Datenbank-Spektrum
https://doi.org/10.1007/s13222-023-00448-z

SCHWERPUNKTBEITRAG

Metadata Extraction from User Queries for Self-Service Data Lake Exploration

Jonas Gunklach[1] · **Sven Michalczyk**[2] · **Mario Nadj**[3] · **Alexander Maedche**[1]

Received: 1 March 2023 / Accepted: 18 May 2023

Abstract
Data catalogs represent a promising solution for semantically classifying and organizing data sources and enriching raw data with metadata. However, recent research has shown that data catalogs are difficult to implement due to the complexity of the data landscape or issues with data governance. Moreover, data catalogs struggle to enable business analysts to find the data they need for their use cases. Against this backdrop, we develop a self-service system that automatically extracts metadata from a data lake and enables business analysts to explore the metadata through an easy-to-use interface. Specifically, instead of implementing the data catalog top-down, our system derives metadata from user queries bottom-up. Hereby, we conduct 15 interviews with business analysts to derive the underlying requirements of the system and evaluate its features with a focus group. Our findings illustrate that participants especially value the possibility to reuse queries from other users and appreciated the support in query validation as data preparation is a complex and time-consuming endeavour.

Keywords Metadata Management · Data Lake · Data Catalog · Self-Service · Data Exploration

1 Introduction

To cope with the large amounts of heterogeneous data as well as the multitude of different use cases (e.g., forecasting demands, optimizing storage, estimating delivery times, etc.), data lakes have emerged in recent years as a new concept for storing, integrating and analyzing all types of data [14]. Data lakes are becoming increasingly popular in practice and are used in various industries and organizations [8]. Ideally, the data lake infrastructure and its tools support both, dedicated experts such as data engineers, as well as less technical users such as business analysts through open up the data on the lake for self-service applications [20]. Self-Service applications aim to enable non-technical users to prepare and analyze data with an easy-to-use interface without being reliant on expert users to perform analytical tasks easier and faster than before [3].

However, to leverage data from data lakes, metadata management including comprehensive documentation is required [6]. Metadata management constitutes activities which involve managing an organizations' knowledge on its data [11]. Data catalogs are seen as a means to address this problem by collecting, creating, and maintaining metadata [18], but organizations struggle with the implementation because of the complexity of data landscapes or issues with data governance [5]. In addition to the implementation, involving business analysts to use data catalogs is challenging. [13] identified that due to a variety of purposes and roles, current data catalogs do not successfully enable business analysts to explore data in data lakes and thus do not increase overall data usage [13].

Data exploration is a tedious process because users must "navigate large datasets by means of sequences of analytical queries" [4]. In this regard, research has investigated

✉ Jonas Gunklach
jonas.gunklach@kit.edu

Sven Michalczyk
sven.michalczyk@de.bosch.com

Mario Nadj
mario.nadj@tu-dortmund.de

Alexander Maedche
alexander.maedche@kit.edu

[1] Human-Centered Systems Lab, Karlsruhe Institute of Technology, Karlsruhe, Germany

[2] Robert Bosch GmbH, Stuttgart, Germany

[3] Business & Information Systems Engineering, TU Dortmund University, Dortmund, Germany

Published online: 26 June 2023

Table 1 Related Work

Studies	Supported Task	Artifact	Source	Target User
[4]	Query Creation	Recommender System	Interests of BI Users	Analysts
[12]	Query Creation	Query Search Engine	Executed Queries	Researcher
[17]	Data Visualization	Architecture	Existing Reports	Business Users
[19]	Data Exploration	Metadata Extraction Architecture	Technical Queries	Researcher
[22]	Query Creation	Query Designer	Business Ontologies	Business Users
[23, 24]	Data Analysis	Sidebar in BI System	Analysis Paths	Business Users

architectures, technical approaches, and systems supporting the extraction and exploration of data in data lakes. For instance, [19] present an approach that creates a metadata model from heterogenous data sources by detecting the data types and parsing its components. Furthermore, [12] discussed the general potential of reusing existing user queries to support students in writing new queries.

In this paper we present requirements and an implementation of a self-service data lake exploration system that follows a bottom-up approach to create relevant metadata and provides business analysts with an easy-to-use interface to explore it [4]. By implementing the system in a real-world environment with our industry partner, we show that it is feasible to extract metadata from user queries following a bottom-up approach.

2 Related Work

To identify related work, we relied on the systematic literature review on self-service Business Intelligence and Analytics by [15]. After scanning the identified papers, we identified two papers [22, 24] that discussed the concept of self-service data exploration. We extracted the keywords “query creation”, “knowledge transfer”, “knowledge management”, “recommender system”, “data exploration”, “data identification”, and “data preparation”. Using Google Scholar, we combined the keywords with “self-service” and identified three more papers [4, 12, 19]. Finally we conducted a forward backward search and identified two additional papers [17, 23]. We grouped related work into (1) supported task, (2) artifact, (3) source, and (4) target user group (see Table 1).

