Information Systems Frontiers
https://doi.org/10.1007/s10796-024-10471-4

Analytic Processing in Data Lakes: A Semantic Query-Driven Discovery Approach

Claudia Diamantini[1] · Domenico Potena[1] · Emanuele Storti[1]

Accepted: 9 January 2024

Abstract
Data integration and discovery are open issues in Data Lakes potentially storing hundreds of data sources. The present paper addresses these issues targeting multidimensional data sources, that is sources containing atomic or derived measures aggregated along a number of dimensions, typically derived from raw data for analytical and reporting purposes. Combining semantic models of metadata with existing data-driven techniques, the paper proposes an approach for the discovery of mappings between source metadata and concepts in a reference knowledge graph, enabling the definition of reasoning-based techniques to discover, integrate, and rank data sources relevant to a given analytical query. The efficiency and effectiveness of the approach is discussed by means of experiments on real-world scenarios.

Keywords Data lake · Query-driven discovery · Knowledge graph · Multidimensional data

1 Introduction

Data integration and data discovery are two traditional challenges in data management. Data integration refers to the problem of providing users with a unified view of several heterogeneous related sources. A longstanding issue in Database research, it has come back to the fore due to the explosion of data generated by modern digital information systems and the adoption of Data Lake architectures. Data discovery is a more recently defined problem, mainly resulting from the widespread interest in data analytics. When analysts have to perform a data analysis activity, very often the first problem they are faced with is that of identifying which data sources are potentially useful for their purpose. This typically requires the understanding of the content of data sources, and the search for sources whose content extends, complements or integrates a data source at hand. For this reason, data integration and discovery are often seen as intertwined operations (Nargesian et al., 2019). To this end, metadata modeling is considered of tantamount importance in Data Lakes to make data actionable and avoid data swamps (see also Sawadogo and Darmont 2021).

Recently, a paradigm called *query-driven discovery* has been proposed to combine the two aspects of data integration and discovery (Miller, 2018). It is based on the idea to find datasets that are "similar" to a query dataset, and that can be integrated with it in some way (either by joins, unions or aggregates). Existing approaches to query-driven discovery only consider raw data. However, several available data sources contain *summary* data, that is statistical measures or indicators derived from raw data. Examples include open data by public bodies[1], e.g., to monitor economic trends or the effectiveness of governmental policies and initiatives like a vaccination campaign. These kinds of data have specific structures and rise specific issues that have not been taken into account by the literature.

This work fits into the field of query-driven discovery, introducing a number of novelties. First of all, we take into account summary data, that typically have the structure of multidimensional data cubes, with atomic or derived measures aggregated along a number of dimensions. Second,

✉ Emanuele Storti
e.storti@staff.univpm.it

Claudia Diamantini
c.diamantini@univpm.it

Domenico Potena
d.potena@univpm.it

1 Dipartimento di Ingegneria dell'Informazione (DII), Universitá Politecnica delle Marche, Ancona, Italy

[1] e.g., Eurostat (https://ec.europa.eu/eurostat), World Bank Open Data (https://data.worldbank.org/), European Centre for Disease Prevention and Control (https://www.ecdc.europa.eu/en/data/downloadable-datasets).

we extend the notion of target query, from that of a single dataset, to general OLAP queries. Third, we integrate data-driven approaches typically adopted to assess source similarity with semantics. The approach builds on an ontology of dimensions, measures (also called indicators in the following) and their computation formulas, whose concepts are used to enrich source metadata. More specifically, the contributions of this work are multifold:

- we model data sources as a *Semantic Data Lake*, which is composed by a knowledge layer, in turn defined by a Knowledge Graph and its ontological schema, a metadata graph model, and a set of mappings between the two;
- we define mechanisms for *integration* of data sources , which are based on *mapping discovery* between elements of data sources and concepts in the Knowledge Graph. In particular, for the identification of dimensions we exploit LSH Ensemble for the efficient evaluation of set containment (Zhu et al., 2016). On the other hand, we rely on a string-similarity for identification of measures. The final integration then directly follows from the graph model and the identified mappings;
- we introduce an algorithm to efficiently evaluate the *degree of joinability index*, which is an estimate of the cardinality of the join among a set of sources;
- we introduce a novel algorithm for *query answering* specifically tailored to analytical processing. Reasoning over the knowledge layer allows to exploit computation formulas to identify a set of sources together with the proper calculation rules collectively capable to provide answer to a user query. For instance, let us suppose a user is interested in analysing the measure *Unemployment_rate*, that is not available in any source. Given that such a measure can be calculated as $\frac{Unemployed}{Labour\_force}$, a response can be obtained by combining sources providing the two measures *Unemployed* and *Labour_force*. The degree of joinability index is then exploited to rank the possible alternative ways to answer the query involving different sources;
- we provide thorough validation of the approach by extensive experimentation to assess efficiency and effectiveness in real-world scenarios.

A preliminary version of the approach has been presented in Diamantini et al. (2022). With respect to that work, the present paper better frames the contribution in the existing literature, formalizes the algorithms for query answering and joinability index evaluation, and provides extensive experimentation.

The rest of the paper is structured as follows: Section 2 is devoted to present related work dealing with metadata management, source integration and query answering in Semantic Data Lakes. In Section 3 a case study is introduced that will be used throughout the paper. Section 4 is aimed to present the data model of our Semantic Data Lake, including the metadata and the knowledge layers. The approach for source integration is discussed in Section 5, while query answering mechanisms are detailed in Section 6, along with an evaluation of the approach on the case study. Section 7 reports on experiments aimed to assess efficiency and effectiveness of the approach. Finally, Section 8 concludes the work and draws future research lines.

2 Related Work

Data Lakes recently emerged as a flexible, scalable, and cost-effective way for organizations to store and process large volumes of data in various formats, making them a valuable tool for modern data-driven businesses. On the other hand, this technology can be challenging to implement due to issues related to data quality, security, governance, integration, discovery, and processing (Hai et al., 2021). Organizations need to develop robust data management and governance policies, implement appropriate security controls, and leverage data management tools and technologies to mitigate these challenges (Nargesian et al., 2019; Giebler et al., 2019).

To this aim, semantic models and technologies can provide a valuable support, given their capability to provide standardized ways to represent and query data. As such, ontologies and knowledge graphs have been fostered in the literature as a promising solution to offer a comprehensive view over the underlying data sources, modelling their relationships and dependencies, and managing metadata. In the following, we discuss solutions for metadata extraction and modeling, source integration, discovery and exploration, focusing on semantic-based approaches.

2.1 Metadata Extraction and Modeling

Extraction of metadata from sources with different formats is a pre-requisite for Data Lake management. The Generic and Extensible Metadata Management System (GEMMS) for Data Lakes (Quix et al., 2016) is a framework that extracts metadata from heterogeneous sources, and stores the metadata in an extensible metamodel.

For what concerns metadata modeling, logic-based approaches such as Quix et al. (2016) allows the separation of metadata containing information about the content, semantics, and structure. DCMI Metadata Terms (Board, 2020) is a set of metadata vocabularies and technical specifications maintained by the Dublin Core Metadata Initiative. It includes generic metadata, represented as RDF properties, on dataset creation, access, data provenance, structure and format. The Vocabulary of Interlinked Datasets (VoID) (Alexander et al., 2011) is an RDF Schema vocabulary that provides terms and patterns for describing RDF datasets. It is used as a bridge between the publishers and the users of RDF data and focuses on general metadata, following the Dublin Core model, access metadata and structural metadata. These last represent the structure and the schema of datasets, mostly used for supporting querying and data integration for a variety of scenarios (both in private or public sectors (Mouzakitis et al., 2017)).

On the other hand, (hyper)graph-based approaches enable a common view over multiple metadata, usually exploiting Linked Data graphs (e.g., Diamantini et al. 2021; Dibowski et al. 2020; Pomp et al. 2018; Diamantini et al. 2021b; Bagozi et al. 2019; Chessa et al. 2022). In Diamantini et al. (2021) authors propose a graph model to uniformly handle unstructured sources along with semi-structured and structured ones, taking into account various types of technical, operational and business metadata, also by exploiting lexical and string similarities and links to semantic knowledge (e.g., from DBpedia). Dibowski et al. (2020) discussed how to address FAIR (Findability, Accessibility, Interoperability, and Re-use) data management in a Data Lake at Bosch Automotive. They showed the benefits provided to a Data Lake through the support of ontologies and knowledge graphs which provide cataloguing of data, tracking provenance, access control, and semantic search. In particular, they built the DCPAC ontology (Data Catalog, Provenance, and Access Control) for managing data in a comprehensive manner. Similarly, Pomp et al. (2018) proposed the ESKAPE semantic data platform for the semantic annotation of the ingested data in a knowledge graph. The system is useful for collection, finding, understanding and accessing of large data sources with the goal of ensuring their real-time availability. In a previous work of Diamantini et al. (2021b), which is extended by the present manuscript, we presented a semantic model for statistical multidimensional data stored into a Data Lake. As a distinguished feature of the approach, the knowledge layer of the Data Lake included the possibility to represent indicators, their calculation formulas and analysis dimensions. It was capable to support logic-based services to guarantee the correct identification of indicators and their manipulation. A similar model was adopted by Bagozi et al. (2019) to support a Data Lake for personalized exploration.

