Preserving domain knowledge in a data lake

Master Thesis
Master of IT, Software Construction

Mads Staberg Thomsen
msth21@student.aau.dk

Supervisor:
Hamdi Ben Hamadou, PhD
Aalborg University, Denmark

Department of Computer Science
Aalborg University
Selma Lagerlöfs Vej 300, 9220 Aalborg, Denmark

2022-12-02

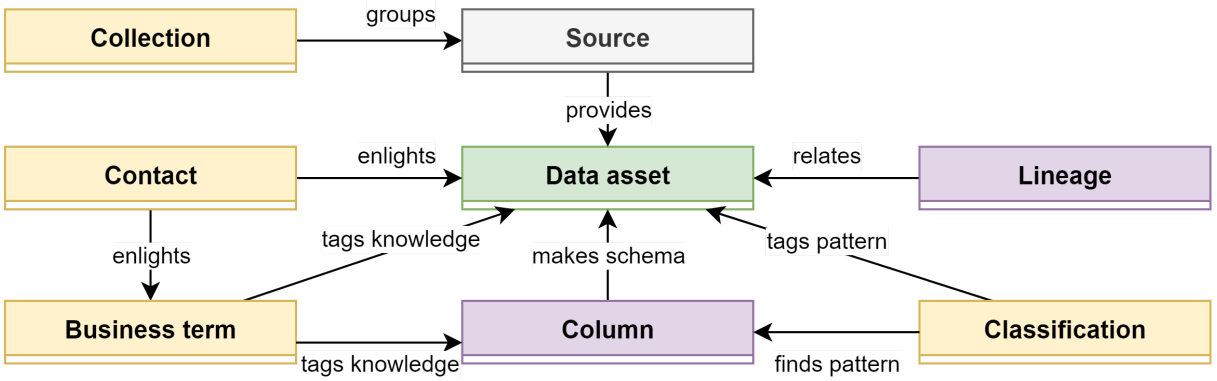


Table of Content

List of figures and tables

Abstract

This thesis is based on real-world challenges for a data-intensive application in a data lake context, e.g. difficulties with understanding the domain and the meaning of the data. It explores how to optimize data discovery by preserving domain knowledge in a time-saving way.

In order to transform domain knowledge into metadata, the thesis contributes with a definition of novel metadata concept. The metadata concept makes the task of implementing a data catalog and building a metadata fundament manageable and accessible. The concept guides the user to navigate through the implementation of metadata in the data catalog and classify the data. The concept is both tool, platform and domain independent.

Sample data and the corresponding metadata is introduced. A data catalog is implemented, which is a collection of metadata, combined with data management and search tools, that helps users to find the data that they need. The catalog serves as an inventory of available data, and provides information of the data quality. A small sandbox environment is set up with some sample data.

A custom analysis model is defined to learn how to handle the challenges. To ensure the analyses are conducted in a reproducible manner, the scenarios are expressed in a Quality Attribute Scenario framework. The scenario template outlines response measures, which are measurable.
Usability and performance qualities are in focus. The usability scenarios cover automatic, manual and semi-automatic infer of metadata into the data catalog. A performance scenario explores how implicit infer of metadata using a structured directory affects the fetching of data from the data lake.

The reader is guided through each analysis with a process map based on the metadata concept and a corresponding logbook. The user actions are marked and explained. The development logbook documents the steps in the analyses. Issues, solutions and learnings are elaborated. A result section at the end of each logbook clarifies the partial results.

All results are combined and grouped by usability and performance scenarios. The results are discussed based on the metadata concept and the problem statement. Learnings from the analyses are highlighted. Based on the usability and performance scenarios, it can be concluded that it is possible to preserve domain knowledge in a time-saving way and thus optimize data discovery. The best usability results are obtained with semi-automatic infer and storage of data in a structured directory provides the best performance.

However, it should be noted that automatic infer in some cases can be a good choice, as it is possible to create searchable technical metadata very quickly. This is a much better starting point than having no information in the data catalog.

Chapter 1 - Motivation

Aarhus Vand is a large water utility company, which handles drinking water in most of Aarhus municipality. The distribution network is divided into zones and monitored by a number of sensors. The sensors are distributed in the network based on domain-specific knowledge. Each sensor measures the water's temperature, flow and pressure every minute, and data is continuously transferred wirelessly to the backend Scada control system. A part of the southern distribution network is illustrated in Figure 1.

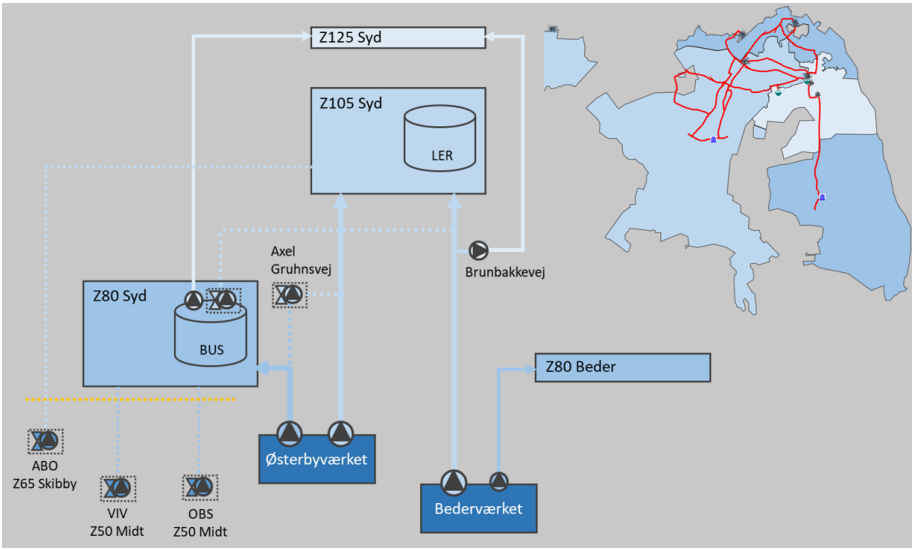


Figure 1*:* Distribution Network

One measurement every minute, 24 hours per day, 365 days per year creates approx 500.000 data rows. Aarhus Vand has more than 4.000 sensors. Every year, this results in billions of data rows being created and must be handled in a big data context. A small part of this data is uploaded to an Azure Data lake, where they are consumed by a third party analysis tool. Every hour approx. 8.100 measurements are uploaded to the data lake. The number of records increases by approx. 6 million measurements every month. The current files in the data lake contain approx. 52 million records. Further details are outlined in Appendix A.

Data is loaded into the data lake in their raw / non-conform state, so it is quick to add new data. Since the sensor measurements are being stored without any corresponding metadata, users find it challenging to inspect, understand and consume the data. Thus, the deep domain knowledge about the distribution network and the measurement's internal relationship is lost and use of the data may require domain expert knowledge. If data was stored in a data warehouse, then the domain experts will interpret and combine data according to the current need in a conforming manner. But it will be a heavy process that will limit future use of data.

Chapter 2 - Problem Statement

The purpose of this project is to investigate how domain knowledge can be preserved in a data lake to improve data discovery. Managing domain knowledge can be a time-consuming process and seems like a difficult task. Therefore, the project intends to answer the following problem statement:

> Can domain knowledge be preserved in a time-saving way into a data lake in order to optimize the data discovery?

The criterias for domain knowledge, metadata and data discovery are determined by a custom analysis model. The problem statement will be elaborated through the hypothesis:

> By constructing a custom analysis model, where automatic, manual, semi-automatic and implicit ingesting of metadata are explored, it can be evaluated whether domain knowledge in the data lake optimizes the data discovery or not.

The current setup of the data lake at Aarhus Vand presents some real-world challenges for the data consumers, eg.:

- Difficult to understand the domain and meaning of the data
- Impossible to know which sensors make sense to compare to get useful information
- Time-consuming to fetch measurements from a specific period or a certain sensor because all data are stored in one file

A metadata concept will help navigating through the exploration. Some metadata and sample data implemented in a sandbox environment will be used as a basis for the analyses, where the challenges are investigated and tested in different scenarios. By performing a series of analyses the aim is to learn more about how the problem can be handled.

2.1 Project Scope

Although the hypothesis, analysis model and analyses will be relevant to many domains, the project will be limited to use only a subset of the data from the drinking water domain at Aarhus Vand. Handling of access control and General Data Protection Regulation (GDPR) issues are not in scope.

Chapter 3 - Terminology

3.1 Data discovery

Data Discovery involves the collection and evaluation of data from various sources and is often used to understand trends and patterns in the data. It requires a progression of steps that organizations can use to understand their data. Data discovery can be treated as a synonym for Knowledge Discovery in Data (KDD) [Han2012]. The KDD process includes collecting data from multiple data sources, cleaning, integration, task-relevant selection, transformation, and performing analysis to gain knowledge and insights into business processes [Nwagu2017].

In this project scope, data discovery includes the workflow of 'find and understand' data, which includes integration of different sources and selection of task-relevant data.

3.2 Data lake

A data lake stores current and historical data for one or more systems for the purpose of analyzing the data. Data may include raw copies of source system data and transformed data used for tasks such as reporting, visualization, advanced analytics and machine learning. Files of different file formats generated from different sources can be stored in a data lake [Hamadou2020]. A data lake can be established on premises within an organization's data centers or be cloud based using cloud services.

The data lake will typically be structured in three zones [Thomsen2022]:

- Landing zone where the raw data enters in different formats including structured, unstructured and binary data
- Work zone where work-in-progress data is stored
- Gold zone with data that has been cleansed and processed

3.3 Data lineage

Data lineage uncovers the life cycle of data and aims to show the complete data flow, from start to finish. Data lineage is the process of understanding, recording, and visualizing data as it flows from data sources to consumption. This includes all transformations the data underwent along the way; where data came from and how the data was transformed [Gorelik2019].

3.4 Domain knowledge

Domain knowledge is knowledge of a specific, specialized discipline or field. People with domain knowledge are often regarded as specialists or experts in their field. Data does not "speak on its own" and attention is needed to also

convey information about the problem domain as well [Videla2021]. In this project scope, domain knowledge will be expressed through metadata.

3.5 Metadata

Metadata is data about the data and provides information about one or more aspects of data, e.g. description, data quality, value ranges, dates for creation and update [Hamadou2020]. Metadata will give the user confidence in the data. Metadata unlocks the value of data by improving that data's usability and findability.

There are different types of metadata [Atlas2022]:

- Technical: schemas, data types, models etc.
- Business: data tags, classifications, mapping to business
- Operational: process output, performance, Extract-Transform-Load jobs

This project explores technical and business metadata. When data is tagged with metadata, it will help the user to become self-serviced in the 'find and understand' data discovery process. The self-service data discovery idea is illustrated in Figure 2.

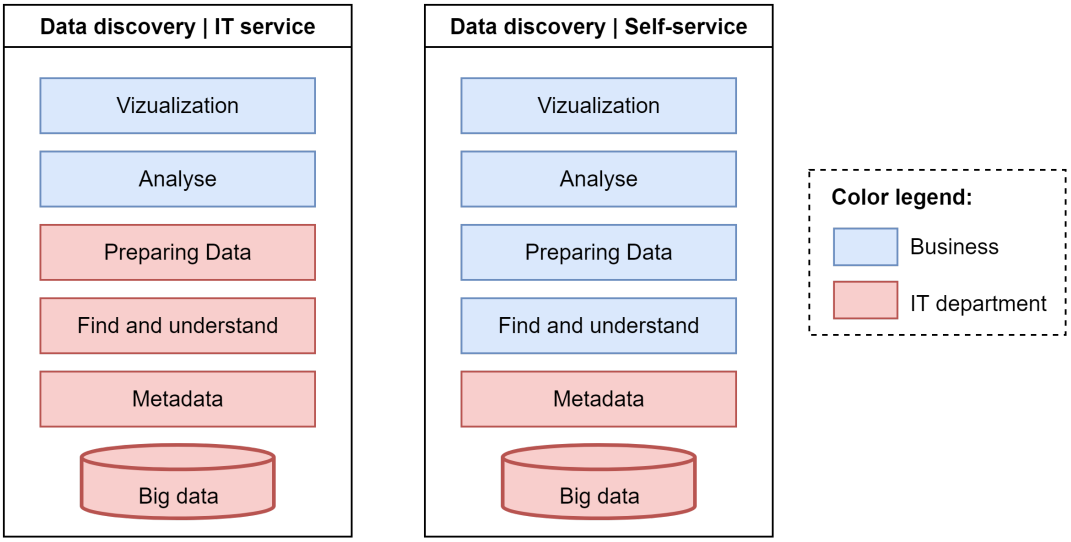


Figure 2: Data discovery | Self-service

Chapter 4 - Methods

This chapter outlines the method for exploring the hypothesis. A metadata concept introduces a conceptual way of interpreting domain knowledge. Some metadata and sample data implemented will be used as a basis for the analyses. The scenarios will be described using a Quality Attribute Scenarios framework.

4.1 Metadata concept

To transform domain knowledge into metadata, this project contributes with the definition of a metadata concept. The concept offers a conceptual way of interpreting domain knowledge into different entities, including metadata types.

- **Asset**: a piece of data information about the data but not the data itself
- **Technical metadata:** column with data type and relation between assets
- **Business metadata**: tags based on classification and business terms
- **Data store**: source containing the asset, e.g. file or database

The entities are included as they precisely describe domain knowledge. The metadata concept is visualized in Figure 3.

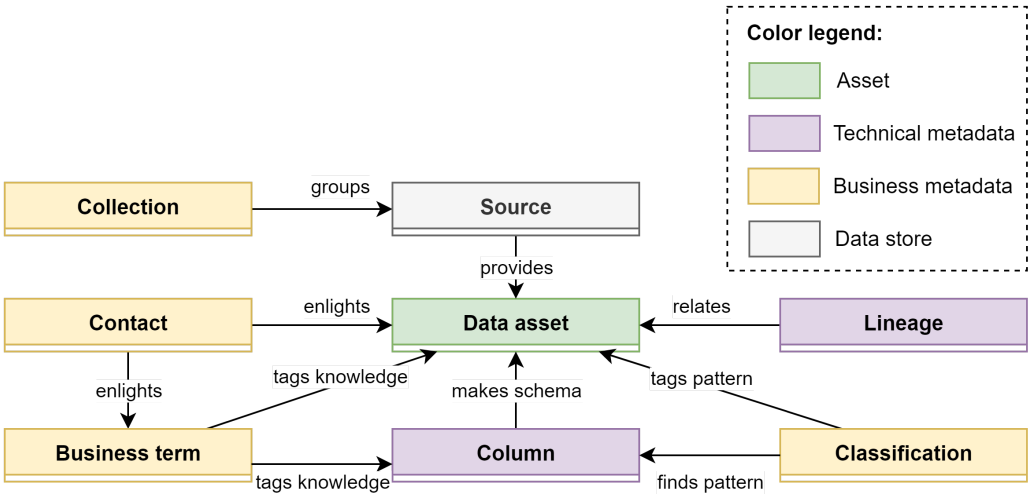


Figure 3: Metadata concept | Entities

Each entity in the metadata concept can be extended with properties that define characteristics of the elements. The reason for the characteristics being chosen is explained in the Table 1.

Type	Entities	Properties | Description
Asset	Data asset	Includes a description, a timestamp for update of the schema and a domain expert validation flag.
Technical	Column	Name, data type and description. Collection of columns makes up the schema. Changes to the schema can be tracked in a schema registry.

Technical	Lineage	Shows the relationship among the data assets.
Business	Classification	Describes patterns to look for as input for tagging.
Business	Business term	Explains domain specific knowledge.
Business	Contact	Person who has ownership of data and an associated role.
Business	Collection	Shows logical boundaries in the organization.
Data store	Source	Contains data in several different formats.

Table 1: Metadata concept | Properties

The metadata concept with entities is outlined as a class UML diagram at the logical level in Figure 4. The metadata concept is tool independent and can be implemented in any data catalog tool. Furthermore, the concept is domain independent.

