# Analytic query answering in a Semantic Data Lake (extended abstract)

Claudia Diamantini, Domenico Potena and Emanuele Storti*

*DII, Università Politecnica delle Marche, via Brecce Bianche 60131*

Abstract

The management of Data Lake technologies is challenged by the increasing flexibility they provide in data storage, as well as the fast-changing and diverse data they handle. In order to effectively identify relevant sources for analysis, it is crucial to make sense of disparate data, which is especially important in data science applications where users need to analyze statistical measures from multiple heterogeneous sources. In the paper, a knowledge-based approach for a Semantic Data Lake is presented to enable efficient integration of data sources and alignment to a Knowledge Graph, which represents indicators of interest, their mathematical formulas, and dimensions of analysis. A query-driven discovery approach is used to dynamically identify, integrate and rank the sources to respond to a given analytical query.

**Keywords**

Data Lake, Query-driven discovery, Knowledge graph, Multidimensional model

## 1. Introduction

Data Lakes (DL) are repositories for storing data in their native format, providing centralized access and the capability to apply data transformations when needed according to an ELT approach. However, the lack of a global schema and the need to make sense of disparate raw data pose challenges related to data management. As recognized by recent literature [1], how to integrate heterogeneous data sources and help users to find the most relevant data are still open issues in this setting.

A variety of solutions have been proposed for DL integration, ranging from raw data management to semantic-enriched frameworks. Traditional techniques based on schema matching typically assume complete metadata, which is not realistic for real-world Data Lakes [1, 2]. Among the latter, Knowledge graphs have been exploited to drive integration, relying on information extraction tools (e.g., [2, 3, 4]), while recent effort focused on combining the Ontology-Based Data Access paradigm with Data Lakes to support uniform access (e.g., [5]).

In order to combine the two aspects of discovery and integration, which are often seen as intertwined operations, a *query-driven discovery* paradigm was recently proposed [6] , aimed to finding datasets that are similar to a query dataset and that can be integrated in some way (e.g., by join, union or aggregates). A related problem is the *correlated dataset search*, in which besides identifying possible joins, it is also necessary to compute, or estimate, the joinability

*SEBD 2023: 31st Symposium on Advanced Database System, July 02–05, 2023, Galzignano Terme, Padua, Italy*

*Corresponding author.

c.diamantini@univpm.it (C. Diamantini); d.potena@univpm.it (D. Potena); e.storti@univpm.it (E. Storti)

among the sources (or their correlation). Algorithms such as JOSIE [7] provide an exact solution, while Lazo [8], LSH Ensemble [9] or GB-KMV [10], focus on approximate solutions at the reduced cost of precision and recall. Aurum [11] exploits hypergraphs to find similarity-based relationships through LSH among tabular datasets. In [12], given an input query table, the aim is finding the top-k tables that are both joinable with it and contain columns that are correlated with a column in the query, through a novel hashing scheme that allows the construction of a sketch-based index to support efficient correlated table search. After the discovery has been performed (through join, union or related-table search), tables can be integrated (e.g., [13]).

However, when dealing with *summary* data, that is statistical measures or indicators derived from raw data, specific issues rise that have not been taken into account by the literature. This is the case of open Data Lakes managed by public bodies, e.g., to monitor economic trends or the effectiveness of governmental policies and initiatives like a vaccination campaign.

In this work, we propose a query-driven knowledge-based approach for integration and discovery in a Data Lake. The approach builds on a Knowledge Graph that includes a formal model of measures and their computation formulas, in which concepts are used to enrich source metadata. The approach defines mechanisms for integration and mapping discovery, based on efficient evaluation of set containment between a source domain and a concept in the Knowledge Graph. It also defines an ontology-based and math-aware query answering function, specifically tailored to analytical processing, capable of identifying the set of sources collectively capable of responding to the user request and the proper transformation rules to make the needed calculation. To quantitatively estimate the quality of such results, we propose an algorithm to efficiently evaluate the degree of joinability index, which estimates the cardinality of the join among a set of sources.