2.2 Data Source Integration

A variety of solutions have been proposed on the topic of data source integration in a Data Lake, ranging from approaches based on raw data (and related metadata) management to semantic-enriched frameworks. By making the meaning of the structural elements of the sources explicit, the latter are intrinsically more suitable to address issues related to data variety/heterogeneity and data quality. In Diamantini et al. (2021) the network-based model to represent metadata is used to drive a topic-view extraction which integrates related sources, starting from a set of user defined topics. Knowledge graphs are exploited in Farid et al. (2016) to drive integration, relying on information extraction tools, e.g., Open Calais, that may assist in linking metadata to uniform vocabularies, while in Fernandez et al. (2018) a graph is built by a semantic matcher, leveraging word embeddings to find links among semantically related data sources. Only a few work deal with end-to-end integration, e.g., Constance (Hai et al., 2016) focuses on a holistic approach capable of addressing a variety of structures, with the aim to discover, extract, and summarize structural metadata from (semi)structured data sources, and annotates (meta)data with semantic information to avoid ambiguities.

Recent research effort focused on combining the OBDA (Ontology Based Data Access) paradigm with Data Lakes, leading to the so-called “Semantic Data Lake”. Here, the development of semantic layers supports physical and logical integration and uniform access to Data Lakes content (e.g., Mami et al. 2019). However, unlike our work, no mapping discovery and catalogs of metadata are discussed. Similarly, Beheshti et al. (2018) refer to the term “Knowledge Lake”, and proposes the CoreKG Data Lake, a platform containing contextualized data as a graph, that offers a number of services for curation (enrichment, linking, annotation), indexing and querying through the SPARQL language.

2.3 Data-Driven Discovery and Exploration

Closely related to integration, a recent novel paradigm called *query-driven discovery* was proposed to address in a combined way the issues of source integration and source discovery (Miller, 2018), following the idea to find datasets that are similar to a query dataset and that can be integrated in some way (either by joins, unions or aggregates). In this sense, a new class of data-driven queries is emerging, which consists of datasets and aims to retrieve, from a large collection, related datasets. In a traditional database setting, this problem is usually referred to as schema matching (Rahm & Bernstein, 2001), a longstanding problem of identifying correspondences among database attributes, which is still investigated through recent algorithmic approaches (e.g., Chen et al. 2018, Shraga et al. 2020). A common assumption for most of this work is the existence of consistent and complete metadata, which is not realistic for real-world Data Lakes (Farid et al., 2016; Nargesian et al., 2019). Some work investigated how to apply schema matching in the Data Lake scenario for dataset discovery (Koutras et al., 2021), with the purpose to offer insights on the strengths and weaknesses of existing techniques. Among the work assuming reliable and complete metadata, some refer to clustering-based approaches to match a set of schemas at the same time

(Alshaikhdeeb & Ahmad, 2015). However, they focus only on schema level and not on values.

An interesting type of query is the so-called join-correlation query: given an input query table T and a dataset collection, the goal is to retrieve tables that are joinable with T. Our proposal can be seen as extending this notion to other kinds of operations since, beside join, target tables can be retrieved on the basis of any algebraic expression query defining the calculation of an indicator. Another related problem is the *correlated dataset search*, where an additional challenge arises: besides identifying possible joins, it is also necessary to compute, or estimate, the joinability (or correlation). Most approaches propose the Jaccard Index and the Jaccard Containment similarity as a measure of joinability. The former is the cardinality of the intersection of the two sets divided by the cardinality of their union, wherease the latter divides the intersection by the cardinality of the first set, resulting in an asymmetric measure, but less biased w.r.t. very large datasets.

Algorithms such as JOSIE (Zhu et al., 2019) provide an exact solution to this problem, although a typical solution to overcome scalability issues is to provide approximate answers, e.g., Lazo (Fernandez et al., 2019), LSH Ensemble (Zhu et al., 2016) or GB-KMV (Yang et al., 2019), balancing precision and recall.

Such techniques apply indexing structures and data sketches (typically through hashing) to reduce the dimensionality of the datasets and perform time-effective estimation of the Jaccard index or the containment set. Among them, Asymmetric MinWise Hashing (ALSH) (Shrivastava & Li, 2015) and LSH Ensemble are both techniques to estimate the containment between a source and a set of tables. The latter applies Locality Sensitivity Hashing to determine, given a column of an input dataset, what columns in other datasets are similar to it beyond a given threshold, and outperforms the former in terms of quality results. On the other hand, Lazo (Fernandez et al., 2019) overcomes some limitations of LSH, in particular by directly estimating the containment values through a cardinality-based approach, and reducing the computational efforts in generating the data sketches.

Aurum Fernandez et al. (2018) exploit hypergraphs to find similarity-based relationships and primary-foreign key candidates among tabular datasets. For each columns, it first builds a profile by adding signatures based on information extracted from values e.g., cardinality, data distribution, a representation of data values (i.e., MinHash). Then, it indexes the signatures through Locality-Sensitive Hashing (LSH): if two column signatures are indexed into the same bucket, an edge is created between the two column nodes, while the similarity value is stored as the weight of the edge.

In Santos et al. (2022), given an input query table, authors aim to find the top-k tables that are both joinable with it and contain columns that are correlated with a column in the query. The approach proposes 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. Khatiwada et al. (2022) propose ALITE, a scalable approach for integration of tables which relaxes previous assumptions, such as that tables share common attribute names, are complete and have acyclic join patterns. The approach first assigns an integration ID to each column and then applies natural full disjunction to integrate the tables.

With respect to the content-driven notion of query-driven discovery previously proposed, 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. As a further difference, user requests are expressed in ontological terms as OLAP-like queries, specifying an indicators of interest and the required dimensions of analysis, while in Miller (2018) a query is expressed as a dataset. As such, we refer to LSH Ensemble (Zhu et al., 2016) not as a means of comparing the query against all possible sources in the Data Lake, but (1) to efficiently support discovery, comparing the new source with the dimensions defined in the Knowledge Graph, and (2) once the possible solutions are identified, to estimate the cardinality of their join. For this latter task, we also complement such an approach through the evaluation of a joinability index, similar to Fernandez et al. (2019).

3 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 (Microsoft, 2023) and Our World in Data repository:

S1) *Bing COVID-19 Data*[2], provides data for countries (in some cases, also regions are included), updated daily for years 2020-2021. The dataset includes three basic measures, namely number of confirmed cases, number of recovered cases and number of deaths, together with the variation with respect to the previous day.

S2) *COVID Tracking Project*[3], with data updated daily for every US state for years 2020-2021. It includes indicators such as: numbers on positive, negative and recovered cases, number of deaths, number of hospital-

[2] https://learn.microsoft.com/it-it/azure/open-datasets/dataset-bing-covid-19

[3] https://github.com/COVID19Tracking/covid-tracking-data

ized people, current and cumulative number of people in ICU (Intensive Care Unit), number of ventilated patients, number of pending tests.

S3) *European Centre for Disease Prevention and Control (ECDC) Covid-19 Cases*[4], which includes 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. In particular, it includes measures on the number of cases and number of deaths.

S4) *Oxford COVID-19 Government Response Tracker (OxCGRT)* (Hale et al., 2020), which contains systematic information on measures against COVID-19 taken by governments, for years 2020-2021,namely the confirmed number of cases and the confirmed number of deaths.

S5) *Our World in Data*[5], which contains data on the number of people in hospitals and in ICU (as well as number of people in hospitals and in ICU per 1 million people) 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. As it can be seen from descriptions and Table 1, sources in the Data Lake contain summary data aggregated along certain, mainly temporal and geographical, perspectives. According to the traditional nomenclature, we term them *measures* and *dimensions* respectively. For what concerns the latter, several of the reported dimensions involve information on the day in which the measures were recorded, e.g., *updated* (S1), *date* (S2, S4 and S5), *date_rep* (S3). On the other hand, other dimensions refer to the location of the measurement. In particular, *continent_exp* (S3) specifies the continent. As for the countries, dimensions *country_region* (S1), *iso_country* (S2), *countries_and_territories* (S3), *countryname* (S4) and *entity* (S5) includes the name of the country, while *iso2* and *iso3* (S1), *state* (S2), *geo_id*, *iso_country* and *country_and_territory_code* (S3), *country_code* and *ISO_country* (S4), *ISO_code* (S5) represent the ISO code for the country, namely an international standard short form, which can be encoded in 2 or 3 letters (e.g., IT for Italy, ES for Spain are 2-letter codes while ITA and ESP are 3-letter codes)[6]. Finally, regions within a country are represented by dimensions *admin_region* (S1) and *state* (S2), while their ISO code counterparts are represented by dimensions *iso_subdivision* (S1 and S2).

[4] https://www.ecdc.europa.eu/en/covid-19/data-collection

[5] https://github.com/owid/covid-19-data

[6] Both formats are are part of the “ISO 3166-1 - Codes for the representation of names of countries and their subdivisions”, which is created and maintained by the International Organization for Standardization (ISO).

Further columns represent attributes and are not reported in the table. In the following section we introduce the semantic data model adopted to represent this kind of data.