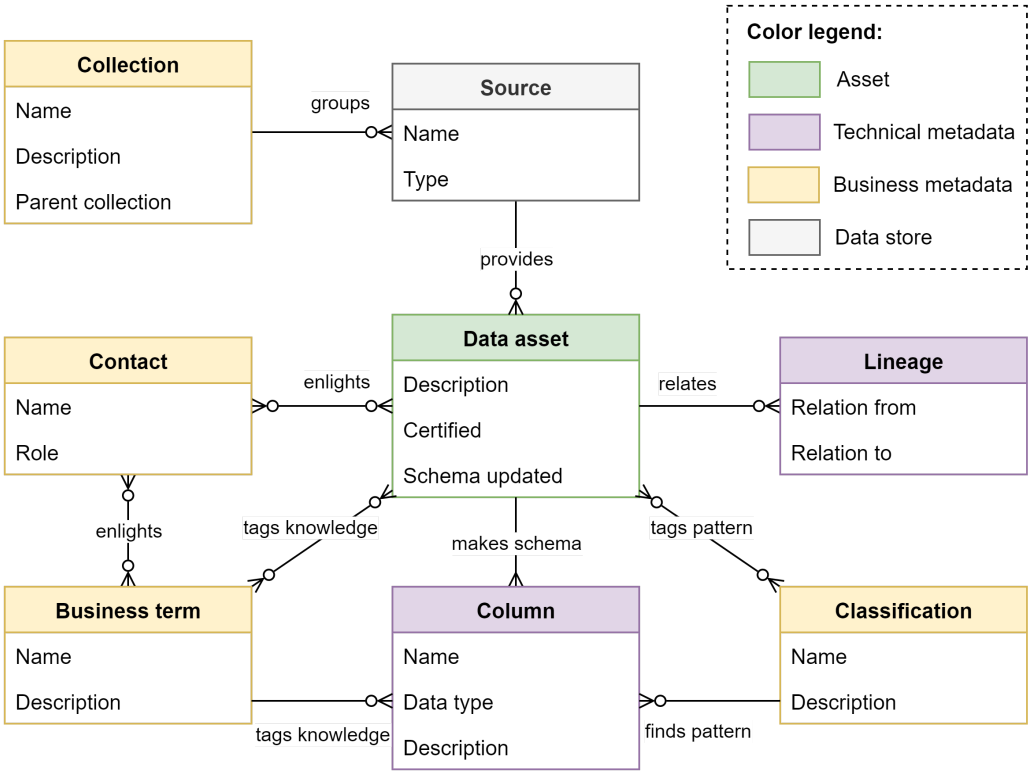


Figure 4: Metadata concept | Logical model

4.2 Sample data

The basis for the analyses are some sample data, which is a subset of the raw data with total measurements as outlined in Appendix A. The subset covers measurements per minute from the period 06.06 - 12.06.2022. Data is divided into three different files:

- Measurement.csv
- SensorConfig.csv

- Sensor.csv

The three files with sample data are uploaded in the project's code repository placed at Azure DevOps as described in Appendix B.

Measurement

The measurement from each sensor with a timestamp. The size of the measurement.csv file is 125 MB and contains approx. 1.450.000 records. The header and the first few records are shown in Figure 5.

Datetime	SensorId	Measure	Updatetime	FileId
2022-06-06 00:00:00.000	ABO.B10_E_W3_BF1-M_FLOW	2.369791746	2022-06-06 01:29:38.000	3
2022-06-06 00:00:00.000	BED.B12_E_B1-M_TRYK	6.62808609	2022-06-06 00:29:53.000	9
2022-06-06 00:00:00.000	BED.B12_E_B2-M_FLOW	219.816482544	2022-06-06 01:29:38.000	3
2022-06-06 00:00:00.000	BED.B12_E_G3_B1-M_TRYK	4.288676739	2022-06-06 00:29:10.000	10
2022-06-06 00:00:00.000	BED.B12_E_G3_BF1-M_FLOW	27.178453445	2022-06-06 11:29:57.000	4
2022-06-06 00:00:00.000	BUS.C04_E_C1_B1-M_NIVEAU	3.723090172	2022-06-06 00:31:42.000	8
2022-06-06 00:00:00.000	BUS.C04_E_G1_BF1-M_FLOW	18.900817871	2022-06-06 11:29:57.000	4
2022-06-06 00:00:00.000	BUS.C04_E_G1_BF1-M_FLOW	18.900817871	2022-06-06 01:30:15.000	2
2022-06-06 00:00:00.000	BUS.C04_E_G1_BP1-M_TRYK	4.951388836	2022-06-06 00:31:42.000	8

Figure 5: Sample data | Measurement.csv

Sensor

Metadata about each sensor including parameter, unit and zone according to Figure 6. File size is 10 KB and contains 93 records.

SensorId	Description	Parameter	Unit	Media	Time se	Zone	Location
OST.B09_E_G1_BF1-M_FLOW	Flow Østerby til Z80S	Q	m3/h	Drikkevand	Sensor	Z80Syd	Østerbyværket
BUS.C04_E_G1_BF1-M_FLOW	Flow fra Bushøj til Z125 Syd	Q	m3/h	Drikkevand	Sensor	Z80Syd	Beholder Bushøj
BUS.C04_E_G2_BF2-M_FLOW	Flow fra Bushøj til Z105 Syd	Q	m3/h	Drikkevand	Sensor	Z80Syd	Beholder Bushøj
BUS.C04_E_W1_BF1-M_FLOW	Flow fra Bushøj til Z80S	Q	m3/h	Drikkevand	Sensor	Z80Syd	Beholder Bushøj
VIV.B08_E_G1_BF1-M_FLOW	Flow udveksling Z50M til Z80S (viv)	Q	m3/h	Drikkevand	Sensor	Z80Syd	Beholder Vibyværket
OBS.C11_E_G1_BF1-M_FLOW	Flow udveksling Z50M til Z80S (obs)	Q	m3/h	Drikkevand	Sensor	Z80Syd	Beholder Observatoriet
DIS.D01_H_B84_BF1-M_FLOW	DB84 Flow S80 Grøndalsvej	Q	m3/h	Drikkevand	Sensor	Z80Syd	DB84: Skanderborgvej 226

Figure 6: Sample data | Sensor.csv

SensorConfig

A sensor configuration file containing a few metadata about the different zones and measurement types as displayed in Figure 7. File size is 1 KB and contains 11 records.

FileId	FileName	DecimalSeparator	ColumnDelimiter
2	Flows_Z80Syd.csv	,	;
3	Flows_Z105Syd.csv	,	;
4	Flows_Z125Syd_Z80Beder.csv	,	;
5	Temperatur_Z80Syd.csv	,	;
6	Temperatur_Z105Syd.csv	,	;
7	Temperatur_Z125Syd_Z80Beder.csv	,	;
8	Tryk_Z80Syd.csv	,	;
9	Tryk_Z105Syd.csv	,	;
10	Tryk_Z125Syd_Z80Beder.csv	,	;
11	FlowTrykTemperatur_Z125N.csv	,	;

Figure 7: Sample data | Sensorconfig.csv

The internal relationship between the files can be described at logical level in a class diagram with Unified Modeling Language (UML) notation. The sample data are .csv format thus without data type, but the types are stated in the class diagram in Figure 8.

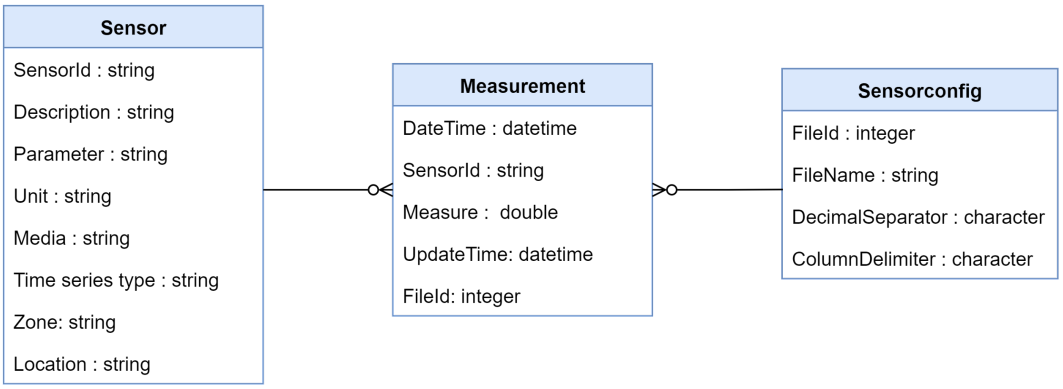


Figure 8: Data model | Sample data

An interview with the domain expert unlocked knowledge hidden in the SensorId, as this uses a unique template. Business specific information can be extracted from the SensorId field. This is shown in Figure 9 where the extracted metadata based on the template is indicated below the dashed line in the 'Sensor' class.

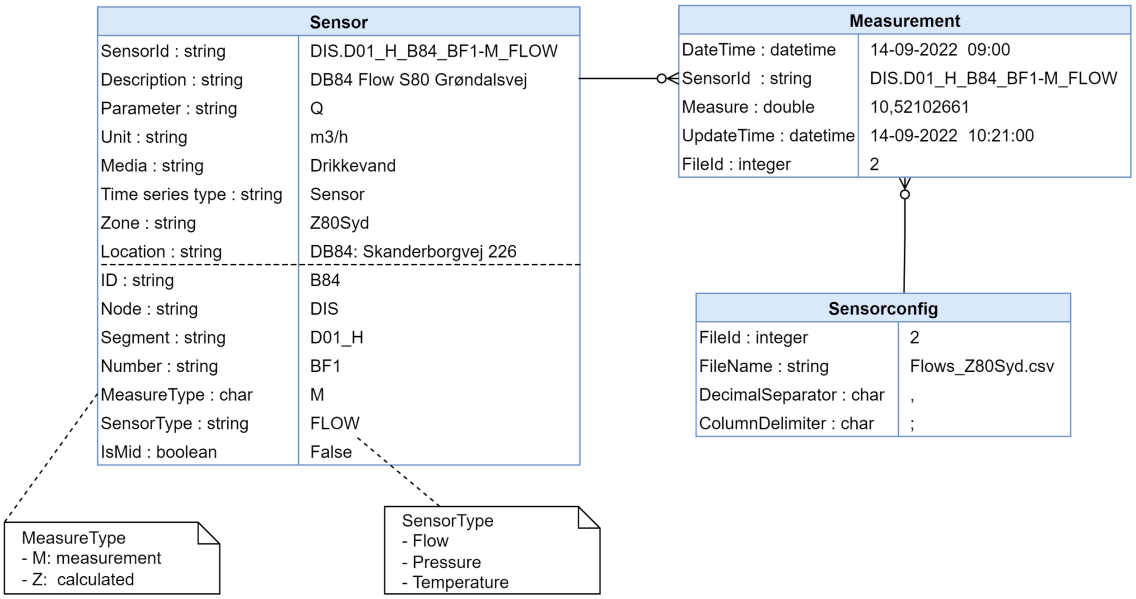


Figure 9: Data model | Sample data with metadata

The interview with the domain expert also revealed some aspects of the network and measurements. This extended domain knowledge is reproduced in Table 2.

Subject	Domain expert knowledge
Flow	Measurements with false 'IsMid' values have more noise as it is a snapshot of a given minute
Pressure	Values can be used to detect pressure shocks and extend lifetime
Temperature	Depending on the distance to the surface
Context	The fluctuations over the day have morning and evening peaks
	Increased flow creates increased pressure
	Increased flow does not immediately result in lower temperature

Table 2: Metadata | Domain expert knowledge

To get an idea of how to transform the domain knowledge and the metadata from the sample data, the classes in the metadata concept logical model can be filled with an example as shown in Figure 10.

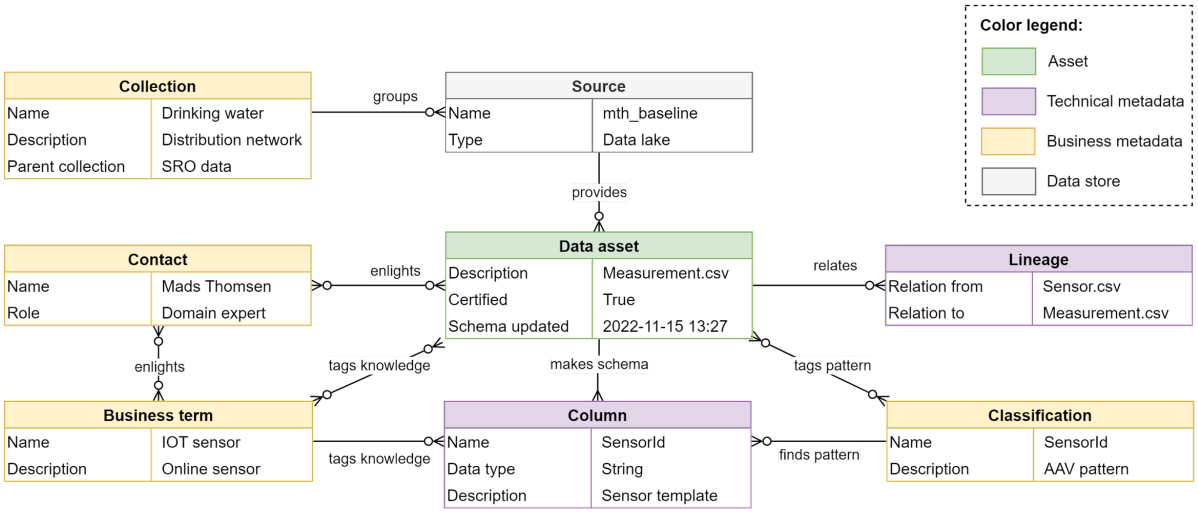


Figure 10: Metadata concept | Logical model with sample data

4.3 Quality Attribute Scenarios

To ensure the analyses are conducted in a reproducible manner, the scenarios are expressed in Quality Attribute Scenarios [Bass2012, p. 196].

> Quality Attribute Scenarios are a way of setting a testable frame around a specific quality the system should have, and it provides a structured approach for the evaluation of requirements.

To measure software we need to define the qualities we measure and some kind of metric. The framework handles quality attributes, metrics, measurements and uses a measurement template. The template captures requirements and measurements in a common format. The composition of a Quality Attribute Scenario (QAS) is illustrated in Figure 11.

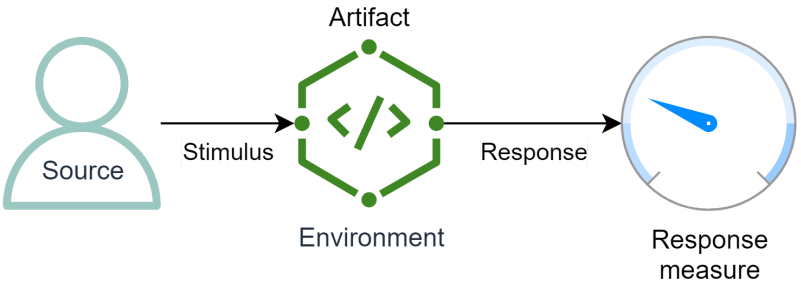


Figure 11: Quality Attribute Scenario | Template

Key points of the QAS template [Christensen2020]:

- a ***source*** generates some ***stimulus***
- arrives as some ***artifact*** in an ***environment***
- must be dealt with a ***response*** with a satisfactory ***response measure***

The custom analysis model constructed in chapter 6 will elaborate Usability and Performance quality attributes in different scenarios.

Chapter 5 - Sandbox environment

The following sections describe the implementation of a data catalog and data lake in the sandbox environment. The setup is documented in the project's code repository located on Azure DevOps and available in Appendix B.

5.1 Data catalog

A data catalog is a collection of metadata, combined with data management and search tools, that helps users to find the data that they need. The catalog serves as an inventory of available data, and provides information of the data quality. Searching a data catalog is a great tool for data discovery, especially for the workflow of 'find and understand' data as illustrated in Figure 12.