Unlike alternative solutions in the literature, our approach takes into account both data and metadata (i.e., mappings to indicators concept in the Knowledge Graph and their formulas) as a support to reformulate the query and determine which sources can be used to respond. This helps in reducing the search space by identifying the most semantically relevant data sources according to the discovery need. Second, in our case the target query is extended to general OLAP queries. As such, it can also be named a *Semantic Data Lakehouse*, following a recent terminological proposal in the literature [14]. The article is an extended abstract of a work submitted to Information Systems Frontiers, a prior version thereof is available at [15].

The rest of the paper is structured as follows: Section 2 is devoted to introducing the Semantic Data Lake model. The approach for source integration is discussed in Section 3, while query answering mechanisms are introduced in Section 4. An evaluation of the approach is discussed in Section 5 while Section 6 concludes the work and draws future research lines.

## 2. Semantic Data Lake: data model

A Semantic Data Lake is defined as a tuple $SDL = \langle \mathcal{S}, \mathcal{G}, \mathcal{K}, m \rangle$, where $\mathcal{S} = \{S_1, \ldots, S_n\}$ is a set of data sources, $\mathcal{G} = \{G_1, \ldots, G_n\}$ is the corresponding set of metadata, $\mathcal{K}$ is a Knowledge Graph and $m \subseteq \mathcal{G} \times \mathcal{K}$ is a mapping function relating metadata to knowledge concepts. Our approach is agnostic w.r.t. both the degree of structuredness of the sources, ranging from structured datasets to semi-structured documents (e.g., XML, JSON), and the specific DL

architecture at hand, e.g., based on ponds vs. zones (see also [16, 17]). If the architecture is pond-based, in fact, the approach is applied to datasets in a single stage, while in zone-based DLs the approach can be applied on any stage of the platform, although it is best suited to the staged area for data exploration/analysis. As a minimum requirement, we assume a data ingestion process to wrap separate data sources and load them into a data storage. The model for a Semantic Data Lake is detailed in the following.

## 2.1. Metadata layer

Different typologies of metadata can be related to a resource, depending on how they are gathered [18]. Hereby, we refer to *technical* metadata, i.e., related to data format and, whenever applicable, to their schema. Since the representation of metadata is highly source-dependent (e.g., the schema definition for a relational table), a uniform representation of data sources in a *metadata layer* is required for the management of a Data Lake. The procedure to represent technical metadata of a given source depends on the typology of data source, e.g., a relational database has tables with attributes, while XML/JSON documents include complex/simple elements and their attributes. For each source $S_k$, metadata are represented as a directed graph $G_k$ that is built incrementally by a *metadata management* system [19], starting from the definition of a node for each metadata element. An edge is defined to represent the *structural* relation between a table and a column of a relational database, or between a JSON complex object and a simple object.

## 2.2. Knowledge layer

The knowledge layer of the Semantic Data Lake is based on *KPIOnto*[1], an OWL2-RL ontology aimed to provide the terminology to model an indicator in terms of description, unit of measurement and mathematical formula for its computation. The ontology also provides classes and properties to fully represent multidimensional hierarchies for dimensions (e.g., level *Province* rolls up to *Country* in the *Geo* dimension) and members.

On this top, a *Knowledge Graph* provides a representation of the domain knowledge in terms of definitions of indicators, dimension hierarchies and dimension members. Concepts are represented in RDF as Linked Data according to the KPIOnto ontology, thus enabling standard graph access and query mechanism. Finally, *Logic Programming rules* are enacted by the XSB[2] logical reasoner providing *algebraic services*. These are capable of performing mathematical manipulation of formulas (e.g., equation solving), which are exploited to infer all formulas for a given indicator. This functionality is used to support query answering (see Section 4).

Figure 1a shows a fragment of a Knowledge Graph representing two dimensions *Time* and *Geo* with the corresponding hierarchy of levels. On the other hand, Figure 1b highlights the mathematical relations among a set of indicators related to monitoring of COVID, some of which are atomic (e.g., *Positive*, *Deaths*, *Recovered*, *ICU*), and others can be calculated from the former (*Cases* and *ICU on Positive Rate*).

