar to a Data Warehouse it must be known in advance which use cases have to be supported. Therefore, *zone models* provide an alternative type of partitioning. Besides a Raw Data Zone, in which the original raw data is held, there are zones which hold transformed versions of the data. Users operate on this pre-processed data. How the data is transformed is defined by the Data Lake [35]. Some models have several zones to provide the data in different processing levels. Users can select the most appropriate zone for their particular use cases [39]. Yet, as the pre-processing is as generic as possible, further transformations are required, which users have to implement and carry out on their own.

Data Catalogs. A *Data Catalog* is a digital inventory of data in the Data Lake, i. e., it contains information about the data source, available attributes, or their data types. Thereby, it facilitates the discovery of data and its usage. While in Data Warehouses a description of all data is possible due to their rigid schemata, this is often not the case in flexible Data Lakes. A thorough and systematic coverage of all data in a Data Lake is still barely addressed in literature [25]. For this purpose, a comprehensive metadata management is required. Sawadogo and Darmont [37] identify two distinct typologies of metadata. *Functional metadata* covers manually generated business metadata, which facilitates the semantic understanding of data, as well as automatically captured operational metadata, which describes the processing of the data, and technical metadata, which contains information about the data formats. For instance, *HANDLE* [9] follows this typology. *Structural metadata* adopts an object-oriented approach by regarding items in Data Lakes as objects with attributes and relations between them. The *CC Data Lake* [38] takes a structural metadata approach using a knowledge graph. However, a Data Catalog only facilitates data discovery, not its demand-driven pre-processing.

Index Structures. While Data Catalogs are designed to improve data discovery by describing the data in more detail, *query-driven discovery* takes the opposite approach. Here, incoming queries are analyzed, and datasets are identified that are similar to the query results. Comprehensive index structures are a key enabler to achieve this. Bogatu et al. [4] leverage hash-based indexes over the features of the data to find similar datasets. By contrast, Zhang and Ives [49] focus on a higher level of granularity. They exploit existing indexes of the source databases of a Data Lake, to recommend additional potentially relevant tables. Yet, similar to Data Catalogs, demand-driven data pre-processing remains the users' responsibility and they are not supported in this process.

Navigation Assistance. This category summarizes approaches that enable users to browse a Data Lake and comprehend its content. To this end, Nargesian et al. [33] organize the content of a Data Lake using a hierarchical tree structure where the actual data is in the leaves. Starting from a root element, the data is divided into finer and finer categories. By navigating the hierarchy, users find similar data in the respective subtrees. *Data Markets* take this concept a step further. Like in a store for physical goods, data providers offer their data and data consumers can obtain them. This enables market basket analyses and recommendations of useful data based

on previous acquisitions or collaborative filtering, i. e., the behavior of comparable users. Fernandez et al. [10] introduce such a Data Market. This requires data providers to be interested in users finding their data and therefore prepare it accordingly. Yet, it can neither be assumed that every data provider invests such effort, nor that an appropriate data offer is available for every demand.

AI-Based Pre-Processing Approaches. The last category of approaches is about using AI techniques to pre-process raw data in a fully automated way in order to make it available for use cases in a tailored form. To this end, Mishra et al. [31] study various ensemble approaches, in which several predefined pre-processing techniques are applied to raw data. All of these approaches have in common that arbitrary combinations of pre-processing techniques are applied to the data in a fully automated manner. For each of the resulting datasets, a machine learning model is computed for the respective use case. The quality of these models (e. g., accuracy or precision) is evaluated. All models with a quality better than a given baseline (e. g., a model trained on the raw data) are pooled into an ensemble. When deployed, all models in the ensemble are evaluated and the weighted average of the results of each model is considered as the overall result. However, the data pre-processing considered here is limited to standard data cleansing tasks, such as handling missing data or removing noise and outliers. Use-case-specific transformations of the raw data are not considered. Chaari Fourati and Ayed [7] present an approach based on federated learning that enables the transformation of data in near real-time. Here, the pre-processing steps are applied as close to the data source as possible. Yet, in order to enable automatic distribution of the transformation tasks, a model of the infrastructure and the data pre-processing workflow is required. This model can be specified, for instance, using an extended version of BPMN [20]. As the key focus of this approach is on the distribution of the transformation tasks, primarily infrastructure knowledge and less knowledge about the data itself (which is the core competence of the domain experts) is included in the model. Also, information is lost in this approach as sources only submit pre-processed data instead of raw data.

All AI-based approaches face the inherent problem that, apart from use-case-independent standard tasks such as data cleansing tasks, predefined pre-processing steps are required. Consequently, use-case-specific domain knowledge has to be integrated in order to improve automated pre-processing (or to enable it in the first place) [24]. Moreover, since the trained models are usually difficult to understand, subsequent adaptations by the domain expert are hardly possible [26]. Due to the use-case-specific pre-processing tasks, the models also cannot be applied to new use cases without further ado. Therefore, domain experts have to be involved in the training of new models for each and every use case, which is contrary to the fully automated idea behind such AI-based approaches.

Besides these five main categories, there are many hybrid approaches that combine these concepts. Also, combinations of dynamic Data Lakes with user-friendly Data Warehouses are possible [27]. For instance, Jensen et al. [19] present a programming framework that enables to easily specify extract-transform-load flows to transform data from heterogeneous sources into a processable format. Here, users only need to implement the three individual steps in Python, while the framework handles the execution of the workflow. Thereby, the flows are parallelized as much as possible, which renders the approach suitable for processing Big Data [46]. As shown in Section 2, the extract-transform-load workflow is applied in both, Data Warehouses as well as Data Lakes. Although the approach by Jensen et al. [19] greatly simplifies the realization of the workflows as a whole, a lot of IT knowledge is still required to implement the individual steps.

In addition, all related work require some kind of *analysis-driven discovery*, providing the essential data for the respective analyses [32]. Without an efficient data provisioning, however, a Data Lake deteriorates substantially [29].

Overall, two opposing trends can be identified in related work: enabling domain experts to contribute to data pre-processing and a full automation of the data preparation process.

While the latter sounds very appealing, as an AI might discover new insights regarding data pre-processing and additionally no expensive and rare domain experts are required [18], studies show that full automation is not effective and a trade-off between both trends has to be achieved [8]. That is, comprehensive tool assistance of domain experts in data pre-processing is needed [1].

To this end, we introduce our approach BARENTS in the following section to solve this problem.

5 A TAILORABLE DATA PREPARATION ZONE

Since related work supports users to identify relevant data within the flood of data available in a Data Lake, with BARENTS we take the next logical step by enabling non-IT experts to pre-process this data in a demand-oriented manner. This is crucial as the human and his or her knowledge should always be first-class citizen in the data pre-processing process [2].

For this, we build upon prior work by Giebler et al. [13], which introduces a holistic zone architecture for Data Lakes. This architecture categorizes a Data Lake in two areas: a use-case-independent area and a use-case-dependent area. At the transition point, data scientists or other IT experts transform the available raw data into refined information. This is where BARENTS comes into play by tailoring the data to the specific use cases for the user.