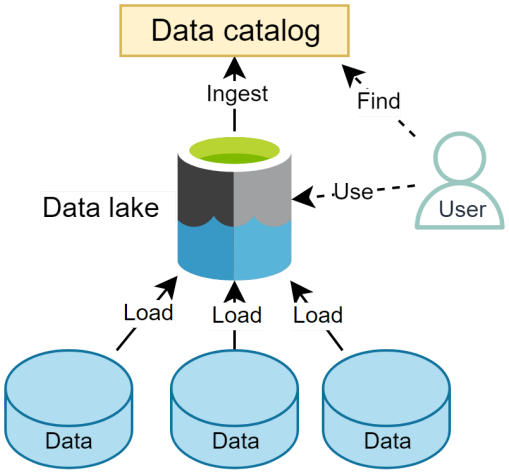


Figure 12: Data catalog | Exploring data

The data catalog in the sandbox is using the Azure Cloud platform, since Aarhus Vand already uses Azure Cloud. However, it is important to highlight that the metadata concept is tool independent and can be implemented in any data catalog tool. There are several cloud providers, e.g. Amazon Web Services, Google Cloud Services and Microsoft Azure Cloud. Exploration of the problem is independent of the cloud provider.

In Azure the data catalog is deployed as a part of Microsoft Purview, which is a unified data governance service. Purview Data Catalog provides a system for registration of enterprise data sources and a discovery service for searching data assets. Purview is based on Apache Atlas, an open source metadata management and data governance tool [Purview2022].

A collection of entities from the metadata concept introduced in the last chapter can be presented in a data catalog as shown in Figure 13.

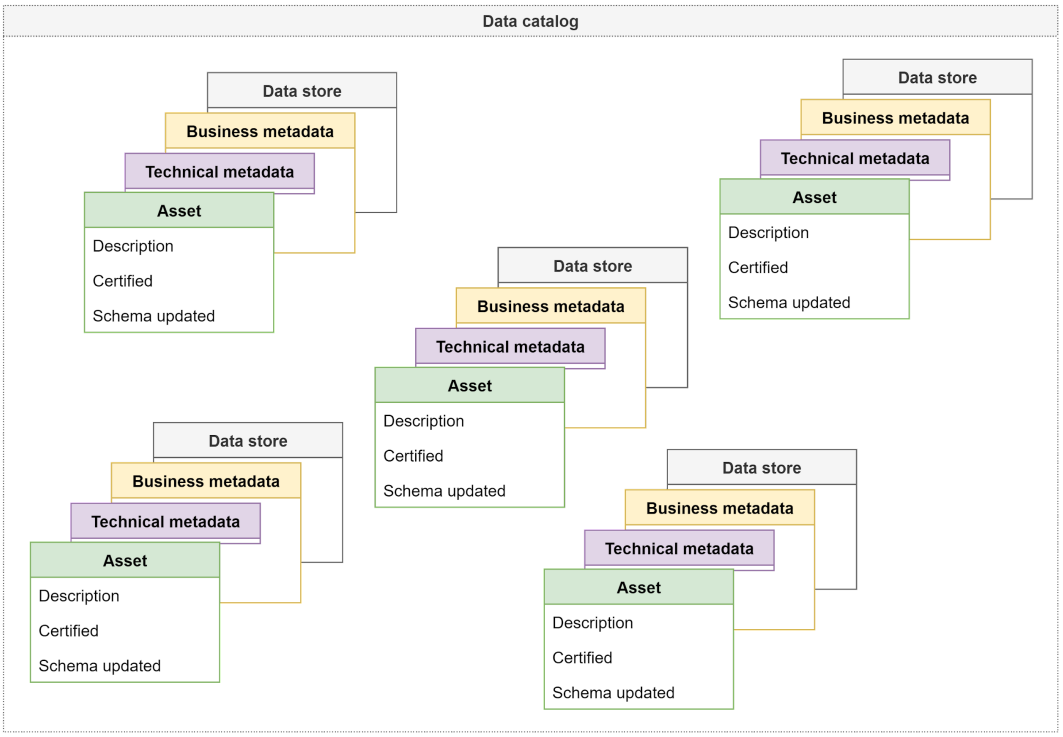


Figure 13: Data catalog | Metadata concept

The data catalog is accessible through a web viewer and the data assets can be found on a data map as shown in Figure 14, where relevant domains are added as collections. The sample data belongs to the collection hierarchy Thesis Purview / SRO data / Drinking water.

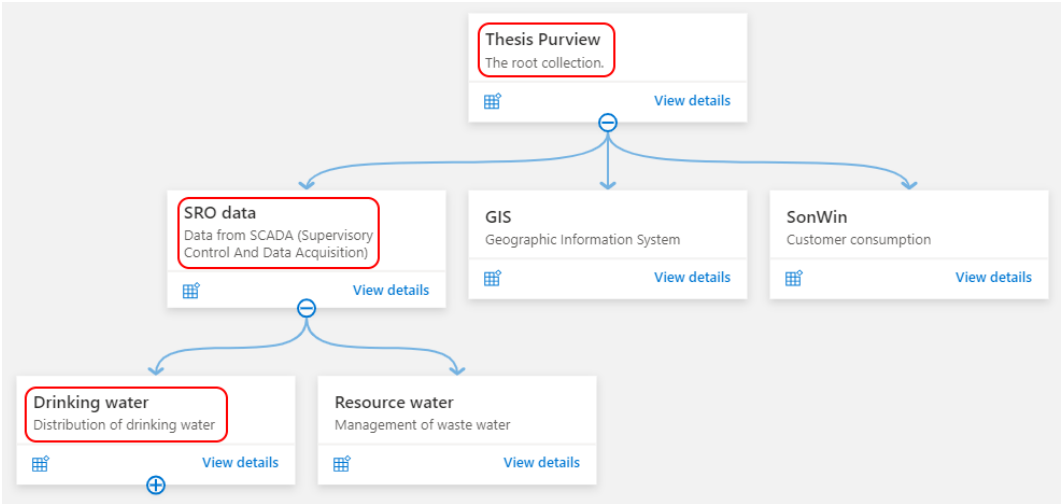


Figure 14: Data catalog | Data map

Data assets are ingested during scans and populated in the data catalog. Some of the processes are performed automatically and others processes require manual

interaction to enrich the metadata. Both types of processes are covered in the analysis model as outlined in the next chapter.

5.2 Data lake

To make the architecture reproducible, the data lake is implemented via a cloud service rather than in an on premises data center. The sample data described are ingested. Focus is on the landing zone in the data lake architecture as shown in Figure 15.

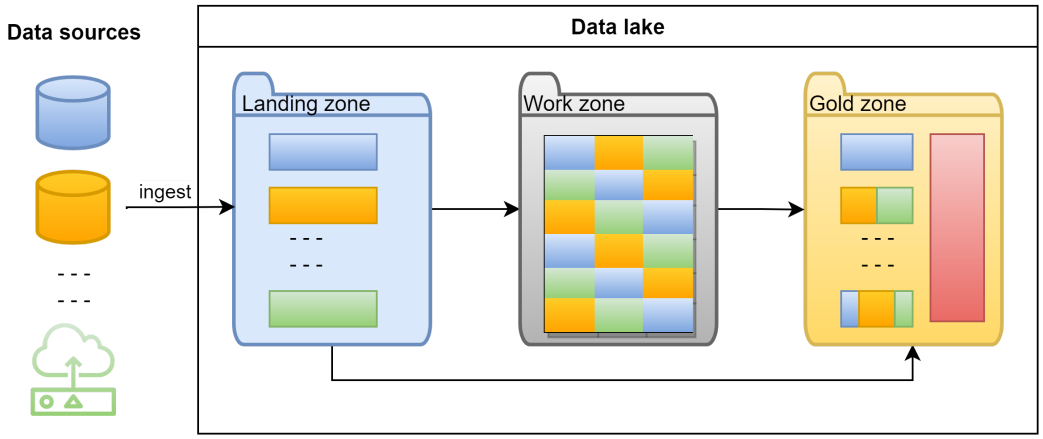


Figure 15: Data lake | Zones

In Azure a data lake is implemented as a standard storage account resource with file blob storage. Content of structured and unstructured data is organized in blob containers, where each container represents a zone [Azure2022].

A data lake storage account is implemented in the sandbox environment as displayed in Figure 16. The data lake acts as a baseline with a landing zone including sample data from the distribution network.

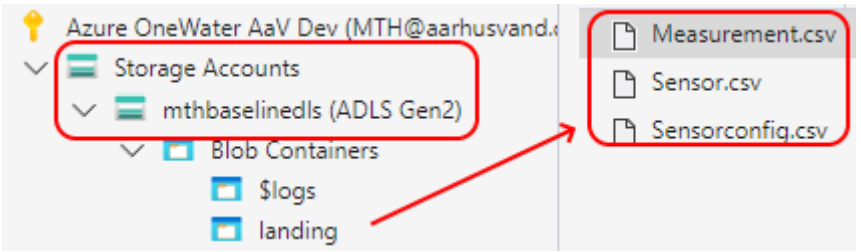


Figure 16: Data lake | Sandbox storage accounts

Chapter 6 - Analysis Model

The analysis model used to evaluate the hypothesis is described in this chapter. The goal is to test if domain knowledge can be preserved in a time-saving way into a data lake in order to optimize the user's data discovery. The model consists of a number of Quality Attribute Scenarios (QAS), which each has a description of why it is relevant and an important architectural consideration.

6.1 Usability

Usability is important when the user searches the data catalog, as it should be easy to understand the data context, relationships and track the data lineage. The response to an usability quality attribute is intended to give the user appropriate feedback or assistance. In Software Architecture in Practice, Bass [Bass2012, p. 175] describes usability as:

> Usability is concerned with how easy it is for the user to accomplish a desired task and the kind of user support the system provides. Over the years, a focus on usability has shown itself to be one of the cheapest and easiest ways to improve a system’s quality.

Metadata can be inferred automatically into the data catalog during a scan. The scanning process establishes a connection to the data source and captures data assets with technical metadata like names, file size and columns. The scan can also apply classification tags on data assets.

However, only a subset of domain knowledge will be captured and ingested. The remaining part of domain knowledge must be added manually to obtain a full set of metadata. Business terms are never inferred automatically, as they are unique to the organization. The following usability scenarios deal with automatic and manual ingestion of metadata.

6.1.1 Automatic infer

This scenario explores how large a subset of domain knowledge can be derived automatically. The data catalog will be used out-of-the box with a minimum of human intervention. The process of automatic infer metadata by running a scan in the data catalog tool is explored in the first usability Quality Attribute Scenario as described in Table 3.

Scenario:		As a data manager, I want to be able to ingest metadata automatically into the data catalog by running a scan.
Quality Attributes:		Usability
Scenario Parts	**Source:**	Data manager
	Stimulus:	Running a scan from the data map.
	Artifact:	Baseline data lake and the data catalog.
	Environment:	Runtime
	Response:	The scan completes and data assets are ingested.
	Response Measure:	All three files with sample data are ingested as data assets.

Table 3: Quality Attribute Scenario | Usability | Automatic infer

6.1.2 Manual infer

Not all metadata can be inferred automatically in the data catalog. Manually creating metadata allows for more detail and it is possible to add more domain knowledge manually into the data catalog. Table 4 outlines how to do this in data catalog tool and is the second usability Quality Attribute Scenario. The effort has been time-limited to reflect a real-world situation with a business demand.

Scenario:		As a data manager, I want to manually enrich metadata with domain knowledge including business terms and classification tags on assets.
Quality Attributes:		Usability
Scenario Parts	**Source:**	Data manager
	Stimulus:	Provide information about business terms, classification and relationship between data assets.
	Artifact:	Baseline data lake and the data catalog.
	Environment:	Runtime
	Response:	Data assets are updated with data types, classification, business terms and lineage. Domain knowledge is searchable in the data catalog.
	Response Measure:	The work is completed within 90 minutes of work.

Table 4: Quality Attribute Scenario | Usability | Manual infer

6.1.3 Semi-automatic infer

Obtaining a full set of domain knowledge captured into the data catalog can be very time-consuming. Finding a time-saving way to populate metadata will be explored in the third Quality Attribute Scenario as shown in Table 5. The effort has been time-limited to reflect a real-world situation with a business demand.

Scenario:		As a data manager, I want to import business terms in a time-saving way and tag with custom classification automatically.
Quality Attributes:		Usability
Scenario Parts	**Source:**	Data manager
	Stimulus:	Import business terms via interface and define custom classification rules. Run a scan and edit.
	Artifact:	Baseline data lake and the data catalog.
	Environment:	Runtime
	Response:	Business terms are imported and the scan will tag data assets with custom clarification.
	Response Measure:	The work is completed within 60 minutes of work.

Table 5: Quality Attribute Scenario | Usability | Semi-automatic infer

6.2 Performance

Performance is the ability to meet timing requirements for an event. The response to a performance quality attribute is intended to be generated within a time constraint as described by Bass [Bass2012, p. 135]:

> The goal of performance tactics is to generate a response to an event arriving at the system within some time-based constraint. The event can be single or a stream and is the trigger to perform computation.

The sample data contains time series data and one of the biggest challenges is that data grows over time. One measurement every minute, 24 hours per day, 365 days per year creates approx 500.000 data rows! Big data is a challenge for relational database management systems (RDBMS). Because of the relations between tables, a selection of data may require a time-consuming table join.

Big data is often generated by the time dimension, but the relation model ignores the order of rows in a table. It is easier to get data in than out of a RDBMS, as queries do not scale linearly with data size[Jacobs2009]. In the baseline data lake, measurements are stored in one big file. Even though there are no relations to other files, extracting data will become increasingly time-consuming.

Structuring a directory in the landing zone might improve performance. The directory can be divided into a hierarchy of subfolders with year / month / day.

6.2.1 Implicit infer

The performance scenario investigates how long time it takes to fetch data from the structured directory compared to the landing zone as shown in Table 6. The scenario contains two test periods, both of which include a specific sensorId and a time span. Each test period is run five times and the average execution time is calculated as a response measure, The measure is compared between structured directory and unstructured landing zone. 25% faster is chosen as a realistic goal.

Scenario:		A user fetches data from the landing zone and the structured directory in these periods: #1: 09.06 06:15 - 06:29 #2: 08.06 06:15 - 06:29 + 09.06 06:15 - 06:29 Only sensorId 'DIS.D01_H_B26_BP1-M_TRYK'
Quality Attributes:		Performance
Scenario Parts	**Source:**	Data analyst
	Stimulus:	The user fetches the data using Python language into an Apache Spark notebook.
	Artifact:	Baseline data lake with a structured directory
	Environment:	Normal mode
	Response:	All data from the period #1 and #2 and the sensorId are retrieved.
	Response Measure:	Average response time is at least 25% faster from the structured directory compared to the landing zone in both periods.

Table 6: Quality Attribute Scenario | Performance | Implicit infer

The structuring strategy based on time is about metadata, as the naming of folders implicitly becomes meaningful as displayed in Figure 17.

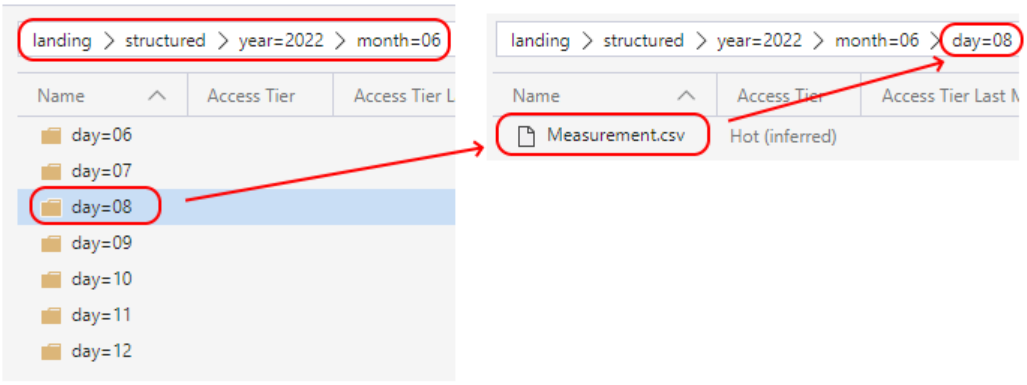


Figure 17: Data lake | Structured directory

Chapter 7 - Analyses and Results

Based on the sample data and scenarios outlined in the previous chapters, the analyses in this chapter will explore how to deal with the challenges of preserving domain knowledge in a data lake. The analyses will focus on whether or not it is possible to optimize the data discovery.