[1] KPIOnto specifications are available at http://w3id.org/kpionto
[2] http://xsb.sourceforge.net/

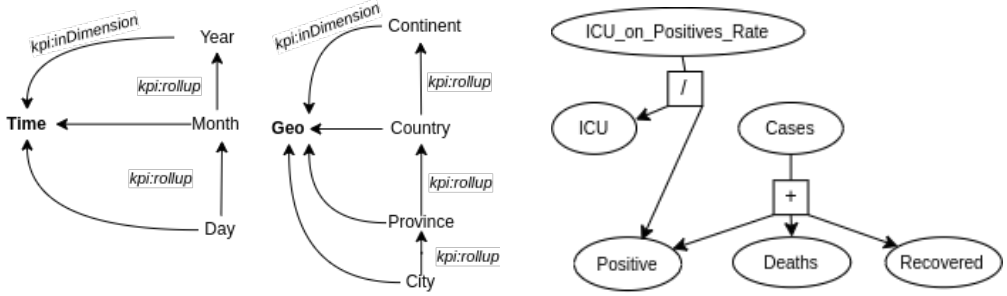


**Figure 1:** Case study: (a) dimensions, levels and (b) indicators with their formulas.

For each source, the nodes in the metadata graphs are aligned with concepts in the Knowledge Graph through the definition of the mapping function $m \subseteq \mathcal{G} \times \mathcal{K}$, which links the metadata to the knowledge layer, following the approach discussed in the next section.

## 3. Integration and mapping discovery

This section is aimed to discuss (a) how to identify dimensions, given a new data source and (b) how to properly map them to the Knowledge Graph. Hereby, we refer to data *domain* as a set of values from a data source, e.g., for relation tables it is the projection of one attribute, while for a JSON collection is the set of values extracted from all the included documents according to a given path.

In order to identify whether a given domain from a data source and a dimensional level represent the same concept, a matching step is required. The Jaccard similarity coefficient is one of the most widely adopted index for comparing sets, however when sets are skewed, i.e., have very different cardinality, this index is biased against the largest one. Given that the cardinality of a domain (without duplicates) is typically much lower than that of a dimensional level, we refer to an asymmetric variant named *set containment*, which is better suited than Jaccard to evaluate whether a domain has intersection with a given level. Given two sets $X, Y$, it is defined as $c(X, Y) = \frac{|X \cap Y|}{|X|}$, i.e. it is independent on the dimension of the second set. As an example, let us consider a domain $A = \{Rome, Berlin, Paris\}$ and a dimensional level *Geo.City* including 100 cities in Europe. In this case, $c(A, Geo.City) = \frac{3}{3}$, meaning that the domain perfectly matches the dimensional level, while $Jaccard(A, Geo.City) = \frac{3}{100}$.

We formalize the problem of mapping a domain of a data source to a dimensional level as a reformulation of the *domain search* problem [9], which belongs to the class of R-nearest neighbor search problems. Here, the goal is to determine what dimensional level, defined in the Knowledge Graph, is the most relevant, i.e. best represent the values in the domain at hand. Formally, given a set of dimensional levels $\mathcal{L}$, a domain $D$, and a threshold $t \in [0, 1]$, the set of relevant dimensional levels from $\mathcal{L}$ is $\{X : c(D, X) \geqslant t, X \subseteq \mathcal{L}\}$. In the following we refer to the most relevant dimensional level as the one having the greatest threshold $t$. As an example, the most relevant dimensional level for a domain *country_region* in a data source, containing

names of countries, is *Geo.Country*.

Comparing a given domain to a dimensional level involves a linear time complexity in the size of the sets. Given the target scenario, which may include data sources with hundreds of thousands or even millions of tuples, the computation of the index may often be not scalable in many practical cases. An improvement discussed in the literature consists in the estimation of the index using MinHash computation [20], which involves performing the comparison on their MinHash signatures instead of on the original sets. If data sources have high dimensionality, however, MinHash is used with a data structure capable of significantly reducing the running time, named Locality Sensitivity Hashing (or LSH) [21], a sub-linear approximate algorithm.