In the following, we first describe how BARENTS adapts this architecture in Section 5.1. We then discuss in Section 5.2 how domain experts are able to specify transformations in BARENTS in a comprehensible manner. Finally, Section 5.3 outlines how BARENTS is implemented.

5.1 BARENTS Zone Architecture

The architecture by Giebler et al. [13] comprises several zones which hold pre-processed data in addition to the raw data. The pre-processing can be highly use-case-specific. Users operate on the refined data. As a result, data can be provided in a demand-driven manner. Yet, profound IT know-how is needed to implement the required transformations, i. e., the effort to support a novel use case is high.

Therefore, in BARENTS we extend this architecture. The result is shown in Figure 2. Here, we differentiate between zones that contain data persistently (depicted in white) and zones that contain data only transiently (depicted in gray).

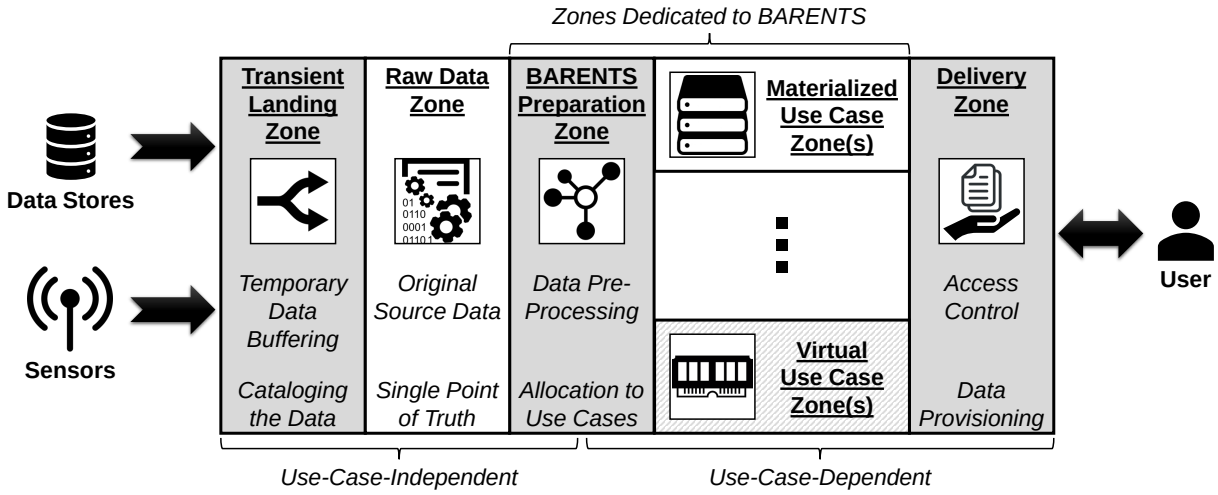


Figure 2: The BARENTS Zone Architecture.

Data from different source systems enters the Data Lake via the *Transient Landing Zone*. This zone acts as a kind of buffer, since data that is ingested into the Data Lake can accrue in large volumes at a high velocity. If the subsequent zones cannot cope with such high ingestion rates, the Transient Landing Zone can compensate for this by forwarding the incoming data staggered in batches. This zone is supposed to fulfill two additional tasks: On the one hand, it is intended to ensure that the incoming data is authentic, and on the other hand it has to annotate incoming data with metadata that may be relevant for later processing or retrieval. Gritti et al. [15] present a lightweight approach that facilitates both tasks. Here, incoming data packets are signed by the sources using an attribute-based signature. Such a signature guarantees the integrity of the data and provides the necessary metadata via the attributes at the same time.

The unmodified data is then loaded into the *Raw Data Zone*. Since the data here is in an unprocessed state[2], this zone is considered the single point of truth for the Data Lake. It is not intended for users to interact directly with the Raw Data Zone. For this reason, a wide range of storage technologies and systems are used in this zone (e. g., the *Hadoop Distributed File System*[3]), since the main focus is on an efficient data storage and not on an easy data discovery and retrieval.

In order to facilitate data discovery and retrieval, the *BARENTS Preparation Zone* harmonizes the data in the Data Lake. That is, it takes the raw data and pre-processes it based on the demands of the users. As a result, this zone represents the transition from the use-case-independent components of the BARENTS Zone Architecture to its use-case-dependent components. A great deal of insight about the intended use cases — i. e., domain knowledge — is therefore required at this point. Section 5.2 outlines an example of how a domain expert can specify the use-case-specific demands.

The transformed and pre-processed data is stored in one of the *Use Case Zones*. These zones later serve as primary data sources and contain the refined data as demanded by the use cases. The user determines how specifically the data is adapted to a particular use case. S/he is also able to use any technology to store the data in the Use Case Zones. For instance, the *Materialized Use Case Zones* can be implemented as relational databases such as the *Oracle Database*[4] or NoSQL data stores such as the *Apache CouchDB*[5], depending on the requirements of the respective use case.

Since such redundant data storage could be an issue in a Big Data context[6], BARENTS also introduces *Virtual Use Case Zones*. These are, e. g., a strictly in-memory database such as the *Raima Database Manager*[7], a hybrid database such as *Oracle Berkeley DB*[8], or a *Kafka Broker*[9] providing the data via data streams — i. e., the data pre-processed by BARENTS is not persisted but forwarded directly to the user. This enables users to choose the type of data delivery demand-driven for their use cases.

The *Delivery Zone* enables demand-driven data provisioning by referring to the appropriate Use Case Zone. To determine an appropriate Use Case Zone[10], an approach such as the one introduced by Stach et al. [42] can be applied. In this approach, users are identified based on their attributes. Access rights are specified in terms of *public patterns* (i. e., data requirements of an authorized user) and *private patterns* (i. e., insights that must not be disclosed). A utility metric that maximizes the number of detectable public patterns and minimizes the number of disclosed private patterns can be used to select an eligible Use Case Zone.

Orthogonally to these *operational zones*, a Data Lake requires a *Management Zone* that contains, e. g., data catalogs, privacy restrictions, or access control policies [39]. Yet, these aspects are not relevant for this paper and are therefore not shown in the figure for the sake of simplicity.

[2]Nota bene: "Unprocessed" refers to the actual data — it is, however, enriched by metadata when it is stored in the Raw Data Zone.
[3]see https://hadoop.apache.org/docs/r1.2.1/hdfs_user_guide.html
[4]see https://www.oracle.com/database/technologies/
[5]see https://couchdb.apache.org/
[6]Note that a single dataset in the Raw Data Zone can be included in multiple Use Case Zones.
[7]see https://raima.com/
[8]see https://www.oracle.com/database/berkeley-db/
[9]see https://kafka.apache.org/
[10]In this context, not only the data requirements of the user, but also his or her access rights as well as data privacy requirements have to be taken into account.