To document the process, a simple template is set up in each section with a development logbook that contains steps, issues, solutions, learnings and results for each scenario. The purpose of the logbooks is to document each step in the Quality Attribute Scenarios and make them reproducible. As a guide for navigating through the scenarios, each analysis section contains a process map showing the various steps in the infer process. The maps are based on the metadata concept and user actions are marked as yellow events.

7.1 Automatic infer

This analysis explores the scenario described in the Quality Attribute Scenario | Automatic infer. The baseline data lake in the sandbox environment is the foundation for the work. The goal is to have metadata automatically registered as data assets, enabling data discovery by searching the catalog. As a starting point, the data catalog is empty and contains no data assets. The automatic infer user actions are illustrated in Figure 18 and includes one event (yellow box):

1. Scan the baseline data lake

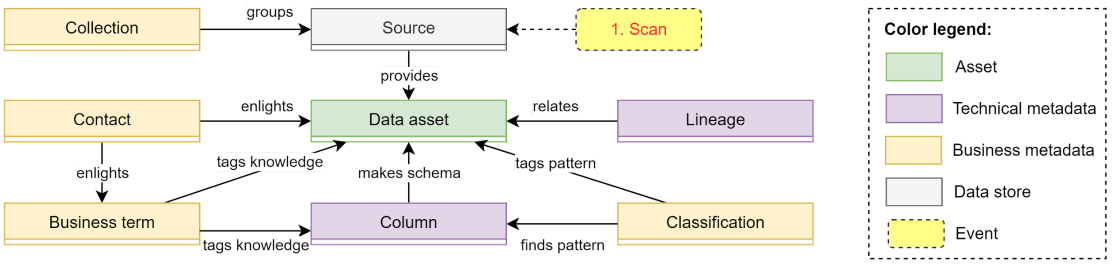


Figure 18: User actions | Automatic infer

7.1.1 Logbook

QAS usability | Automatic infer
Using the baseline data lake, the data manager ingests metadata automatically into the data catalog. All three files with sample data are ingested as data assets.
Development Logbook

Date	Subject
2022-11-05	Register baseline data lake as a source to ‘Drinking Water’ collection. Setup scan connector, use default scan rule set and run the scan. Three assets candidates for classification and schema are extracted. Search the data catalog for ‘sensorId’ and get two results.

Usability Issues	
Issue	**Solutions and learnings**
Schema insufficient	The schema inferred automatically is not sufficient. Only column names, but not descriptive tags, are derived. All data types are set to strings, since the source files are csv. This must be handled manually.
Metadata types	The automatic out-of-the-box scanning only discovers technical metadata. There is a lack of business metadata, which has to be provided manually.
Classification missing	The default scan rule set includes all supported file types for schema extraction and classification. The system classifications rules look for passport and credit card numbers, date of birth, email addresses and passwords and so on in order to tag the data assets. However, the sample data in the baseline data lake contains domain specific data, hence none of the data sets are classified. Custom classifications rules need to be created.
Schema registry	The schema is automatically inferred. If the schema is changed before the next scan, the column names of the data asset will be overridden without any warning (see Appendix C). It is actually possible to be alerted during ingesting via Azure event hub and messaging services to be aware about the change [AzureEvent2022]. To be able to use messaging in the data catalog, event hubs have to be enabled in the settings. This is not investigated further as it is out of project scope.
Relation not clear	The three data assets are marked as related, but it is not stated which properties link the files together. The data manager must discover the lineage himself.

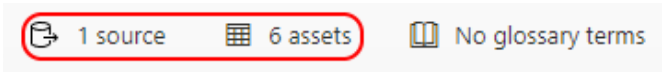
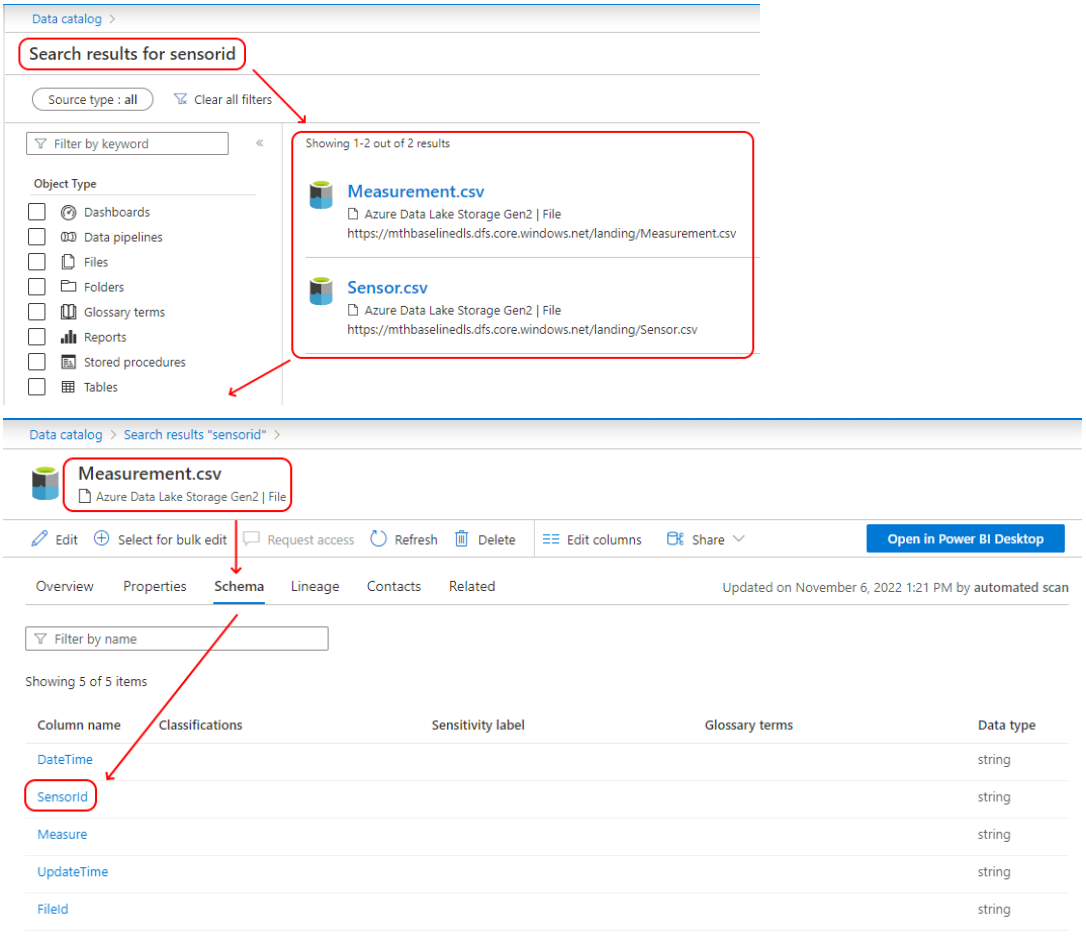
Screenshots
Additional screenshots are added in Appendix C. Starting point:    After initial scan:    Data discovery by searching the data catalog for 'sensorId': 
Results
The automatic scan is capable of ingesting all data assets. But the inferred schemas are not efficient and no business metadata tags are obtained. The subset of domain knowledge preserved in the data lake is small. However the user action is conducted fast and the domain knowledge can be discovered in the data catalog.

Table 7: Logbook | Usability | Automatic infer

7.2 Manual infer

The previous Quality Attribute Scenario pinpointed that the automatic out-of-the-box scanning only creates a small subset of domain knowledge with some missing elements such as:

- column data types not specified
- no classification tags of patterns
- business metadata missing
- lineage with relations between files not explicit

This analysis explores how to ingest the missing elements as described in the Quality Attribute Scenario | Manual infer. The baseline data lake in the sandbox environment is the foundation for the work. As a starting point, the data catalog is populated with three data assets from the automatic scan.

The user actions are illustrated in Figure 19 and includes the following events (yellow boxes):

1. Create business terms in the business glossary.
2. Create custom classification with description.
3. Edit data assets to update descriptions, tags and certified labels.

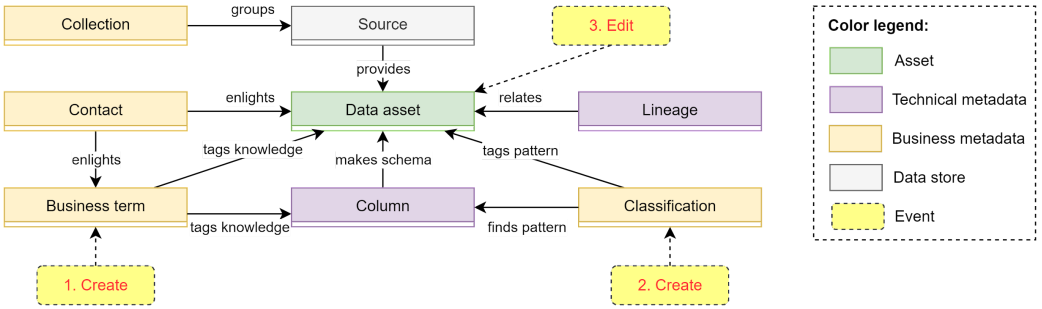
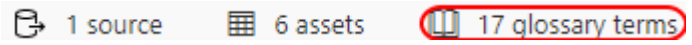
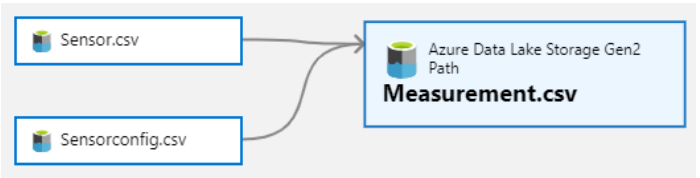


Figure 19: User actions | Manual infer

7.2.1 Logbook

QAS usability | Manual infer	
The data manager enriches metadata manually with domain knowledge and updates data assets with data types, classification, business terms and lineage. Domain knowledge is searchable in the data catalog. The work should be completed within 90 minutes of work.	
Development Logbook	
Date	**Subject**
2022-11-07	Create business glossary terms in the data catalog.

	Create custom classifications for columns in data assets. Relate business glossary to assets, change data type, add lineage and finally label enriched data assets as 'Certified'. Search the data catalog for 'sensorId' and get two results. Time consumption: 125 minutes
Usability Issues	
Issue	**Solutions and learnings**
Glossary taxonomy	Defining business terms cannot be inferred from data assets. It is time consuming to enter this information. Metadata extraction from Figure 9 and domain expert knowledge in Table 2 is captured as 'Sensor ID template' with related terms (see Appendix D).
Edit data assets	Tagging fields is done easily by picking glossary terms from a dropdown menu. Data type is a free text field, which may result in incorrect values. By adding lineage the relation between files becomes explicit.
Classification	After creating custom classification they are manually added to data asset fields in the same way as glossary terms.
Enable trust for the user	When data asset information is validated, it can be labeled with 'Certified' and regarded as reliable.
Data discovery	Business glossary terms and custom classification can be filtered by when searching the data catalog.
Screenshots	
Additional screenshots are added in Appendix D. <u>After creating business glossary terms:</u>  <u>Lineage:</u> 	

Custom classification:

Display name	Formal name	Description
Aarhus Vand	Aarhus Vand	Assest owned by Aarhus Vand
Configuration	Configuration	Metadata for consuming a time serie file
Delimiter	Delimiter	Column delimiter
Description	Description	Explanation
File name	File name	Name of file
FileID	FileID	Foreign key to file
Location	Location	Geographical directions
Measure	Measure	Value from time serie recorded on created datetime
Media	Media	Utility type: Drinking water | Rain water | Resource water
Parameter	Parameter	Measure parameter
SensorID	SensorID	Unique sensor template: ID | Node | Segment | Number | Measure type | Sensor type | IsMid.
Sensors	Sensors	Wireless Internet of Things (IOT) sensor
Separator	Separator	Separator between measurement values
Time serie	Time serie	Time serie data from wireless Internet of Things (IOT) sensor
Time serie type	Time serie type	Source of time serie
Timestamp	Timestamp	Timestamp with date and time
Unit	Unit	Mathematical unit
Zone	Zone	Section in the distribution network

Results

All metadata are added by manual processes, including data types, tagging, business metadata and relation between files. The metadata from the data model of the sample data is also applied and even the domain expert knowledge in a searchable way. In fact, all domain knowledge is preserved in the data lake in a searchable way . But the process is very time consuming and infeasible in a real-world scenario with several data assets. The time limit of 90 minutes of work was not met, as the scenario took 125 minutes to conduct.

Table 8: Logbook | Usability | Manual infer

7.3 Semi-automatic infer

This analysis explores the scenario described in the Quality Attribute Scenario | Semi-automatic infer. The analysis will explore how to properly define business terms, so it can be imported in a time-saving manner.
The user actions are illustrated in Figure 20 and includes the following events (yellow boxes):

1. Import business terms in the business glossary using a template
2. Create custom classification with a pattern defined as regular expression.
3. Add scan rule using the created custom classifications and run scan
4. Edit data assets to update description, check tags and set certified labels.

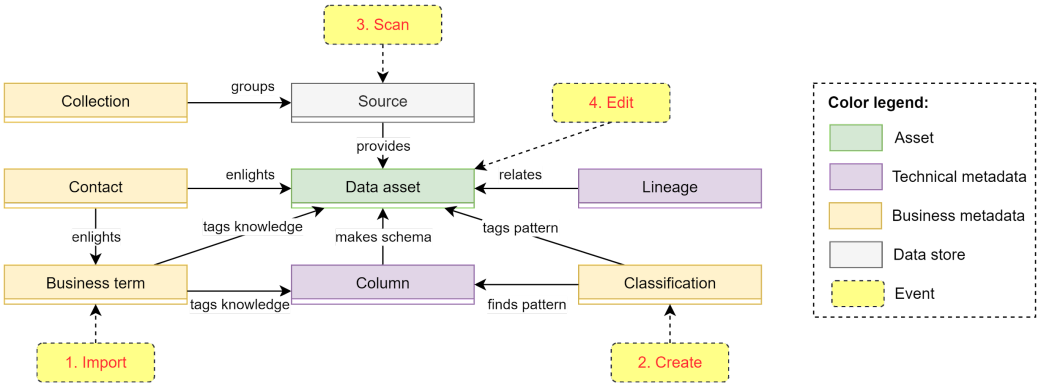


Figure 20: User actions | Semi-automatic infer

7.3.1 Logbook

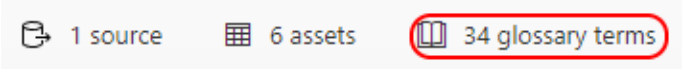
QAS usability | Semi-automatic infer	
The data manager imports business terms in a time-saving way. After defining custom classification rules, the tagging is done automatically by running a scan. The work should be completed within 60 minutes of work.	
Development Logbook	
Date	**Subject**
2022-11-15	Fill out template and bulk import business terms in the catalog. Create custom classification rules and define data patterns with a regular expression or column patterns by name. Add scan rule set using the new custom classification rules. New scan with custom scan rule set. Edit data assets to update description, terms and certified labels Time consumption: 85 minutes
Usability Issues	
Issue	**Solutions and learnings**
Business terms import	Import can be done using either REST API endpoints for create, read, update or delete actions or a term template in .csv format [PurviewAPI2022]. In this scenario, the .csv template is used. After a few trials with validation errors, it was very quick to import the total glossary. The import template is added to the code repository placed at Azure DevOps as described in Appendix B.