[19] presented an approach creating a metadata model from heterogenous data sources. Their approach consists of a metadata parser that detects data types and parses its components to extract metadata. They further discussed the possibility of annotating metadata with semantic information. For future work, they envisioned the metadata to support query creation. [24] used analysis paths of expert users, defined as a sequence of analysis steps stored in a knowledge repository. A sequence matcher finds similar sequences to predict the subsequent analysis steps based on what the user has already conducted. Drushku et al. [4] suggested an approach for data preparation using a sequence of executed queries to recommend queries to users with similar interests relying on collaborative filtering. Further, [17] suggested a self-service business intelligence architecture that assisted users in building their reports, thus supporting the task of data visualization. The architecture includes a knowledge base and collaboration rooms for report creation. A unified access layer should simplify data access. They follow the rationale of a high likelihood that other departments have already created the required reports and can then be reused.

To support self-service data exploration, [23] used a knowledge repository to offer users “guided recommendations” in a sidebar of a business intelligence system. The sidebar supports the entire analytical investigation process. [22] developed an ontology browser with visual query creation functionalities on top of a business ontology. The ontology is based on technical metadata that is extracted from database schemas. Tables are represented as concepts, columns as attributes of concepts, and foreign keys as relationships between concepts. For simplification, [22] suggested omitting technical but important attributes for query creation like key columns. The ontology is displayed as a graph presenting concepts as nodes and relations between concepts as directed edges. In addition, queries can be created by a drag-and-drop of nodes, executed returning a data preview, exported to Excel, and deployed directly. [12] also supported data preparation by offering a full-text search in executed queries; however, they targeted researchers. Their study found that query reuse could accelerate and facilitate data preparation and query creation significantly. A session-based search is developed since queries are usually created iteratively, with a high likelihood that queries at the end of a user session will be the final ones with the highest quality.

After analyzing related work, we identified that no study supported the exploration of data lakes, the data identification, and preparation simultaneously in one holistic system (gap 1). Moreover, no study investigated query validation, a complex and time-consuming sub-task of data preparation (gap 2). As gap 3, we identified a lacking approach for creating and enriching business terminologies with technical metadata from the bottom-up automatically. We identified paper creating business ontologies based on technical metadata [22], however they do not necessarily reflect the business terminologies used by analysts. In this regard, executed queries of expert users were enriched with a full-text search, a first attempt to provide business terminologies in conjunction with technical data preparation scripts in the form of queries [12]. However, queries cannot be searched by explicitly specifying contained tables or columns. Further, only one paper targeted analysts [4] by offering a recommendation approach but did not offer a concrete system (gap 4).

3 Requirements

To identify requirements for a self-service system for exploring data in data lakes, we first conducted 15 semi-structured interviews [21] with business analysts of our industry partner, a leading global supplier of technology and services. Via snowball sampling, we acquired users from various business functions (see Table 2). Interviews took place virtually and at different company sites and lasted between 0:56 h to 1:38 h. We conducted, recorded, and transcribed the interviews in the participants' native language. The interviews were three-parted: (1) First, we asked participants about their daily tasks, used tools, and current job role. (2) Furthermore, we asked more specifically about the source systems used and if they have experience with the data lake in their area of responsibility. In this regard, we also raised questions about how they analyze data and what are the most complex and time-consuming tasks. (3) We collected the requirements they have for a self-service system that supports exploration of data in data lakes. Specifically, we requested to sketch out possible inputs and outputs.

We observed that business analysts have difficulties exploring data in data lakes, but are pretty familiar with the underlying business domain and processes. Thus, the system should render the data in the data lake in business terminology (**R1**) as used in the common source systems (e.g., SAP). This requirement seems critical since the data lake contains mainly raw data described with technical column names, which does not directly correspond to the view that business analysts are used to. Moreover, users requested to ease the tedious search for data. For instance, they often go back and forth between source systems and an internet search to identify technical names, which can be found in the data lake. In this regard, users requested an enriched search in the data lake to ease this process. The system should provide a search for column and table names in the data lake (**R2**). In this regard, they raised the third requirement that the system should indicate how tables in the data lake can be joined (**R3**). Furthermore, users reported that it

Table 2 Participants of Semi-Structured Interviews

PNo.	Domain	Duration
P01	Logistics	01:32:02
P02	Logistics	01:28:57
P03	Logistics	01:30:16
P04	Manufacturing	01:38:26
P05	Purchasing	01:37:54
P06	Controlling	01:26:06
P07	Logistics	00:56:52
P08	Logistics	01:35:03
P09	Logistics	01:31:57
P10	Logistics	01:18:42
P11	Purchasing	01:28:29
P12	Logistics	00:59:35
P13	Manufacturing	00:58:43
P14	Controlling	01:20:22
P15	Purchasing	01:22:02

would be helpful to reuse queries from other users because multiple plants work on similar use cases. One participant explained, "we could copy the SQL code from other departments and use it for our use cases". Currently, users miss an option to efficiently share queries within the organization. As users find the creation of queries complex and time-consuming, starting with an existing query would ease the creation. In this line, the system should recommend queries for reuse (**R4**). To further support users in the creation of the query, the system should embed a SQL editor with syntax highlighting and auto-completions for syntax, column, and table names (**R5**). Next, users reported that the validation of queries is one of the most complex and time-consuming tasks. As many users lack detailed knowledge about the source systems, they validate queries in small increments. To support them in query validation, several users indicated that the system should provide a preview of the query results using a random result sample of rows (**R6**).

