A Knowledge-Based Approach to Support Analytic Query Answering in Semantic Data Lakes

Claudia Diamantini, Domenico Potena, and Emanuele Storti(✉)

DII, Polytechnic University of Marche, Ancona, Italy
{c.diamantini,d.potena,e.storti}@univpm.it

Abstract. The increased flexibility brought by Data Lake technologies, along with size and heterogeneity of quickly changing data sources, bring novel challenges to their management. Making sense of disparate data and supporting users to identify the most relevant sources for a given analytic request are indeed critical requirements to make data actionable. This is particularly relevant in data science applications, where users want to analyse statistical measures from a variety of data sources. To this aim, in the paper we introduce a knowledge-based approach for a Semantic Data Lake, capable of supporting efficient integration of data sources and their alignment to a Knowledge Graph representing indicators of interest, their mathematical formulas and dimensions of analysis. By leveraging manipulation of indicator formulas, a query-driven discovery approach is exploited to dynamically identify the sources, along with the needed transformations, to respond a given .

Keywords: Data Lake · Query-driven discovery · Knowledge Graph · Multidimensional model

1 Introduction

Data Lakes (DL) have recently emerged as schema-agnostic repositories for storing data in their native format, providing centralized access and the capability to apply data transformation when needed according to an ELT (Extraction, Load, Transformation) approach. This increased flexibility, along with size and heterogeneity of growing data sources bring novel challenges related to data management. In particular, the lack of a global schema and the need to make sense of disparate raw data require proper modeling of their metadata, to make data actionable and avoid data swamps (see also [15]). As recognized by recent literature (e.g., [13]), how to integrate heterogeneous data sources and help users to find the most relevant data are still open issues in this setting and are often seen as intertwined operations. This is particularly relevant in data science applications, where users want to analyse statistical measures from a variety of data sources. Examples include Open Data Lakes managed by public bodies, e.g., to monitor the effectiveness of governmental initiatives like a vaccination campaign, or analysing outcomes from Open Science collaborative projects.

To address such challenges, a novel paradigm called *query-driven discovery* was proposed to combine the two aspects [12], following the idea to find datasets

© Springer Nature Switzerland AG 2022
S. Chiusano et al. (Eds.): ADBIS 2022, LNCS 13389, pp. 179–192, 2022.
https://doi.org/10.1007/978-3-031-15740-0_14

that are similar to a query dataset and that can be integrated in some way (either by joins, unions or aggregates).

In the literature, a variety of solutions have been proposed for DL integration, ranging from approaches based on raw data (and related metadata) management to semantic-enriched frameworks, the latter being intrinsically more suitable to address issues related to data variety/heterogeneity and data quality. Among them, some work focused on holistic models capable of addressing a variety of structures, e.g., in [8] a DL system is proposed, that discovers, extracts, and summarizes the structural metadata from the (semi)structured data sources, and annotates (meta)data with semantic information to avoid ambiguities, while in [2] a network-based model to represent technical metadata of structured, semi-structured and unstructured data sources is proposed. Knowledge graphs are exploited in [5] to drive integration, relying on information extraction tools, e.g., Open Calais, that may assist in linking metadata to uniform vocabularies, while in [6] a graph is built by a semantic matcher, leveraging word embeddings to find links among semantically related data sources.

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 including a formal model of measures (also named indicators) and their computation formulas [4], in which concepts are used to enrich source metadata. On top of the model, the contributions of this work are multifold:

- We define mechanisms for *integration* of data sources into the Semantic Data Lake and *mapping discovery*, based on efficient evaluation of set containment [16] between a source domain and a concept in the Knowledge Graph.
- We define an ontology-based and math-aware *query answering* function, capable of identifying the set of sources collectively capable of responding the user request, and the proper transformation rules to make the needed calculation. For instance, let us suppose a user is interested in analysing measure $CO_2PerPerson$, but it is not available in any source. Given that such a measure can be calculated as $\frac{TotalCO_2}{Population}$, a response can be obtained by combining sources providing the two components $TotalCO_2$ and $Population$ measures.
- To quantitatively estimate the quality of such results, we define a *degree of joinability* index, that evaluates to what extent the sources are joinable, i.e., how much they share the same values over the same attributes.