Classification rules	Defining data patterns with a regular expression to match the data stored in a data field takes a little practice. Using https://regex101.com (online regex builder), a regex for the unique sensor ID template is created. But for the untrained it takes time to build a solid regex pattern. Classification rules can also label sensitive data in order to restrict access. However, this is not in scope of this project.
Scan with new rule set	A new scan with the custom scan rule set attached should tag the data assets with classifications according to the classification rule - the first few times without result! After refactoring the classification rules and using the 'test classification rule' option, the assets were tagged. This is much more time-saving than re-running a scan.
Edit	Need to do tagging of imported business terms manually.

Screenshots

Additional screenshots are added in Appendix E.

After creating business glossary terms:



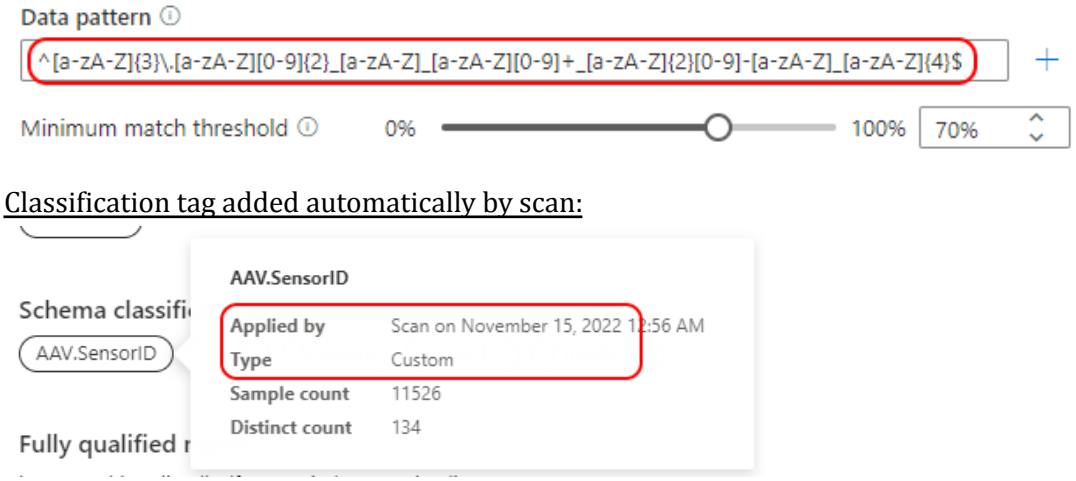
Template for importing business terms:

Name	Definition	Status	Parent Term Name	Term T	[Attribute
_Created date time	Created date time	Draft		Thesis;	Import
_Metro maps	Function map showing sensor locations	Draft		Thesis;	Import
_IOT sensor measurement	Measurement from IOT sensor	Approved		Thesis;	Import
_Sensor ID template_Segment	Subset of Scada network	Approved	_Sensor ID template	Thesis;	Import
_IOT sensor	Wireless Internet of Things (IOT) sensor	Approved		Thesis;	Import
_Sensor ID template_Number	Consecutive numbering of sensors in specific segment	Draft	_Sensor ID template	Thesis;	Import
_Sensor value	Sensor measurement	Draft		Thesis;	Import
_File reference	Relation to file	Draft		Thesis;	Import
_Sensor ID template_Node	Scada netvwork	Approved	_Sensor ID template	Thesis;	Import
_Utility type	Types: Drinking water - Rain water	Draft		Thesis;	Import
_Config file	Settings for using a measurement	Approved		Thesis;	Import
_Sensor ID template_Measure type	Posible values: M: measurement Z: calcula	Approved	_Sensor ID template	Thesis;	Import
_Sensor ID template_IsMid	Boolean value: True: minute average of approx. 2	Approved	_Sensor ID template	Thesis;	Import
_Uploaded date time	Timestamp vwith date and time	Draft		Thesis;	Import
_Sensor ID template_Sensor type	Posible values: - Flow - Pressure - Tem	Approved	_Sensor ID template	Thesis;	Import
_Sensor ID template	Unique sensor ID using template with information about	Approved		Thesis;	Import
_Sensor ID template_ID	Only ID of the Seonsor ID template	Approved	_Sensor ID template	Thesis;	Import

Import validation:

Column	Guidance	Issue to resolve
[Attribute][Thesis]Source *	The field is required. Select a value from: I...	This required column is missing from the template, please modify the file and try again.

Custom classification rule | Pattern of SensorId e.g. OST.B09_E_G1_BF1-M_FLOW:



Classification tag added automatically by scan:

Results

The scan results in tagging the columns with classifications. Importing the business terms makes it fast to tag data assets with business knowledge. The time limit of 60 minutes of work was not met, as the scenario took 85 minutes to conduct. The time-consuming part was creating regular expressions. One way to make the process faster is to use column patterns with names, but it is not as robust as it does not look at the actual data content. Over time, the regular expressions will be reusable for the users, thus skipping the overhead.

Table 9: Logbook | Usability | Semi-automatic infer

7.4 Implicit infer

This analysis explores the scenario described in the Quality Attribute Scenario | Implicit infer. A structured directory divided into a hierarchy of subfolders with year / month / day is added to the baseline data lake. The user fetches the data using Python programming language into an Apache Spark notebook running on Azure. The cluster attached to the notebook has 14 GB memory and 4 cores. The user actions are illustrated in Figure 21 and includes one event (yellow box):

1. Fetch data from source into an Apache Spark notebook

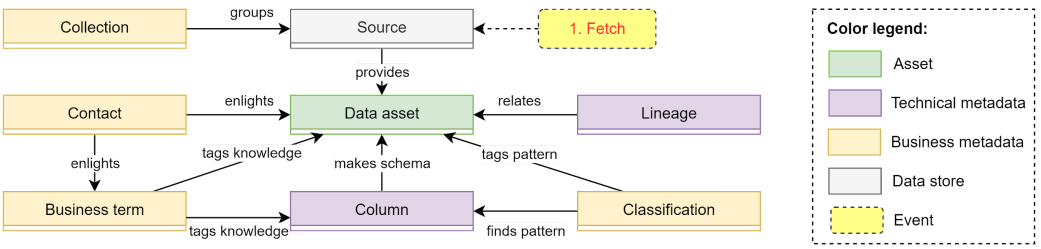
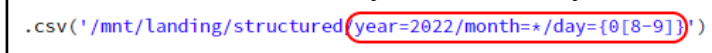
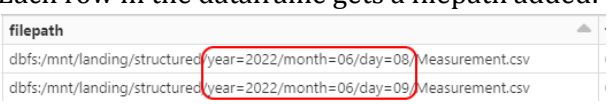


Figure 21: User actions | Semi-automatic infer

7.4.1 Logbook

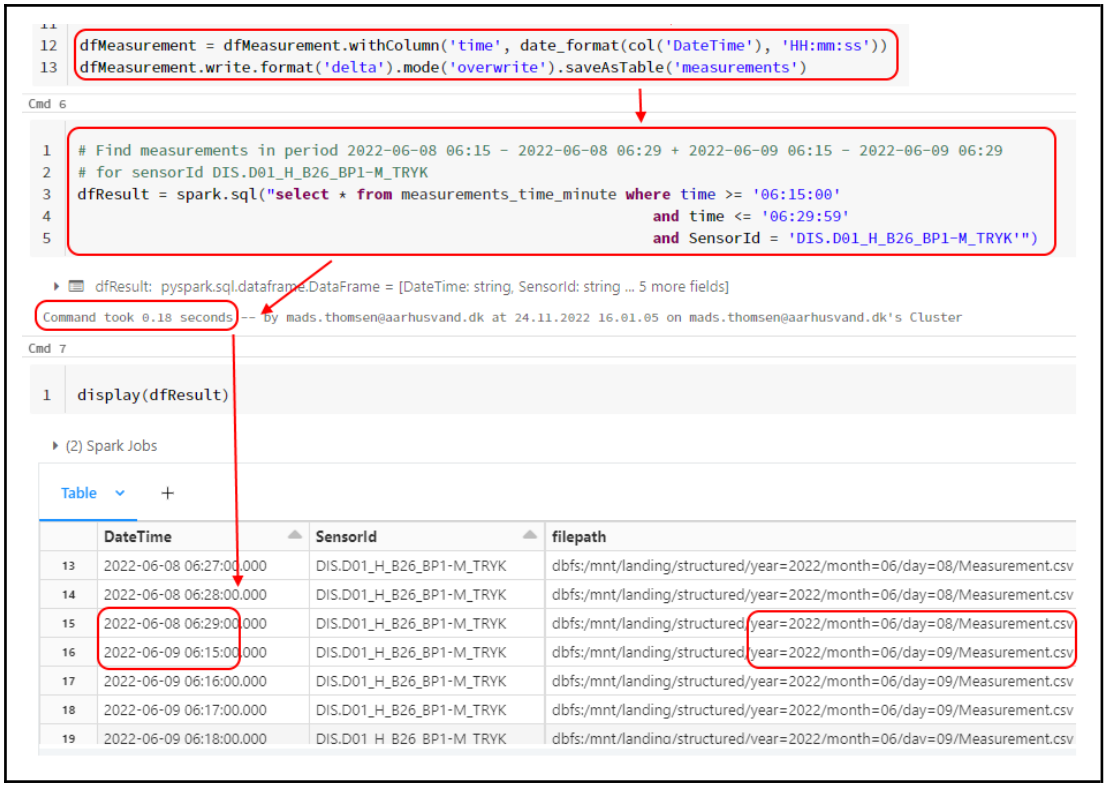
<table>
<tr><th colspan="2">QAS performance | Implicit infer</th></tr>
<tr><td colspan="2">The data analyst fetches data from a structured directory and the landing zone in the period #1: 09.06 06:15 - 06:29 and #2: 08.06 06:15 - 06:29 + 09.06 06:15 - 06:29. Each test period is run five times and the average execution time is calculated. The goal for the structured directory is to be 25% faster than the landing zone.</td></tr>
<tr><th colspan="2">Development Logbook</th></tr>
<tr><td colspan="2"></td></tr>
<tr><th>Date</th><th>Subject</th></tr>
<tr><td>2022-11-22</td><td>Period #1: 09.06 06:15 - 06:29
- Fetch data from period #1 into a dataframe using ‘loc’ which selects data by columns.
- Average faster execution time: 58,6%</td></tr>
<tr><td>2022-11-23</td><td>Periode #2: 08.06 06:15 - 06:29 + 09.06 06:15 - 06:29
- Two files from the structured directory need to be combined in order to find the proper results.
- Average faster execution time: 69,8%</td></tr>
<tr><th colspan="2">Performance Issues</th></tr>
<tr><th>Issue</th><th>Solutions and learnings</th></tr>
<tr><td>Combine files</td><td>After some failed attempts, the Python ‘glob’ module solves the challenge [GlobModule2022]. Glob patterns specify sets of filenames with wildcard characters similar to regular expressions [GlobPattern2022].
This enables iterating over the structured directory and loading only files of interest. This glob pattern find filenames from all month in year 2022 on days 08+09:
  </td></tr>
<tr><td>SQL-like search</td><td>A Python panda dataframe loads all data to the notebook before selecting the result. To avoid this, the PySpark library processes and combines data before they are loaded to the notebook [PySpark2022].
The PySpark library applies spark.sql() to run arbitrary SQL-like queries on a dataframe saved as a table.
Each row in the dataframe gets a filepath added:
  </td></tr>
</table>

Screenshots

Additional screenshots are added in Appendix F.
Python notebooks are uploaded in the code repository (Appendix B)
Structured landing zone - test #1:

Structured landing zone - test #2:

```
dfMeasurement = (spark.read
                    .option('header', True)
                    .option('delimiter', ',')
                    .option('recursiveFileLookup', True)
                    .csv('/mnt/landing/structured/year=2022/month=06/day={0[8-9]}')
                    .withColumn('filepath', input_file_name())
                )
```



Results			
Test period #1	**Structured (ms)**	**Unstructured (ms)**	**Difference (%)**
Run #1	110	350	68,6
Run #2	160	300	46,7
Run #3	110	290	62,1
Run #4	120	300	60,0
Run #5	150	340	55,9
Average	**130**	**316**	**58,6**
Test period #2	**Structured (ms)**	**Unstructured (ms)**	**Difference (%)**
Run #1	200	600	66,7
Run #2	200	560	64,3
Run #3	140	610	77,0
Run #4	160	560	71,4
Run #5	180	590	69,5
Average	**176**	**584**	**69,8**

Table 10: Logbook | Performance | Implicit infer

7.5 Discussion

To get an overview, all the results found in the above analyses based on the Quality Attribute Scenarios formulated in the analysis model, are gathered together and discussed in this section. The purpose of the analyses is to test if ***domain knowledge*** can be preserved in a ***time-saving*** way into a data lake in order to optimize the user's ***data discovery***. Each analysis has its starting point in the ***metadata concept***. Therefore, the overview is divided according to the highlighted words in bold and placed on a scale, where 'High' is best.

7.5.1 Usability

The usability quality is investigated through three Quality Attribute Scenarios with different user actions. The response measure for the scenarios indicates how easy it is for the user to accomplish the scenarios as described in Table 11.

Metadata concept		**Infer type**		
Entities	**Type**	**Automatic**	**Manual**	**Semi**
Data asset	Asset			
Column	Technical			
Lineage	Technical			
Business term	Business			
Classification	Business			
Total scale		Low	High	Medium
Domain knowledge				
Time consumption				
Data discovery				
Total scale		Low	Medium	High

Legend
Full
Partly
None

Table 11: Results | Usability

Looking at the metadata concept, automatic infer is at the low end of the scale, while manual infer is at the high end and in the middle is semi-automatic infer.
In terms of optimizing the data discovery, the picture is different: at the high end is semi-automatic infer, in the middle is manual infer while automatic infer still is at the low end. The above result table is based on the result section from the logbooks:

Automatic infer

The automatic scan is capable of ingesting all data assets. However, the inferred schemas are not efficient and no business metadata tags are

obtained. The subset of domain knowledge preserved in the data lake is small. However the user action is conducted fast and the technical metadata is searchable and can be discovered in the data catalog.

Manual infer

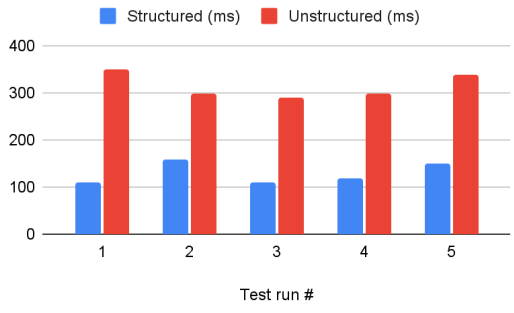
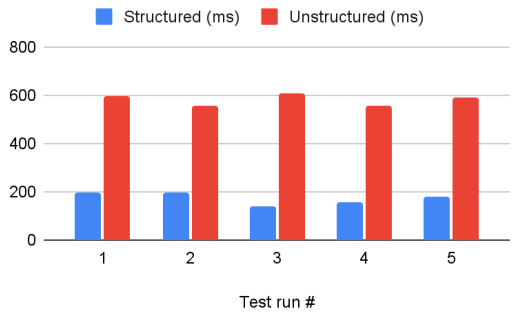
All metadata are added by manual processes, including data types, tagging, business metadata and relation between files. The metadata from the data model of the sample data is also applied and even the domain expert knowledge in a searchable way. In fact, all domain knowledge is preserved in the data lake in a searchable way . But the process is very time consuming and infeasible in a real-world scenario with several data assets. The time limit of 90 minutes of work was not met, as the scenario took 125 minutes to conduct.