5.2 BARENTS Configuration

To enable domain experts without IT know-how to describe the data requirements of their use cases, we adopt an ontology-based approach in BARENTS. There are many alternative approaches towards the representation of knowledge, e. g., *linguistic knowledge bases*, *expert knowledge bases*, or *cognitive knowledge bases*. However, in our opinion, ontologies are best suited for the configuration of BARENTS, as they are well suited to persist expert knowledge in a machine-processable manner. Since a lot of research work has been done in the area of ontologies, they have reached a high maturity level. Thus, many methods, tools, languages, and systems have been developed in recent years which greatly facilitate the manual creation of ontologies. In addition, machine learning and natural language processing approaches also enable an automatic or semi-automatic creation of ontologies [48]. However, the main benefit of ontologies, which makes them particularly appealing for our purposes, is that they are primarily designed to facilitate the sharing and reuse of knowledge [3]. Thus, the knowledge of domain experts can not only be persisted, but also successively adapted in order to address novel use cases. This constantly reduces the configuration effort for the domain experts since they can leverage the already established knowledge base.

The BARENTS ontology is inspired by the *Data Pyramid* [36], i. e., a model for relationships between *data*, *information*, and *knowledge* [50]. Data is a disjointed collection of facts. It is heterogeneous, unorganized, and not assigned to a particular context. If this data is processed, combined, and assigned to a specific purpose, information is obtained. The preparation steps that are used in this process are commonly validations (i. e., filtering out false or irrelevant data), transformations (i. e., mapping a function to a set of data), and aggregations (i. e., reducing a set of data to a result value). If the obtained information is used to achieve a goal, it is referred to as knowledge. That is, in order to reach the highest level in terms of information value, the information has to be assigned to a specific use case in which it can be profitably used.

Translated to the configuration of BARENTS, at the *data level*, users select the data required for their analyses from the Raw Data Zone and model which data features have to be included in the pre-processing. Data features are modeled as resources in the ontology and data sources are descriptive literals that are linked to these resources via relationships. Additional literals provide further information, e. g., access credentials. The discovery and selection of suitable data sources is supported by the metadata available for the Raw Data Zone.

At the *information level*, the transformations are specified. Initially, all the relevant data features are grouped in a resource. Then, the function which has to be applied to the data, is described in a literal linked to this resource. Additionally, it is modeled which type of transformation is used. BARENTS supports four different types of transformations:

I) A *filter* operator deals with the elimination or selection of facts from the specified data sets at the data level. For this purpose, a filter function has to be specified that has the following mathematical signature:

$$f : D \to \mathbb{B}$$

This function is applied to a set of data of data type D and maps each of them to a boolean value. All facts selected at the data level where this boolean value evaluates to `false` are filtered out of the set of data.

II) A *map* operator deals with the transformation or modification of facts from the specified data sets at the data level. For this purpose, a map function has to be specified that has the following mathematical signature:

$$f : D \to E$$

This function is applied to a set of data of data type D and each fact contained in the set is processed accordingly. In this process, the facts selected at the data level are cast to data type E (which can be identical with data type D).

III) A *reduce* operator deals with the aggregation of facts from the specified data sets at the data level. For this purpose, a reduce function has to be specified that has the following mathematical signature:

$$f : E \times D \to E$$

This binary function is applied to a set of data of data type D. All facts contained in the set are combined to a single output value of data type E. Similar to the accumulator in the *von Neumann Architecture*, this binary function takes one fact from the input data set and the current intermediate result — i. e., the data is processed progressively. After processing all facts selected at the data level, the last intermediate result is the result of the reduction.

IV) Even though many of the pre-processing tasks can be carried out with the three operators presented above, BARENTS also provides support for the use of user-defined functions. For this purpose, the transformation type *procedure* can be used. Although it cannot be assumed that domain experts are able to implement arbitrary transformations independently, implementations for frequently used pre-processing operations can be provided by the IT department in programming libraries that can be integrated and used in BARENTS.

The *knowledge level* is used to model where the results are stored. Here, a literal is used to denote in which Use Case Zone this data sink is located. Similar to the data level, further literals can describe the data sink in more detail, e. g., in terms of access rights.

Sample Use Case Scenario. To demonstrate the configuration of BARENTS, we selected a real-world sample scenario from the field of food chemistry. As food allergies are on the rise in Western countries, it is very important to test food for allergens on a regular basis. In this context, nut seeds are among the most prevalent food allergies which can trigger severe allergic shocks. Hence, in the considered use case, it has to be analyzed, whether a food sample contains allergens. More precisely, it has to be ascertained whether chocolate samples contain traces of hazelnut or walnut [6].

In the analysis considered in this paper, the samples are examined at the molecular level. That is, the samples are searched for allergenic peptides, i. e., fragments of proteins. Thereby, it is possible to detect even the smallest traces of nuts in the samples. To this end,

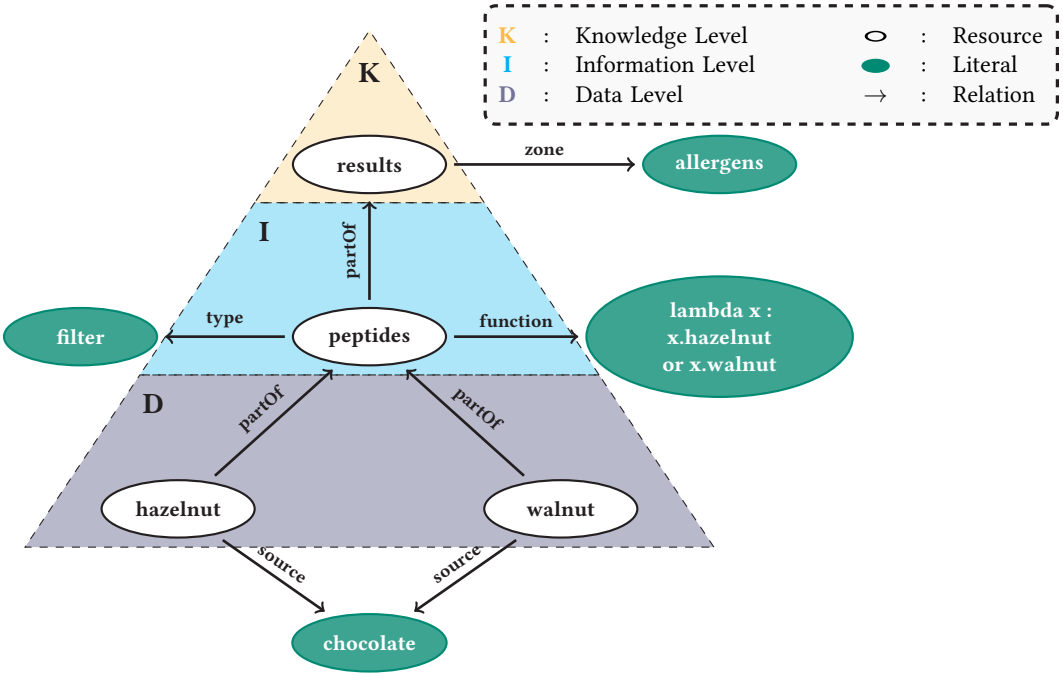


Figure 3: Excerpt of an Instance of the BARENTS Ontology.

the samples are analyzed with a *LTQ Orbitrap XL* mass spectrometer[11] paired with an *Accela HPLC* system[12]. The thereby acquired raw data is cross-checked against a protein sequence database[13] to determine whether hazelnut or walnut peptides are contained in the sample. The samples in which a match occurred are marked. To reduce the data volume for subsequent analyses, unmarked samples are then filtered out. The remaining samples, i. e., all chocolates in which traces of nut seeds were detected, are loaded into a peptide analysis software[14] for in-depth analysis [23].

