Midas: Towards an Interactive Data Catalog

Patrick Holl(✉) and Kevin Gossling

Technical University of Munich, 85748 Garching b. Muenchen, Germany
{patrick.holl,kevin.gossling}@tum.de

Abstract. This paper presents the ongoing work on the Midas polystore system. The system combines data cataloging features with ad-hoc query capabilities and is specifically tailored to support agile data science teams that have to handle large datasets in a heterogeneous data landscape. Midas consists of a distributed SQL-based query engine and a web application for managing and virtualizing datasets. It differs from prior systems in its ability to provide attribute level lineage using graph-based virtualization, sophisticated metadata management, and query offloading on virtualized datasets.

Keywords: Polystore · Data catalog · Metadata management

1 Introduction

To provide data consumers, e.g., analysts and data scientists, with the data they need, enterprises create comprehensive data catalogs. These systems crawl data sources for metadata, manage access rights and provide search functionality. Such catalogs are the starting point for almost every analytical task. Once a data scientist has found a potentially interesting dataset in the catalog, he/she has to move to another tool in order to prepare it for analysis. This is because data catalogs often cannot interact with their referenced data sources directly. Instead, engineers have to build ETL pipelines to move and shape data in a way that it is ready for analysis. This process is time consuming, costly, unscalable, and can even lead to the insight that the dataset is unsuitable for the intended task because it is hard to asses the data quality based on raw metadata. Even highly sophisticated systems like Goods from Google require such processes [6]. Another challenge for data catalogs is tracking the provenance of derived datasets, specifically when the schema and the location is different from the origin data. In such cases, the datasets need to be registered manually back to the catalog.

In this paper, we present the ongoing work on the polystore system Midas that tackles the stated problems by providing a large scale data virtualization environment that combines ad-hoc analytical query access with sophisticated metadata management features. Midas is an interactive data catalog designed for data science teams working in heterogeneous data landscapes. In this context, we define interactive as the ability for a data scientist to run large scale ad-hoc queries within the same application that manages the metadata of connected

V. Gadepally et al. (Eds.): DMAH 2019/Poly 2019, LNCS 11721, pp. 128–138, 2019.
https://doi.org/10.1007/978-3-030-33752-0_9

data stores. This approach enables data science teams to share schema details, comments, and other important information in the same place where they access, prepare and analyze the data.

Midas builds upon the concepts of Google's Dremel and other in-situ polystore systems like Apache Drill and Presto [7,12]. It uses a SQL-based query engine as the backbone to provide uniform ad-hoc access to a heterogeneous data landscape and implements a novel approach to represent and virtualize datasets. We are working on graph-based views enriched with arbitrary meta information to represent virtual datasets. This approach allows global lineage tracking down to the attribute level. To achieve interactive performance on large scale datasets and to provide query offloading, we implement an adaptive, columnar-oriented cache that partitions entity attributes based on their occurrence in virtual datasets. Additionally, Midas offers an intuitive web-based user interface to allow for easy data preparation, curation, and sharing among data science teams.

The core concepts of Midas are: Virtualized and sharable datasets, sophisticated metadata management, comprehensible data lineage, interactive performance through adaptive columnar-oriented caches, and ease-of-use.

2 Differentiation to Existing Systems

In the following, we compare Midas to similar systems we have identified. First, we compare it with polystore query engines. Second, we compare it to related data catalog systems.

The publication of the Dremel concept led to several open source implementations of SQL-based query engines that provide ad-hoc access to large, distributed datasets for OLAP use cases [10]. Apache Drill and Presto are the most known open source implementations [7,12]. Both query engines focus on ad-hoc analytical tasks without providing sophisticated user interfaces or metadata management features.

Data catalog systems like Goods from Google [6] or the dataspace concept by Franklin et al. [5] are very close to the goal of Midas. However, Midas is provisioning datasets in a way that a user can directly interact with it and query the actual data without being limited on certain metadata.

The closest system to Midas is Dremio [2]. Dremio is a data management platform based on Apache Drill and Apache Arrow. Similar to Midas, Dremio implements cataloging and lineage features. However, compared to Midas, Dremio does not allow global lineage tracing on attribute level. Furthermore, Dremio does not allow the attachment of arbitrary metadata to attributes.

Extensive research on data lineage has already been done [3,4,13]. Most approaches use annotations to keep track of data transformations and schema changes. Midas does not annotate data but maintains an attribute graph for tracking the lineage.