While the previous approach is targeted to the Jaccard index, an estimation of the set containment can be obtained through LSH Ensemble [9], which is proved to be suitable for skewed sets and more performing than alternative solutions in terms of accuracy and execution time. In our approach, given a domain of a source, we rely on LSH Ensemble to obtain the dimensional level(s) that are estimated to have a containment score above a certain threshold. In the following, given a domain $D$ from a data source $S$, given a set of dimensional levels $\mathcal{L}$, and a threshold $t \in [0, 1]$, we refer to $LSH\_Ensemble$ as a function returning the set of relevant dimensional levels for $D$.

For what concerns measures, they are particular domains which are purely quantitative. As such, unlike dimensional levels, they are not constrained to a finite number of possible values. For this reason, solutions for evaluating domain similarity through containment such as LSH Ensemble cannot be applied. Several approaches can be considered ranging from string-based ones to those based on dictionary, semantic similarity (e.g., [19]) or frequency distribution and will be discussed in future work.

## 4. Query answering

The mappings defined between the metadata graphs and the Knowledge Graph are exploited to support query-driven discovery and query answering in the Data Lake context. This requires to determine what data sources are needed and how to combine them for a given request. A user query $Q$ is expressed as a tuple $Q = \langle ind, \{L_1, \ldots, L_n\}\rangle$, where $ind$ is an indicator and $\{L_1, \ldots, L_n\}$ is a set of levels, each belonging to a different dimension.

A data source $S$ has a compatible dimensional schema with respect to a query if $S$ contains a subset of the levels in the query. For all dimensions of the query that are not included in $S$, the source is assumed to supply such dimensions at the most aggregate level. A data source can respond a query if its dimensional schema is compatible and if it provides the requested indicator. On the other hand, if the indicator is not provided by any source but it can be calculated from other indicators, a set of data sources may collectively answer the query if they have a compatible dimensional schema and provide all the component indicators. In the latter case, the actual calculation of the indicator requires to join the needed data sources.

It is worth noting that multiple formulas may exist to calculate an indicator and also for each formula there may be multiple sets of sources that have the necessary measures. Clearly, the different solutions must be compared to assess the quality of the query result. To this end, it is necessary to join the sources considered in each solution. This is highly inefficient in the

**Algorithm 1** Computing degree of joinability

```
1: function COMPUTE_JOINABILITY(⟨S_1, ..., S_m⟩, {L_1, ..., L_n})
2:     τ = 1
3:     Search S* ∈ {S_1, ..., S_m} s.t. |S*| = min_{i=1,...,m} |S_i|
4:     flag = True
5:     while flag do
6:         Λ = LSH_Ensemble(S*, {S_1, ..., S_m}\S*, τ)
7:         if |Λ| < (m − 1) then τ = τ − τ_step
8:         else flag = False
9:         end if
10:    end while
11:    return τ
12: end function
```

context of a Data Lake. Therefore, we propose an efficient algorithm to estimate the quality of the query result, in terms of its cardinality. The outcome of the algorithm is then used to choose which sources will be joined to compute the query result.

The algorithm takes as input a query $Q = \langle ind, \{L_1, \ldots, L_n\}\rangle$ and returns the list of possible solutions, in terms of the formula to be applied and sources to be considered, enriched with the estimated cardinality of the result. First, using the reasoning services defined over the KPIOnto, the algorithm searches for all formulas $f(ind_1, \ldots, ind_m)$ for $ind$ that can be derived from $\mathcal{K}$, such that each component measure $ind_i, i = 1, \ldots, m$ is provided by a data source with a dimensional schema compatible with $Q$. For each formula in $f(.)$, the sets of sources that can provide $ind_1, \ldots, ind_n$ are also returned. On these sets the *degree of joinability* is calculated, which is used to estimate the cardinality of the query result. Such index measures the likelihood to produce a result out of a join among a set of domains.