Figure 3 illustrates how a food chemist can specify a configuration for the previously described use case in the BARENTS ontology. However, for the sake of simplicity and clarity, only an excerpt of the complete configuration is shown here. This excerpt contains only the last pre-processing step, in which all irrelevant food samples — i. e., the chocolates that do not contain hazelnut or walnut traces — are filtered out.

To this end, the domain expert initially selects the database containing the chocolate samples. This is modeled as a literal in the ontology, which is the data source for the specified transformation. From the records contained in it, only the features indicative of hazelnuts or walnuts traces are selected for filtering at the data level. At the information level, a resource is defined that takes these two features as input and applies a filter operator to them. The filter is described as a lambda expression that evaluates to true if one of the two features is present in a sample, i. e., if a corresponding marker is found. Finally, at the knowledge level, it is specified that all samples forwarded by the filter operator have to be inserted into the use case zone "allergens".

Prior to the depicted excerpt, a similarly structured map operator is needed, which assigns a corresponding marker to all samples, for which the comparison with the protein sequence database indicates that they require an in-depth analysis. This also highlights another key feature of our ontology — all transformations can be arbitrarily combined and composed. That is, the outcome of a transformation, i. e., the resources at the knowledge level, can be used as input, i. e., as literals at the data level, in subsequent transformation steps.