With respect to the content-driven notion of query-driven discovery that was proposed in [12], our approach also considers 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. The rest of the paper is structured as follows: in Sect. 2 a case study is introduced that will be used throughout the paper. Section 3 is devoted to introduce the Semantic Data Lake model. The approach for source integration is discussed in Sect. 4, while query answering mechanisms are detailed in Sect. 5. Section 6 discusses an evaluation of the approach. Finally, Sect. 7 concludes the work and draws future research lines.

2 Case Study: Azure COVID-19 Data Lake

In this work we take as example a set of COVID-19 related datasets from the Microsoft Azure Covid-19 Data Lake [11] and Our World in Data repository:

S1) *Bing COVID-19 Data*[1], which includes confirmed, fatal, and recovered cases per country/region, updated daily for years 2020–2021.
S2) *COVID Tracking Project*[2], with numbers on tests, confirmed cases, hospitalizations daily from every US state for years 2020–2021.
S3) *European Centre for Disease Prevention and Control (ECDC) Covid-19 Cases*[3], which includes the latest available public data on COVID-19 cases worldwide from the European Center for Disease Prevention and Control (ECDC), reported per day and per country for year 2020.
S4) *Oxford COVID-19 Government Response Tracker (OxCGRT)* [9], which contains systematic information on measures against COVID-19 taken by governments, for years 2020–2021.
S5) *Open World in Data*[4], which contains data on the number of people in hospitals and ICU per day and country, for years 2020–2021.

In Table 1 we summarize relevant detail about the sources, that are derived from the source metadata provided by the publishers.

Table 1. Details of sources S1-S5.

Source	# Rows	# Cols	Measures	Dimensions
S1	3051712	17	confirmed, confirmed_change, deaths, deaths_change, recovered, recovered_change	updated, country_region, admin_region, iso2, iso3, iso_subdivision
S2	22261	31	positive, negative, death, recovered, hospitalized_currently, in_icu_currently, in_icu_cumulative, on_ventilator_currently, on_ventilator_cumulative, pending	date, iso_country, state, iso_subdivision
S3	61900	14	cases, deaths	date_rep, continent_exp, countries_and_territories, iso_country, geo_id, country_and_territory_code
S4	231192	38	confirmedcases, confirmeddeaths	countryname, countrycode, date, ISO_country
S5	28661	8	daily ICU occupancy, daily ICU occupancy per million, daily hospital occupancy, daily hospital occupancy per million	entity, ISO_code, date

[1] https://www.bing.com/covid.
[2] https://github.com/COVID19Tracking/covid-tracking-data.
[3] https://www.ecdc.europa.eu/en/covid-19/data-collection.
[4] https://github.com/owid/covid-19-data.

3 Semantic Data Lake: Data Model

In this Section, we briefly review the model for a Semantic Data Lake that was discussed in [4], on top of which the source integration, mapping discovery and query answering mechanisms will be defined, as discussed in next sections.

We define a Semantic Data Lake 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 (e.g., XML, JSON) documents, and the specific DL architecture at hand, e.g., based on ponds vs. zones (see also [7,15]). 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.

3.1 Metadata Layer

Different typologies of metadata can be related to a resource, depending on how they are gathered [14]. 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 $G_k = \langle N_k, A_k, \Omega_k \rangle$, where N_k are nodes, A_k are edges and $\Omega_k : A_k \rightarrow \Lambda_k$ is a mapping function s.t. $\Omega_k(a) = l \in \Lambda_k$ is the label of the edge $a \in A_k$. The graph is built incrementally by a *metadata management* system [2], starting from the definition of a node $n \in N_k$ for each metadata element. An edge $(n_x, n_y) \in A_k$ is defined to represent the *structural* relation existing between the elements o_x, o_y, e.g., this corresponds to the relations between a table and a column of a relational database, or between a JSON complex object and a simple object. Further details on this modeling approach are available in [2].

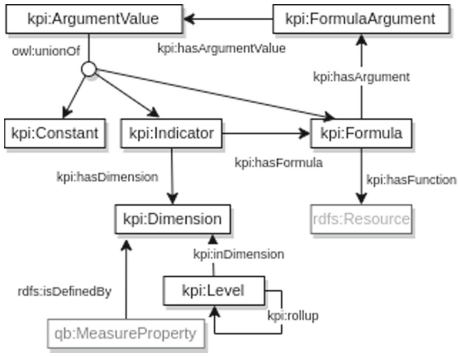

Fig. 1. Main classes and properties in KPIOnto ontology.