4 Implementation

Based on the requirements identified, we developed and integrated the system into the industry partner's existing data lake architecture. The underlying data lake architecture follows the architecture proposed by [7] and is composed of four layers: data source layer, data integration layer, analysis and optimization layer, and data consumer layer. The metadata extraction and exploration system proposed consists of three main layers that interact with the data lake (see Fig. 1): (1) the logging layer accesses the active session history and logs the active session history of the databases. This layer saves the active sessions including SQL full-text queries in the active session data store. (2) Next, the extrac-

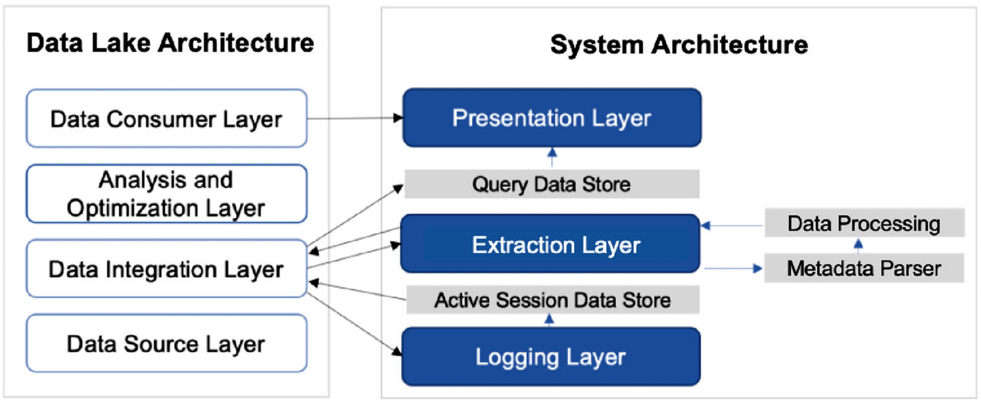


Fig. 1 Architecture

tor layer consists of two modules, the metadata parser, and the data processing workflow. The metadata parser parses the user queries and the data processing workflow extracts their metadata (e.g., columns, tables, or joins used in the query). (3) The presentation layer accesses the metadata information and provides a web interface for the user.

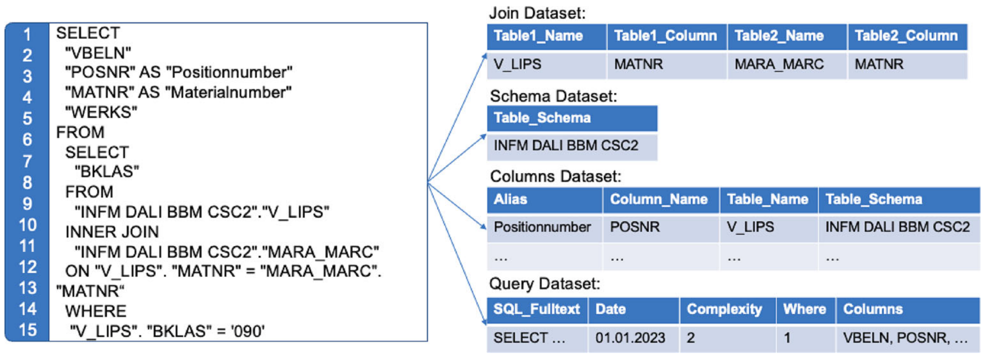
4.1 Logging Layer

To instantiate our approach of extracting metadata, the user queries need to be logged and stored in a table. In our case, the system is relying on user queries captured in the database active session history . The database samples all activities and saves the information session-based in a table. For that, the database takes a snapshot of all active database sessions every second. Further information, such as the SQL full-text which is necessary to extract all metadata, are stored in another table. To obtain them, we joined the action session history with the SQL full-text table. After that, we store the user queries in the active session data store. In total, we are relying on the following columns: (1) sampleID: every active database session is provided with a unique sample identifier, (2) sampleTime: time at which the sample was taken, (3) sessionID: uniquely identifies the session, (4) userID: hash of user, and (5) sqlQuery: full-text of the SQL query raised by the user.

4.2 Extraction Layer

The goal of the next layer is to extract metadata from SQL full-text queries such as columns, column aliases, tables and table joins. We implemented the data processing workflow in KNIME to combine different technologies, such as Java (for the SQL parser using the library 'JsqlParser') and Python (for the data processing). We run the workflow daily and analyze the queries that were run on the data lake the previous day. Similar to the knowledge store proposed by [17], each time the workflow is executed, the extracted metadata is compared to the previously extracted metadata in the query data stores to remove duplicates and update accordingly.