Listing 1: RDF/XML Representation of an Instance of the BARENTS Ontology.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  ↪ xmlns:dl="http://barents.dl/">
   <rdf:Description rdf:about="http://barents.dl/walnut">
      <dl:layer>Data Layer</dl:layer>
      <dl:source>chocolate</dl:source>
      <dl:partOf rdf:resource="http://barents.dl/peptides"/>
   </rdf:Description>
   <rdf:Description rdf:about="http://barents.dl/hazelnut">
      <dl:layer>Data Layer</dl:layer>
      <dl:source>chocolate</dl:source>
      <dl:partOf rdf:resource="http://barents.dl/peptides"/>
   </rdf:Description>
   <rdf:Description rdf:about="http://barents.dl/peptides">
      <dl:layer>Information Layer</dl:layer>
      <dl:function>lambda x : x.hazelnut or x.walnut</dl:function>
      <dl:type>filter</dl:type>
      <dl:partOf rdf:resource="http://barents.dl/results"/>
   </rdf:Description>
   <rdf:Description rdf:about="http://barents.dl/results">
      <dl:layer>Knowledge Layer</dl:layer>
      <dl:zone>allergens</dl:zone>
   </rdf:Description>
</rdf:RDF>
```

[11] see https://www.selectscience.net/products/thermo-scientific-ltq-orbitrap-xl?prodID=81295
[12] see https://www.selectscience.net/products/thermo-scientific-accela?prodID=20507
[13] see https://www.uniprot.org/
[14] see https://www.bioinfor.com/peaks-studio/

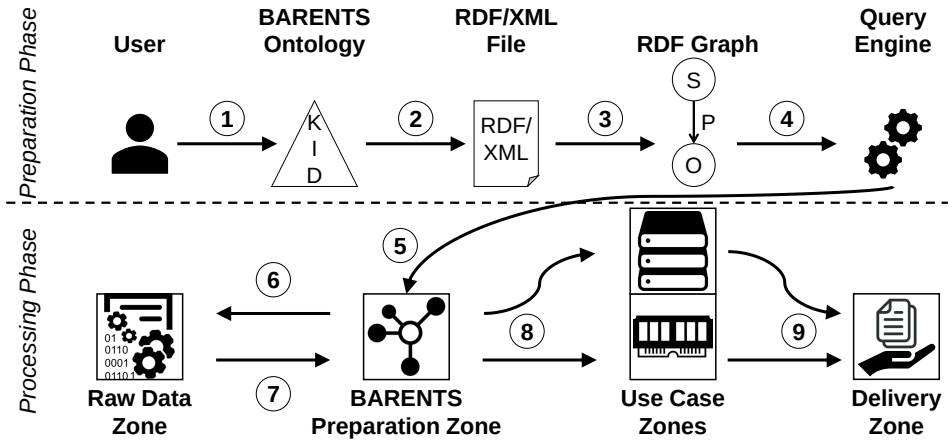


Figure 4: Two-Phase Configuration and Operation Process of BARENTS.

5.3 BARENTS Implementation

We use *RDF/XML*[15] for the internal representation of the ontology. RDF is well suited to describe ontology instances in a formal way and for its XML representation, a lot of processing libraries are available. In addition, XML files can be easily shared among users. In Listing 1, the instance of our ontology modeled in Figure 3 is given in this notation. To enable efficient queries on these RDF/XML files, an equivalent RDF graph has to be generated first. As shown in Figure 4, they are processed in two phases: an initial *preparation phase* and a subsequent *processing phase*.

① First, a domain expert models the pre-processing steps required for his or her use cases, e. g., with the help of a graphical editor. To this end, s/he can make use of existing RDF triples and adopt or extend them. In the example presented in Figure 3, the identified marker peptides could be cross-checked against an external protein sequence database or information emphasizing scripts as presented by Stach et al. [44] could be applied to the data.

② The sum of all triples constitutes the BARENTS ontology. An RDF/XML file is used internally to store it. ③ In order to derive the required pre-processing steps, the XML file is parsed, and the corresponding RDF graph is generated. ④ Efficient query engines are available to traverse such graphs. With them, all subgraphs can be retrieved which describe a transformation for entities at the data level to entries at the knowledge level.

⑤ After this preparation phase, the derived pre-processing steps are then transferred to the BARENTS Preparation Zone as configuration for the processing phase. ⑥ There, all affected tuples are requested from the Raw Data Zone. ⑦ BARENTS applies the required transformations sequentially to the raw data. Filter, map, and reduce operators are translated into stream operators, while procedures are applied to the datasets using iterators.

⑧ Since the BARENTS Preparation Zone is a transient zone, pre-processed data is forwarded to the respective Use Case Zone specified in the ontology, i. e., either a materialized or a virtual one. ⑨ Users can access the prepared data via the Delivery Zone.

[15] see https://www.w3.org/TR/rdf-syntax-grammar/

For the data stores at the data and knowledge level, adapters handle the actual access. The information given in the ontology (e. g., access credentials) is used as configurations for these adapters. To support a new data store, BARENTS only needs to be extended by a corresponding adapter. Then, domain experts can use them in their transformation descriptions as sources or sinks without having to worry about technical details.

If needed, a second BARENTS Preparation Zone can be deployed between the Use Case Zones and the Delivery Zone to enable further data pre-processing prior to its release. For instance, use case data can be filtered or blurred to comply with privacy policies and enable fine-grained data sharing in the process [41].

A proof-of-concept implementation for the use case outlined in Section 5.2 is depicted in Figure 5. The databases used in the Raw Data Zone and in the Use Case Zone are only examples that have proven to be suitable for the given use case. Depending on the available infrastructure, other databases and storage technologies can be used as well. One only has to define appropriate connectors for BARENTS that handle the required data accesses. More details on the prototypical implementation of BARENTS as well as an evaluation of our approach are given in the following section.

6 EVALUATION

After introducing our tailorable data preparation zone, BARENTS is now evaluated. For this purpose, first the runtime overhead as well as the memory requirements of the BARENTS prototype are determined in Section 6.1. Then, in Section 6.2, the scope of functionality of the BARENTS approach in general regarding the six requirements specified in Section 3 is assessed. These two aspects provide indicators for the soundness of our concept regarding the support of demand-driven data provision.

6.1 Performance Measurement

In order to evaluate the performance of our approach, we implemented the use case outlined in Section 5.2 with BARENTS. The result is shown in Figure 5. However, it has to be noted that in this prototypical implementation created specifically for performance measurement, both the data import (i. e., the Transient Landing

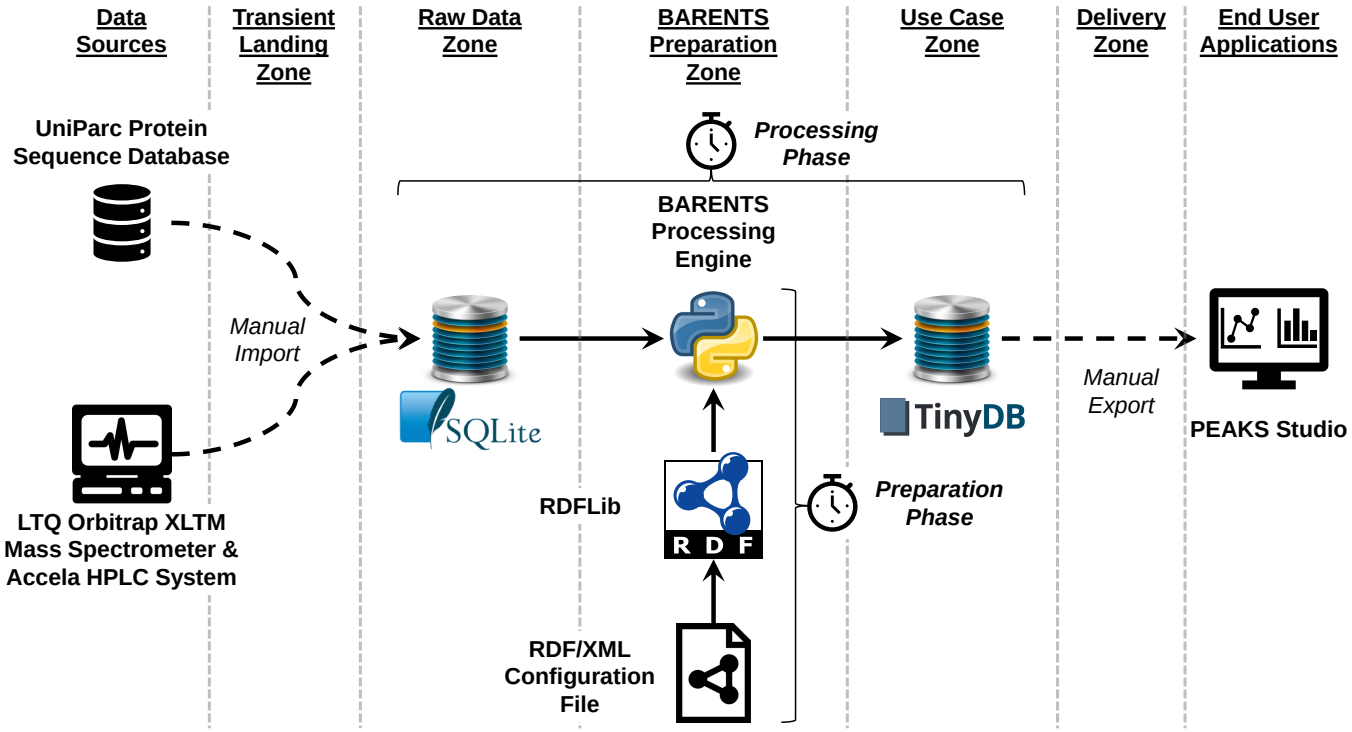


Figure 5: A Prototypical Implementation of BARENTS used for the Evaluation.

Zone) as well as the data export (i. e., the Delivery Zone) are handled manually. That is, both the data transfer from the metering devices (i. e., the LTQ Orbitrap XLTM mass spectrometer and the Accela HPLC system) and the UniParc protein sequence database into the Raw Data Zone as well as the data access of the analytics software (i. e., PEAKS Studio) to the (Virtual) Use Case Zone are carried out manually by the domain experts. This is due to the fact that these operations are not within the scope of BARENTS. For more information on how these two zones can be realized, please refer to the work of Giebler et al. [13].

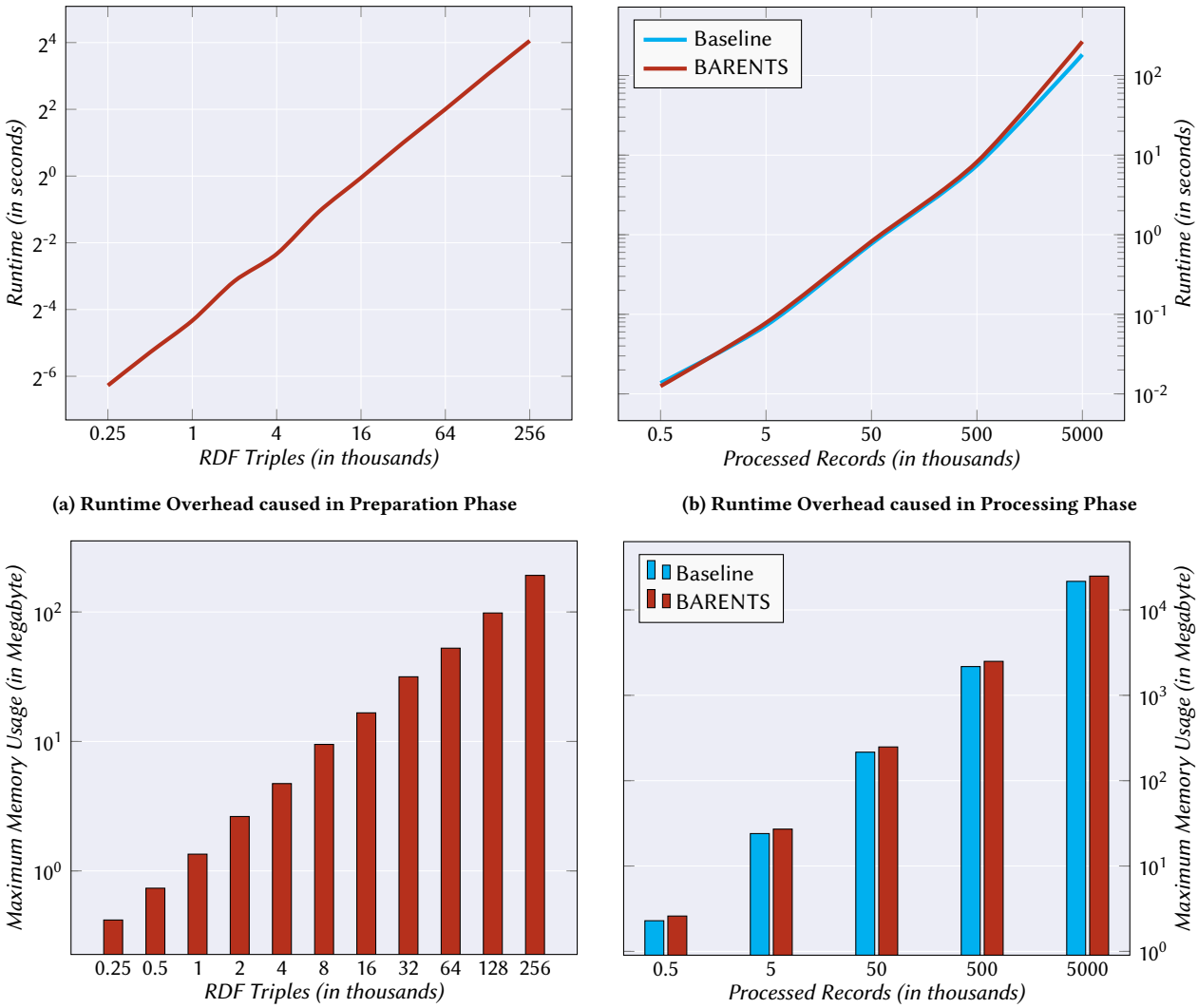