3.2 Knowledge Layer

The knowledge layer of the Semantic Data Lake comprises:

- *KPIOnto*[5] 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. The main classes and properties, including those aimed at representating a formula in terms of operands and operator, are shown in Fig. 1.
- a *Knowledge Graph* $\mathcal{K} = \langle K_N, K_A, K_\Omega \rangle$, where K_N and K_A respectively represent nodes and edges, while K_Ω is a mapping function assigning labels to edges. It 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.
- *Logic Programming rules*, which are enacted by a logical reasoner (namely, XSB[6]) to automatically provide *algebraic services*, 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 Sect. 5).

Figure 2 shows (a) a fragment of the Knowledge Graph for the case study representing dimensions *Time* and *Geo* with the corresponding levels, and (b) highlights the mathematical relations among a set of indicators. The full

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

list of measures defined in the Knowledge Graph is as follows: *Positive*, *Cumulative_Positive*, *Negative*, *Deaths*, *Cumulative_Deaths*, *Recovered*, *Cases*, *ICU*, *Cumulative_ICU*, *ICU_on_Positive_Rate*.

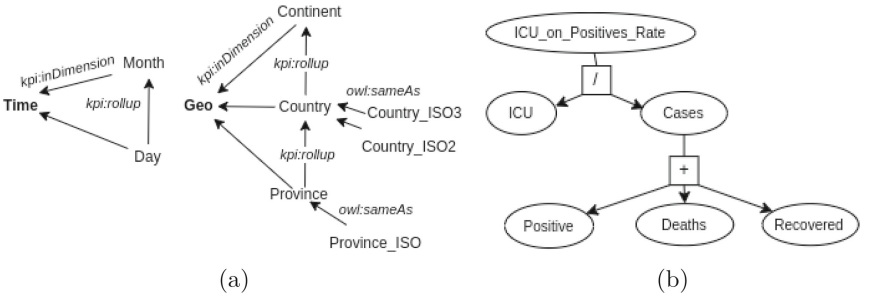

Fig. 2. Case study: (a) dimensions, levels and (b) indicators with their formulas.

4 Integration and Mapping Discovery

This section is aimed to discuss (a) how to identify, given a new data source, dimensions and measures, and (b) how to properly map them to the Knowledge Graph. In the following, we refer to data *domain* as a set of values from a data source. If the data source is a relation table, a domain can be seen as the projection of one attribute. Conversely, if the data source is a JSON collection, a domain is the set of values extracted from all the included documents according to a given path (e.g., using JSONPath expression). As a result, a data source corresponds to a set of domains.

Identification of Dimensions. In order to identify whether a given domain from a data source (e.g., the attribute *countryname* in S4) and a dimensional level (e.g., *Geo.Country*) represent the same concept, a matching step is required. One of the most widely adopted index for comparing sets is the Jaccard similarity coefficient, aimed at measuring the similarity between finite sets as the ratio between their intersection and their union. When sets are skewed, i.e., have very different cardinality, this index is however biased against the largest one. In such contexts an asymmetric variant can be used, namely the *set containment*, that is independent on the dimension of the second set.

Definition 1 *(Set Containment). Given two sets X,Y, the set containment is given by* $c(X,Y) = \frac{|X \cap Y|}{|X|}$.

Given that the cardinality of a domain (without duplicates) is typically much lower than that of a dimensional level, this index is better suited than Jaccard to evaluate whether a domain has intersection with a given level[7]. For this reason, we rely on set containment in our work. 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 [16], which belongs to the class of R-nearest neighbor search problems. We first give the definition of relevant dimensional level for a given domain as follows.

Definition 2 *(Relevant dimensional levels for a domain). 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 $\{X : c(D, X) \geqslant t, X \subseteq \mathcal{L}\}$.*

The number of relevant dimensional level for a domain may be greater than one, although in practice we are interested in the level with the greatest threshold t, i.e., the most relevant dimensional level. As an example, the most relevant dimensional level for domain *country_region* in data source S1 is *Geo.Country*, while for *iso_subdivision* is *Geo.Province_ISO*.