First, to extract unique daily queries from the dataset, we group the active sessions by sessionID and userID. Next, the unique daily queries can be parsed using the metadata parser. For that, we developed a SQL parser relying on "JSqlParser" [1]. The metadata parser first transforms the full-text query into a tree structure and then walks through the tree structure to extract the metadata from each SQL keyword (e.g., And-Expression, Or-Expression, etc.). The metadata parser iterates over all Select-Statements and extracts the following information: (1) queryComplexity: number of Select-Statements inside the query, (2) columnNames: list of all column names in the main Select-Statement of the query, (3) nestedColumnNames: list of all column names in the nested queries, (4) allTables: list of all tables, (5) tableJoins: table joins used in query, (6) countWhere: number of where-conditions used in the query. Afterwards, we removed non-parseable queries and queries with more than 10 nested select statements. For the second part, we developed a data processing workflow to create four datasets based on the previously extracted metadata information (see Fig. 2). The first dataset **joins** contain all unique joins of the past. Second, the dataset **schemas** contain all database schemas, and the parser matches related projects at the industry partner. Hence, schemas enrich the business domain structure of the industry partner. Third, the **columns** dataset contains all unique columns, including their descriptions and associated database schemas. Fourth, the **query** dataset consists of all unique queries executed; calculated query attributes like the number of nested select statements and the number of used where clauses. The query dataset contains a time series about when, by whom, and which query was executed. To create this dataset, we implemented a deduplication routine in the parser. Personal information is anonymized. The datasets are saved in the query data store and can be accessed by the presentation layer.



```
SELECT
  "VBELN"
  "POSNR" AS "Positionnumber"
  "MATNR" AS "Materialnumber"
  "WERKS"
FROM
  SELECT
    "BKLAS"
  FROM
    "INFM DALI BBM CSC2"."V_LIPS"
  INNER JOIN
    "INFM DALI BBM CSC2"."MARA_MARC"
  ON "V_LIPS". "MATNR" = "MARA_MARC".
"MATNR"
  WHERE
   "V_LIPS". "BKLAS" = '090'
```

Join Dataset:

Table1_Name	Table1_Column	Table2_Name	Table2_Column
V_LIPS	MATNR	MARA_MARC	MATNR

Schema Dataset:

Table_Schema
INFM DALI BBM CSC2

Columns Dataset:

Alias	Column_Name	Table_Name	Table_Schema
Positionnumber	POSNR	V_LIPS	INFM DALI BBM CSC2
…	…	…	…

Query Dataset:

SQL_Fulltext	Date	Complexity	Where	Columns
SELECT …	01.01.2023	2	1	VBELN, POSNR, …

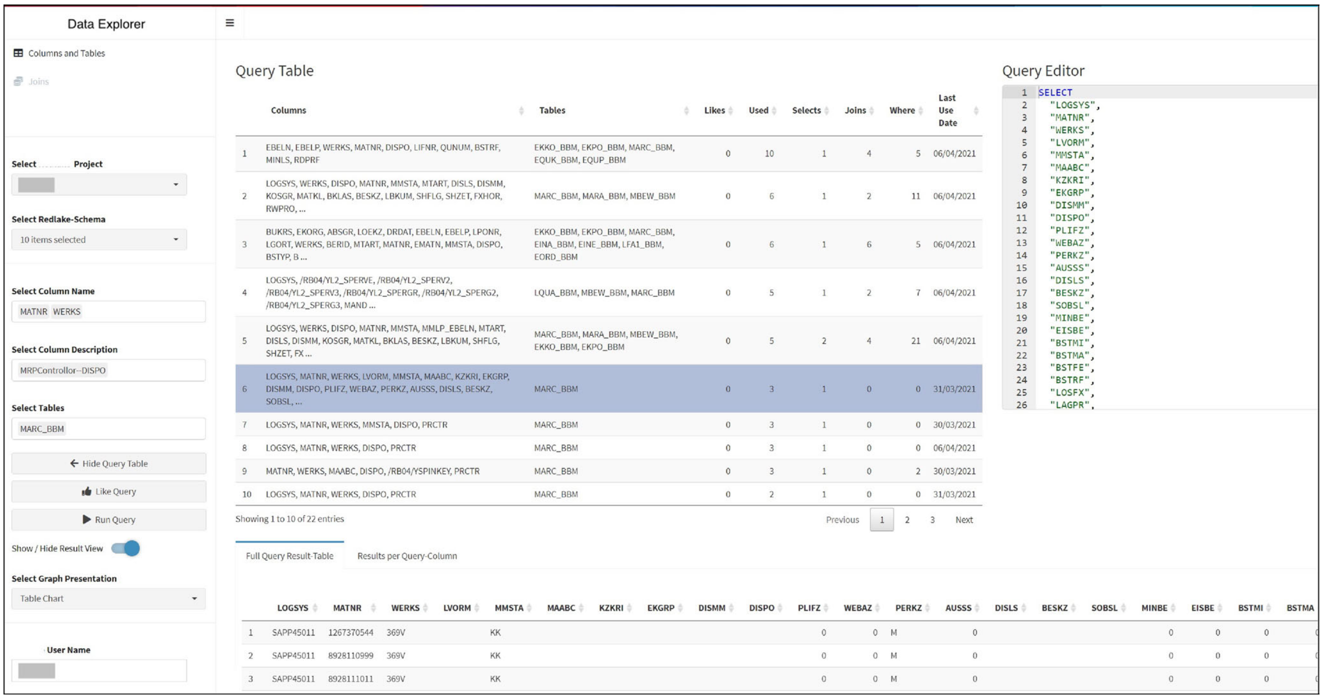
Fig. 2 Exemplary extraction of SQL query into four datasets

4.3 Presentation Layer

The presentation layer consists of two parts (see Fig. 3). The left side contains a sidebar for navigation and global filtering options. The content depends on the main page selected at the top of the sidebar navigation. For that, we implemented (1) Columns and Tables and (2) Joins.