As both the data from the metering devices as well as the comparison data from the protein sequence database are structured data, a relational database is suitable for implementing the Raw Data Zone in this use case. Due to the given data volumes and for reasons of simplicity, we decided to use *SQLite DB 3.35.5*[16]. Since only samples that are subjected to an in-depth analysis are forwarded to the Use Case Zone, we opted for a Virtual Use Case Zone — it is sufficient to keep the samples in their original form in the Raw Data Zone permanently. The document-oriented database *TinyDB 4.5.1*[17] proved to be appropriate for this intermediate storage. However, the data stores in both zones can be substituted as needed, without any significant impact on the validity of the subsequent measurements.

The core of BARENTS, the BARENTS Preparation Zone, is implemented in *Python 3.9.6*[18]. All transformations specified in the ontology are handled by the BARENTS Processing Engine (partially supported by *pandas 1.3.1*[19]). The ontology, which is provided as an RDF/XML file, is parsed using *RDFLib 6.0.0*[20] and converted into a processable RDF graph. As Python allows to execute dynamically created program code, all user-defined transformations specified in the ontology can be applied to the raw data straightforwardly.

In the performance measurement, both the preparation phase as well as the processing phase are considered, as indicated in Figure 5. In order to determine the overhead of the preparation phase — i. e., the overhead caused by compiling and processing the RDF graph — we supplemented the ontology containing the transformations introduced in Section 5.2 by artificially generated ones. By doubling the generated triples gradually, we obtained eleven ontologies consisting of 250 RDF triples (i. e., 20 transformations) up to 256k RDF triples (i. e., over 21k transformations). To determine the overhead of the processing phase — i. e., the overhead caused by the actual transformations — we incrementally populated the Raw Data Zone with 500 up to 5,000k entries (i. e., approximately 15 GB of data), increasing the data volume tenfold at each step.

The runtime overhead in the preparation phase is defined as the time elapsing between reading in the RDF/XML file and completely traversing the compiled graph, i. e., it indicates how long it takes to identify all relevant transformations within the ontology. This overhead is entirely accruing due to the usage of BARENTS, since such a preparation phase would not occur if the transformations were performed manually. That is, no baseline can be provided for this overhead.

[16] see https://www.sqlite.org/
[17] see https://github.com/msiemens/tinydb/
[18] see https://www.python.org/
[19] see https://pandas.pydata.org/
[20] see https://rdflib.readthedocs.io/



(a) Runtime Overhead caused in Preparation Phase

(b) Runtime Overhead caused in Processing Phase

(c) Memory Overhead caused in Preparation Phase

(d) Memory Overhead caused in Processing Phase

Figure 6: Overhead caused by BARENTS.

The runtime overhead in the processing phase is defined as the time elapsing until the transformations are applied to all raw data and the pre-processed data is available in the corresponding Use Case Zone, i. e., it indicates how long it takes until the refined data can be provided to users. For benchmark purposes, we have determined the runtime of the best possible manual realization of the transformations as a baseline for the overhead of the processing phase. Here, for instance, the filtering was applied directly to the raw data in the database via selection operators to exploit existing index structures and also reduce the data volume at an early stage.

In addition to the runtime overhead, we have also determined the memory consumption of BARENTS for both phases in a similar way. We identified the memory peak usage of BARENTS during compiling and processing the RDF graph as well as transforming the raw data. For the processing phase, we also compared the consumption of BARENTS to a baseline consisting of the best possible manual implementation.

Each measurement was performed five times on an Intel Core i7-1165G7 (11$^{\text{th}}$ gen.) at 2.80 GHz with 32 GB RAM. After each run, all databases were reset to avoid side effects caused by warm caches. Figure 6 shows the medians for each measurement. We use the median to exclude side effects caused by background processes.

It is particularly promising that in both phases the overhead — both in terms of runtime as well as memory consumption — is in $\mathcal{O}(n)$, i. e., it grows linearly with the number of RDF triples or data records that are processed, respectively. Furthermore, it is remarkable that the overhead in the processing phase is competitively viable with respect to the baseline.

A detailed review of the performance data also reveals that the runtime overhead caused by BARENTS is low. 21k transformations are traversed in approximately 16.6 s and 15 GB of data are transformed in 267.3 s. It is important to keep in mind that both overheads are one-time costs per use case and accrue in advance of the actual analyses.

Table 2 lists the detailed consolidated runtime costs (i. e., the sum of the overhead of the preparation phase and processing phase). It is evident that when dealing with Big Data, it does not greatly differ whether a small ontology (20 transformations) or a large ontology (21k transformations) is used. The main overhead is therefore caused during the processing phase. However, when comparing these costs to the baseline, it is evident that the consolidated runtime overhead caused by BARENTS is approximately 155% higher.

This overhead is partly due to the fact that pandas is not designed to process Big Data [34]. This can be improved, e. g., by approaches such as *Grizzly* [22], in which part of the computation is carried out in the databases of the source system — i. e., the transformations performed by BARENTS get considerably more in line with the manually optimized baseline. An alternative optimization strategy is to use a highly parallelized computation library, such as the extension for *Dask*[21] presented by Böhm and Beránek [5], which scales better with Big Data. This also has a positive effect on the memory consumption.

6.2 Feature Discussion

As the performance evaluation indicates that the overhead caused by BARENTS is compelling, we assess its functionality next.