3 System Design

We are making design decisions specifically tailored to the requirements of agile data science teams. The most important metric is fast access to data in a processable format. Additionally, it must work together with already established workflows and tools like Spark, Jupyter Notebooks, and Tableau. The main components of the Midas system are a query engine that enables users to create virtualizations even on massive scale datasets and the user interface as an interactive data catalog. Figure 1 depicts the overall architecture of the Midas system.

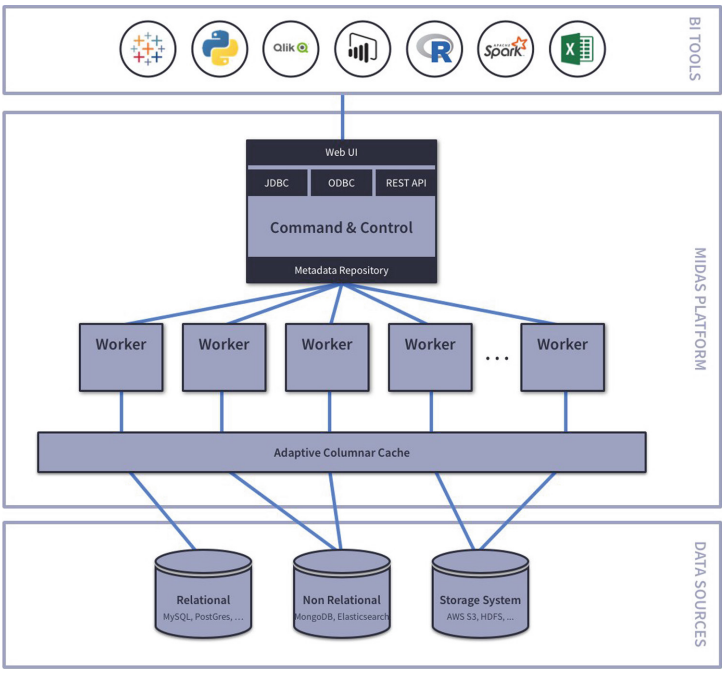

Fig. 1. Midas system architecture

3.1 Query Engine

The primary component of Midas is an extensible, distributed, SQL-based query engine written in Java that follows a similar architecture as Dremel, Presto, and Apache Drill [7,10,12]. The *Command & Control* node takes incoming queries and creates a logical execution plan which is then distributed to the worker nodes that materialize the data.

All three systems are practically proven and show that they can handle analytical queries even on multiple terabyte large datasets [7,10,12]. Internally, Midas uses a columnar-oriented data representation based on Apache Arrow that supports complex data models [1]. The query engine materializes data through an extensible set of connectors, currently, it supports: MySQL, MongoDB, HBase, Hive, Amazon S3, MapR-DB, JSON - File, CSV and Parquet.

All of the listed sources can be virtualized, joined, and queried via SQL.

The main reason for choosing SQL as the main language for interacting with the system is that almost any data scientist knows it and can work with it. Additionally, with open standards like ODBC, almost any tool and framework can directly use it.

3.2 Graph-Based Data Virtualization

Managing metadata and tracking data lineage on attribute level is challenging for polystore systems. Especially when data from multiple stores is combined. Even sophisticated metadata management systems like Goods from Google track provenance on a dataset-basis, i.e., upstream and downstream datasets are tracked as a whole [6]. Combining data from multiple sources makes it complex to reconstruct the origin of a specific attribute. Another challenge for data catalogs is the inheritance of metadata like schema information to derived datasets, especially when attributes are renamed.

To tackle the stated problems, we are working on a novel approach to represent and virtualize datasets. In Midas, a virtual dataset is a view on one or multiple datasets defined by a SQL statement. Technically, Midas implements a rooted graph-based approach to represent these views.

Each dataset D consists of a name N, a list of arbitrary metadata objects $META$ and a set of attribute graphs SAG:

$$D := (N, META, SAG)$$

The name N is an arbitrary string which is usually a reference to the name of a table, a file, or a collection. $META$ is a JSON document that contains arbitrary metadata for a dataset like a description or access rights.

The attribute graph $AG \in SAG$ represents the provenance and metadata of a particular attribute $a \in D$ and denotes as follows:

$$AG := (V, E)$$

The vertex V consists of a name N_a and a list of arbitrary metadata objects $META_a$:

$$V := (N_a, META_a)$$

The edges E denote operations on an attribute.