Sources are joinable if they have the same values for domains that are mapped to the same dimensional levels. To check this condition, the corresponding domains should be compared in order to determine how many values are shared between the sources through set containment. However, a full comparison is not practical in a Data Lake scenario. For this reason, we resort to the LSH Ensemble to provide an estimated evaluation of the joinability of $m$ data sources. Typical use of LSH Ensemble is based on single join attribute at a time (similarity between sets), while in our case the match needs to be performed on sets of dimensional levels. Hence, we apply a combination function (e.g., a concatenation of strings) to the domains representing levels, in order to map them into a single domain before applying the hashing function. In the following, we refer to combined MinHashes, that can be pre-computed at source loading time in order to speed up the evaluation of the joinability index.

The procedure for computing the degree of joinability is summarized in Algorithm 1. Given the set of sources $\{S_1, \ldots, S_m\}$ with $S^*$ being the one with the lowest cardinality, the algorithm returns the portion of elements of $S^*$ that will be considered in computing the join with the other sources. Since the set $\{L_1, \ldots, L_n\}$ defines a unique identifier for each $S_i$, multiplying the degree of joinability by $|S^*|$ yields the estimation of the cardinality of the join. In case the indicator is already available in a source, the cardinality of the query result is equal to the cardinality of the source. As a first step (line 2), the threshold $\tau$ is set to the maximum value. Then, after identifying the source $S^*$ with the lowest cardinality (line 3), the function *LSH_Ensemble* is called to obtain the set of sources with which S* is estimated to have a

| Source | Cardinality | # Domains | Mapped levels | Mapped measures |
|---|---|---|---|---|
| S1 | 3051712 | 17 | Time.Day, Geo.Country | Positive, Recovered, Deaths |
| S2 | 22261 | 31 | Time.Day, Geo.Province | ICU, Positive, Negative, Recovered, Deaths |
| S3 | 61900 | 14 | Time.Day, Geo.Country | Positive, Deaths |
| S4 | 231192 | 38 | Time.Day, Geo.Country | Cumulative_Positive, Cumulative_Deaths |
| S5 | 28661 | 8 | Time.Day, Geo.Country | ICU, Cumulative_ICU |

**Table 1**
Details on the data sources for the case study, mapped levels and measures.

containment score above $\tau$. If there is at least one source for which this does not hold, then the degree of joinability is less than $\tau$ and the threshold is decreased by a given step (line 7).

It is noteworthy that Algorithm 1 returns an overestimate of the degree of joinability of $m$ sources. To give an example, if $S_x = \{a, b, c\}$, $S_y = \{a, b, d, e\}$ and $S_z = \{b, c, d, e\}$, the *compute_joinability* returns $\frac{2}{3}$, but the cardinality of the join is 1, so the degree of joinability should be $\frac{1}{3}$ of $S_x$. To get a more accurate result MinHash could be directly used to estimate the set containment, and then to perform the join among the $m$ sources. Clearly this solution lengthens the computation time, so for the scenario of this work we consider the approximation proposed above.

## 5. Evaluation

An evaluation[3] of the approach is proposed here on a case study based on the Microsoft Azure Covid-19 Data Lake[4] and Our World in Data repository[5]. The Data Lake contains 5 sources reporting several measures on COVID, aggregated by temporal and geographical dimensions (basic informations are available in the first columns of Table 1).

The Knowledge Graph was setup by defining dimensions and levels from available online resources. For any loaded data source, initialization includes computation of MinHashes for any domain, mapping with the dimensional levels and computation of the combined MinHashes for domains mapped to dimensional levels. For LSH Ensemble we set the number of hashing permutations to 256 and number of parts to 32. The average execution time for hashing computation for a domain ranges from 0.076 s (S3) to 28.125 s (S1). The mapping discovery phase always requires less than 0.001 s per domain, while the time for computation of combined MinHashes ranges from 0.151 s (S2) to 21.235 s (S1) per domain. Overall, domains are processed in less than 1.6 s on average.

