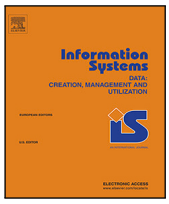
Information Systems 142 (2026) 102791

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is



Query-driven explainable join discovery of analytic cubes based on profiling

Claudia Diamantini [a], Alessandro Mele [a], Domenico Potena [a], Cristina Rossetti [a,b], Emanuele Storti [a,*]

[a] DII, Università Politecnica delle Marche, via Brecce Bianche, Ancona, 60131, Italy
[b] DAUIN, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino, 10129, Italy

ARTICLE INFO

Keywords:
Join discovery
Query-driven discovery
Analytic cubes
Data profile
Joinability estimation
Explainable analytics
Big data

ABSTRACT

The rapid growth of data repositories limits user's awareness of their content and the understanding of interrelationships between diverse datasets, increasing the demand for effective data discovery tools that help analysts locate relevant data. Join discovery, the process of finding tables that can be joined based on common attributes, is one possible strategy. This paper introduces a novel join discovery approach tailored to data structures commonly used in analytics: multidimensional or analytic cubes. By leveraging the unique structure of analytic cubes, the approach utilizes the concept of *profile*, which refers to metadata providing information about data distribution. Specifically, we introduce a profile-based, query-driven join discovery model that allows users to identify alternative joinable cubes starting from a declarative query, as well as to rank and qualitatively assess alternatives. We validate the efficiency and effectiveness of our approach through extensive experimentation. Our dual approach, which combines schema-level and content-level information, enhances both the expressivity of join discovery and the clarity of join discovery results, making the analysis process more intuitive and interpretable.

1. Introduction

Several challenges are involved in Big Data management, particularly when it comes to making data actionable. Among them, the rapid growth of Big Data environments, such as open data repositories and Data Lakes, prevents the definition of a unified model describing relationships among heterogeneous datasets from different sources. Nevertheless, users lack awareness of both the datasets stored in repositories and their content. This has increased the demand for dataset discovery tools to assist analysts in locating relevant data. One form of dataset discovery has been formulated as join discovery [1–4], which is the problem of finding tables that can be joined with a query table on a join key.

Plenty of Big Data resources come in the form of analytic cubes. Analytic cubes are characterized by a set of *indicators*, quantitative data of interest for the analysis that are aggregated along a number of *dimensions*, typically at different granularities called *levels*. The integration of different analytic cubes may be done by joining data items on the set of dimensions, in order to compare and analyze different indicators. Inconsistent, sparse, or noisy data cubes are more the rule than an exception, and can make data discovery and integration time-consuming and frustrating endeavors.

Example 1. The United Nations global database monitors Sustainable Development Goals (SDGs) across more than 230 indicators.[1] Analyzing the relationship between the poverty rate (Indicator 1.1.1) and new HIV infections (Indicator 3.3.1) highlights significant data heterogeneity. While the HIV dataset contains 42,851 samples, the two poverty-related datasets contain 9211 and 12,879 rows, respectively. When these sources are joined to cross-reference the data, the results shrink drastically: merging the HIV data with the first poverty dataset yields only 1162 samples, while merging it with the second produces just 3099. The reason is that data are collected starting from different years and in a discontinuous manner for the different countries, resulting in sparse cubes, or different indicators adopt different aggregation levels (e.g. "BOTHSEX", instead of "MALE/FEMALE"). Joining poverty indicators with some other disease indicator could provide more representative data. Unfortunately, we can discover it only after the whole data processing pipeline (download and analysis of tables, join execution) is performed.

The problem becomes even more challenging in Big Data scenarios when multiple, potentially complementary data sources can be combined. In such cases, users must identify the subset of sources

* Corresponding author.
E-mail addresses: c.diamantini@univpm.it (C. Diamantini), a.mele@pm.univpm.it (A. Mele), d.potena@univpm.it (D. Potena), cristina.rossetti@polito.it (C. Rossetti), e.storti@univpm.it (E. Storti).

[1] https://unstats.un.org/sdgs/dataportal/database.

https://doi.org/10.1016/j.is.2026.102791
Received 20 February 2025; Received in revised form 13 July 2026; Accepted 10 August 2026
Available online 14 August 2026
0306-4379/© 2026 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

S_{cs}

city	day	AQI
Firenze	2021-04-24	40
Perugia	2021-04-25	52
Roma	2021-09-01	47
...	...	...