4 Semantic Data Lake: Data Model

In this Section, we review the model for a Semantic Data Lake that was presented in Diamantini et al. (2022), 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 Data Lake architecture at hand, e.g., based on ponds vs. zones (see also Giebler et al. 2019, Sawadogo and Darmont 2021). If the architecture is pond-based, in fact, the approach is applied to datasets in a single stage, while in zone-based Data Lakes 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.

4.1 Knowledge Layer

The knowledge layer of the Semantic Data Lake comprises the following components.

KPIOnto It is an OWL2-RL ontology[7] aimed to provide the terminology to model an indicator in terms of its details and relations among them. 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 the representation of a formula in terms of operands and operator, are detailed below and shown in Fig. 1:

- `Indicator`: the class of Performance Indicators.
- `Dimension`: the class includes the definition of dimensions along which indicators are measured. It is aligned to the class `qb:MeasureProperty` in the RDF Data Cube Vocabulary (Consortium, 2014).

[7] KPIOnto specifications are available at http://w3id.org/kpionto

Table 1 Details of case study 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

- `Level`: the class represents a specific level, which is related to the corresponding dimension through the property `inDimension`. A level is also linked to the corresponding upper level through property `rollup`, which enables to encode the hierarchical relations among levels.
- `Formula`: it represents a mathematical expression specifying how to calculate the indicator. A formula is defined as the application of an operator (property `hasFunction` on some `FormulaArguments`. Each argument has an `ArgumentValue` which may represent an other indicator, a constant or, in turn, another formula).

Further object properties are defined to link an indicator to a unit of measurement (instances of units are defined externally) and to an instance of `BusinessObjective`.

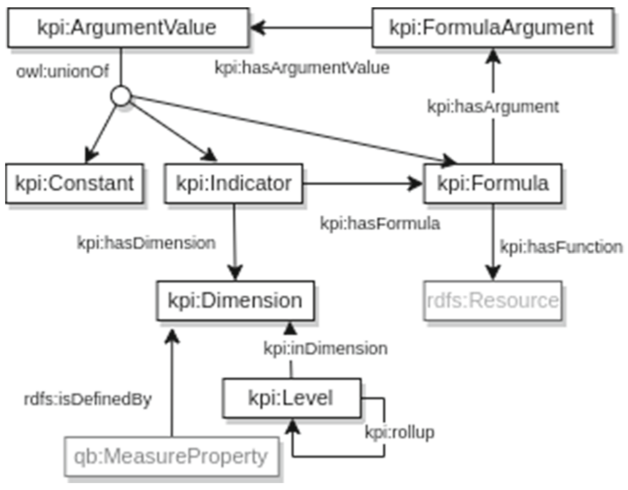

Fig. 1 Main classes and properties in KPIOnto ontology

Knowledge Graph It is defined as a tuple $\mathcal{K} = \langle K_N, K_A, K_\Omega \rangle$, 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. Figure 2(a) shows a fragment of the Knowledge Graph for the case study representing dimensions *Time* and *Geo* with the corresponding levels.

In the following, we refer to the notion of *formula graph*, as a view over the Knowledge Graph which is focused on the mathematical relations of indicators. In the formula graph, a node represents either an indicator, a constant or an operator. On the other hand, an edge connects an indicator node to its corresponding operator for its formula, while this last is connected to other indicator nodes (and/or constants) which represent its operands. Please note that we use this notion just to graphically represent the indicator formulas more clearly, as the indicator and its components are part of the Knowledge Graph. Figure 2(b) shows the formula graph for the case study. The full list of measures defined in the Knowledge Graph for the case study is as follows: ICU, $ICU\_on\_Positives\_Rate$, $Positive$, $Cumulative\_Positive$, $Negative$, $Deaths$, $Cumulative\_Deaths$, $Recovered$, $Cases$.

Logic Programming rules A set of logic programming rules, enacted by a logical reasoner (namely, XSB[8]), have been developed 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 Section 6).

[8] http://xsb.sourceforge.net/

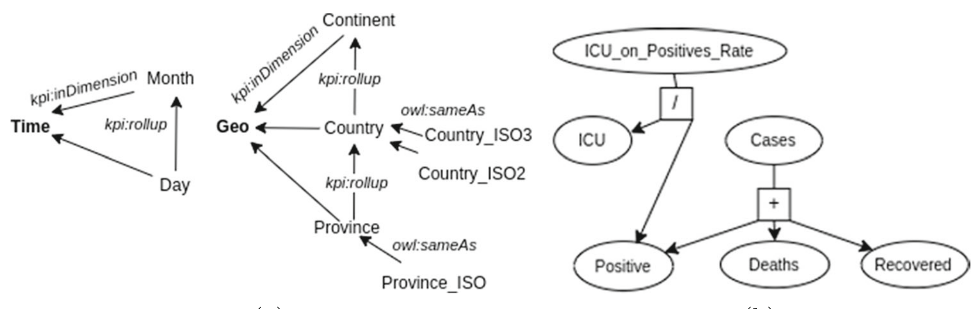

Fig. 2 Case study: (a) dimensions, levels and (b) indicators with their formulas

4.2 Metadata Layer

Different typologies of metadata can be related to a resource, depending on how they are gathered. Oram (2015) introduces a classification in three categories, namely business, operational and technical. Business metadata include business rules (e.g., the upper and lower limit of a particular field, integrity constraints). Operational metadata represent information automatically generated during data processing (e.g., data quality, data provenance, executed jobs). Finally, technical metadata include information about data format and schema. Other categorizations are possible and some overlaps among them may exists (Diamantini et al., 2021), as also shown in Fig. 3. For instance, since business metadata contain all business rules that are mainly expressed in terms of data fields, and since the data schema is included in the technical metadata, we can conclude that data fields represent the perfect intersection between these two subsets. Analogously, technical metadata contain the data type and length, the possibility that a field can be NULL or auto-incrementing, the number of records, the data format and some dump information. These last three things are in common with operational metadata, which contain information like sources and target location, and the file size as well. Finally, the intersection between operational and business metadata represents information about the dataset license, the hosting server and so forth (e.g., see the DCMI Metadata Terms).

Hereby, we refer to the intersection of business and technical metadata, i.e., related to data fields, both domain description and technical details. 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.

Whatever the nature of the source, we refer to a graph model in which the nodes are the elements of the source schema, e.g., tables and attributes in relational databases, complex/simple elements and their attributes in XML/JSON documents. 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 (Diamantini et al., 2021), starting from the definition of a node $n \in N_k$ for each element of the schema of S_k. 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 Diamantini et al. (2021).

4.3 Mapping Function

The data model is completed by defining the mapping function $m \subseteq \mathcal{G} \times \mathcal{K}$, which links the metadata layer to the knowledge layer. We consider three types of mapping, to semantically enrich source elements representing indicators, dimension levels, and their members, respectively.

Specifically, let $G_i \in \mathcal{G}$ be the metadata graph of the source S_i:

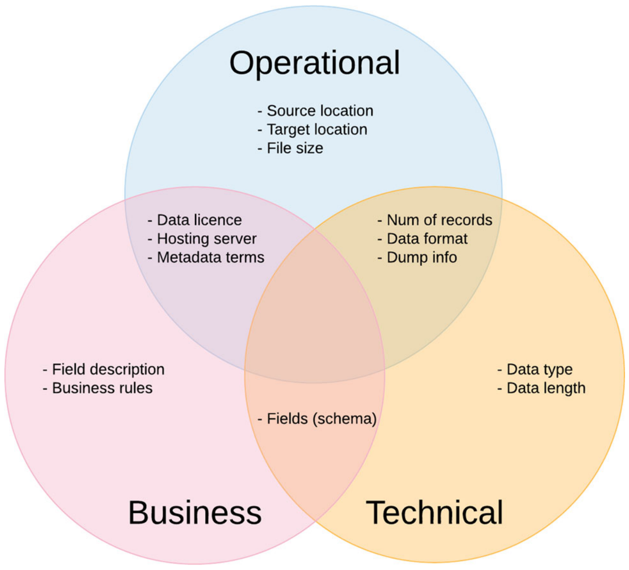

Fig. 3 Metadata classification (Diamantini et al., 2021)