In the following, we report an example of the application of the algorithms on the case study. The result of the mapping discovery is shown in Table 1, where mapped levels and measures are reported for each source. Let us assume the user is interested in analysing measures *ICU_on_Positives_Rate* and *Positive* at *Geo.Country* and *Time.Day* levels. As for the first measure, the algorithm returns $(Positive, \{\{S1\}, \{S3\}\})$. In this case, no join is needed as the measure is directly available from multiple sources. Therefore, the degree of joinability is equal to 1.

[3]Tests have been carried out on an Intel Core i5-1135G7, 8 cores @ 2.40GHz, x86_64 architecture, with 8 GB RAM running Linux Fedora 34.
[4]https://docs.microsoft.com/en-us/azure/open-datasets/dataset-covid-19-data-lake
[5]https://github.com/owid/covid-19-data

As for the second measure, the function returns $(\frac{ICU}{Positives}, \{\{S5\}, \{S1, S3\}\})$. Combination of sources are produced and two alternative solutions are available by combining S5 with either S1 or S3. They are checked for joinability as follows, considering that the cardinality of S5 is 28661:

- $\langle S5, S1\rangle$: the degree of joinability between S5 and S1 is 0.78. Hence, the estimated join cardinality is 0.78 * 28661 = 22355 with a query time equal to 3.109 s;
- $\langle S5, S3\rangle$: the degree of joinability between S5 and S3 is 0.31. Hence, the estimated join cardinality is 0.31 * 28661 = 8884, with a query time equal to 3.283 s.

As a result, the solution (S5,S1) is preferred over (S5,S3). This is motivated by the fact that S5 and S1 include data for both years 2020 and 2021, while S3 includes data only on year 2020. Therefore, the degree of joinability of S3 with S5 is lower than that of S1, as the former shares a smaller subset of data with the latter.

## 6. Conclusion

This paper has introduced a knowledge-based approach for analytic query-driven discovery in a Data Lake, which is characterized by the formal representation of indicators' formulas and efficient mechanisms for source integration and mapping discovery. Given a query ontologically expressed as a measure of interest and relevant analysis dimensions, the framework identifies sources capable of collectively responding by utilizing math-aware reasoning on indicator formulas. The joinability of sources is quantitatively evaluated through the degree of joinability index. With respect to previous work on query-driven discovery, which requires a number evaluations among sources increasing linearly with their number, our approach reduces such a number to only the relevant sources by performing a preliminary evaluation based on mapping to domains in the Knowledge Graph and formula rewriting.

Future work will be devoted to define a more comprehensive metadata model for the Data Lake, including also operational and business metadata. We also aim to extend the query answering approach towards interesting research directions. In particular, the degree of joinability could be adapted to evaluate the completeness of a data source with respect to the Knowledge Graph concepts. This would enable to determine the scope of a source and paves the way for an efficient evaluation of the overlapping or complementarity among sources, and possible more efficient indexing approaches. Merging capabilities could also be beneficial to find unionable sources and hence to vertically integrate data providing the same measures. Finally, dynamic calculation of indicators can be envisaged for a variety of analytical tasks, including interactive data exploration [22] or navigation [23]. Furthermore, we plan to individuate real case studies for an extensive evaluation, which will help in more precisely identify potential benefits and limitations for specific application contexts.

## References

[1] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, P. C. Arocena, Data lake management: challenges and opportunities, Proceedings of the VLDB Endowment 12 (2019) 1986–1989.