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 hundred 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 as for the Jaccard index consists in its estimation using MinHash computation [1], which involves performing the comparison on their MinHash signatures instead of on the original sets. Given a hash function h, a domain can be mapped to a corresponding set of integer hash values of the same length. For a domain X, let $h_{min}(X)$ be the minimum hash value. Given two sets X,Y, the probability of their minimum hash values being equal is the Jaccard index, i.e., $P(h_{min}(X) == h_{min}(Y)) = J(X, Y)$. Since the comparison can only be true or false, this estimator has a too high variance for a useful estimation of the Jaccard similarity. However, an unbiased estimate can be obtained by considering a number of hashing functions and averaging results: this is done by counting the number of equivalences in the corresponding minimum hash values and dividing by the total number of hash values for a set.

If data sources have high dimensionality, however, pair-wise comparison is still highly time consuming. In our scenario, for a source with N domains and M dimensional levels the time complexity is in $O(N * M)$. For such a reason, in practice MinHash is used with a data structure capable of significantly reducing the running time, named Locality Sensitivity Hashing (or LSH) [10], a sub-linear approximate algorithm.

[7] Under this assumption, the set containment is equivalent to the *overlap* (or Szymkiewicz–Simpson) coefficient, i.e., $\frac{|X \cap Y|}{|min(X,Y)|}$.

While the previous approach is targeted to the Jaccard index, an estimation of the set containment can be obtained through LSH Ensemble [16], 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, we rely on LSH Ensemble to obtain the dimensional levels with which a given domain of a source is estimated to have a containment score above a certain threshold.

Definition 3 *(LSH Ensemble). Given a domain D from a data source S, given a set of dimensional levels $\mathcal{L}$, and a threshold $t \in [0,1]$, LSH_Ensemble is a function returning the set of relevant dimensional levels for D.*

Identification of Measures. In terms of dataset attributes, measures are particular domains which are purely quantitative. As such, unlike dimensional levels, a measure belongs to a certain data type but is 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.

In this work we rely on a string-similarity approach, namely LCS (Longest Common Subsequence) in the comparison of the attribute names of a data source with the list of measure names in the Knowledge Graph. For each domain, the measure names that have the highest value of LCS, i.e., that are most similar, are returned. This is useful to propose only a subset of the measures defined in the Knowledge Graph to the DL Manager. To make an example, for S2 the measure *in_icu_cumulative* is mapped to the Knowledge Graph measure *Cumulative_ICU*, which is the closest syntactically.

We'd like to note that however a manual revision is ultimately required, as the recognition can be affected by homonyms and unclear or ambiguous wording of the domain names. For instance, for S3 the measure *cases* gets mapped to the Knowledge Graph measure *Cases*, but its meaning is different: indeed, by reviewing the publisher metadata, it is clear that instead it actually accounts for the number of positive cases. As such, it needs to be mapped to the measure *Positive*. More advanced approaches could be considered for this step, including some based on dictionary, semantic similarity (e.g., [2]) or frequency distribution and will be discussed in future work.

Representation of Mappings. Given a domain of a data source and the most relevant dimensional level with respect to a given threshold, the domain is mapped to the corresponding level in the Knowledge Graph.

Definition 4 *(Set of mappings). Let $\mathcal{K}$ be the Knowledge Graph, G_S be a metadata graph for a source S, $D \subseteq S$ be a domain, $L \in \mathcal{L}$ be the most relevant dimensional level for D, the mapping between D and L is defined as a tuple $m{=}(n_D, n_L, c(D,L))$. The set of mappings M_{G_S} includes all mappings for dimensions in S.*

Similarly, given a domain and a related identified measure, a mapping between the corresponding nodes is created. In the following, we represent by

Dim_S the set of dimensional levels available in a source S and by $Ind_S \subseteq \mathcal{I}$ the set of measures available in S, where $\mathcal{I}$ is the set of indicators defined in $\mathcal{K}$.

5 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.

Definition 5 *(Compatible dimensional schemas). Given a data source S, given a query $Q = \langle ind, \{L_1, \ldots, L_n\}\rangle$, the dimensional schema of S is compatible with Q iif $Dim_S \subseteq \{L_1, \ldots, L_n\}$.*

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. For instance, if a query requires indicator *Positive* for *Country*, *Day* and *Age group*, data source $S2$ has a compatible dimensional schema: the missing level *Age group* is assumed to be aggregated at the highest level and therefore not reported. 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.