Semi-automatic infer

The scan results in tagging the columns with classifications. Importing the business terms makes it fast to tag data assets with business knowledge. The time limit of 60 minutes of work was not met, as the scenario took 85 minutes to conduct. The time-consuming part was creating regular expressions. One way to make the process faster is to use column patterns with names, but it is not as robust as it does not look at the actual data content. Over time, the regular expressions will be reusable for the users, thus skipping the overhead.

7.5.2 Performance

Performance quality is explored through one Quality Attribute Scenario with two test periods. The results are evaluated by comparing the average execution time in milliseconds from the structured directory versus the unstructured landing zone as outlined in Table 12.

Results period #1	Results period #2
	

Metadata concept	Average results		
Source	**Structured**	**Unstructured**	**Difference**
Period #1	130 ms	316 ms	58,6 %
Period #2	176 ms	584 ms	69,8 %
Total scale	High	Low	
Domain knowledge			
Time consumption			
Data discovery			
Total scale	High	Low	

Legend
Full
Partly
None

Table 12: Results | Performance

Looking at the metadata concept, fetching data from the structured directory is at the high end of the scale, unstructured landing zone is at the low end. In terms of optimizing the data discovery, the picture is the same: at the high end is structured directory while unstructured landing zone still is at the low end. The above result table is based on the logbook:

Structured directory

Selection from the structuring directory based on time provides the user with useful knowledge about when the data is created. Combined with glob pattern, the fetching process only loads files of interest and serves the proper file paths. This approach is extremely efficient, very scalable and offers the user a quick data discovery with finding data. In addition, the user benefits from the SQL-like syntax. It is also possible to introduce additional parameters in the glob pattern, e.g. sensorId.

Unstructured landing zone

No implicit knowledge is offered from the unstructured landing zone. When storing all the measurements in one large file in the landing zone, the user must load the entire file to fetch the desired data. This is inefficient and the performance degrades significantly over time, as time series data grows over time. Thus the data discovery will be worse. Querying data from the dataframe columns is more time consuming to write than the Pyspark SQL-like syntax.

Chapter 8 - Conclusion

The purpose of this project is to investigate how domain knowledge can be preserved in a data lake to improve the user data discovery. Managing domain knowledge can be a time-consuming process and seems like a difficult task. Therefore, the project intends to answer the following problem statement:

> Can domain knowledge be preserved in a time-saving way into a data lake in order to optimize the user's data discovery?

In order to answer the problem statement, a hypothesis is set up, which involves constructing a metadata concept and an analysis model with scenarios. The concept provides the user with an overview of the different metadata entities. The hypothesis is evaluated using a small sandbox environment with sample data.

> By constructing a custom analysis model, where automatic, manual, semi-automatic and implicit ingesting of metadata are explored, it can be evaluated whether domain knowledge in the data lake optimizes the data discovery or not.

The hypothesis is verified, as it is possible to evaluate the extent to which ***domain knowledge*** can be preserved in a ***time-saving*** way into a data lake in order to optimize the user's ***data discovery***.

Usability
The result of the usability part of the analysis model is visualized using a scale. At the high end is semi-automatic infer, in the middle is manual infer while automatic infer is at the low end. For each infer type, the domain knowledge and time consumption is analyzed to determine whether data discovery has improved.

> The best optimization of data discovery is obtained by manual and semi-automatic infer. The most time-saving method is semi-automatic.

However, it should be noted that automatic infer in some cases can be a good choice, as it is possible to create searchable technical metadata very quickly. This is a much better starting point than having no information in the data catalog.

Performance
For the performance part of the analysis model, the result is very significant. Fetching data from a structured directory versus the unstructured landing zone is more than twice as fast. Looking at the metadata concept, fetching data from the structured directory is at the high end of the scale, unstructured landing zone

is at the low end. Combined with the glob pattern, the structured directory performs great. The faster data can be retrieved, the faster discovery.

> The structured directory with implicit domain knowledge is extremely efficient, very scalable and offers the user a quick data discovery.

Depending on data use, it can be considered to introduce additional parameters in the glob pattern, e.g. sensorId.

Answer to Problem Statement
Based on the results from the usability and performance part of the analysis model, the following answer to the problem statement is given:

> Based on the usability and performance scenarios, it can be concluded that it is possible to preserve domain knowledge in a time-saving way and thus optimize the data discovery. The best results are obtained with semi-automatic infer and storage of data in a structured directory.

8.1 Reflections

The work to investigate the problem statement has been a learning journey in how to handle some real-world challenges for a data-intensive application in a data lake. The journey has not been straightforward, thus it is valuable to highlight some learnings.

Key takeaways:

- One of the most important learnings has been the importance of developing a concept that is both tool, platform and domain independent.
- Another learning is the value of enabling the user to become self-serviced.
- The metadata concept makes the task of implementing a data catalog and building a metadata fundament manageable and accessible.
- The concept guides the user to navigate through the implementation of metadata in the data catalog, classify the data and add business knowledge.
- Users can expand the metadata concept according to their experiences.
- Costs of running a data catalog in Azure are billed per capacity hour unlike other cloud resources.

Bibliography

[Atlas2022] Apache Atlas. *Metadata Management and Data Governance*. https://atlan.com/what-is-apache-atlas/. Accessed 25.10.2022.

[Azure2022] Microsoft Learn. *Azure Blob Storage documentation*. https://learn.microsoft.com/en-us/azure/storage/blobs/. Accessed 08.10.2022.

[AzureEvent2022] Microsoft Learn. *Azure Schema Registry in Azure Event Hubs*. https://learn.microsoft.com/en-us/azure/event-hubs/schema-registry-overview. . Accessed 05.11.2022.

[Bass2012] Bass, Len, Paul Clements, and Rick Kazman. *Software architecture in practice*. SEI Series in Software Engineering, 2012.

[Christensen2020] Christensen, Henrik Bærbak. *Quality Attribute Scenarios (Slide deck)*. Software Architecture in Practice (Master of IT course), 2020.

[GlobModule2022] Towards Data Science. *The Python Glob Module*. https://towardsdatascience.com/the-python-glob-module-47d82f4cbd2d. Accessed 22.11.2022.

[GlobPattern2022] GeeksforGeeks. *How to use Glob function to find files.* https://www.geeksforgeeks.org/how-to-use-glob-function-to-find-files-recursively-in-python/. Accessed 23.11.2022.

[Gorelik2019] Gorelik, Alex. *The enterprise big data lake: Delivering the promise of big data and data science*. O'Reilly Media, 2019.

[Hamadou2020] Hamadou, Hamdi Ben, Torben Bach Pedersen, and Christian Thomsen. "The Danish National Energy Data Lake." *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020.

[Han2012] Han, Jiawei, Jian Pei, and Hanghang Tong. *Data mining: concepts and techniques*. Morgan Kaufmann, 2012.

[Jacobs2009] Jacobs, Adam. "The pathologies of big data." *Communications of the ACM* 52.8 (2009): 36-44.

[Nwagu2017] Nwagu et al. "Knowledge Discovery in Databases (KDD): an overview." *Int J Comput Sci Inf Secur (IJCSIS)* 15.12 (2017): 13-16.

[Purview2022] Microsoft Learn. *Microsoft Purview governance documentation*. https://learn.microsoft.com/en-us/azure/purview/. Accessed 08.10.2022.

[PurviewAPI2022] Microsoft Learn. *Microsoft Purview Resource Rest API*. https://learn.microsoft.com/en-us/rest/api/purview/. Accessed 08.10.2022.

[PySpark2022] Apache Spark. *PySpark Documentation*. https://spark.apache.org/docs/latest/api/python/. Accessed 23.11.2022.

[Thomsen2022] Thomsen, Christian. *Data warehousing & OLAP (Slide deck)*. Business Intelligence - Analysis of large databases (Master of IT course), 2022.

[Videla2021] Videla, Alvaro. "Meaning and Context in Computer Programs: Sharing domain knowledge." *Queue* 19.5 (2021): 60-68.

Appendix

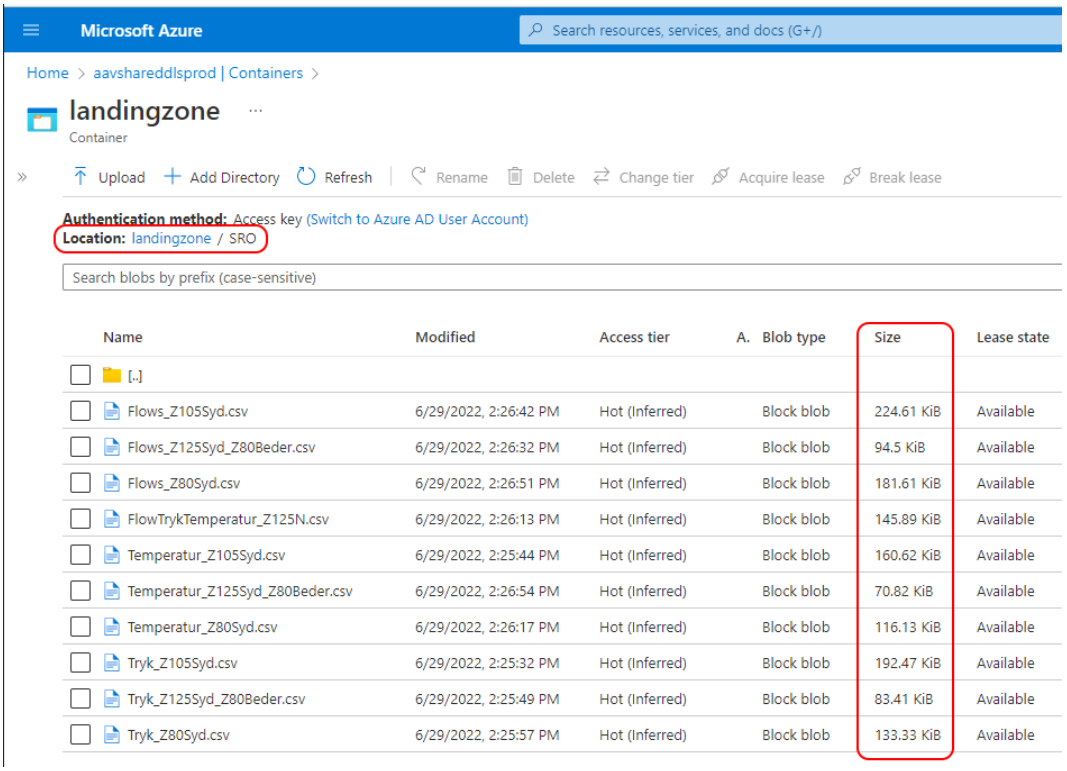
A. Production environment | Size

The data lake was started up in Azure in September 2021 where a small amount of sensor measurements were uploaded to the landing zone. Since November 2021 and until the summer of 2022, measurements from 45 sensors have been uploaded. It corresponds to approx. 6 million measurements every month (60 minutes x 24 hours x 31 days x 45 sensors x 3 types).

Every hour measurements per minute are uploaded for the last six hours. The size of files placed in the landing zone are 0,7 MB and contain 8.100 records.

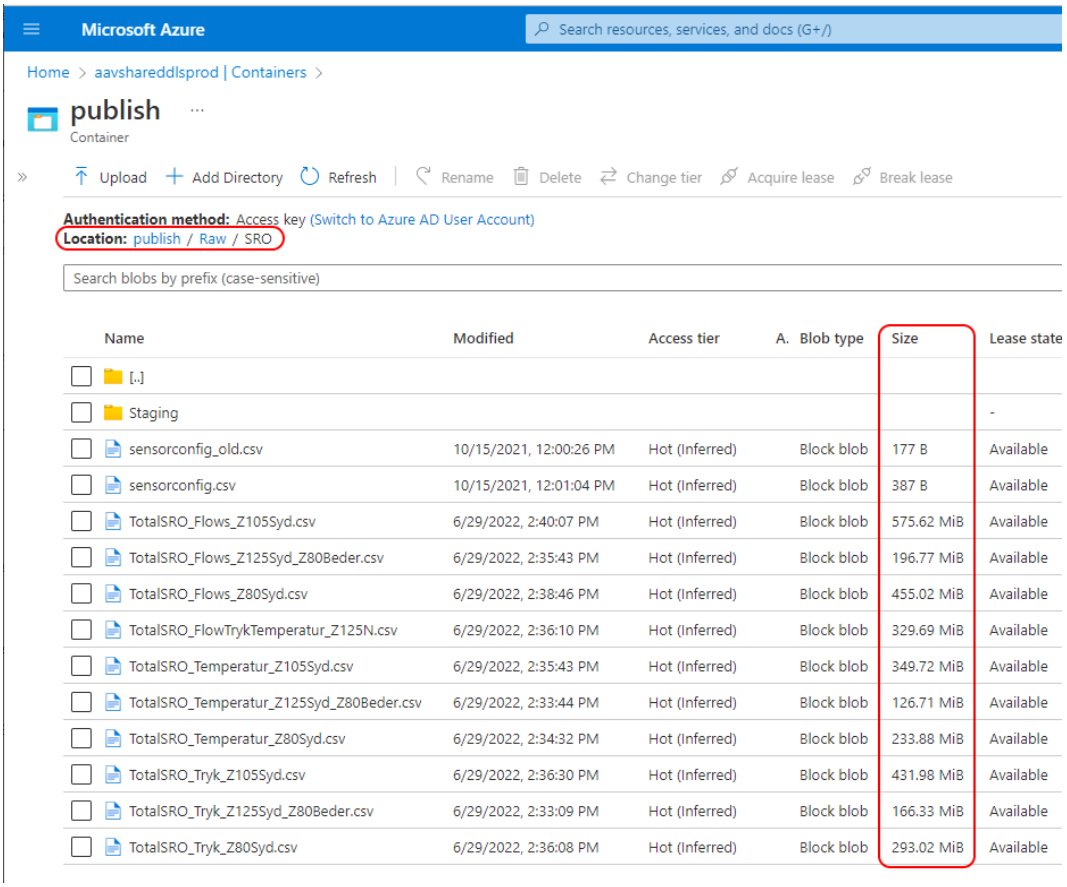
The uploaded files are handled in a data factory, where an Extract - Transform - Load (ETL) process appends delta measurements to the files in the publish zone with raw data. The current size of files placed in the publish zone are 3,2 GB and contain approx. 52.000.000 records.