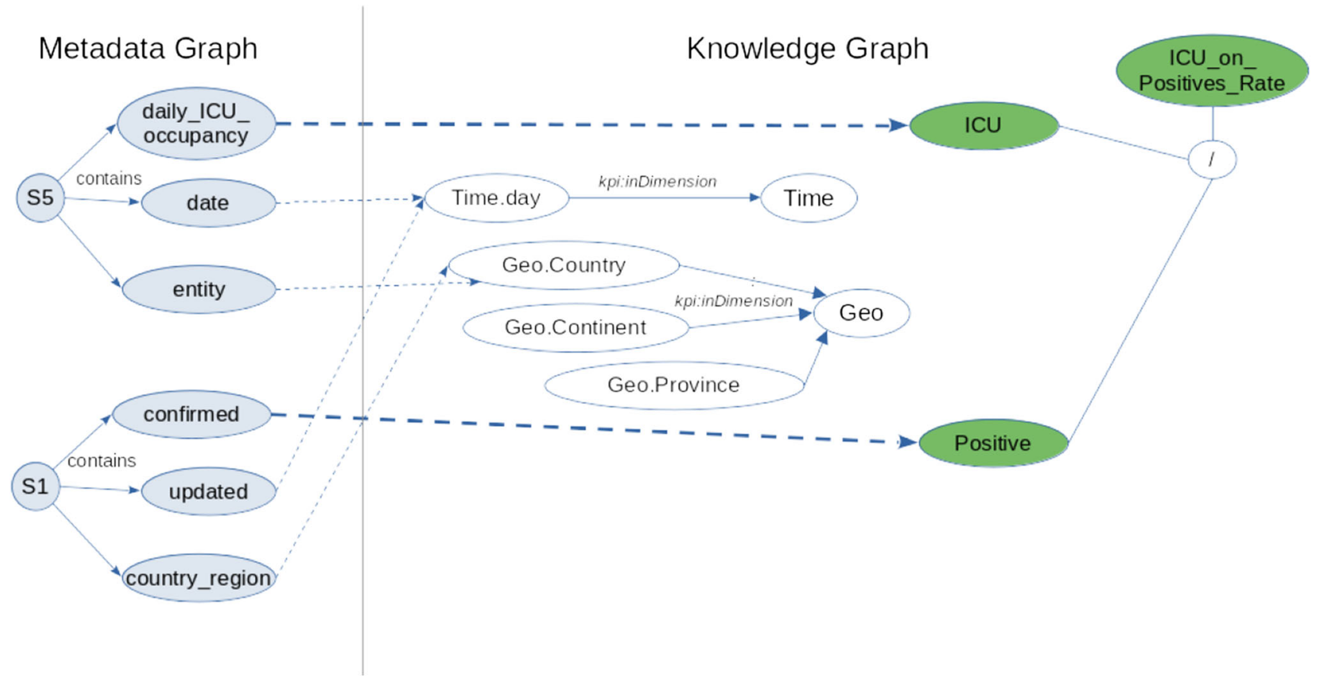

Fig. 4 A fragment of the graph for sources S1 and S5 of the case study

- if the node $n_m \in G_i$ represents a measure, the pair $< n_m, ind >$ will be added to m, where $ind \in \mathcal{K}$ represents the indicator semantically equivalent to n_m;
- if the node $n_d \in G_i$ represents a dimension, the pair $< n_d, l >$ will be added to m, where $l \in \mathcal{K}$ represents the dimension level semantically equivalent to n_d.

In Fig. 4 we report an example of the mappings between two sources of the case study and the Knowledge Graph. On the left, fragments of the metadata graphs for sources S1 and S5 are shown, where the edges represent structural relations between sources and their attributes. On the right, we report a fragment of the Knowledge Graph representing two dimensional hierarchies (nodes with white background) and a formula for the indicator ICU_on_Positives_Rate. Dotted edges, linking nodes of Metadata and Knowledge Graphs, represent the mappings of dimension levels, while dashed edges link attributes to semantically equivalent measures.

5 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 entry. 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 expressions). As a result, a data source corresponds to a set of domains.

5.1 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[9]. For this reason, we rely on set containment in our work. As an example, let us consider a domain $A = \{Rome, Berlin, Paris\}$

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

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 (Zhu et al., 2016), 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 $\{X : c(D, X) \geq t, X \in \mathcal{L}\}$.

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 (Broder, 1997), 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, pairwise 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) (Indyk & Motwani, 1998), a sub-linear approximate algorithm.

While such an approach is targeted to the Jaccard index, an evolution of LSH, namely LSH Ensemble (Zhu et al., 2016), is a technique capable to efficiently estimate the set containment. In particular, it provides an approximate solution to the domain containment search problem, aimed to match a given set d against n other sets, and producing as output the list of those sets that maximally contain d. It is characterized by efficient indexing (almost constant with the domain cardinality) and efficient search (constant with respect to the cardinality of the query domain and sub-linear with respect to the number of sets used for the comparison). To these aims, LSH Ensemble partitions domains into disjoint partitions based on the domain cardinality. Then, a MinHash LSH index is defined for each partition. In this way, LSH Ensemble provides an optimal partitioning function that maximizes the overall pruning power of the index. As a result, the technique proved to be suitable for comparing up to hundreds of millions of sets, even with highly skewed distribution of their cardinality.

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 $t \in [0, 1]$, $LSH\_Ensemble(D, \mathcal{L}, t)$ is a function returning the set of relevant dimensional levels for D.

5.2 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 Data Lake Manager. To make an example, for S2 the measure *in_icu_currently* is mapped to the Knowledge Graph measure *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., Diamantini et al. 2021) or frequency distribution and will be discussed in future work.

5.3 Representation of Mappings

Given a domain of a data source and the most relevant dimensional level with respect to a given threshold (see Definition 3), 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_k be a metadata graph for a source S_k, $D \subseteq S_k$ 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=$(n_D, n_L, c(D, L))$, where $n_D \in G_k$ and $n_L \in \mathcal{K}$ are nodes representing the domain D and the level L respectively, and the estimated value for set containment $c(D, L)$ is used to weigh the mapping. The set of mappings M_{G_k} includes all mappings for dimensions in S_k.

Similarly, given a domain and a related identified measure, a mapping between the corresponding nodes is created. In the following, we represent by $Dims_S \subseteq \mathcal{L}$ the set of dimensional levels available in a source S and by $Inds_S \subseteq \mathcal{I}$ the set of measures available in S, where $\mathcal{I}$ is the set of indicators defined in $\mathcal{K}$.

6 Query Answering

The mappings defined between the metadata graphs and the Knowledge Graph are exploited to support 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, following a query-driven discovery approach where however the query is not represented as a dataset but as a general multidimensional query. A user query Q is expressed as a tuple $Q = \langle ind, \{L_1, \ldots, L_n\}\rangle$, where ind is an indicator (i.e., measure) and $\{L_1, \ldots, L_n\}$ is a set of levels, each belonging to a different dimension.

Definition 5 (Compatible dimensional schema) 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 $Dims_S = \{L_1, \ldots, L_n\}$.

In this definition the source is constrained to contain exactly the same levels of the query. This assumption has been made to keep the paper simpler, and does not limit the generality of the approach. As a matter of fact, we can relax the definition to include sources with more dimensions than in the query. In this case, rollup relations can be exploited following the approach in Diamantini et al. (2018) to provide data at the correct aggregation level.

A data source can respond a query Q if its dimensional schema is compatible with Q and if it provides the requested indicator. On the other hand, taking advantage of the reasoning mechanisms defined on the Knowledge Graph, a formula is searched for calculating ind from other indicators that are all available in other sources of the Data Lake, provided that such sources have a dimensional schema compatible with Q.

In the current framework, the derivation of a formula for an indicator relies on the reasoning services introduced in Section 4. In order to search a formula for ind, the logical reasoner browses the graph of indicators' formulas in $\mathcal{K}$, returning all possible rewriting $f(ind_1, \ldots, ind_m)$ of ind, where $ind_i \in \mathcal{K}, i = 1, \ldots, m$. Considering the example in Fig. 2(b), possible rewritings for the indicator $Positive$ are: $Positive = Cases - Deaths - Recovered$ and $Positive = \frac{ICU}{ICU\_on\_Positives\_Rate}$. A detailed discussion of the working mechanism for the services is available in Diamantini et al. (2021a).

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 $ind \in Inds_{S_x} \wedge Dims_{S_x} = \{L_1, \ldots, L_n\}$, or (2) $\exists$ a formula $f(ind_1, \ldots, ind_m)$ for ind such that $\forall ind_i$ $(\exists S_i \in \mathcal{S}$ such that $ind_i \in Inds_{S_i} \wedge Dims_{S_i} = \{L_1, \ldots, L_n\})$.

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 context of a Data Lake. Therefore, in Algorithm 1 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.

As first step (line 2) the *find_rewriting(ind,*$\{L_1, \ldots, L_n\}$*)* function is executed, which returns 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 (Definition 6(2)). In case ind is already available in one source, the function will return the identity function (Definition 6(1)). For each formula, the *find_rewriting* returns also the set $\{\Phi_1, \ldots, \Phi_m\}$, where $\Phi_i \subseteq \mathcal{S}$ is the set of sources that can provide the component ind_i. In other terms,

Φ_i includes the (alternative) data sources from which ind_i can be retrieved.

For each pair $\langle f(ind_1, \ldots, ind_m), \{\Phi_1, \ldots, \Phi_m\}\rangle$ (line 4), 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 $\langle S_1, \ldots, S_m\rangle$ (line 6). The estimation of the cardinality of the query result is obtained by means of the function *compute_joinability* (line 9), which computes the *degree of joinability*. Such an index, discussed below, measures the likelihood to produce a result out of a join among a set of domains. Specifically, given the set of sources $\{S_1, \ldots, S_m\}$ with S^* being the one with lowest cardinality, *compute_joinability* 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 (line 7).