Definition 6 *(Existence of a solution). Given a query $Q = \langle ind, \{L_1 \ldots, L_n\}\rangle$ and a set $\mathcal{S}$ of data sources, Q has a solution iif: either (1) $\exists S_x \in \mathcal{S}$ such that $ind \in Ind_{S_x} \wedge Dim_{S_x} \subseteq \{L_1, \ldots, L_n\}$, or (2) $\exists$ a formula $f_\alpha = ind = f(ind_1, \ldots, ind_m)$ such that $\forall ind_i$ ($\exists S_i \in \mathcal{S}$ such that $ind_i \in Ind_{S_i} \wedge Dim_{S_i} \subseteq \{L_1, \ldots, L_n\}$).*

In the current framework, the derivation of a formula for an indicator relies on the reasoning services introduced in Sect. 3. A detailed discussion of the working mechanism for the services is available in [3]. Query answering is aimed to retrieve a (sorted set of) solution(s) from a user query and involves the following steps, that are summarized in Algorithm 1:

- (Line 1) the algorithm takes as input the query $Q = \langle ind, \{L_1, \ldots, L_n\}\rangle$ and an integer k representing the number of solutions to retrieve.
- (Line 2) the *find_rewriting(ind,*$\{L_1, \ldots, L_n\}$*)* function is executed, which returns a formula for *ind* such that all its component measures $\{ind_1, \ldots, ind_m\}$ are provided by some data source according to Definition 6,

and such sources are compatible with the dimensional schema of the query. For this task, reasoning services are exploited that are capable of manipulating the mathematical relations among indicators to retrieve alternative rewritings of a formula. The function returns the retrieved formula f_α and, for each component of the formula ind_i, a set $\Phi_i \subseteq \mathcal{S}$ of sources that provide ind_i. In other terms, Φ_i includes the (alternative) data sources from which ind_i can be retrieved.
- (Line 3) the cartesian product of all the sets Φ_i is computed in order to list all combinations of data sources that can be used to calculate the formula, where a combination is a tuple $\langle S_1, \ldots, S_m \rangle$. The set $\Phi_\bowtie$ includes all alternative sets of sources capable of providing a solution.
- (Lines 4–7) given that more than a single solution may be available, due to the fact that multiple sources can provide the same measure, sorting them according to a quality index is needed. Although a set of sources may provide the needed measures, their join does not necessarily produce a non-empty result. Here, we refer to the *degree of joinability*, discussed below, which measures the likelihood to produce a result out of a join between two (set of) domains. For each tuple $\langle S_1, \ldots, S_m \rangle$ in $\Phi_\bowtie$, a new tuple $\langle x, \{S_1, \ldots, S_n\} \rangle$ is produced, where $x \in [0, 1]$ is the degree of joinability among sets $S_1, \ldots, S_m$.
- (Line 8) the set Ψ is sorted in descending order by the degree of joinability.
- Finally, the formula f_α and the k-top solutions in Ψ_{sort} are returned.

Algorithm 1. Query answering

```
1: function FINDSOLUTION(⟨ind, {L_1, ..., L_n}⟩,k)
2:     (fα(ind_1, ..., ind_m), {Φ_1, ..., Φ_n})=find_ rewriting(ind,{L_1, ..., L_n})
3:     Φ_⋈ = ×_{i=1}^{n} Φ_i
4:     Ψ = ∅
5:     for ⟨S_1, ..., S_m⟩ ∈ Φ_⋈ do
6:         Ψ ← compute_joinability({S_1, ..., S_m}, {L_1, ..., L_n})
7:     end for
8:     Ψ_sort ← sort(Ψ)
9:     return ⟨f_α, Ψ_{sort,k}⟩
10: end function
```

In the following, we discuss the degree of joinability index and the procedure for its computation. Two 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 two 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 that represent the dimensional levels, in order to map them onto a single domain before applying the hashing function. To give an example, if the query requires levels *Geo.Country* and *Time.Day*, the hash will be calculated on the concatenation of domains *country_ region + updated* for source S1 (a possible value is "Italy 2020-11-30"). Such "combined MinHashes" corresponding to concatenated domains for each source are precomputed during the integration and mapping step and stored in order to speed up the query answering.