Example: A data scientist defines a virtual dataset *VD* by creating a view that joins the two dataset D_1 with the attributes a_{1_1} and a_{1_2}, and D_2 with the attributes a_{2_1}, a_{2_2}, and a_{2_3} together. D_1 and D_2 are combined based on their common join key a_{1_1} and a_{2_1}, respectively. The SQL statement looks like the following:

```
CREATE VIEW VD AS (SELECT (a_1_1+a_1_2) as `sum`, a_1_2, a_2_2,
a_2_3 FROM D1, D2 WHERE D1.a_1_1 = D2.a_2_1)
```

Midas takes the incoming query and creates the attribute graphs for *VD*. Figure 2 depicts the set of attribute graphs for *VD*.

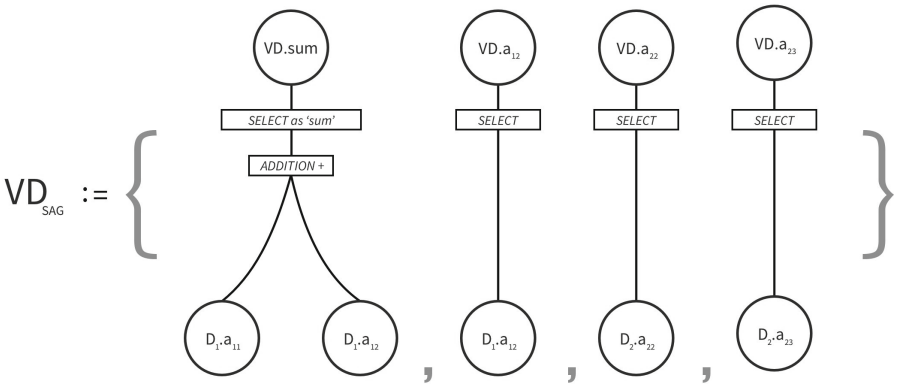

Fig. 2. The set of attribute graphs (SAG) for the virtual dataset VD

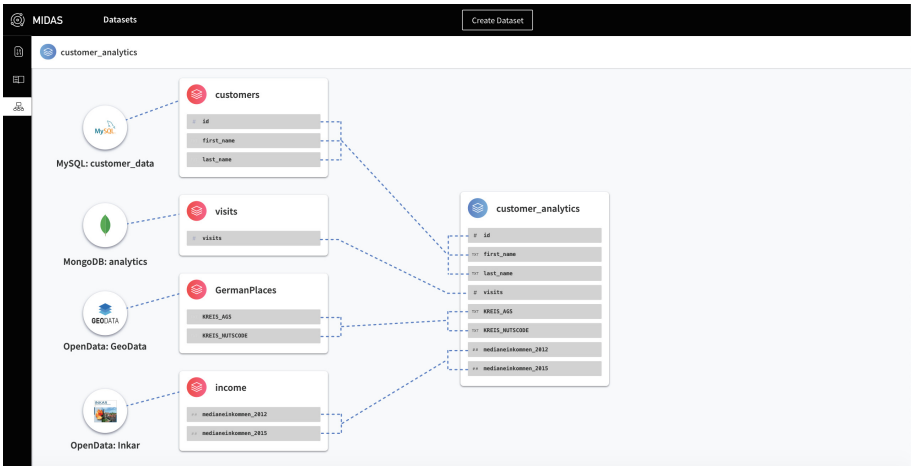

Fig. 3. Full lineage graph of a virtual dataset. The red icons on the left to the dataset names indicate a physical dataset and the blue icon on *customer_analytics* a virtual one. (Color figure online)