In the sidebar, users can select a project they are working for or have an initial indication that the data they need will be used in that project. Next, they can specify to which database schema the data might belong. If this is unclear, users can also select all schemas. In the three input fields, users can search on a detailed level for technical column names, column descriptions, or table names. Multiple selections within and across fields are possible and desirable since they narrow down the search. In the default state, the system will only search for column names and alias in the highest order of the query's select statement, or the entire query by toggling on the switch search in complete query. The sidebar supports the exploration of the data lake by providing a search in the data lake's metadata. By adjusting the filters in the sidebar, respective queries were displayed in the system's body. Further, the sidebar has the functionality like a query and hide the sidebar, which is useful for small screens. To support query validation, users can execute queries (with the run query button in the sidebar) after entering their data lake user and password. Subsequently, users can select query result visualizations through the select graph presentation dropdown.

The query table lists queries according to the sidebar's search result. In this regard, the system recommends queries to users for their use case. The query table displays all used columns and tables per query. It offers various sorting criteria to ease the selection of a suitable query. First, users can



Query Table

	Columns	Tables	Likes	Used	Selects	Joins	Where	Last Use Date
1	EBELN, EBELP, WERKS, MATNR, DISPO, LIFNR, QUNUM, BSTRF, MINLS, RDPRF	EKKO_BBM, EKPO_BBM, MARC_BBM, EQUK_BBM, EQUP_BBM	0	10	1	4	5	06/04/2021
2	LOGSYS, WERKS, DISPO, MATNR, MMSTA, MTART, DISLS, DISMM, KOSGR, MATKL, BKLAS, BESKZ, LBKUM, SHFLG, SHZET, FXHOR, RWPRO, ...	MARC_BBM, MARA_BBM, MBEW_BBM	0	6	1	2	11	06/04/2021
3	BUKRS, EKORG, ABSGR, LOEKZ, DRDAT, EBELN, EBELP, LPONR, LGORT, WERKS, BERID, MTART, MATNR, EMATN, MMSTA, DISPO, BSTYP, B ...	EKKO_BBM, EKPO_BBM, MARC_BBM, EINA_BBM, EINE_BBM, LFA1_BBM, EORD_BBM	0	6	1	6	5	06/04/2021
4	LOGSYS, /RB04/YL2_SPERVE, /RB04/YL2_SPERV2, /RB04/YL2_SPERV3, /RB04/YL2_SPERGR, /RB04/YL2_SPERG2, /RB04/YL2_SPERG3, MAND ...	LQUA_BBM, MBEW_BBM, MARC_BBM	0	5	1	2	7	06/04/2021
5	LOGSYS, WERKS, DISPO, MATNR, MMSTA, MMLP_EBELN, MTART, DISLS, DISMM, KOSGR, MATKL, BKLAS, BESKZ, LBKUM, SHFLG, SHZET, FX ...	MARC_BBM, MARA_BBM, MBEW_BBM, EKKO_BBM, EKPO_BBM	0	5	2	4	21	06/04/2021
6	LOGSYS, MATNR, WERKS, LVORM, MMSTA, MAABC, KZKRI, EKGRP, DISMM, DISPO, PLIFZ, WEBAZ, PERKZ, AUSSS, DISLS, BESKZ, SOBSL, ...	MARC_BBM	0	3	1	0	0	31/03/2021
7	LOGSYS, MATNR, WERKS, MMSTA, DISPO, PRCTR	MARC_BBM	0	3	1	0	0	30/03/2021
8	LOGSYS, MATNR, WERKS, DISPO, PRCTR	MARC_BBM	0	3	1	0	0	06/04/2021
9	MATNR, WERKS, MAABC, DISPO, /RB04/YSPINKEY, PRCTR	MARC_BBM	0	3	1	0	2	30/03/2021
10	LOGSYS, MATNR, WERKS, DISPO, PRCTR	MARC_BBM	0	2	1	0	0	31/03/2021