Algorithm 1 Find solutions.

```
1: function FINDSOLUTION($\langle ind, \{L_1, \ldots, L_n\}\rangle$)
2:     $\Theta$=find_rewriting(ind,$\{L_1, \ldots, L_n\}$)
3:     $\rho = \emptyset$
4:     for each $\langle f(ind_1, \ldots, ind_m), \{\Phi_1, \ldots, \Phi_m\}\rangle \in \Theta$ do
5:         $\Psi = 0$
6:         for each $\langle S_1, \ldots, S_m\rangle \in \times_{i=1}^{m} \Phi_i$ do
7:             if $m = 1$ then $\Psi = |S_1|$
8:             else
9:                 $\Psi \leftarrow compute\_joinability(\langle S_1, \ldots, S_m\rangle, \{L_1, \ldots, L_n\}) \cdot \min_{i=1,\ldots,m} |S_i|$
10:            end if
11:            $\rho \leftarrow \langle f(ind_1, \ldots, ind_m), \langle S_1, \ldots, S_m\rangle, \Psi\rangle$
12:        end for
13:    end for
14:    return $\rho$
15: end function
```

In the following, we discuss the *degree of joinability* index and the procedure for its computation. 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 a 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 into 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"). In the following of the work, we refer to the hashing of the concatenation of dimensional domains as *combined MinHash*. The generation of such hashes can be performed off-line after the mapping discovery phase has been done.

The procedure for computing the degree of joinability is summarized in Algorithm 2. As a first step (line 2), the threshold is set to the maximum value. Then, after identifying the source S^* with lowest cardinality (line 3), the function *LSH_Ensemble* is called to obtain the set of sources with which S* is estimated to have a containment score above the given threshold (see Definition 3). If there is at least one source for which this does not hold, then the degree of joinability is less than τ and the threshold is decreased by a given step (line 7).

It is noteworthy that Algorithm 2 returns an overestimate of the degree of joinability of m sources. To give an example, if $S_1 = \{a, b, c\}$, $S_2 = \{a, b, x, y\}$ and $S_3 = \{b, c, x, y\}$, 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_1.

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 in Algorithm 2.

Algorithm 2 Computing degree of joinability.

```
1: function COMPUTE_JOINABILITY($\langle S_1, \ldots, S_m\rangle, \{L_1, \ldots, L_n\}$)
2:     $\tau = 1$
3:     Search $S^* \in \{S_1, \ldots, S_m\}$ $s.t.$ $|S^*| = \min_{i=1,\ldots,m} |S_i|$
4:     $flag = True$
5:     while $flag$ do
6:         $\Lambda = LSH\_Ensemble(S^*, \{S_1, \ldots, S_m\} \setminus S^*, \tau)$
7:         if $|\Lambda| < (m-1)$ then $\tau = \tau - \tau_{step}$
8:         else $flag = False$
9:         end if
10:    end while
11:    return $\tau$
12: end function
```

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 2, 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 *find_rewriting* 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.

Table 2 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

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 3.109s;
- $\langle S5, S3\rangle$: 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 3.283s.

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 the years 2020 and 2021, while S3 includes data only for the 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.

7 Experiments

A set of experiments have been carried out to evaluate the approach. In particular, they have been designed to investigate the following features:

(E1.1) the *efficiency of the mapping discovery*. The experiment is devoted to asses the cost of plugging a new source. To this end, we calculate the time needed for the mapping discovery for dimensions, investigating how this is affected by the size of the source and its noise, i.e., the amount of values that cannot be mapped to Knowledge Graph concepts;

(E1.2) the *effectiveness of mapping discovery*, in terms of ratio of source domains that are correctly mapped to the corresponding dimensions, evaluating the impact of dataset size and noise;

(E2.1) the *efficiency of the degree of joinability calculation*, in terms of running time. The experiment is performed on couples of datasets of different sizes (cardinality and domains in the dimensional schema), and aims to compare the efficiency w.r.t. performing an extensive join between the sources;

(E2.2) the *accuracy of the degree of joinability*, with the purpose to estimate the capability of the index to estimate the cardinality of the join between two sources. The quality of the degree of joinability as an estimator of the join is measured through the Mean Absolute Percentage Error (MAPE) of the estimated cardinality.

Details and results for each experiment are discussed in the forthcoming subsections. All tests have been carried out on a virtual machine running on the departmental cluster, with the following configuration: QEMU Virtual CPU version 1.5.3 - 2.26 GHz (8 processors), 32 GB RAM, 64-bit Windows 10 Pro. Tests have been written in Python 3, making use of specific 3rd-party libraries, including datasketch 1.5.7 (Zhu and Markovtsev, 2017) for the implementation of MinHash and LSH Ensemble, Pandas 1.3.3 for manipulation of data structures, Numpy for statistical functions and rdflib for management of RDF graphs.

7.1 Knowledge Graph Generation

As a preliminary step, a synthetic Knowledge Graph has been generated in order to provide the dimensional hierarchies for all experiments. The graph was automatically generated by setting its size as follows:

- number of dimensions: 10
- number of levels per dimension: 5
- initial number of members in the first level: 10
- drill-down factor: 10

In total, the graph includes 50 levels (5 levels * 10 dimensions), arranged in a hierarchy for each dimension. Given a dimension, each member of a level is connected to 10 members of the corresponding lower level. As such, a level at depth i of the hierarchy includes 10^i members, for a total of $\sum_{i=1}^{5} 10^i$ members for each dimension, with a total of 555550 members for the whole knowledge base. The resulting graph is comparable in size with real-world multidimensional models, which in practical usages include

4-12 dimensions on average (Pedersen, 2009). The graph is implemented as an RDF graph, where each concept (i.e., a dimension, a level and a member) is represented as a node, whereas edges represent relations among them (e.g., the membership between a member and a level, or a level and a dimension, the hierarchical relation between a level and its upper level, and between a member and its corresponding upper-level member). Combined MinHashes for all dimensional levels have been pre-generated and stored, in order to speed up the mapping discovery as previously discussed.

7.2 Mapping Discovery (E1.1, E1.2)

The tests described in this subsection are devoted to assess the efficiency and effectiveness of the mapping discovery phase. The datasets used for these tests have been synthetically generated from the Knowledge Graph, by setting a number of parameters:

- the cardinality: from 10^3 to 10^7;
- the number of domains: from 10 to 50;
- the percentage of domains that are dimensions was set to 20% according to the ratio found in empirical studies already cited (Pedersen, 2009);
- the percentage of noise on domains, i.e., the percentage of values in the domains that cannot be aligned to members of the corresponding dimension. This value ranges from 0% to 90%.

Given a value of cardinality, a number of domains and a percentage of noise, the procedure for the generation of a dataset follows these steps: (1) for the 20% of domains which are to be mapped to a dimension, a dimension is randomly picked (with no repetition) from the Knowledge Graph, and one of its levels is randomly selected. (2) In order to generate the values of the domain, a number of members equal to the cardinality value are extracted from the level and randomly sorted. (3) A percentage of values equal to the noise are then replaced with random integers. Conversely, for the 80% of domains that are not to be mapped to a dimension, values are generated as random integers.

For what concerns the configuration of LSH Ensemble, its parameters were initialized by using the following values: 128 permutations, 32 parts, with a threshold of 0.8.

The analysis for E1.1 has been performed by focusing on different aspects. *Execution Time by Dataset Size* The execution time was measured by varying the dataset sizes (in terms of cardinality and number of domains), with no noisy data. Here, the running time for each step of the mapping discovery has been recorded, namely:

- time for *hashing*, i.e., for the generation of the MinHash for each domain;
- time for *querying*, which involves the execution of LSH Ensemble and the identification of the potential mapping;
- time for computation of *combined* MinHashes for the dimensional schema: as discussed in Section 6, the domains of the dimensional schema are combined together (e.g., through concatenation) in order to apply LSH Ensemble, which compares pairs of sets.

Furthermore, the overall running time has been recorded. For any combination of cardinality and number of domains, 10 datasets have been generated and the corresponding results have been averaged. This helped in reducing the bias in the random choice of the dimensions and levels that are included in the datasets. Results are reported in Table 3. Figure 5(a) summarizes a comparison of the overall running time for mapping discovery, for datasets with an increasing cardinality and number of domains. As it is clearly shown, the overall running time linearly depends on the cardinality and number of domains. In particular, small datasets (i.e., up to cardinality 10.000) require less than 3 seconds for a complete mapping, while medium-sized datasets (up to cardinality 100.000) reach 30 seconds and larger datasets takes more than 30 seconds.

The specific contribution to running time of each step of the mapping discovery process is shown in Fig. 5(b): the largest contribution to the overall running time is due to the hashing phase, while querying and profiling take a few seconds even for larger datasets.

In order to provide a more detailed analysis on the hashing time, we also report in Fig. 5(c) the average running time distinguishing between type of domains: dimensions and attributes. As expected, the running time for attributes is longer, as it requires a comparison against all dimensional levels. Running times have been averaged over 10 executions, in order to reduce the bias in the random choice of the dimensional domains.

Execution Time by Noise and Cardinality The impact of noise in the execution time has been investigated for datasets with an increasing cardinality. Since the process is iterative, as the datasets are analysed one domain at a time, the number of domains has been set to a constant of 30, with 6 dimensions. Results, reported in Fig. 5(d), clearly show that, given a dataset with a given cardinality, the execution time for the overall mapping discovery process is almost constant (only slightly decreasing for increasing values of percentage of noise).