A1 Landing zone



Name	Modified	Access tier	A.	Blob type	Size	Lease state
[..]						
Flows_Z105Syd.csv	6/29/2022, 2:26:42 PM	Hot (Inferred)		Block blob	224.61 KiB	Available
Flows_Z125Syd_Z80Beder.csv	6/29/2022, 2:26:32 PM	Hot (Inferred)		Block blob	94.5 KiB	Available
Flows_Z80Syd.csv	6/29/2022, 2:26:51 PM	Hot (Inferred)		Block blob	181.61 KiB	Available
FlowTrykTemperatur_Z125N.csv	6/29/2022, 2:26:13 PM	Hot (Inferred)		Block blob	145.89 KiB	Available
Temperatur_Z105Syd.csv	6/29/2022, 2:25:44 PM	Hot (Inferred)		Block blob	160.62 KiB	Available
Temperatur_Z125Syd_Z80Beder.csv	6/29/2022, 2:26:54 PM	Hot (Inferred)		Block blob	70.82 KiB	Available
Temperatur_Z80Syd.csv	6/29/2022, 2:26:17 PM	Hot (Inferred)		Block blob	116.13 KiB	Available
Tryk_Z105Syd.csv	6/29/2022, 2:25:32 PM	Hot (Inferred)		Block blob	192.47 KiB	Available
Tryk_Z125Syd_Z80Beder.csv	6/29/2022, 2:25:49 PM	Hot (Inferred)		Block blob	83.41 KiB	Available
Tryk_Z80Syd.csv	6/29/2022, 2:25:57 PM	Hot (Inferred)		Block blob	133.33 KiB	Available

A2. Gold zone



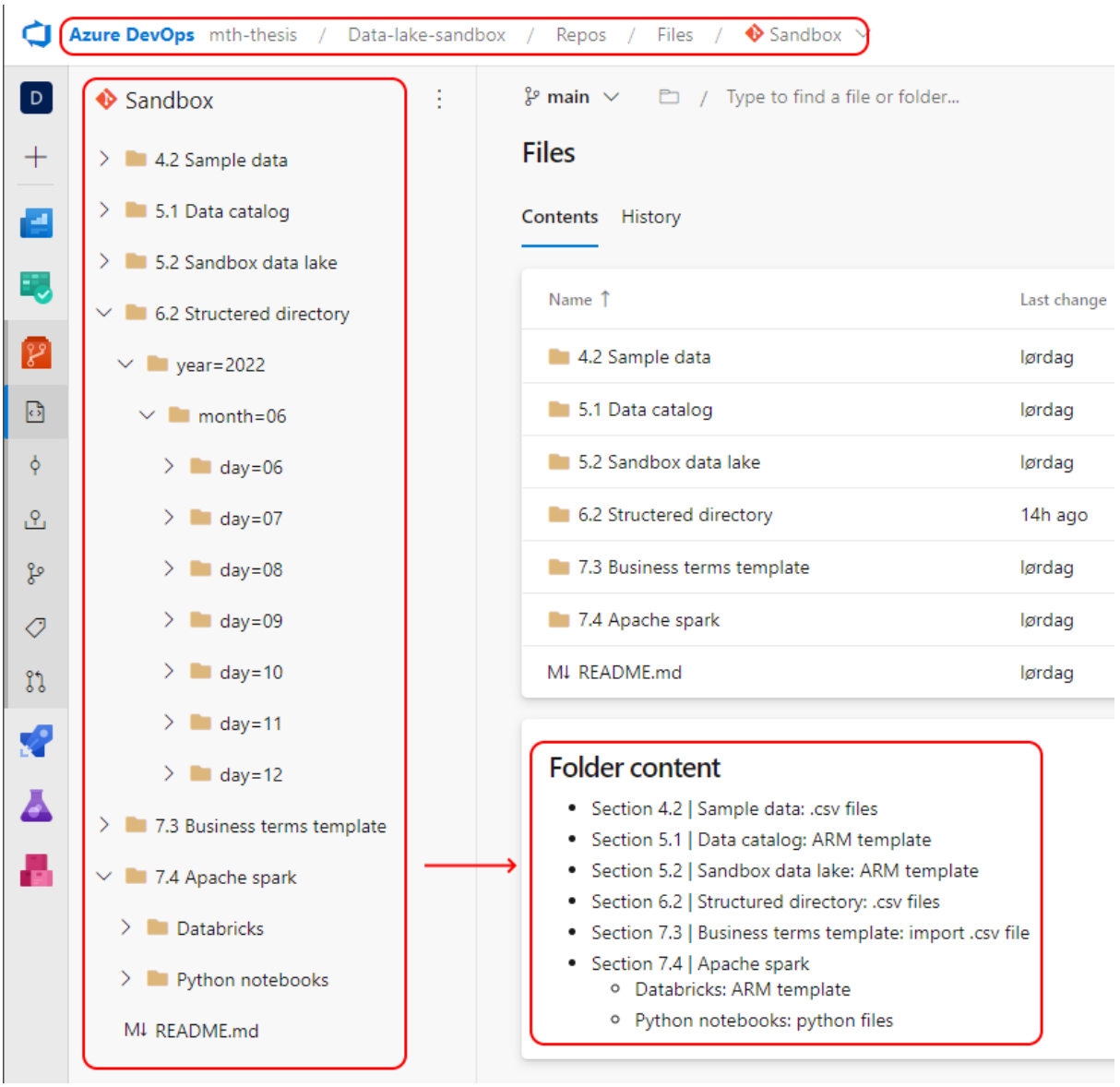
Name	Modified	Access tier	A.	Blob type	Size	Lease state
[..]						
Staging						-
sensorconfig_old.csv	10/15/2021, 12:00:26 PM	Hot (Inferred)		Block blob	177 B	Available
sensorconfig.csv	10/15/2021, 12:01:04 PM	Hot (Inferred)		Block blob	387 B	Available
TotalSRO_Flows_Z105Syd.csv	6/29/2022, 2:40:07 PM	Hot (Inferred)		Block blob	575.62 MiB	Available
TotalSRO_Flows_Z125Syd_Z80Beder.csv	6/29/2022, 2:35:43 PM	Hot (Inferred)		Block blob	196.77 MiB	Available
TotalSRO_Flows_Z80Syd.csv	6/29/2022, 2:38:46 PM	Hot (Inferred)		Block blob	455.02 MiB	Available
TotalSRO_FlowTrykTemperatur_Z125N.csv	6/29/2022, 2:36:10 PM	Hot (Inferred)		Block blob	329.69 MiB	Available
TotalSRO_Temperatur_Z105Syd.csv	6/29/2022, 2:35:43 PM	Hot (Inferred)		Block blob	349.72 MiB	Available
TotalSRO_Temperatur_Z125Syd_Z80Beder.csv	6/29/2022, 2:33:44 PM	Hot (Inferred)		Block blob	126.71 MiB	Available
TotalSRO_Temperatur_Z80Syd.csv	6/29/2022, 2:34:32 PM	Hot (Inferred)		Block blob	233.88 MiB	Available
TotalSRO_Tryk_Z105Syd.csv	6/29/2022, 2:36:30 PM	Hot (Inferred)		Block blob	431.98 MiB	Available
TotalSRO_Tryk_Z125Syd_Z80Beder.csv	6/29/2022, 2:33:09 PM	Hot (Inferred)		Block blob	166.33 MiB	Available
TotalSRO_Tryk_Z80Syd.csv	6/29/2022, 2:36:08 PM	Hot (Inferred)		Block blob	293.02 MiB	Available

B. Sandbox environment | DevOps

The infrastructure behind the sandbox environment can be described as code using the Azure Resource Manager templates and is available in the project's code repository placed at Azure DevOps. Sample data, python notebooks and the business terms template are also uploaded to the repository.

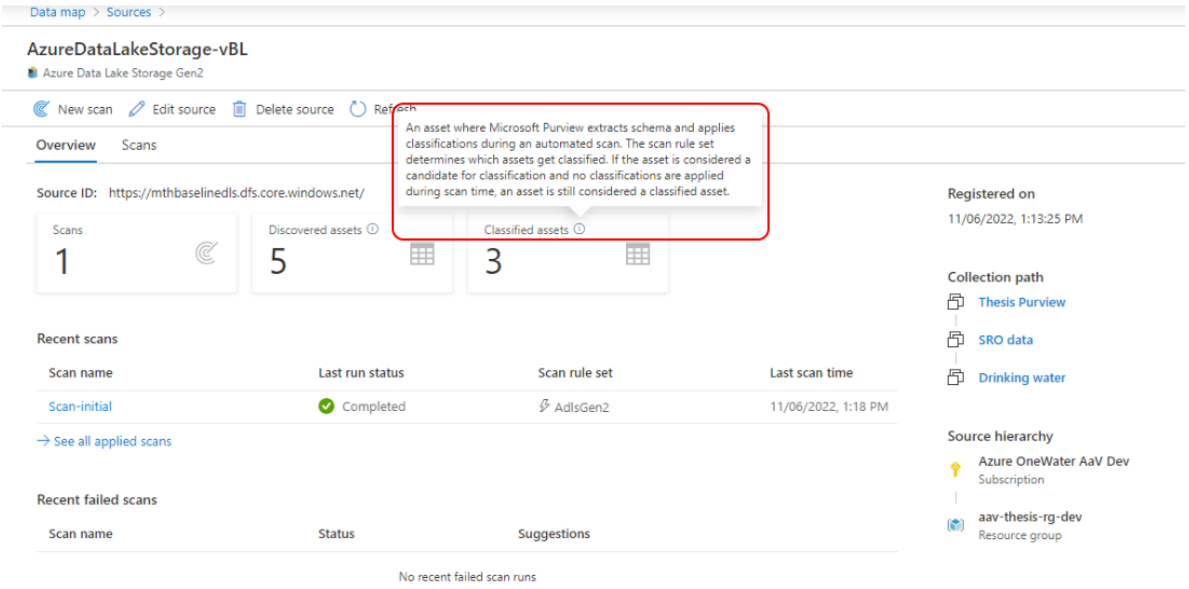
Until the exam at 16th December 2022 there is granted anonymous access to the DevOps: https://dev.azure.com/mth-thesis/Data-lake-sandbox/_git/Sandbox