S_{nw}

city	day	AQI
Genova	2020-02-13	43
Bologna	2020-03-15	50
Torino	2020-03-23	51
...	...	...

S_{tc}

location	timestamp	VAT
Firenze	2021-04-17	712
Siena	2021-04-17	875
Siena	2021-09-18	900
...	...	...

S_{cn}

city	day	AQI
Firenze	2021-04-17	52
Bologna	2021-05-17	47
Ancona	2021-06-19	59
...	...	...

S_{ne}

city	day	AQI
Bologna	2021-01-17	40
Trento	2021-01-17	56
Rimini	2021-02-01	47
...	...	...

S_{tn}

location	timestamp	VAT
Bologna	2020-02-15	659
Genova	2020-02-12	599
Torino	2020-06-15	657
...	...	...

Fig. 1. Excerpt of data sources. Tables with blue names correspond to GreenSense organization, while the others correspond to PureAir.

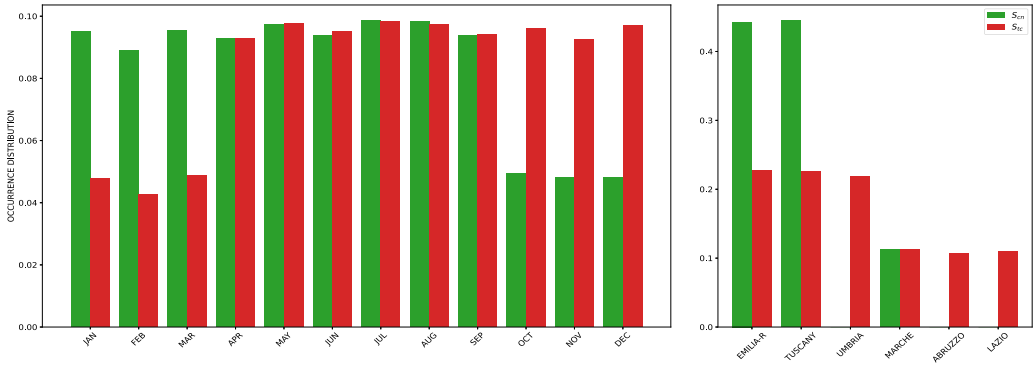


Fig. 2. S_{cn} (green bars) and S_{tc} (red bars) distributions across temporal (left chart) and geographical (right chart) dimensions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

that provides the highest support and relevance for their analysis needs. As the number of sources increases, an exhaustive search of the joins among sources can become intractable, due to the combinatorial growth of possible alternatives. Let us consider the following running example, specifically conceived to best illustrate the issue, exemplify the proposed approach, and assess it.

Example 2. PureAir and GreenSense are two independent companies that provide solutions for collecting and monitoring environmental pollution data through sensors distributed across Italy. PureAir gathers air quality data daily, where each instance contains the city where the sensor is located (*city*), the date of the measurement (*day*), and its average value (*AQI*). Data are partitioned into four different sources, i.e., *center south*, *north west*, *center north*, and *north east*. GreenSense, likewise, collects data on vehicle traffic, where each instance contains the city where the sensor is located (*location*), the date of the measurement (*timestamp*), and its average daily value (*VAT*). Similarly, the measurements are collected into two different sources, i.e., *traffic center*, and *traffic north*. From this point onward, to simplify notation, we will refer to the sources respectively as: S_{cs} = *center south*, S_{nw} = *north west*, S_{cn} = *center north*, S_{ne} = *north east*, S_{tc} = *traffic center*, S_{tn} = *traffic north*. Fig. 1 reports excerpts from the data sources.