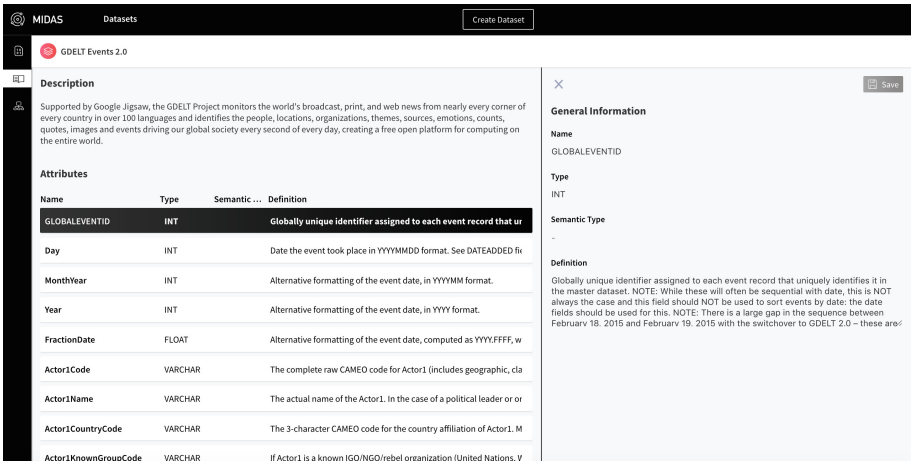

Fig. 4. Interface for editing metadata

By traversing the attribute graph using a depth-first search (DFS) algorithm, we can now visualize the full provenance of an attribute. Additionally, it is possible to add weights to edges for each operation done on an attribute. Having a weighted graph could potentially be used to do cost-based query optimization. However, the determination of the actual costs is tricky since the underlying data sources could be running on different systems in different locations, which makes it hard to do proper cost estimations. Figure 3 shows the full lineage graph of a virtualized dataset in the Midas application.

3.3 Initial Graph Creation and Metadata Management

Midas triggers a discovery process and creates a new graph-based representation of a dataset whenever a user queries it for the first time. During this discovery process, the attribute graphs are created based on the dataset's schema information. For self-describing data stores like Parquet or MySQL, the schema is taken directly from the store. For non-self-describing formats like CSV, the user has to define where to find the schema manually, e.g., on the first row. The actual implementation of the discovery method depends on the individual data store and is defined in the corresponding adapter. For future versions, we are working on the implementation of a crawler-based approach like Goods from Google to facilitate the discovery process on massive heterogeneous data landscapes.

The metadata for datasets and attributes is added separately by other applications via API or manually by users through the Midas catalog interface. Figure 4 shows this catalog interface for data owners and scientists. Currently, the Midas interface supports arbitrary descriptions and the attachment of a semantic type which is a reference to a class or an attribute in an ontology. Creating these links is either done manually by the data owner or automatically by making a lookup in a pre-defined ontology. For now, this lookup is a simple

string match on the DBpedia ontology. The long term goal is to enable data scientists to do semantic queries over an enterprise's data ontology. There has been extensive research on how to tackle the technical challenge of creating ontologys on top of data [9]. However, building these ontologies is not only a technical but also a process challenge. With Midas, we are investigating a human-in-the-loop approach by using a semi-automated mechanism that makes it easy for users to contribute in building the ontology.

3.4 Adaptive Caching Layer

Data virtualization and query federation comes with the advantage that a user always works on a current view of the actual source data. However, achieving interactive performance on massive datasets in such a federated setup is a challenging problem. For achieving interactive query performance, Midas implements an adaptive columnar-oriented cache similar to column caches in Apache Spark.

The implementation is straightforward and the algorithm is as follows:

1. Scan the referenced columns in the logical execution plan.
2. Store the selected columns in a columnar format (Parquet) and add a freshness indicator.
3. For all upcoming queries, do not query the actual source but use the Parquet reference files if the freshness is above a certain threshold.

The cache files are not linked to a certain query but rather adapt to selected columns, i.e., other queries referencing the same columns can use the same cache reference.

For improving the caching behavior, we are currently working on a more sophisticated approach based on query predictions. Recent research in information retrieval shows how search intents and queries can potentially be predicted by using pre-search context and user behavior like past search queries [8]. Querying a dataset using SQL is very similar to querying a search engine, both will lead to a result set of data based on some input parameters. We believe that a similar approach can potentially lead to better caching behavior by pre-calculating result sets based on predicted queries. Current observations in our prototype usage indicate, that more than 75% of all queries contain some aggregate function on one or more attributes. Specifically, we are working on a query prediction model based on past queries of a user, queries of the data science team as a whole, and queries on a certain dataset. The goal of this model is to pre-generate caches for dataset columns that are most likely to be accessed in a query. This approach could potentially lead to more responsive queries but also to offload production systems from analytical workloads in critical times.

3.5 User Experience

The web client is the primary interface for building, managing, and querying datasets. Figure 5 depicts the workspace of a logged-in user where all virtual and physical datasets he/she has access to are listed.