Effectiveness by Noise and Cardinality Results for effectiveness (E1.2) are shown in Fig. 6, in terms of percentage of dimensions that have been correctly identified over the total number of dimensions, for datasets of different sizes and percentage of noise in domains. For larger datasets, i.e., with cardinality 1 million and 10 millions, the pres-

Table 3 Execution time by dataset size (values are averaged over 10 repetitions)

Cardinality	#domains	Hashing (s)		Querying (s)		Combined (s)		Overall (s)	
		avg	std	avg	std	avg	std	avg	std
10^3	*10*	0,037	±0,01	<0,001	±<0,001	0,012	±0,004	0,118	±0,027
	20	0,071	±0,012	<0,001	±<0,000	0,013	±0,002	0,221	±0,022
	30	0,099	±0,026	<0,001	±0,001	0,016	±0,004	0,303	±0,074
	40	0,118	± 0,012	<0,001	±0,001	0,018	±0,002	0,367	±0,029
	50	0,142	±0,014	<0,001	±0,001	0,02	±0,002	0,445	±0,042
10^4	*10*	0,38	±0,065	<0,001	±0,001	0,11	±0,032	0,703	±0,213
	20	0,789	±0,149	<0,001	±<0,001	0,13	±0,013	1,259	±0,164
	30	1,122	±0,168	0,001	±0,001	0,146	±0,019	1,755	±0,176
	40	1,566	±0,139	<0,001	±0,001	0,162	±0,014	2,392	±0,17
	50	1,884	±0,197	0,001	±0,001	0,172	±0,015	2,861	±0,29
10^5	*10*	4,419	±1,322	<0,001	±0,001	1,179	±0,471	6,981	±2,367
	20	6,771	±0,773	0,001	±0,001	1,329	±0,1	10,35	±1,045
	30	9,882	±0,688	<0,001	±<0,001	1,405	±0,095	15,47	±1,724
	40	13,03	±1,763	0,001	±<0,001	1,585	±0,095	19,63	±2,84
	50	15,73	±1,8	<0,001	±<0,001	1,738	±0,183	23,35	±3,06
10^6	*10*	34,57	±6,021	<0,001	±<0,001	8,256	±5,16	51,02	±7,46
	20	71,69	±11,67	<0,001	±<0,001	13,15	±1,693	100,1	±12,57
	30	92,58	±8,565	<0,001	±0,001	14,41	±0,687	129,7	±10,37
	40	129	±11,82	<0,001	±<0,001	16,64	±1,151	176,7	±13,69
	50	159,6	±14,95	0,001	±0,001	17,72	±1,241	215,5	±16,55
10^7	*10*	344,7	±51,6	<0,001	±<0,001	44,19	±41,3	461,6	±55,23
	20	675,3	±119,9	<0,001	±<0,001	126	±3,915	947,9	±123,7
	30	1015	±110,8	<0,001	±<0,001	145	±8,343	1381	±117,5
	40	1211	±103,5	<0,001	±<0,001	156,3	±6,836	1653	±97,03
	50	1552	±149	<0,001	±<0,001	172,4	±7,662	2076	±153,3

Experiments that completes in less than 5 seconds on average are highlighted in green, while those that completes in more than 5 minutes on average are in red

ence of noisy data does not significantly affect the capability to recognize domains. On the other hand, for small to medium-sized dataset, i.e., from cardinality 1.000 to 100.000, the effectiveness appears to be more affected by the presence of noise. In general, the capability of the algorithm to detect the corresponding dimension increases with the dataset size. Even in the case of completely clean data (i.e., noise=0%), some dimensions may not be recognized for small datasets, i.e., when a dimension only have 1.000 values. In this case, dimensions are identified with a probability less than 60%, whereas it increases up to 100% for datasets with cardinality 1 million or more. This is a well-known behaviour of LSH Ensemble when dealing with a query domain that is much smaller than the considered domains.

7.3 Degree of Joinability Index (E2.1, E2.2)

This subsection is aimed to discuss the tests evaluating the efficiency and effectiveness of the degree of joinability index calculation. To this purpose, the tests present a comparison between the execution of a complete join and the evaluation of the joinability index for pairs of datasets. Given that the focus is on a pair of datasets, we report the procedure followed for their generation and comparison:

- the first dataset (S1) of the pair is generated, with cardinality ranging from 10^6 to 10^7 and domains ranging from 20 to 50;
- the second dataset (S2) of the pair is generated from S1, by randomly picking a number of items ranging from 10^3 to the whole set of items (e.g., for a dataset S1 with cardinality 10^6 we generate 4 pairs: $\langle 10^6, 10^3\rangle$, $\langle 10^6, 10^4\rangle$, $\langle 10^6, 10^5\rangle$, $\langle 10^6, 10^6\rangle$);
- a percentage of 20% of items in S2 has been altered randomly, in order to reduce the items that can be joined together.

By this approach, we generate a number of pairs (S1,S2). Each test involves the computation of the full join between the two dataset of a pair and the estimation of the join through the degree of joinability.

Execution Time of Join vs. Estimated Joinability The execution time was measured for all pairs of datasets (S1,S2), with a number of repetitions of 10. Values have been averaged

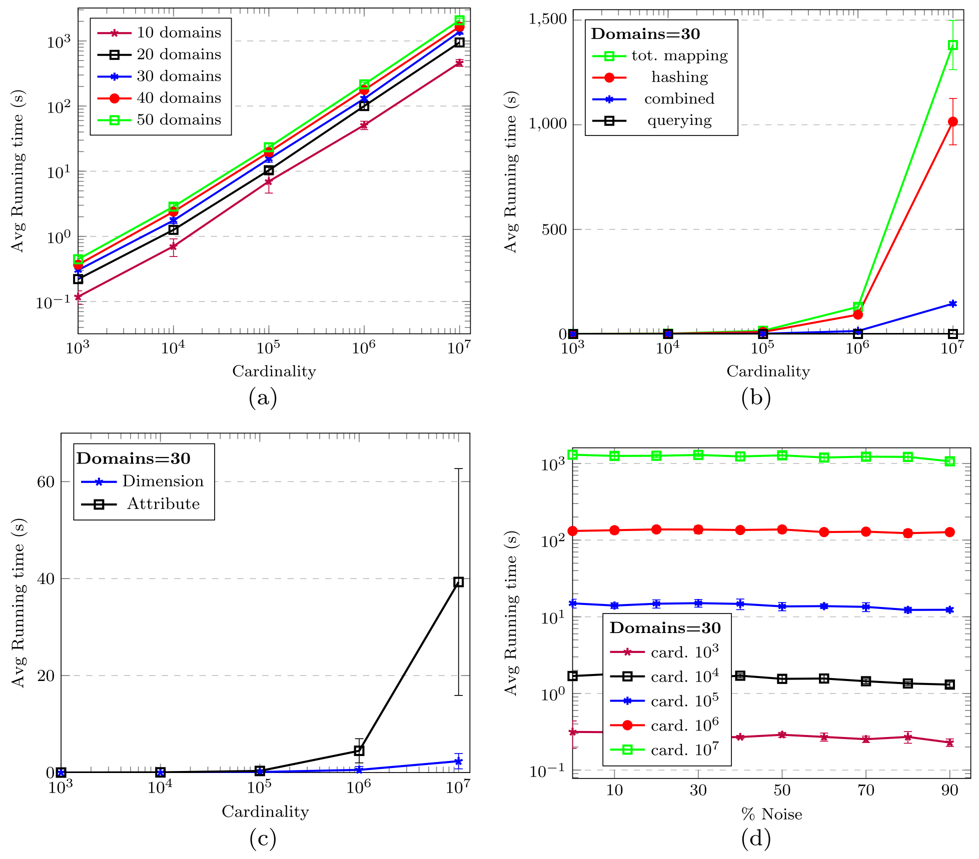

Fig. 5 Efficiency of mapping discovery. (a) total mapping time (log) by dataset cardinality (log) for different number of domains (20% of which are dimensions), no noise; (b) detail of running time for different steps of the mapping process, by cardinality, with domains=30, no noise (averaged over 10 repetitions); (c) average hashing time detailed for dimension and attribute domains, with domains=30, no noise (averaged over 10 repetitions); (d) running time by percentage of noise, for different cardinalities, with domains=30. Average values over 10 repetitions

and reported in Fig. 7. The average running time is shown for pairs having as a first source a dataset with cardinality 10^6 (subfigures on top) and 10^7 (subfigures on bottom). The trend proves that the running time for the computation of the degree of joinability is almost constant over cardinality and

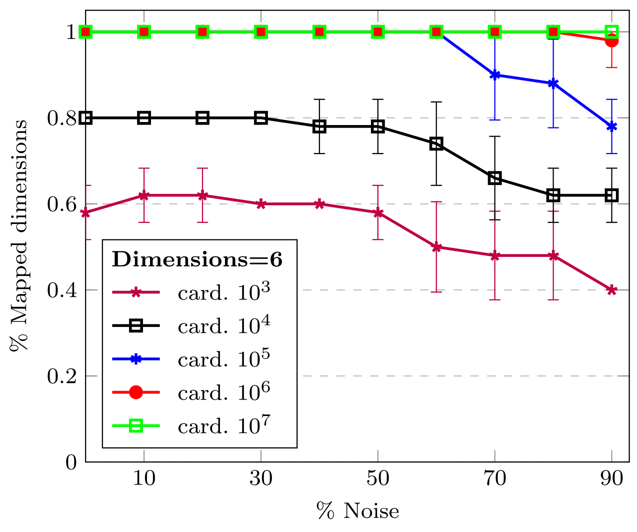

Fig. 6 Effectiveness of mapping discovery by noise rate, for datasets with an increasing cardinality and 6 dimensions. Average values over 10 repetitions

dimensions of the second source. This makes the approach particularly suitable in cases of join between large datasets (in terms of cardinality and domains) where computing the index can reduce the execution time by more than 65%.