A key requirement is that the pre-processing is **adaptable** (R_1). In BARENTS this is achieved as user-defined functions can be specified in the ontology at the information level. Due to the dynamic evaluation of these functions, external scripts can also be invoked here. This way, even complex transformations can be applied to the data. The transformations also must be **versatile** (R_2). Since any source and sink can be included at the data and knowledge level via adapters, this is fulfilled in BARENTS. Moreover, the BARENTS Zone Architecture supports both, materialized and virtual sinks. The virtual sinks reduce the generated data volume, which makes BARENTS **big-data-ready** (R_3).

[21] see https://dask.org/

Table 2: Consolidated Runtime Overhead caused by BARENTS (Preparation and Processing Phase).

Records	Baseline	BARENTS	
		small	*big*
500	0.01 s	0.02 s	16.61 s
$5k$	0.07 s	0.09 s	16.67 s
$50k$	0.77 s	0.84 s	17.42 s
$500k$	7.43 s	8.23 s	24.81 s
$5,000k$	183.34 s	267.35 s	283.94 s

Evidently, an approach targeting non-IT experts also has to be **easy-to-use** (R_4). The BARENTS ontology enables domain experts to model the data requirements of their use cases in an accessible manner without the need for profound IT know-how. A graphical editor such as *CoModIDE* [40] could further facilitate the modeling process. The runtime measurement indicated that BARENTS is **resource-efficient** in terms of performance (R_5). Also, the modeling process is efficient, as parts of the RDF graphs can be reused for other use cases. Stach and Steimle [43] introduce an approach to recommend such available patterns to modelers depending on the modeling context.

Lastly, a practicable approach also has to be **compatible** with existing infrastructures and must be seamlessly embeddable into them (R_6). BARENTS can be seamlessly embedded in any zone model for Data Lakes. Even without a Data Lake, BARENTS can be used whenever sources have to be connected to sinks and transformations must be applied to the data during transmission. The materialized and virtual zones also support a hybrid batch and stream processing, which is common in a Big Data environment [12]. Thus, BARENTS meets all requirements towards a data management infrastructure enabling demand-driven data provisioning.

7 CONCLUSION

The economic value of data is on the rise. Yet in order to fully exploit it, data needs to be refined first. This pre-processing must be tailored to the intended use case. While domain experts have the expertise to decide which pre-processing steps are needed, they often lack the IT skills required to implement them and apply them to the raw data.

To this end, we introduce a tailorable data preparation zone for Data Lakes called BARENTS that enables a demand-driven data provisioning. This is achieved by means of three factors:

1.) BARENTS introduces an **ontology** to specify data pre-processing steps. Domain experts can use this ontology to describe which data transformations have to be applied to the raw data from source systems in order to make it exploitable in their use cases.
2.) This ontology is used to configure the **BARENTS Preparation Zone**, an extension to a zone architecture for Data Lakes. Data Lakes are well suited for managing Big Data. This is further enhanced by the addition of **virtual zones** in BARENTS.
3.) By means of **adapters**, the BARENTS Preparation Zone can be connected to any data source or sink — i. e., BARENTS can be seamlessly integrated into any IT landscape.

Performance measurements and feature analyses show that BARENTS is not only performant, but also fulfills all requirements towards such a data provisioning approach.

As part of future work, the performance of BARENTS can be further enhanced, e. g., by carrying out part of the computation in the databases of the source system or by applying parallel computation approaches.

REFERENCES

[1] Michael Behringer, Pascal Hirmer, Manuel Fritz, and Bernhard Mitschang. 2020. Empowering Domain Experts to Preprocess Massive Distributed Datasets. In

Proceedings of the 23rd International Conference on Business Information Systems (BIS '20), 61–75.

[2] Michael Behringer, Pascal Hirmer, and Bernhard Mitschang. 2018. A Human-Centered Approach for Interactive Data Processing and Analytics. In *Enterprise Information Systems: 19th International Conference, ICEIS 2017, Porto, Portugal, April 26-29, 2017, Revised Selected Papers*. Slimane Hammoudi, Michał Śmiałek, Olivier Camp, and Joaquim Filipe, editors. Springer, Cham, 498–514.

[3] Andrew Thomas Bimba et al. 2016. Towards knowledge modeling and manipulation technologies: A survey. *International Journal of Information Management*, 36, 6, Part A, 857–871.

[4] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *Proceedings of the 2020 IEEE 36th International Conference on Data Engineering* (ICDE '20), 709–720.

[5] Stanislav Böhm and Jakub Beránek. 2020. Runtime vs Scheduler: Analyzing Dask's Overheads. In *Proceedings of the 2020 IEEE/ACM Workflows in Support of Large-Scale Science* (WORKS '20), 1–8.

[6] Julia Bräcker and Jens Brockmeyer. 2018. Characterization and Detection of Food Allergens Using High-Resolution Mass Spectrometry: Current Status and Future Perspective. *Journal of Agricultural and Food Chemistry*, 66, 34, 8935–8940.

[7] Lamia Chaari Fourati and Samiha Ayed. 2021. Federated Learning toward Data Preprocessing: COVID-19 Context. In *Proceedings of the 2021 IEEE International Conference on Communications Workshops* (ICC Workshops '21), 1–6.

[8] Tijl De Bie et al. 2021. Automating Data Science: Prospects and Challenges. *Communications of the ACM*, 1–19. *Accepted for publication (April 2021).* https://arxiv.org/abs/2105.05699.

[9] Rebecca Eichler et al. 2020. HANDLE–A Generic Metadata Model for Data Lakes. In *Proceedings of the 22nd International Conference on Big Data Analytics and Knowledge Discovery* (DaWaK '20), 73–88.

[10] Raul Castro Fernandez, Pranav Subramaniam, and Michael J. Franklin. 2020. Data Market Platforms: Trading Data Assets to Solve Data Problems. *Proceedings of the VLDB Endowment*, 13, 12, 1933–1947.

[11] Samuel Flender. 2019. Data is not the new oil. *Medium – towards data science*, (Feb. 2019). https://towardsdatascience.com/data-is-not-the-new-oil-bdb31f61bc2d.

[12] Corinna Giebler, Christoph Stach, Holger Schwarz, and Bernhard Mitschang. 2018. BRAID — A Hybrid Processing Architecture for Big Data. In *Proceedings of the 7th International Conference on Data Science, Technology and Applications* (DATA '18). Vol. 1, 294–301.

[13] Corinna Giebler et al. 2020. A Zone Reference Model for Enterprise-Grade Data Lake Management. In *Proceedings of the 2020 IEEE 24th International Enterprise Distributed Object Computing Conference* (EDOC '20), 57–66.

[14] Alex Gorelik. 2019. *The Enterprise Big Data Lake: Delivering the Promise of Big Data and Data Science*. Andy Oram, editor. O'Reilly Media, Sebastopol, CA.