Finally, as summarized by Algorithm 2 (lines 3–7), the degree of joinability is iteratively calculated by executing the LSH_Ensemble for each pair of data sources (S_i, S_{i+1}) (line 4) and considering the product of the obtained values (line 6). Given that the containment is an asymmetric index, we consider the application of LSH_Ensemble in both directions (from source i to source $i+1$ and vice versa, according to the semantics of an inner join) and check for the highest threshold t (line 5). The search of such a threshold is done through binary search.

Algorithm 2. Computing degree of joinability

1: **function** COMPUTEDEGREEOFJOINABILITYSCORE($\{S_1, \ldots, S_m\}, \{L_1, \ldots, L_m\}$)
2: $MH \leftarrow retrieve\_combined\_MinHashes(\{S_1, \ldots, S_m\}, \{L_1, \ldots, L_m\})$
3: joinability = 1
4: **for** i=1..n-1 **do**
5: get the highest t such that max of $|LSHEnsemble(MH_i, MH_{i+1})|$ and $|LSHEnsemble(MH_{i+1}, MH_i)|$ is > 0
6: $joinability = joinability * t$
7: **end for**
8: **return** $\langle j, \{S_1, \ldots S_m\}\rangle$
9: **end function**

6 Evaluation

An evaluation of the approach on the case study is proposed here. Tests have been carried out on an Intel Core i5-1135G7, 8 cores @ 2.40 GHz, x86_64 architecture, with 8 GB RAM running Linux Fedora 34. A single-thread implementation of the approach has been used, relying on the Python library datasketch 1.5.7 [18] for the implementation of MinHash and LSH Ensemble and on pandas 1.3.3 for manipulation of data structures.

Integration and Mapping Discovery. A preliminary setup of the Knowledge Graph for the Data Lake has been performed by defining dimensional levels and measures. Members of levels have been defined programmatically from available online resources (e.g., list of countries and corresponding ISO alpha 2 and alpha 3 codes[8]). For any loaded data source, initialization includes computation of

[8] https://gist.github.com/tadast/8827699.

MinHashes for any domain, mapping with the dimensional levels and precomputation 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. We report the average and the total execution time in Table 2 and some of the mappings in Table 3. Domains are processed in less than 1.6 s on average.

Table 2. Case study: execution time for MinHash generation and mapping.

		S1	S2	S3	S4	S5
Hashing computation	Avg [s]	1.654	0.050	0.005	0.009	0.075
	Total [s]	28.125	1.557	0.076	0.376	0.375
Mapping to dimensional levels	Avg [s]	<0.001	<0.001	<0.001	<0.001	<0.001
	Total [s]	0.006	0.012	0.011	0.033	0.003
Precomputation of dimensional MinHashes for querying	Total [s]	21.235	0.151	0.423	1.610	0.918

Table 3. The set of Knowledge Graph levels and measures which source domains are mapped to.

Source	K levels	K measures
S1	Time.Day, Geo.Country	Positive, Recovered, Deaths
S2	Time.Day, Geo.Province	ICU, Positive, Negative, Recovered, Deaths
S3	Time.Day, Geo.Country	Positive, Deaths
S4	Time.Day, Geo.Country	Cumulative_Positive, Cumulative_Deaths
S5	Time.Day, Geo.Country	ICU, Cumulative_ICU

Query Answering. Let us assume the user is interested in analysing measures *Positive* and *ICU_on_positives_rate* at *Geo.Country* and *Time.Day* levels. As for the first measure, the *find_rewriting* returns $(Positive, \{\{S1\}, \{S3\}\})$. In this case, no join is needed as the measure is directly available from multiple sources.

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

- S5,S1: the degree of joinability is 0.78, with a query time equal to 3.109 s;
- S5,S3: the degree of joinability is 0.31, 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 is lower than that of S1, as the former shares a smaller subset of data with S5.

Discussion. The approach proposed in [16] requires, for a given query, a number of set containment evaluations increasing linearly with the number of sources. On the other hand, the present approach enables to reduce such a number to only the relevant sources (2 in the example) by performing a preliminary evaluation based on formula rewriting. In general, by considering M measures and N sources, the approach requires a number of evaluations equal to $\frac{N}{M}$, on average. If indicators are not available at the requested dimensional schema, decomposing indicators in components requires a further number of evaluations. By considering an average number s of dependencies per indicator and a number l of hierarchical levels in the formula graph, the overall number of components to check for an indicator can analytically be estimated as $(1 + \sum_{i=1}^{l} s^i)\frac{M}{N}$, e.g., for $\mathrm{M} = 200$, $\mathrm{N} = 10000$, $\mathrm{s} = 3$, $\mathrm{l} = 2$, corresponding to average formula graphs for real-world frameworks of indicators, the number of evaluations amounts to 500.