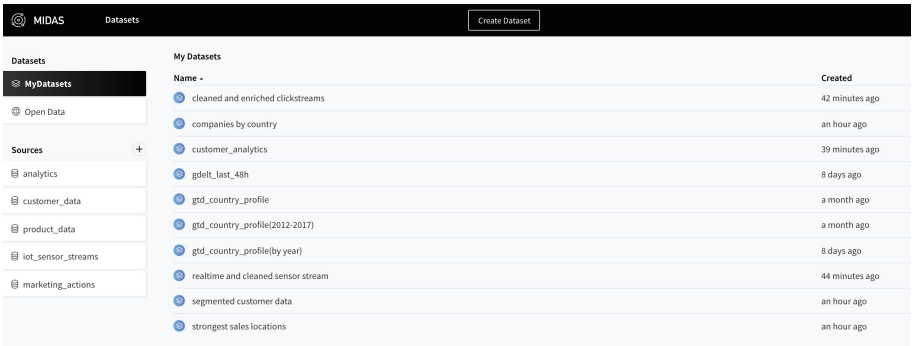

Fig. 5. Workspace containing 10 virtual datasets, 5 physical data stores, and access to third party data hub

Studies show that in the majority of data science teams there is no established process for sharing data [11]. The lack of a clearly defined process often leads to datasets send via email and shadow analytics environments. Midas tackles this problem by providing data science teams a shared workspace for discovering and sharing already integrated data in an analytics-ready format.

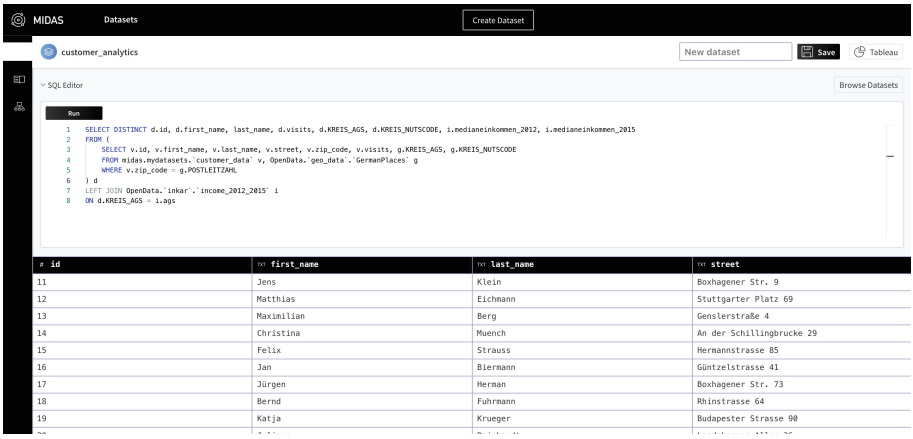

Fig. 6. Interface for running ad-hoc queries and creating new virtual datasets

The main interface for creating a new virtual dataset is shown in Fig. 6. Datasets are created by defining SQL statements which can be saved as a virtual dataset.

Data Discovery. An efficient process to find and explore data is crucial to enable data science teams to fulfill their job. In Midas, we are implementing several approaches to tackle the discovery problem:

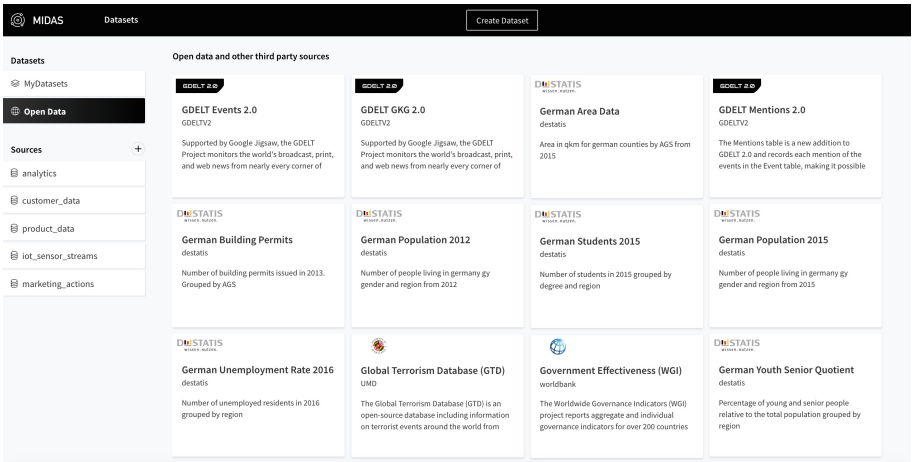

Fig. 7. Interface for the centrally shared open data hub in Midas

1. Providing a search interface for the dataset and attribute metadata.
2. Mapping ontologies to datasets, which potentially allows semantic queries.
3. Implementing a centrally shared hub for open data as shown in Fig. 7.