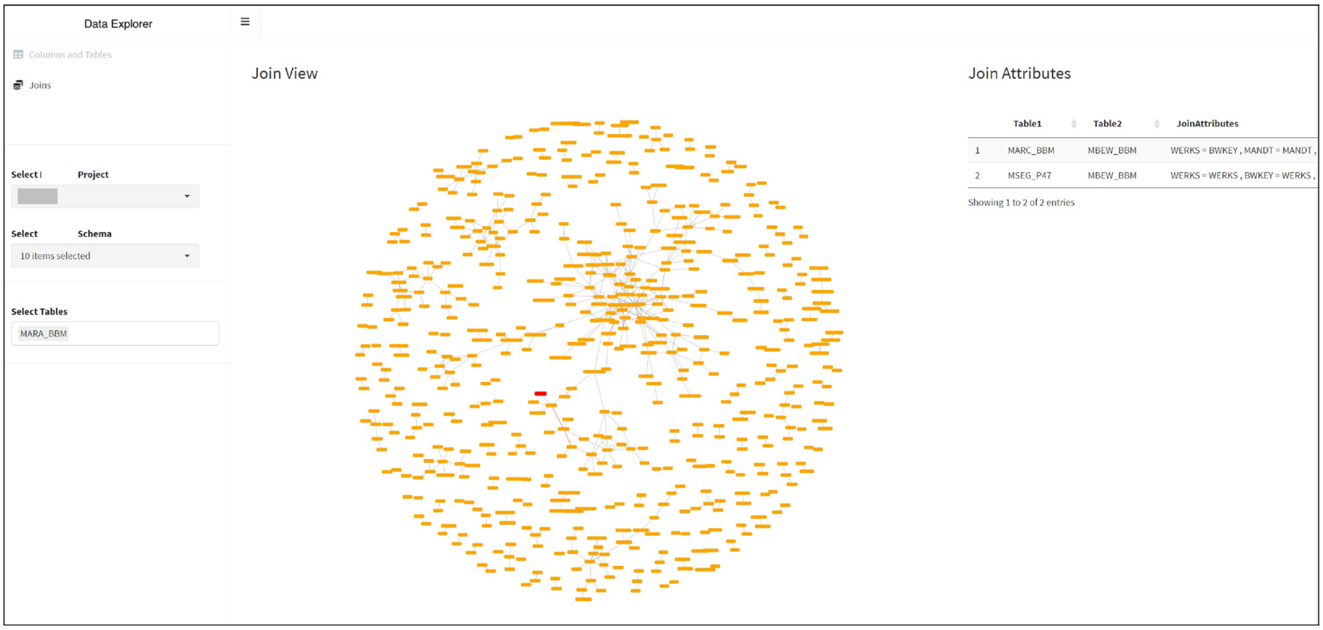
Showing 1 to 10 of 22 entries

Query Editor

```
SELECT
  "LOGSYS",
  "MATNR",
  "WERKS",
  "LVORM",
  "MMSTA",
  "MAABC",
  "KZKRI",
  "EKGRP",
  "DISMM",
  "DISPO",
  "PLIFZ",
  "WEBAZ",
  "PERKZ",
  "AUSSS",
  "DISLS",
  "BESKZ",
  "SOBSL",
  "MINBE",
  "EISBE",
  "BSTMI",
  "BSTMA",
  "BSTFE",
  "BSTRF",
  "LOSFX",
  "LAGPR",
```

	LOGSYS	MATNR	WERKS	LVORM	MMSTA	MAABC	KZKRI	EKGRP	DISMM	DISPO	PLIFZ	WEBAZ	PERKZ	AUSSS	DISLS	BESKZ	SOBSL	MINBE	EISBE	BSTMI	BSTMA
1	SAPP45011	1267370544	369V		KK						0	0	M	0				0	0	0	
2	SAPP45011	8928110999	369V		KK						0	0	M	0				0	0	0	
3	SAPP45011	8928111011	369V		KK						0	0	M	0				0	0	0	

Fig. 3 View 1: Columns and Tables including Sidebar, Query Table, Query Editor, and Query Results



Select | Project
[redacted]

Select Schema
10 items selected

Select Tables
MARA_BBM

Join Attributes

	Table1	Table2	JoinAttributes
1	MARC_BBM	MBEW_BBM	WERKS = BWKEY , MANDT = MANDT ,
2	MSEG_P47	MBEW_BBM	WERKS = WERKS , BWKEY = WERKS ,

Showing 1 to 2 of 2 entries

Fig. 4 View 2: Joins including Sidebar, Join View, and Join Attributes

sort queries by likes and number of used times. These criteria indicate whether a query was identified as applicable and might reach a certain quality level. The column "Used" displays the number of executions of a query. Hence, users can sort by the latest executed queries. The following three columns, "Selects", "Joins", and "Where", are extracted from the query and count how many of the respective SQL keywords are used. For instance, a query using more than one select keyword is nested. A query containing more than one join keyword combines several tables. Thus, the counts indicate how complex a query is and, therefore, how difficult to understand. Finally, the column "Last used date" indicates when the query was last used. The query is shown in the query editor by clicking on a row in the query table. The query is formatted with SQL syntax highlighting and indented to increase readability. Users can edit the query according to their needs, supported by syntax and names autocompletion. Further, line numbers are shown. The system guides users with the syntax highlighting, code indention, and auto-completion functions. Subsequently, users can run the query for final validation. We offer several previews of the query results. Users can either rely on the complete query result or aggregations per column, the frequency of each column value. Users can access the aggregations per column through navigation tabs generated for each column. In the sidebar, users can select whether they would like to inspect the value frequencies in a table, through a tree map, or a pie chart. Hence, the system supports query validation, with the option to preview adapted query results. The results can be shown as a complete table or as frequencies of values per column. The system displays 1.000 rows randomly to account for fast query execution. In summary, users can use the first page of the system to search for technical column names, in column descriptions, and table names. Subsequently, users can preview, adjust, and validate queries.