7 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. Starting from a query, which is expressed ontologically as a measure of interest and relevant analysis dimensions, the framework determines the set of sources that are capable of collectively responding, by exploiting math-aware reasoning on indicator formulas. A quantitative evaluation of the result, in terms of joinability of sources, is provided through the degree of joinability index. Future work will be devoted to individuate real case studies for extensive evaluation and to extend the 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 or navigation [17].

References

1. Broder, A.Z .: On the resemblance and containment of documents. In: Proceedings of Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171), pp. 21–29. IEEE (1997)
2. Diamantini, C., Lo Giudice, P., Potena, D., Storti, E., Ursino, D.: An approach to extracting topic-guided views from the sources of a data lake. Inf. Syst. Front. **23**, 243–262 (2021)

3. Diamantini, C., Potena, D., Storti, E.: Analytics for citizens: a linked open data model for statistical data exploration. Concurr. Comput. Pract. Exp. **33**(8), e4186 (2021)
4. Diamantini, C., Potena, D., Storti, E.: A semantic data lake model for analytic query-driven discovery. In: The 23rd International Conference on Information Integration and Web Intelligence, iiWAS2021, pp. 183–186. Association for Computing Machinery, New York, NY, USA (2021)
5. Farid, M., Roatis, A., Ilyas, I.F., Hoffmann, H., Chu, X.: CLAMS: bringing quality to Data Lakes. In: Proceedings of the International Conference on Management of Data (SIGMOD/PODS 2016), pp. 2089–2092. ACM, San Francisco, CA, USA (2016)
6. Fernandez, R.C.: Seeping semantics: linking datasets using word embeddings for data discovery. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE), pp. 989–1000. IEEE (2018)
7. Giebler, C., Gröger, C., Hoos, E., Schwarz, H., Mitschang, B.: Leveraging the data lake: current state and challenges. In: Ordonez, C., Song, I., Anderst-Kotsis, G., Tjoa, A.M., Khalil, I. (eds.) Big Data Analytics and Knowledge Discovery. pp, pp. 179–188. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-27520-4_13
8. Hai, R., Geisler, S., Quix, C.: Constance: an intelligent data lake system. In: Proceedings of the International Conference on Management of Data (SIGMOD 2016), pp. 2097–2100. ACM, San Francisco, CA, USA (2016)
9. Hale, T., Webster, S., Petherick, A., Phillips, T., Kira, B.: Oxford COVID-19 government response tracker. Technical report, Blavatnik School of Government (2020)
10. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 604–613 (1998)
11. Microsoft. Covid-19 data lake. https://docs.microsoft.com/en-us/azure/open-datasets/dataset-covid-19-data-lake. Accessed 23 Feb 2022
12. Miller, R.J.: Open data integration. Proc. VLDB Endow. **11**(12), 2130–2139 (2018)
13. Nargesian, F., Zhu, E., Miller, R.J., Pu, K.Q., Arocena, P.C.: Data lake management: challenges and opportunities. Proc. VLDB Endow. **12**(12), 1986–1989 (2019)
14. Oram, A.: Managing the Data Lake. O'Reilly, Sebastopol (2015)
15. Sawadogo, P., Darmont, J.: On data lake architectures and metadata management. J. Intell. Inf. Syst. **56**(1), 97–120 (2020). https://doi.org/10.1007/s10844-020-00608-7
16. Zhu, E., Nargesian, F., Pu, K.Q., Miller, R.J.: LSH ensemble: internet-scale domain search. Proc. VLDB Endow. **9**(12), 1185–1196 (2016)
17. Zhu, E., Pu, K.Q., Nargesian, F., Miller, R.J.: Interactive navigation of open data linkages. Proc. VLDB Endow. **10**(12), 1837–1840 (2017)
18. Zhu, E., Markovtsev, V.: ekzhu/datasketch: first stable release, February 2017. https://doi.org/10.5281/zenodo.290602