[2] M. Farid, A. Roatis, I. Ilyas, H. Hoffmann, X. Chu, CLAMS: bringing quality to Data Lakes, in: Proc. of the International Conference on Management of Data (SIGMOD/PODS'16), San Francisco, CA, USA, 2016, pp. 2089–2092. ACM.
[3] R. C. Fernandez, E. Mansour, A. A. Qahtan, A. Elmagarmid, I. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, N. Tang, Seeping semantics: Linking datasets using word embeddings for data discovery, in: 2018 IEEE 34th International Conference on Data Engineering (ICDE), IEEE, 2018, pp. 989–1000.
[4] R. Hai, S. Geisler, C. Quix, Constance: An intelligent data lake system, in: Proc. of the International Conference on Management of Data (SIGMOD 2016), San Francisco, CA, USA, 2016, pp. 2097–2100. ACM.
[5] M. N. Mami, D. Graux, S. Scerri, H. Jabeen, S. Auer, J. Lehmann, Uniform access to multiform data lakes using semantic technologies, in: Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services, 2019, pp. 313–322.
[6] R. J. Miller, Open data integration, Proc. VLDB Endow. 11 (2018) 2130–2139.
[7] E. Zhu, D. Deng, F. Nargesian, R. J. Miller, Josie: Overlap set similarity search for finding joinable tables in data lakes, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 847–864.
[8] R. C. Fernandez, J. Min, D. Nava, S. Madden, Lazo: A cardinality-based method for coupled estimation of jaccard similarity and containment, in: 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, 2019, pp. 1190–1201.
[9] E. Zhu, F. Nargesian, K. Q. Pu, R. J. Miller, Lsh ensemble: Internet-scale domain search, Proc. VLDB Endow. 9 (2016) 1185–1196.
[10] Y. Yang, Y. Zhang, W. Zhang, Z. Huang, Gb-kmv: An augmented kmv sketch for approximate containment similarity search, in: 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, 2019, pp. 458–469.
[11] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, M. Stonebraker, Aurum: A data discovery system, in: 2018 IEEE 34th International Conference on Data Engineering (ICDE), IEEE, 2018, pp. 1001–1012.
[12] A. Santos, A. Bessa, C. Musco, J. Freire, A sketch-based index for correlated dataset search, in: 2022 IEEE 38th International Conference on Data Engineering (ICDE), IEEE, 2022, pp. 2928–2941.
[13] A. Khatiwada, R. Shraga, W. Gatterbauer, R. J. Miller, Integrating data lake tables, Proc. VLDB Endow 16 (2022) 932–945.
[14] M. Armbrust, A. Ghodsi, R. Xin, M. Zaharia, Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics, in: Proceedings of CIDR, 2021, p. 8.
[15] C. Diamantini, D. Potena, E. Storti, A knowledge-based approach to support analytic query answering in semantic data lakes, in: Advances in Databases and Information Systems: 26th European Conference, ADBIS 2022, Turin, Italy, September 5–8, 2022, Proceedings, Springer, 2022, pp. 179–192.
[16] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, B. Mitschang, Leveraging the data lake: Current state and challenges, in: C. Ordonez, I. Song, G. Anderst-Kotsis, A. M. Tjoa, I. Khalil (Eds.), Big Data Analytics and Knowledge Discovery, Springer International Publishing, Cham,

2019, pp. 179–188.
[17] P. Sawadogo, J. Darmont, On data lake architectures and metadata management, Journal of Intelligent Information Systems 56 (2021) 97–120.
[18] A. Oram, Managing the Data Lake, O'Reilly, Sebastopol, CA, USA, 2015.
[19] C. Diamantini, P. L. Giudice, D. Potena, E. Storti, D. Ursino, An approach to extracting topic-guided views from the sources of a data lake, Information Systems Frontiers 23 (2021) 243–262.
[20] A. Z. Broder, On the resemblance and containment of documents, in: Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171), IEEE, 1997, pp. 21–29.
[21] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, in: Proceedings of the thirtieth annual ACM symposium on Theory of computing, 1998, pp. 604–613.
[22] C. Diamantini, D. Potena, E. Storti, H. Zhang, An ontology-based data exploration tool for key performance indicators, Lecture Notes in Computer Science 8841 (2014) 727–744.
[23] E. Zhu, K. Q. Pu, F. Nargesian, R. J. Miller, Interactive navigation of open data linkages, Proc. VLDB Endow. 10 (2017) 1837–1840.