The second view of the system (see Fig. 4) accounts for the inspection of performed table joins. Therefore, users specify a project and schema like before. We adapted the sidebar and removed the column search fields. Once projects and schemas are selected, all tables joined in the past are displayed as a graph with the tables as nodes and joins as edges. Users can zoom in the graph and explore table connections. Users can search a table in the table input field of the sidebar. Afterward, the table is marked red in the graph. Users can click on a node in the graph to identify join columns respectively keys in a relation. The join attributes table on the right shows the relationship details. Multiple entries are shown if a table was joined with multiple other tables.

5 Evaluation

To evaluate our system, we conducted a focus group evaluation [16] with four experts from our industry partner. Specifically, we included two analysts from quality management, one from logistics and an architect from global governance. We showed participants the features of the system and asked them to assess their usefulness.

Participants described the core strength of the system as providing a quick, graphical overview of schemas, tables, and columns. They liked that columns can be identified through their technical name and descriptions **(R2)**. This in turn would save time and create transparency in the data models used, making the system not only beneficial for business analysts, but also for source system experts and data engineers. Source system experts, for instance, who typically have less experience with data lakes, would be enabled to identify schemas in the data lake. The join view of the system might be even more helpful as the underlying table joins of transactional views are not identifiable through source systems **(R3)**.

Participants saw another strength in the visibility of the data lake's entire metadata **(R1)**. Moreover, they agreed that the join graph is compelling but challenging to use. One participant suggested to highlight related tables in a different color and not only the selected table. Furthermore, the query table should be enriched with information about key columns and column types.

Next, participants recognized and liked the system's rationale that less savvy users could learn and reuse queries from more savvy ones **(R4)**. For instance, they discussed that users could learn the use of advanced SQL keywords (e.g., group by) at the example of queries based on their data. In this regard, they emphasized the strength of copy-and-paste queries and were optimistic about their adoption within the organization. Furthermore, users might also learn to write optimized queries by inspecting frequently executed ones **(R5)**. Participants discussed that the system still requires some skill in data engineering to create queries and basic knowledge of the data lake structure is also required (e.g., how schemas are used). Participants concluded that frequently executed queries might be scheduled, hence, being part of productive processes and optimized for efficiency. However, they pointed out that knowledge sharing can potentially lead to problems when users just copy the query and do not validate the result.

Participants liked the functionalities for query validation **(R6)**. Because the system supports query execution and previews the query result in different forms, no sub-queries with group by or windowing and counts are necessary to validate them. Furthermore, they liked that the aggregated column values can be filtered. Hence, participants praised that the system saves additional and manual effort for query validation. Finally, it was emphasized that the tooltips are suitable for quickly providing support. In particular, participants found the last used date and the likes of other users most useful when identifying queries.

6 Discussion

Due to the paucity of codified information about the purpose, value, and provenance of data stored in a data lake, data tends to disappear into silos within organizations, resulting in significant lost productivity and opportunity, duplication of effort, and mishandling of data [10]. In theory, data catalogs provide support in data discovery, inventory, governance, documentation and collaboration [25]. However, data collaboration is often overlooked by current data catalogs but can provide the most benefits for end-users [13]. In software development, reusing code fragments has a long tradition [9]. As software is typically based on many pre-existing components, software engineers benefit from sharing code through public repositiories like Github. To support data collaboration and organizational knowledge management (i.e., process of aquiring, organizing, and communicating knowledge of employees in a way that other employees can use it to increase their productivity [2]), we argue that reusing queries from other users is a promising endeavor to support query creation. In this line, we envision our system to be used in addition to current data catalogs to further optimize the time needed for data identification and preparation. As data lakes evolve in the future and potentially include further features, we think that our system and data catalogs can benefit from each other.

To implement our system, the queries of users need to be logged and saved to extract metadata. We estimate the concept of logging queries and extracting metadata also to be transferable to a wide range of mono to multi-zone data lake architectures including data lakes that base on the Hadoop distributed file system. To be more concrete, this approach can also be transfered to NoSQL databases or simple data warehouses. In addition, this approach only makes sense when there are enough data lake users or queries available. Our evaluation further showed that users benefit most from the system if data from various source systems or tables has to be joined.

In our system, we focus on extracting business metadata (e.g., where can I find specific business metadata such as vendor number and what are the descriptions of cryptic column names such as "BWTTY") to help business analysts explore and prepare data for their use cases. In general, however, our presented approach can extract all types of metadata from SQL queries. This mainly depends on the purpose of the query (e.g., find all suppliers in a specific plant or create a new table for additional supplier information). According to [20], there are three types of metadata in a data lake: (1) technical metadata for data type, format and data structure, (2) operational metadata for process history and (3) business metadata for the descriptions of business objective. To support technical and operational metadata, only the data processing workflow needs to be adjusted – as the metadata parser extracts all types of metadata used in the query. For example, when data engineers create new tables in the data lake, they specify the columns and data types for them in the query – so our metadata parser can extract all the metadata used (including technical metadata such as the data type used). To extract the technical metadata, the data processing workflow then needs to be extended to create an additional table with the table columns and their data types.

7 Summary and Conclusion

In this paper, we presented and developed a self-service system that extracts metadata from a data lake and enables business analysts to explore the metadata through an easy-to-use interface (addressing gap 1-4). Hereby, we conducted 15 interviews with business analysts to derive the underlying requirements of the system and evaluated its features with a focus group. Our results showed that participants especially liked the possibility to reuse queries from other users and appreciated the support in query validation as data preparation is a complex and time-consuming endeavour.