For modern data science workloads, solely focusing on internal enterprise data might not be enough. For example, whenever data is sparse or lacks features to build proper analytical models. Using open data from the web is a common practice to enrich internal data with additional features. However, integrating publicly available datasets using ETL pipelines is costly, time-consuming and usually requires a search process on the web. Additionally, column names are often cryptic and hard to understand without further information. For making open data more accessible to data scientists, we are working on a central data hub that is shared across teams and organizations. Through the data hub, users can formulate ad-hoc queries on any integrated dataset and collaboratively fill missing attribute descriptions. Figure 7 shows the current user interface of the open data hub.

4 Lessons and Challenges

In this section, we discuss challenges that are occurring while developing Midas and highlight areas that require future work. One of the biggest challenges that we are facing in developing Midas is providing interactive query performance on virtual datasets that are compound of complex queries and span across multiple and different source systems. Limited and sub-optimal query push-downs to the source systems lead to expensive materializations and overhead of used computational resources. Currently available open source implementations of the Dremel concept like Apache Drill and Presto suffer from the same problem. Even though

data virtualization is not the primary use case for those systems, it is similarly essential for ad-hoc analytical queries. Using an efficient and optimized caching structure can help in achieving interactive query performance even on massive virtualized datasets by simultaneously offloading the underlying sources. Query predictions may lead to a significantly better cache hit ratio and therefore to a better overall performance. However, we are currently in the beginning of building and evaluating such algorithms and propose to explore this area further in future research. In addition to that, for reaching the next level in data virtualization, we see it as important to build query federation layers that can leverage the core abilities of a high variety of data store technologies and formats. A potential research direction could be the investigation of learned system components for pushdowns.

Midas focuses on analytical use cases and does, therefore not support write federation to the underlying data stores. It follows the "one size does not fit all" principle, and the expressive power of SQL limits its capability. However, for future work, the system has the potential to support more languages like a limited, read-only subset of SPARQL.

5 Conclusion

In this paper, we outlined the current status of the Midas polystore system tailored to analytical use cases and some challenges for future work. We are currently evaluating the system together with data science teams in three large enterprises (>300.000 combined employees). A video demonstrating the current version of Midas is available at https://demo.midas.science/poly19.

References

1. Apache arrow homepage. https://arrow.apache.org/. Accessed 15 Mar 2019
2. Dremio is the data-as-a-service platform. - dremio. https://www.dremio.com/. Accessed 15 Dec 2018
3. Aggarwal, C.C.: Trio a system for data uncertainty and lineage. In: Aggarwal, C. (ed.) Managing and Mining Uncertain Data, vol. 35, pp. 1–35. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-09690-2_5
4. Cui, Y., Widom, J.: Lineage tracing for general data warehouse transformations. VLDB J. Int. J. Very Large Data Bases **12**(1), 41–58 (2003)
5. Franklin, M., Halevy, A., Maier, D.: From databases to dataspaces: a new abstraction for information management. ACM SIGMOD Rec. **34**(4), 27–33 (2005)
6. Halevy, A., et al.: Goods: organizing Google's datasets. In: Proceedings of the 2016 International Conference on Management of Data, pp. 795–806. ACM (2016)
7. Hausenblas, M., Nadeau, J.: Apache drill: interactive ad-hoc analysis at scale. Big Data **1**(2), 100–104 (2013)
8. Kong, W., Li, R., Luo, J., Zhang, A., Chang, Y., Allan, J.: Predicting search intent based on pre-search context. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 503–512. ACM (2015)

9. Lenzerini, M.: Ontology-based data management. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 5–6. ACM (2011)
10. Melnik, S., et al.: Dremel: interactive analysis of web-scale datasets. Proc. VLDB Endow. **3**(1–2), 330–339 (2010)
11. Tenopir, C., et al.: Data sharing by scientists: practices and perceptions. PLoS ONE **6**(6), e21101 (2011)
12. Traverso, M.: Presto: Interacting with petabytes of data at facebook (2013). Accessed 4 Feb 2014
13. Woodruff, A., Stonebraker, M.: Supporting fine-grained data lineage in a database visualization environment. In: Proceedings 13th International Conference on Data Engineering, pp. 91–102. IEEE (1997)