Effectiveness of Joinability Index A further experiment has been performed to evaluate how much the degree of joinability index can be used to effectively estimate the cardinality of the join between two datasets. To this end, the absolute difference between the cardinalities of the join and the estimated cardinality through the index is taken into account, and then divided by the cardinality of the join. Values are averaged over 10 repetitions. This corresponds to the Mean Absolute Percentage Error (MAPE), which is calculated as follows: $MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|card(join)_t - card(estim)_t|}{card(join)_t}$. Results, that are reported in Fig. 8, show that the error decreases with the dataset size. In particular, LSH Ensemble is demonstrated to produce better estimates with pairs of datasets of similar size. For instance, when the cardinality of the two datasets is 10^6 vs. 10^5 or 10^6, the MAPE reaches values below 20% that are in line with the typical performances of LSH Ensemble as documented in Zhu et al. (2016).

We would like to note that better performances can be obtained by increasing the number of parts and hashing functions in the configuration of LSH Ensemble, which increases both running times and memory usage as a consequence. Par-

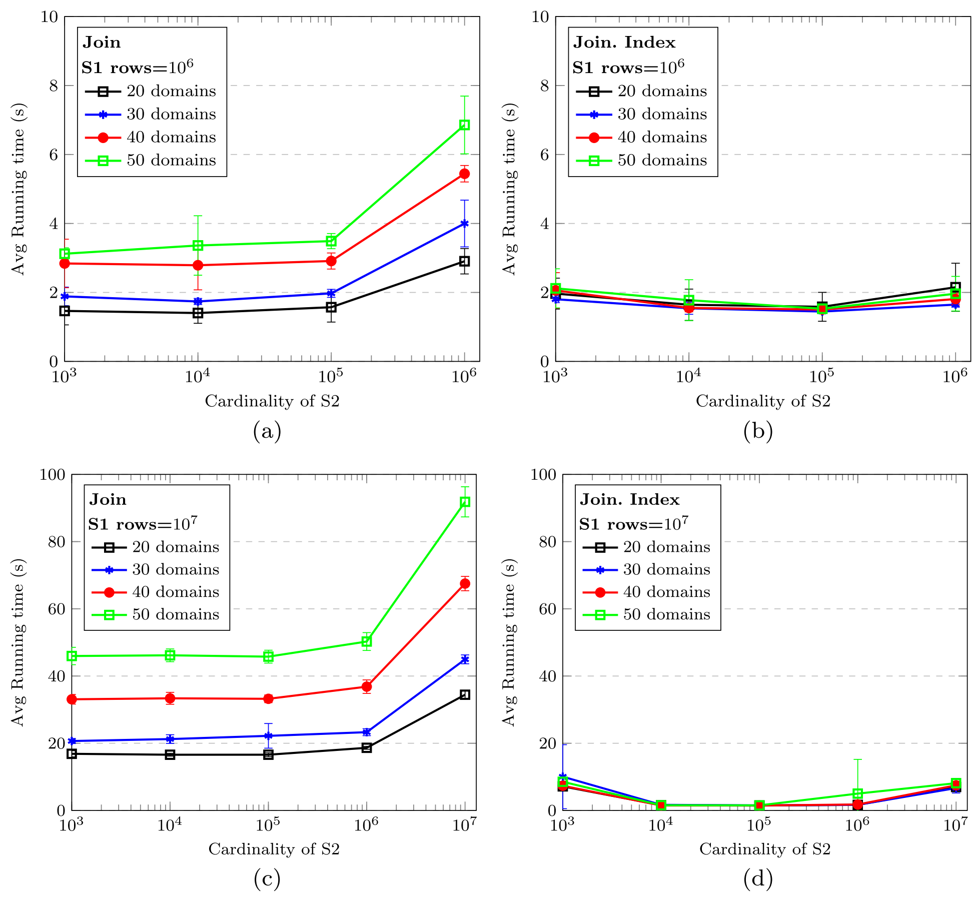

Fig. 7 Execution time of join (a) and (c) and computation of degree of joinability (b) and (d) for datasets with cardinality 10^6 and 10^7 and number of domains from 20 to 50 (the 20% of which are dimensions). Average values over 10 repetitions

allel and distributed solutions for computations, e.g., Hadoop instance with multiple nodes, can be used to keep under control these two parameters.

8 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 a set of dimensions relevant for the analysis at hand, the framework determines the set of sources that are capable of collectively responding, by exploiting math-aware reasoning on indicator formulas. The results of the experimental evaluation demonstrate the validity of the approach in significantly reducing the time needed to evaluate the joinability of different analytical sources, at the same time keeping a reasonable accuracy

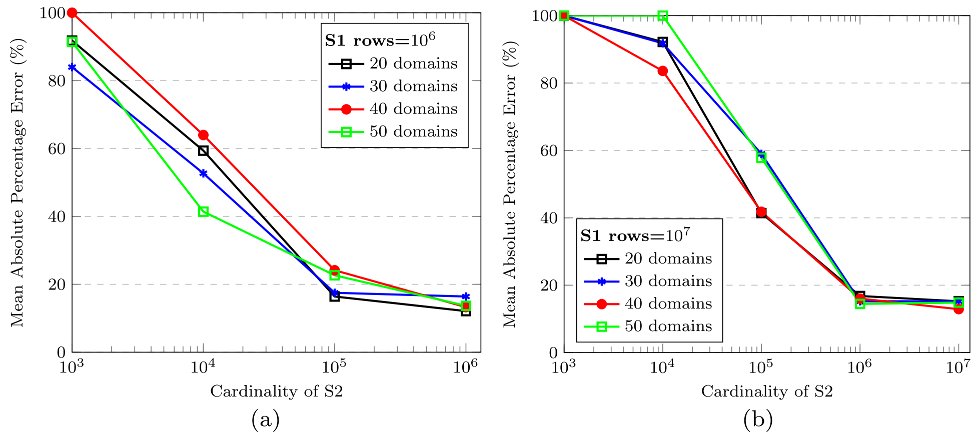

Fig. 8 Mean Absolute Percentage Error between the cardinality of the join and the estimated cardinality though the degree of joinability, for reference datasets of cardinality 10^6 and 10^7, and number of domains from 20 to 50 (20% of which are dimensions). Average values over 10 repetitions

especially with very large datasets. The approach extends standard query-driven discovery, e.g., Zhu et al. (2016), which requires, for a given query, a number of set containment evaluations increasing linearly with the domains in the Data Lake. A peculiarity of our solution is indeed that it enables to reduce such a number to only the relevant sources 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 $(1 + \sum_{i=1}^{l} s^i)\frac{N}{M}$, e.g., for M=200, N=10000, s=3, l=2, corresponding to average formula graphs for real-world frameworks of indicators, the number of evaluations amounts to 650.

Future work will be focused to test the approach on real case studies and to extend it towards interesting research directions. First, besides the join among sources, we are planning to extend the approach by supporting union of alternative solutions to a query. In several cases, indeed, a query may return multiple valid solutions which may be merged (integrated vertically): this would be especially useful in order to complement their content. This however requires several issues to be investigated and managed, among which the following ones:

– the need to take into account, besides the schema, also data contained in domains. This can be achieved by relying on approaches such as Table Union Search (Miller, 2018), which is an efficient although purely data-driven technique exploiting the full search space, or by profiling the domain content and developing mechanisms for comparison of source profiles, capable to evaluate complementarity and overlapping among different sources.
– Partial overlapping between alternative solutions requires to address possible inconsistencies on the overlapping part, e.g., related to different values for measures.

As a further extension, through the notion of data profile several quality measures can be devised, based on the evaluation of how much the content of a source can be aligned to the Knowledge Graph. Quality measures such as completeness, coverage or consistency can be used for documentation of data sources and ETL/ELT processes, and to enrich the response to a query, by providing users with more insights on a result set. Finally, dynamic calculation of indicators can be envisaged for a variety of analytical tasks, including interactive data exploration or navigation in a Data Lake (Zhu et al., 2017).

Acknowledgements Not applicable.

Author Contributions Claudia Diamantini: Methodology, Formal Analysis, Investigation, Writing - original draft, Writing - review & editing. Domenico Potena: Methodology, Formal Analysis, Investigation, Writing - original draft, Writing - review & editing. Emanuele Storti: Conceptualization, Methodology, Formal Analysis, Investigation, Writing - original draft, Writing - review & editing, Software.

Funding Open access funding provided by Universitá Politecnica delle Marche within the CRUI-CARE Agreement. The authors received no financial support for the research, authorship, and/or publication of this article.

Availability of data and material The code used during this study and for the generation of datasets is available at the following link: https://github.com/KDMG/DataLakes_material

Declarations

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

References

Alexander, K., Cyganiak, R., Hausenblas, M., & Zhao, J. (2011). Describing linked datasets with the void vocabulary

Alshaikhdeeb, B., & Ahmad, K. (2015). Integrating correlation clustering and agglomerative hierarchical clustering for holistic schema matching. *Journal of Computer Science, 11*(3), 484.