In conclusion, we have shown through the productive implementation of our system in a real-world environment with our industry partner that it is feasible to extract metadata from user queries following a bottom-up. This approach has the advantage that metadata does not need to be maintained manually. Therefore, we believe that organizations with data lakes could use this approach as an efficient way to create data catalogs. The proposed architecture and prototype can be used as a guideline for implementing such a system. Finally, our focus group evaluation showed that the system is useful for data exploration tasks.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the article.

References

1. (2023) JSqlParser (4.5 Stable or 4.6 Snapshot). JSQLParser
2. Alavi M, Leidner DE (2001) Review: knowledge management and knowledge management systems: conceptual foundations and research issues. MISQ 25(1):107. https://doi.org/10.2307/3250961 (https://arxiv.org/abs/3250961)
3. Alpar P, Schulz M (2016) Self-service business intelligence. Bus Inf Syst Eng 58(2):151–155. https://doi.org/10.1007/s12599-016-0424-6
4. Drushku K, Aligon J, Labroche N et al (2019) Interest-based recommendations for business intelligence users. Inf Syst 86:79–93. https://doi.org/10.1016/j.is.2018.08.004
5. Ehrlinger L, Schrott J, Melichar M et al (2021) Data catalogs: a systematic literature review and guidelines to implementation. In: Database and expert systems applications – DEXA 2021 workshops, S 148–158 https://doi.org/10.1007/978-3-030-87101-7_15
6. Eichler R, Giebler C, Gröger C et al (2020) HANDLE – A generic metadata model for data lakes. In: Big data Analytics and knowledge discovery, S 73–88 https://doi.org/10.1007/978-3-030-59065-9_7
7. Gröger C (2018) Building an industry 4.0 Analytics platform: practical challenges, approaches and future research directions. Datenbank Spektrum 18(1):5–14. https://doi.org/10.1007/s13222-018-0273-1
8. Gröger C, Hoos E (2019) Ganzheitliches metadatenmanagement im data lake: Anforderungen, IT-werkzeuge und herausforderungen in der praxis. BTW. https://doi.org/10.18420/BTW2019-26
9. Haefliger S, Von Krogh G, Spaeth S (2008) Code reuse in open source software. Manage Sci 54(1):180–193
10. Halevy AY, Korn F, Noy NF et al (2016) Managing Google's data lake: An overview of the Goods system. IEEE Data Eng Bull 39(3):5–14
11. International D (2017) DAMA-DMBOK: data management body of knowledge. Technics Publications, LLC
12. Khoussainova N, Kwon Y, Liao WT et al (2011) Session-based browsing for more effective query reuse. Sci Stat Database Manag. https://doi.org/10.1007/978-3-642-22351-8_47
13. Labadie C, Legner C, Eurich M et al (2020) FAIR enough? Enhancing the usage of enterprise data with data catalogs. In: 2020 IEEE 22nd Conference on Business Informatics (CBI), S 201–210 https://doi.org/10.1109/CBI49978.2020.00029
14. Mathis C (2017) Data lakes. Datenbank Spektrum 17(3):289–293. https://doi.org/10.1007/s13222-017-0272-7
15. Michalczyk S, Nadj M, Azarfar D et al (2020) A state-of-the-Art overview and future research avenues of self-service business intelligence & analytics
16. Paetsch F, Eberlein A, Maurer F (2003) Requirements engineering and agile software development. In: WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, S 308–313 https://doi.org/10.1109/ENABL.2003.1231428
17. Passlick J, Lebek B, Breitner MH (2017) A self-service supporting business intelligence and big data analytics architecture. In: Wirtschaftsinformatik und angewandte Informatik
18. Quimbert E, Jeffery K, Martens C et al (2020) Data cataloguing. In: Towards Interoperable research infrastructures for environmental and earth sciences. Springer, Berlin Heidelberg, S 140–161
19. Quix C, Hai R, Vatov I (2016) Metadata extraction and management in data lakes with GEMMS. CSIMQ. https://doi.org/10.7250/csimq.2016-9.04
20. Ravat F, Zhao Y (2019) Data lakes: trends and perspectives. In: Database and expert systems applications, S 304–313 https://doi.org/10.1007/978-3-030-27615-7_23
21. Seaman C (1999) Qualitative methods in empirical studies of software engineering. IEEE Trans Softw Eng 25(4):557–572. https://doi.org/10.1109/32.799955
22. Spahn M, Kleb J, Grimm S et al (2008) Supporting business intelligence by providing ontology-based end-user information self-service. In: Proceedings of the first international workshop on ontology-supported business intelligence, S 1–12

23. Sulaiman S (2019) Knowledge transfer-based recommendations to enable self-service business intelligence. Shaker, Aachen
24. Sulaiman S, Gómez JM (2018) Recommendation-based business intelligence architecture to empower self service business users. In: Multikonferenz Wirtschaftsinformatik
25. Zaidi E, De Simoni G, Edjlali R et al (2017) Data catalogs are the new black in data management and analytics. https://www.gartner.com/en/documents/3837968, last accessed on 2023-01-10