[15] Clémentine Gritti, Melek Önen, and Refik Molva. 2019. Privacy-Preserving Delegable Authentication in the Internet of Things. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing* (IoT '19), 861–869.

[16] Wilfriedand Grossmann and Stefanie Rinderle-Ma. 2015. Data Provisioning. In *Fundamentals of Business Intelligence*. Springer, Berlin, Heidelberg. Chap. 3, 87–118.

[17] Saurabh Gupta and Vonkayala Venkata Giri. 2018. *Practical Enterprise Data Lake Insights: Handle Data-Driven Challenges in an Enterprise Big Data Lake*. Apress, New York, NY.

[18] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. 2019. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, Cham.

[19] Søren Kejser Jensen, Christian Thomsen, Torben Bach Pedersen, and Ove Andersen. 2021. pygrametl: A Powerful Programming Framework for Easy Creation and Testing of ETL Flows. In *Transactions on Large-Scale Data- and Knowledge-Centered Systems XLVIII: Special Issue In Memory of Univ. Prof. Dr. Roland Wagner*. Abdelkader Hameurlain and A. Min Tjoa, editors. Springer, Berlin, Heidelberg, 45–84.

[20] Ameni Kallel, Molka Rekik, and Mahdi Khemakhem. 2021. IoT-fog-cloud based architecture for smart systems: Prototypes of autism and COVID-19 monitoring systems. *Software: Practice and Experience*, 51, 1, 91–116.

[21] Ralph Kimball and Margy Ross. 2013. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. (3rd Edition ed.). Wiley, Indianapolis, IN.

[22] Steffen Kläbe and Stefan Hagedorn. 2021. Applying Machine Learning Models to Scalable Dataframes with Grizzly. In *Tagungsband der 19. Fachtagung für Datenbanksysteme für Business, Technologie und Web* (BTW '21), 195–214.

[23] Robin Korte, Julia Bräcker, and Jens Brockmeyer. 2017. Gastrointestinal digestion of hazelnut allergens on molecular level: Elucidation of degradation kinetics and resistant immunoactive peptides using mass spectrometry. *Molecular Nutrition & Food Research*, 61, 10, 1700130:1–1700130:16.

[24] Maxat Kulmanov, Fatima Zohra Smaili, Xin Gao, and Robert Hoehndorf. 2020. Semantic similarity and machine learning with ontologies. *Briefings in Bioinformatics*, 22, 4, 1–18.

[25] Clément Labadie, Christine Legner, Markus Eurich, and Martin Fadler. 2020. FAIR Enough? Enhancing the Usage of Enterprise Data with Data Catalogs. In *Proceedings of the 2020 IEEE 22nd Conference on Business Informatics* (CBI '20), 201–210.

[26] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. 2021. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy*, 23, 1, 18:1–18:45.

[27] Marilex Rea Llave. 2018. Data lakes in business intelligence: reporting from the trenches. *Procedia Computer Science*, 138, 516–524.

[28] Julián Luengo et al. 2020. *Big Data Preprocessing: Enabling Smart Data*. Springer, Cham.

[29] Christian Mathis. 2017. Data Lakes. *Datenbank-Spektrum*, 17, 3, 289–293.

[30] Amol Mavuduru. 2020. Is Data Really the New Oil in the 21st Century? *Medium*, (Dec. 2020). https://www.towardsdatascience.com/is-data-really-the-new-oil-in-the-21st-century-17d014811b88.

[31] Puneet Mishra et al. 2020. New data preprocessing trends based on ensemble of multiple preprocessing techniques. *TrAC Trends in Analytical Chemistry*, 132, 116045:1–116045:12.

[32] Fatemeh Nargesian et al. 2019. Data Lake Management: Challenges and Opportunities. *Proceedings of the VLDB Endowment*, 12, 12, 1986–1989.

[33] Fatemeh Nargesian et al. 2020. Organizing Data Lakes for Navigation. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (SIGMOD '20), 1939–1950.

[34] Devin Petersohn et al. 2020. Towards Scalable Dataframe Systems. *Proceedings of the VLDB Endowment*, 13, 12, 2033–2046.

[35] Franck Ravat and Yan Zhao. 2019. Data Lakes: Trends and Perspectives. In *Proceedings of the 30th International Conference on Database and Expert Systems Applications* (DEXA '19), 304–313.

[36] Jennifer Rowley. 2007. The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science*, 33, 2, 163–180.

[37] Pegdwendé Sawadogo and Jérôme Darmont. 2021. On data lake architectures and metadata management. *Journal of Intelligent Information Systems*, 56, 97–120.

[38] Stefan Schmid, Cory Henson, and Tuan Tran. 2019. Using Knowledge Graphs to Search an Enterprise Data Lake. In *Proceedings of the 16th European Semantic Web Conference* (ESWC '19), 262–266.

[39] Ben Sharma. 2018. *Architecting Data Lakes: Data Management Architectures for Advanced Business Use Cases*. Rachel Roumeliotis, editor. (2nd Edition ed.). O'Reilly Media, Bejing *et al.*

[40] Cogan Shimizu, Karl Hammar, and Pascal Hitzler. 2020. Modular Graphical Ontology Engineering Evaluated. In *Proceedings of the 17th European Semantic Web Conference* (ESWC '20), 20–35.

[41] Christoph Stach, Clémentine Gritti, and Bernhard Mitschang. 2020. Bringing Privacy Control Back to Citizens: DISPEL — A Distributed Privacy Management Platform for the Internet of Things. In *Proceedings of the 35th ACM/SIGAPP Symposium On Applied Computing* (PDP '20), 1272–1279.

[42] Christoph Stach, Clémentine Gritti, Dennis Przytarski, and Bernhard Mitschang. 2020. 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* (PerCom '20), 50:1–50:4.

[43] Christoph Stach and Frank Steimle. 2019. Recommender-based Privacy Requirements Elicitation – EPICUREAN: An Approach to Simplify Privacy Settings in IoT Applications with Respect to the GDPR. In *Proceedings of the 34th ACM/SIGAPP Symposium On Applied Computing* (PDP '19), 1500–1507.

[44] Christoph Stach et al. 2020. How to Provide High-Utility Time Series Data in a Privacy-Aware Manner: A VAULT to Manage Time Series Data. *International Journal on Advances in Security*, 13, 3 & 4, 88–108.

[45] Michael Stonebraker and Ugur Cetintemel. 2005. "One Size Fits All": An Idea Whose Time Has Come and Gone. In *Proceedings of the 21st International Conference on Data Engineering* (ICDE '05), 2–11.

[46] Christian Thomsen and Torben Bach Pedersen. 2011. Easy and Effective Parallel Programmable ETL. In *Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP* (DOLAP '11), 37–44.

[47] Wil M. P. van der Aalst. 2014. Data Scientist: The Engineer of the Future. In *Proceedings of the 5th International Conference on Interoperability for Enterprise Systems and Applications* (I-ESA '14), 13–26.

[48] Wei Yun et al. 2021. Knowledge modeling: A survey of processes and techniques. *International Journal of Intelligent Systems*, 36, 4, 1686–1720.

[49] Yi Zhang and Zachary G. Ives. 2020. Finding Related Tables in Data Lakes for Interactive Data Science. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (SIGMOD '20), 1951–1966.

[50] Chaim Zins. 2007. Conceptual approaches for defining data, information, and knowledge. *Journal of the American Society for Information Science and Technology*, 58, 4, 479–493.