B1. Repository content

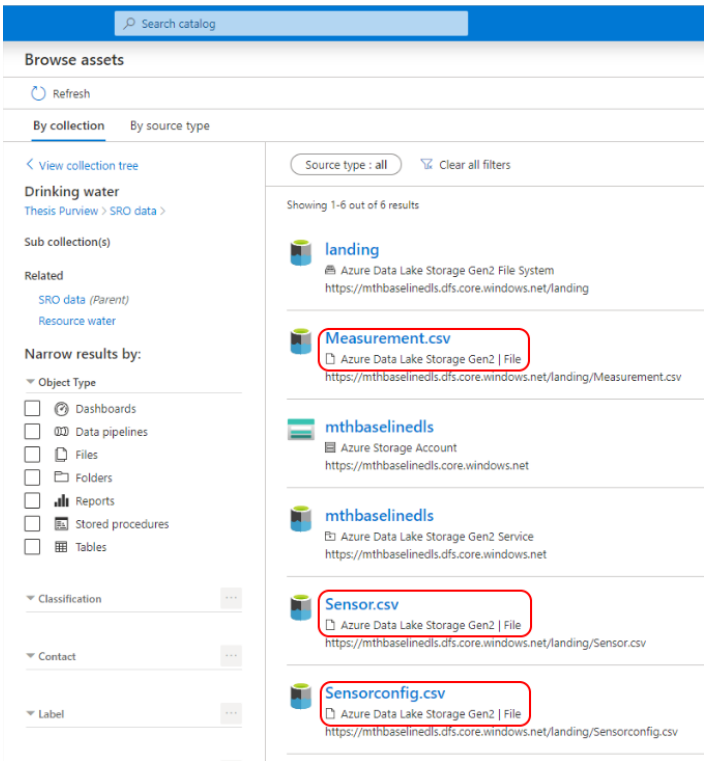


C. Automatic infer | Screenshots

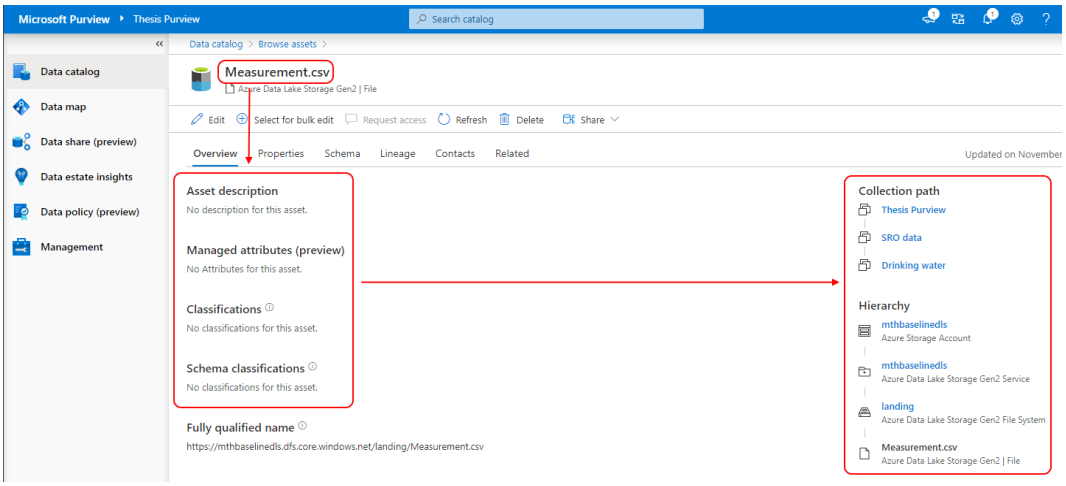
C1. Initial scan



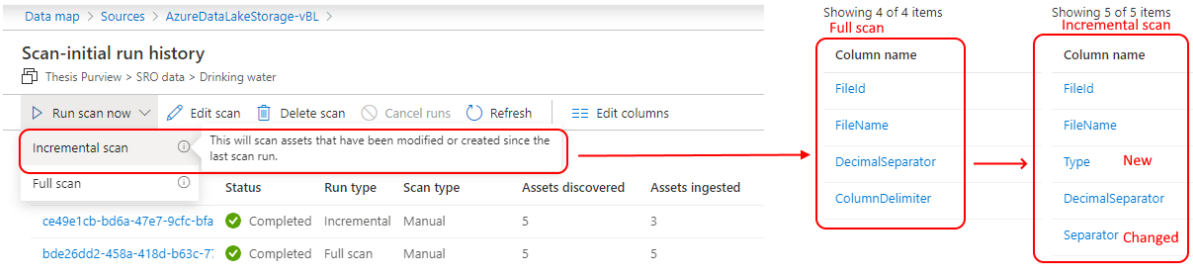
C2. Data discovery | Browse assets



C3. Data discovery | Asset overview

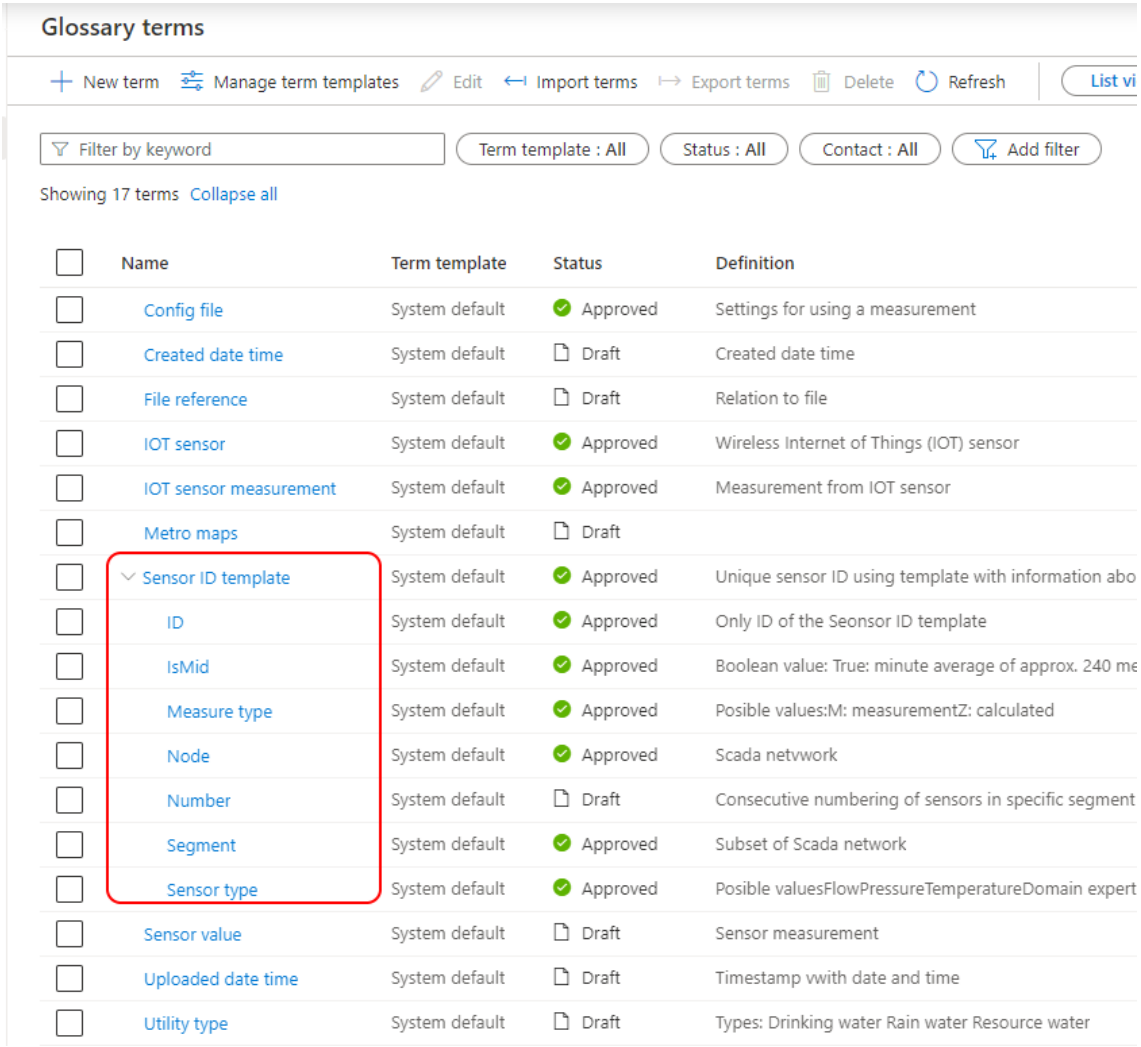


C4. Schema registry after change

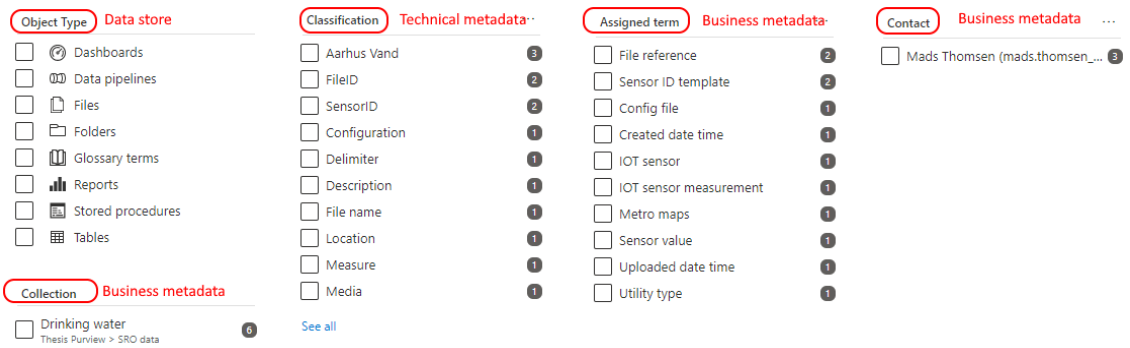


D. Manual infer | Screenshots

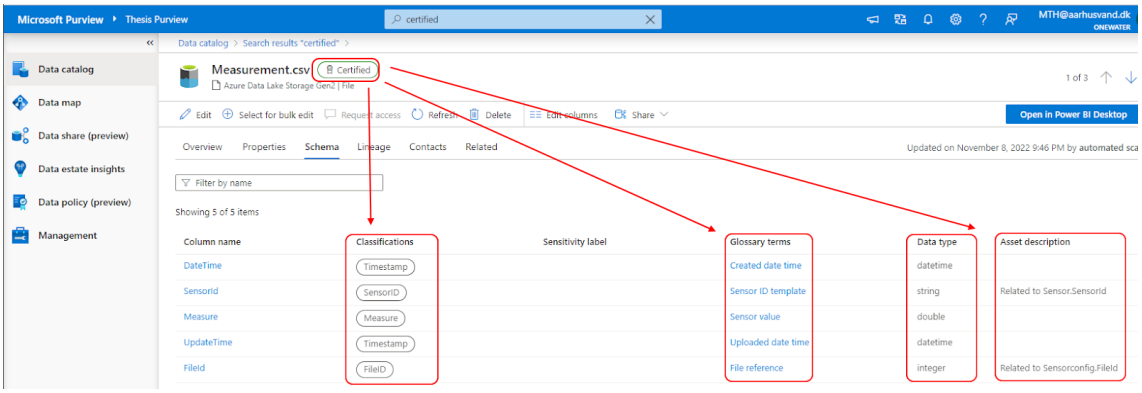
D1. Business glossary terms



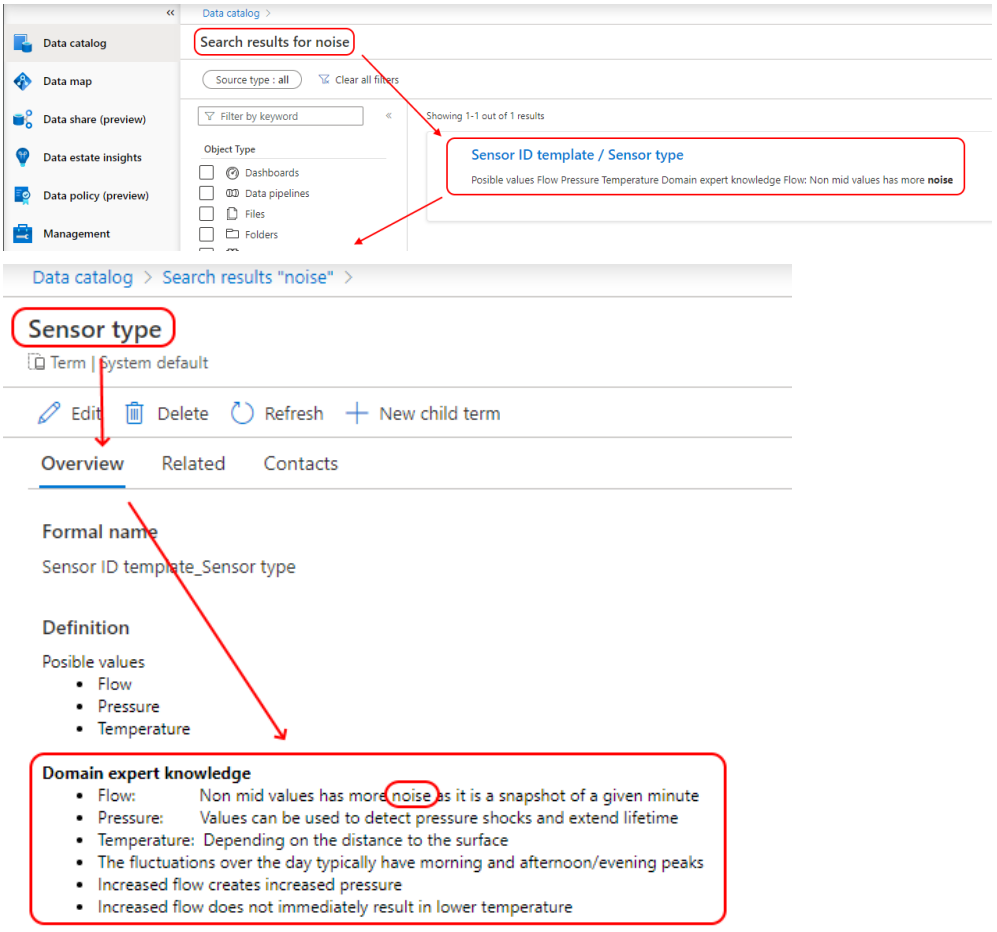
D2. Data discovery | Filters



D3. Data discovery | Asset schema

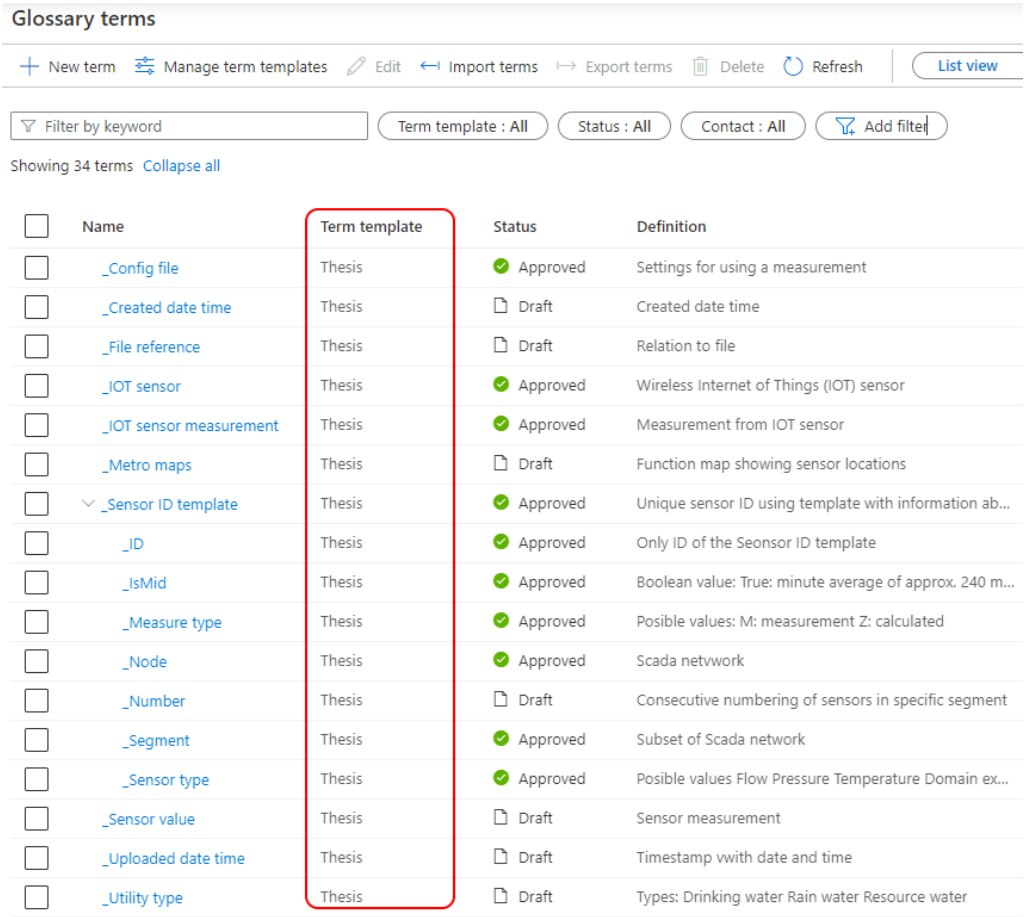


D4. Data discovery | Domain expert knowledge



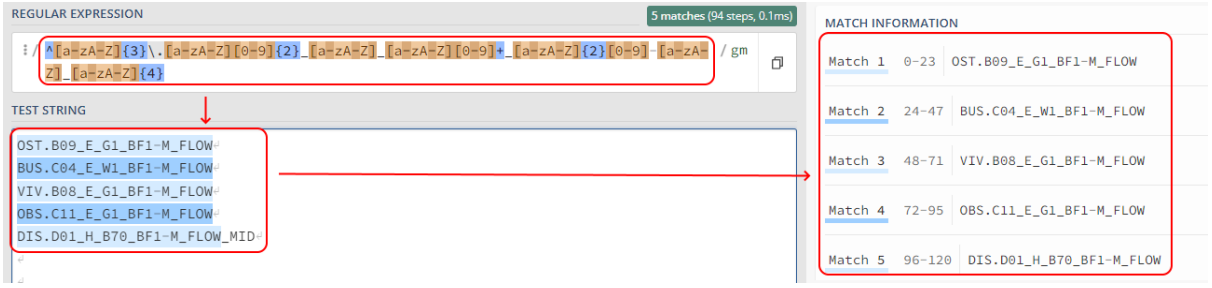
E. Semi-automatic infer | Screenshots

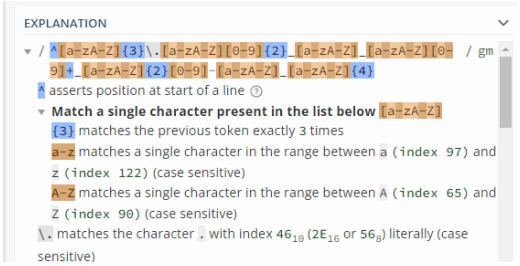
E1. Business glossary terms



Name	Term template	Status	Definition
_Config file	Thesis	Approved	Settings for using a measurement
_Created date time	Thesis	Draft	Created date time
_File reference	Thesis	Draft	Relation to file
_IOT sensor	Thesis	Approved	Wireless Internet of Things (IOT) sensor
_IOT sensor measurement	Thesis	Approved	Measurement from IOT sensor
_Metro maps	Thesis	Draft	Function map showing sensor locations
_Sensor ID template	Thesis	Approved	Unique sensor ID using template with information ab...
_ID	Thesis	Approved	Only ID of the Seonsor ID template
_IsMid	Thesis	Approved	Boolean value: True: minute average of approx. 240 m...
_Measure type	Thesis	Approved	Posible values: M: measurement Z: calculated
_Node	Thesis	Approved	Scada netvwork
_Number	Thesis	Draft	Consecutive numbering of sensors in specific segment
_Segment	Thesis	Approved	Subset of Scada network
_Sensor type	Thesis	Approved	Posible values Flow Pressure Temperature Domain ex...
_Sensor value	Thesis	Draft	Sensor measurement
_Uploaded date time	Thesis	Draft	Timestamp vwith date and time
_Utility type	Thesis	Draft	Types: Drinking water Rain water Resource water

E2. Classification rules





F. Implicit infer | Screenshots

F1. Unstructured landing zone | Test #1

F2. Unstructured landing zone | Test #2

Cmd 5

```
# Dataframe management
import pandas as pd

# Unstructured landing zone
dfMeasurement = pd.read_csv('/dbfs/mnt/landing/Measurement.csv')
```

Command took 1.92 seconds -- by mads.thomsen@aarhusvand.dk at 24.11.2022 16.02.02 on mads.thomsen@aarhusvand.dk's Cluster

Cmd 6

```
# Find measurements in period 2022-06-08 06:15 - 2022-06-08 06:29 + 2022-06-09 06:15 - 2022-06-09 06:29
# for sensorId DIS.D01_H_B26_BP1-M_TRYK
dfResult = dfMeasurement.loc[(dfMeasurement['DateTime'] >= '2022-06-08 06:15:00') &
                             (dfMeasurement['DateTime'] <= '2022-06-08 06:29:59') &
                             (dfMeasurement['SensorId'] == 'DIS.D01_H_B26_BP1-M_TRYK')]
dfResult_09_06 = dfMeasurement.loc[(dfMeasurement['DateTime'] >= '2022-06-09 06:15:00') &
                             (dfMeasurement['DateTime'] <= '2022-06-09 06:29:59') &
                             (dfMeasurement['SensorId'] == 'DIS.D01_H_B26_BP1-M_TRYK')]
dfResult = dfResult.append(dfResult_09_06)
```

Command took 0.60 seconds -- by mads.thomsen@aarhusvand.dk at 24.11.2022 16.02.14 on mads.thomsen@aarhusvand.dk's Cluster

Cmd 7

```
dfResult.head()
```

	DateTime	SensorId	Measure	UpdateTime	FileId
469922	2022-06-08 06:15:00.000	DIS.D01_H_B26_BP1-M_TRYK	4.202184	2022-06-08 07:30:51.000	10
469930	2022-06-08 06:16:00.000	DIS.D01_H_B26_BP1-M_TRYK	4.273755	2022-06-08 07:30:51.000	10
469938	2022-06-08 06:17:00.000	DIS.D01_H_B26_BP1-M_TRYK	4.295518	2022-06-08 07:30:51.000	10
469946	2022-06-08 06:18:00.000	DIS.D01_H_B26_BP1-M_TRYK	4.331699	2022-06-08 07:30:51.000	10
469954	2022-06-08 06:19:00.000	DIS.D01_H_B26_BP1-M_TRYK	4.194826	2022-06-08 07:30:51.000	10

Command took 0.04 seconds -- by mads.thomsen@aarhusvand.dk at 24.11.2022 16.02.26 on mads.thomsen@aarhusvand.dk's Cluster

Cmd 8

```
# Check expected numbers: 2 * 15 minutes
print('Number of measurements')
print('- Expected: ' + str(2 * 15))
print('- Actual:   ' + str(len(dfResult)))
print('- Total:    ' + str(len(dfMeasurement)))
```

```
Number of measurements
- Expected: 30
- Actual:   30
- Total:    1445943
```