Bagozi, A., Bianchini, D., Antonellis, V. D., Garda, M., & Melchiori, M., et al. (2019). Personalised exploration graphs on semantic data lakes. In H. Panetto (Ed.), *On the Move to Meaningful Internet Systems: OTM 2019 Conferences* (pp. 22–39). Cham: Springer International Publishing.

Beheshti, A., Benatallah, B., Nouri, R., & Tabebordbar, A. (2018). Corekg: a knowledge lake service. *Proceedings of the VLDB Endowment, 11*(12), 1942–1945.

DCMI Usage Board (2020). DCMI Metadata Terms. https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

Broder, A.Z. (1997). On the resemblance and containment of documents. In: *Proceedings. Compression and Complexity of Sequences* 1997 (Cat. No. 97TB100171), pp. 21–29. IEEE .

Chen, C., Golshan, B., Halevy, A. Y., Tan, W. C., & Doan, A. (2018). Biggorilla: An open-source ecosystem for data preparation and integration. *IEEE Data Eng. Bull., 41*(2), 10–22.

Chessa, A., Fenu, G., Motta, E., Osborne, F., Reforgiato Recupero, D.A.G., Salatino, A., & Secchi, L., et al. (2022). Enriching data lakes with knowledge graphs. In: *CEUR Workshop Proceedings*, vol. 3184, pp. 123–131

Diamantini, C., Potena, D., & Storti, E. (2022). 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, pp. 179–192. Springer.

Diamantini, C., Lo Giudice, P., Potena, D., Storti, E., & Ursino, D. (2021). An approach to extracting topic-guided views from the sources of a data lake. *Information Systems Frontiers, 23*, 243–262.

Diamantini, C., Potena, D., & Storti, E. (2018). Multidimensional query reformulation with measure decomposition. *Information Systems, 78*, 23–39

Diamantini, C., Potena, D., & Storti, E. (2021). Analytics for citizens: A linked open data model for statistical data exploration. *Concurrency and Computation: Practice and Experience, 33*(8), e4186.

Diamantini, C., Potena, D., & Storti, E. (2021). A semantic data lake model for analytic query-driven discovery. iiWAS2021*The 23rd International Conference on Information Integration and Web Intelligence* (pp. 183–186). New York, NY, USA: Association for Computing Machinery.

Dibowski, H., Schmid, S., Svetashova, Y., Henson, C., & Tran, T. (2020). Using semantic technologies to manage a data lake: Data catalog, provenance and access control. In: *SSWS@ ISWC*, pp. 65–80. Athen.

Farid, M., Roatis, A., Ilyas, I., Hoffmann, H., & Chu, X. (2016). CLAMS: bringing quality to Data Lakes. In: *Proc of the International Conference on Management of Data (SIGMOD/PODS'16),* pp. 2089–2092. San Francisco, CA, USA . ACM

Fernandez, R.C., Abedjan, Z., Koko, F., Yuan, G., Madden, S., & Stonebraker, M. (2018). Aurum: A data discovery system. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE), pp. 1001–1012. IEEE.

Fernandez, R.C., Mansour, E., Qahtan, A.A., Elmagarmid, A., Ilyas, I., Madden, S., Ouzzani, M., Stonebraker, M., & Tang, N. (2018). Seeping semantics: Linking datasets using word embeddings for data discovery. In: *2018 IEEE 34th International Conference on Data Engineering (ICDE),* pp. 989–1000. IEEE.

Fernandez, R.C., Min, J., Nava, D., & Madden, S. (2019). Lazo: A cardinality-based method for coupled estimation of jaccard similarity and containment. In: *2019 IEEE 35th International Conference on Data Engineering (ICDE),* pp. 1190–1201. IEEE.

Giebler, C., Gröger, C., Hoos, E., Schwarz, H., & Mitschang, B. (2019). 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* (pp. 179–188). Cham: Springer International Publishing.

Hai, R., Geisler, S., & Quix, C. (2016). Constance: An intelligent data lake system. In: *Proc of the International Conference on Management of Data (SIGMOD 2016),* pp. 2097–2100. San Francisco, CA, USA . ACM.

Hai, R., Quix, C., & Jarke, M. (2021). Data lake concept and systems: a survey. arXiv preprint arXiv:2106.09592

Hale, T., Webster, S., Petherick, A., Phillips, T., & Kira, B. (2020). *Oxford covid-19 government response tracker*. Blavatnik School of Government: Tech. rep.

Indyk, P., Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing,* pp. 604–613.

Khatiwada, A., Shraga, R., Gatterbauer, W., & Miller, R. J. (2022). Integrating data lake tables. *Proc. VLDB Endow, 16*(4), 932–945.

Koutras, C., Siachamis, G., Ionescu, A., Psarakis, K., Brons, J., Fragkoulis, M., Lofi, C., Bonifati, A., & Katsifodimos, A. (2021). Valentine: Evaluating matching techniques for dataset discovery. In: *2021 IEEE 37th International Conference on Data Engineering (ICDE),* pp. 468–479. IEEE.

Mami, M.N., Graux, D., Scerri, S., Jabeen, H., Auer, S., Lehmann, J. (2019). Uniform access to multiform data lakes using semantic technologies. In: *Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services,* pp. 313–322

Microsoft Covid Data Lake (2023) Covid-19 data lake. https://docs.microsoft.com/en-us/azure/open-datasets/dataset-covid-19-data-lake. Accessed: 23-02-2022

Miller, R. J. (2018). Open data integration. *Proc VLDB Endow, 11*(12), 2130–2139.

Mouzakitis, S., Papaspyros, D., Petychakis, M., Koussouris, S., Zafeiropoulos, A., Fotopoulou, E., Farid, L., Orlandi, F., Attard, J., & Psarras, J. (2017). Challenges and opportunities in renovating public sector information by enabling linked data and analytics. *Information Systems Frontiers, 19*, 321–336.

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

Oram, A. (2015). *Managing the Data Lake*. Sebastopol, CA, USA: O'Reilly.

Pedersen, T.B. (2009) Multidimensional Modeling, pp. 1777–1784. Springer US, Boston, MA.

Pomp, A., Paulus, A., Kirmse, A., Kraus, V., & Meisen, T. (2018). Applying semantics to reduce the time to analytics within complex heterogeneous infrastructures. *Technologies, 6*(3), 86.

Quix, C., Hai, R., Vatov, I. (2016). Gemms: A generic and extensible metadata management system for data lakes. In: *CAiSE forum,* vol. 129.

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal, 10*, 334–350.

Santos, A., Bessa, A., Musco, C., & Freire, J. (2022). A sketch-based index for correlated dataset search. In: *2022 IEEE 38th International Conference on Data Engineering (ICDE),* pp. 2928–2941. IEEE.

Sawadogo, P., & Darmont, J. (2021). On data lake architectures and metadata management. *Journal of Intelligent Information Systems, 56*(1), 97–120.

Shraga, R., Gal, A., & Roitman, H. (2020). Adnev: Cross-domain schema matching using deep similarity matrix adjustment and evaluation. *Proceedings of the VLDB Endowment, 13*(9), 1401–1415.

Shrivastava, A., & Li, P. (2015). Asymmetric minwise hashing for indexing binary inner products and set containment. In: *Proceedings of the 24th international conference on world wide web,* pp. 981–991.

World Wide Web Consortium (2014). The rdf data cube vocabulary. *World Wide Web Consortium: Tech. rep.*

Yang, Y., Zhang, Y., Zhang, W., & Huang, Z. (2019). Gb-kmv: An augmented kmv sketch for approximate containment similarity search. In: *2019 IEEE 35th International Conference on Data Engineering (ICDE),* pp. 458–469. IEEE.

Zhu, E., Deng, D., Nargesian, F., & Miller, R.J. (2019). Josie: Overlap set similarity search for finding joinable tables in data lakes. In: *Proceedings of the 2019 International Conference on Management of Data,* pp. 847–864

Zhu, E., Markovtsev, V. (2017). ekzhu/datasketch: First stable release. https://doi.org/10.5281/zenodo.290602

Zhu, E., Nargesian, F., Pu, K. Q., & Miller, R. J. (2016). Lsh ensemble: Internet-scale domain search. *Proc. VLDB Endow., 9*(12), 1185–1196.

Zhu, E., Pu, K. Q., Nargesian, F., & Miller, R. J. (2017). Interactive navigation of open data linkages. *Proc. VLDB Endow., 10*(12), 1837–1840.

Claudia Diamantini Ph.D. (member IEEE, senior member ACM) is Full Professor at Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, where she also holds the role of Vice Dean of the Faculty of Engineering. Her research interests include data mining and knowledge discovery, process modelling and mining, data semantics and knowledge graphs. On these topics she has worked within national and international projects, and authored more than 170 publications.

Domenico Potena received the Ph.D. in Information Systems Engineering from Università Politecnica delle Marche, Italy, in 2004. At present, he is an associate professor at Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche. His research interests include process mining, knowledge discovery in databases, data mining, data warehousing, information systems and service oriented architectures.

Emanuele Storti received the Ph.D. degree in Computer Engineering from Università Politecnica delle Marche in 2012 and currently works as an assistant professor at Dipartimento di Ingegneria dell'Informazione. His research interests include knowledge graphs, semantic technologies, knowledge management, and data integration.