Let us suppose the two companies share their data to investigate the correlation between the VAT and AQI indicators across multiple cities and days. The pipeline involves several steps, starting with the identification of relevant data sources. Specifically, it requires the identification of sources containing the requested indicators measured at the specified dimensional levels, namely city and day. This task is challenging because identifying suitable data sources requires exploring heterogeneous datasets that differ in format, structure, and naming conventions. In our scenario, both companies partition data sources by geographical area. A natural strategy is therefore to correlate sources covering the same regions, e.g., S_{tc} and S_{cn}. However, geographical coverage alone does not guarantee a meaningful join. As illustrated in Fig. 2, the two sources share only a limited set of cities from a subset of regions because the corresponding sensors are deployed in different locations. Consequently, relying on prior assumptions rather than on the actual data distribution may lead to poor source combinations. On the other hand, exhaustively evaluating all candidate joins is computationally impractical in large-scale Data Lakes. Using a reference dataset as the starting point would simplify join discovery, however it does not reflect typical analytics scenarios, where analyses are driven by business questions rather than by specific datasets.

To solve these issues, the present paper proposes a query-driven join discovery framework for analytic cubes. Starting from a declarative specification of the desired indicators and dimensions, the framework combines schema-level and content-level information to automatically identify candidate cubes, estimate the usefulness of their joins, ranking alternative solutions, and explain the expected results prior to join execution.

Specifically, the contribution of the paper is multi-fold:

- We present a formalization of the data model for multidimensional data sources, that we extend with the synthetic representation of data source content in the form of *profiles*, efficiently encoding their data distribution at different granularity levels. Mappings are defined between attributes of sources and a global schema, which includes definitions of indicators and dimensions.

- We propose a new formulation of the problem of join discovery, specifically a profile-based, query-driven multi-key join discovery. This formulation offers a goal-oriented, declarative perspective, and enables the specification of conditions for source identification and selection based on profiles.
- We also extend the exploitation of profiles to enhance explainability, by enriching the answer to a join discovery query with the estimated data distribution of the result, which is efficiently computed from profiles.
- When multiple solutions are available for a query, we propose a ranking approach based on efficient estimation of the cardinality of the join.
- We provide theoretical results and an extensive experimentation to assess efficiency and effectiveness of the approach in several scenarios.

A distinctive feature of the approach is the combined use of schema-level information and source profiles to drive query answering and explainability of results. This dual approach allows for more flexible join discovery and improves the transparency of the results, by offering clear explanations of how they were derived, fostering greater understanding of data.

The approach elaborates upon previous work by the authors [5] which presented a similar problem formulation purely based on schema-level information. This paper extends it with the exploitation of source profiling, to enable a more expressive specification of the analysis goal through content-level information.

The rest of the paper is structured as follows: Section 2 is devoted to framing the proposed approach within the literature. Section 3 discusses the methodology, including the formalization of the data model, the algorithm for query answering and the approach for result explanation. An implementation of the solution based on semantic technologies is proposed in Section 4, while Section 5 reports on experiments performed to assess efficiency and effectiveness of the approach. Finally, Section 6 concludes the work and draws future research lines.

2. Related work

To contextually position our query-driven join discovery framework within the state of the art, this section reviews relevant literature within dataset discovery and exploration. Dataset discovery is the process of identifying datasets that may meet an information need. This process can be challenging for several reasons, such as the heterogeneities of data sources or the huge amount of stored data. Additionally, users might not know in advance what to specifically search, therefore tools effectively supporting them in discovering new data sources are needed. We compare the approaches along four fundamental dimensions, as summarized in Table 1: the *search type*, which classifies how users express their information needs (spanning basic keyword-search, table-based dataset-search, and query-based approaches); the *focus of the approach*, which categorizes methods by their core architectural goals (distinguishing search-centric, join discovery, and hybrid models); the exploitation of *metadata*; and the system's *explainability* capabilities. For more details, we refer the interested reader to a recent survey on dataset discovery and exploration [6].

2.1. Search type

The simplest approach to dataset discovery relies on *Keyword-search*. This method can be applied to metadata or, when accessible, to the dataset's content (see e.g., the survey [17]). Among the recent solutions, the Google Dataset Search [7] is a tool enabling a keyword-based search over published annotation of web datasets, which are extracted and reconciled with a Knowledge Graph to identify replicated sources. On the other hand, TRSS [8] proposes a semantic matching between keywords and tables, where both are represented in a semantic space, enabling their similarity evaluation. A second group of approaches is based on *Dataset-search*: the search starts from a given dataset, which can be provided by the user or be the result of a previous analysis task. In this way, source integration and source discovery are addressed in a combined way, following the idea to find datasets that are similar to the query dataset and that can be integrated in some way. Most approaches in this group can be categorized into union search and joinable search, depending on how tables are combined. The former aims to find and integrate datasets with compatible schemas (e.g., Table Union Search [18]). Regarding the latter, most approaches are formulated as a domain search problem: given a query domain Q (defining the set of values of an attribute), find all the domains that are highly relevant to Q. Examples include JOSIE [4], D3L [9], PEXESO [3], and the frameworks proposed by Putrama et al. [12] and Santos et al. [10]. With a broader focus, the Auctus framework [11] for data discovery and augmentation enables several types of queries, including keyword-based search as well as joinable dataset discovery. Finally, *Query-based search* relies on a specific query language and a model, typically a graph, to discover datasets. For example, in SEDAR [15] the query is expressed in ontological terms through the OBDA (Ontology Based Data Access) approach, while the Source Retrieval Query Language (SRQL) and SPARQL are respectively proposed by Aurum [2] and KGLac [16] to query an Enterprise Knowledge Graph. Our work falls within this category, yet adopting a different and original perspective. As a matter of fact, *none of the proposals take analytic cubes into account*, whose characteristic structure requires specialized multidimensional models and OLAP-based query languages.

2.2. Focus of the approach

Related literature can be broadly categorized into three paradigms: *search-centric* approaches focus primarily on dataset retrieval and exploration via keyword or table-based search. Their main goal is to return standalone, relevant datasets rather than establishing exact relational intersections between them. On the other hand, *join* discovery, for most Dataset-search approaches, are strictly tailored to optimize join operations. Their architectures are explicitly engineered to find tables that can be joined with a given input table Q, whose similarity to it is above a given threshold. Some work focuses on efficiently estimating the overlap (Jaccard similarity), e.g., D3L [9] through MinHash signatures, but variants are explored [12]. Other work focuses instead on the estimation of set containment, an asymmetric measure that is less biased for very large datasets. Examples include Asymmetric MinWise Hashing (ALSH) [19] and LSH Ensemble [1], which we adopt, that applies Locality Sensitivity Hashing (LSH). An alternative is Lazo [20], adopted by [11], which directly estimates the containment through a cardinality-based approach, thus reducing the computational efforts in generating the data sketches. Outside of hashing-based methods, alternative methods for join cardinality estimation struggle to balance accuracy and efficiency: Machine Learning methods [21] capture complex distributions well but scale poorly and require high training costs, while traditional solutions inspired by the Selinger model [22] are less resource-intensive but frequently fail on correlated or skewed data. *Most of this related work focuses on a single join key*. While this simplifies the join discovery process and reduces computational complexity, it also limits the scope of integration, and fails to address analytical cube scenarios, where the semantic alignment of two indicators requires considering the complete set of dimensions along which data is aggregated. In contrast, the approach proposed in the present work does not impose any limit on the number of join keys that can be specified. Recent approaches explore the adoption of self-supervised representation learning to tackle both the multi-key join and the join key semantic similarity issues [13]. These approaches cannot be applied in query-based scenarios, due to the lack of a query table.

Table 1
Feature matrix of dataset discovery methods. For metadata, T, C, and S represent Technical/schema-level, Content-level, and Semantic-level metadata, respectively.

Approach	Search	Focus	Metadata	Expl.
Google Dataset Search [7]	Keyword	Search	T,C,S	–
TRSS [8]	Keyword	Search	T,C,S	–
JOSIE [4]	Dataset	Join	T,C	–
D3L [9]	Dataset	Search, join, union	T,C	–
Santos et al. [10]	Dataset	Join	T,C	–
Auctus [11]	Keyword, dataset	Search, join (multi)	T,C	Source-level (snippets)
PEXESO [3]	Dataset	Join	T,C	–
Putrama et al. [12]	Dataset	Search	T,C	–
PolyJoin [13]	Dataset	Search, join (multi)	T,C	–
Constance [14]	Keyword, query	Search	T,C,S	Source-level (maps)
SEDAR [15]	Keyword, query	Search, join (multi)	T,C,S	Source-level (profiles)
Aurum [2]	Keyword, dataset, query	Search, path	T,C	Lake-level (paths)
KGLac [16]	keyword, dataset, query	Search, path	T,C,S	Lake-level (paths)
Our	Query	Search, join (multi)	T,C,S	Result-level (profile)

2.3. Metadata representation

Metadata representation has evolved from basic structural information to complex semantic models. Dataset-search approaches mostly focus on metadata on technical (i.e. schema-based) aspects and content, which include technical metadata like field names and types, probabilistic sketches for efficient similarity or correlation estimation [3,9,10], or data summaries [11]. Beyond these metadata, most Query-search approaches manage more expressive semantic metadata in graph-based representation, to support source exploration, search and join. They include mappings to graph entities, or similarity-based relations and primary-foreign key candidates [2], and inclusion dependencies [16]. Similarly, our work exploits schema- and content-level information, e.g., source profiles, along with mappings between source attributes and a global multidimensional model to address heterogeneity.

2.4. Explainability

Only a few data discovery systems explicitly support explainability, with approaches ranging from granular source-level transparency to structural lake-level navigation. At the source level, Auctus [11] and Constance [14] offer content snippets, statistical value distributions, and spatial maps to help users manually verify a dataset's relevance, while SEDAR [15] also provides transformation lineage and version history to explain a dataset's evolution. Other solutions shift the focus to lake-level explainability by modeling data relationships. Aurum [2] records a provenance graph for every query, allowing tracing the entire discovery path. Similarly, KGLac [16] facilitates exploration by discovering multi-hop join paths between disparate tables, enabling users to trace the logical connections across the entire data lake. To the best of our knowledge, our work is the first to support explainability at result level, supporting interpretability of results through the estimated data distribution of alternative solutions.

3. Methodology

Fig. 3 summarizes the overall workflow of the proposed methodology. The user submits a declarative analytical query expressed as a set of desired indicators, aggregation levels of a set of dimensions, and aggregate conditions on the source content. An example of query in natural language is[2]:

Find sources with AQI and VAT indicators aggregated by CITY and DAY, where at least 10% of data are about cities in the Lazio region.

The approach progressively identifies candidate data sources, estimates the usefulness of alternative joins without executing them, and finally explains the expected characteristics of each alternative.

The methodology consists of three main phases. (i) Source discovery, where metadata are exploited to reconcile heterogeneous schemas and efficiently identify combinations of analytic cubes matching the requested indicators and dimensions. (ii) Join cardinality estimation, where LSH Ensemble is used to estimate join cardinality and rank alternative solutions without materializing joins. (iii) Solution profile estimation, where source profiles are combined to estimate the distribution of the expected join result, helping analysts understand the characteristics of each recommendation before performing expensive data integration operations. The remainder of this section first presents the conceptual metadata model underlying the proposed framework, followed by a detailed description of the three phases of the methodology.

3.1. Conceptual metadata model

Rather than operating directly on heterogeneous data sources, the proposed approach models each source through conceptual metadata describing its content. This representation enables the framework to reason about schema compatibility, efficiently answer user queries, and explain the characteristics of candidate joins without executing them.

In the following, a data source S_k is modeled as a set of attributes: $S_k = \{a_1, \ldots, a_n\}$. Each attribute a_i is associated with a set of values. We denote by $\nu(a_i) = \{v_1, \ldots, v_m\}$ the multiset of values of the attribute a_i. Examples of data sources, their attributes and associated values are provided in Fig. 1. We hasten to note that, although for the sake of simplicity throughout the paper we refer to tabular data, the proposed approach can be applied to semi-structured data as well (e.g., JSON documents) following, for instance, the approach in [23].

Since the paper focuses on analytic cubes, the conceptual metadata model relies on three key concepts: (i) dimensions and indicators, which describe the multidimensional semantics of data sources; (ii) mappings between source attributes and multidimensional concepts; (iii) source profiles, which summarize the distribution of dimensional values across hierarchy levels. These concepts constitute the foundation for the query-driven join discovery algorithms presented in the next subsections, and are defined below.

3.1.1. Multidimensional representation

An analytic cube organizes data through dimensions and quantitative indicators. Dimensions define the perspectives of analysis and are organized into hierarchical levels, while indicators represent measures aggregated at a given level of granularity. The following formalization introduces these concepts. In the multidimensional model, a dimension

[2] We will provide the formal model of a query later on.

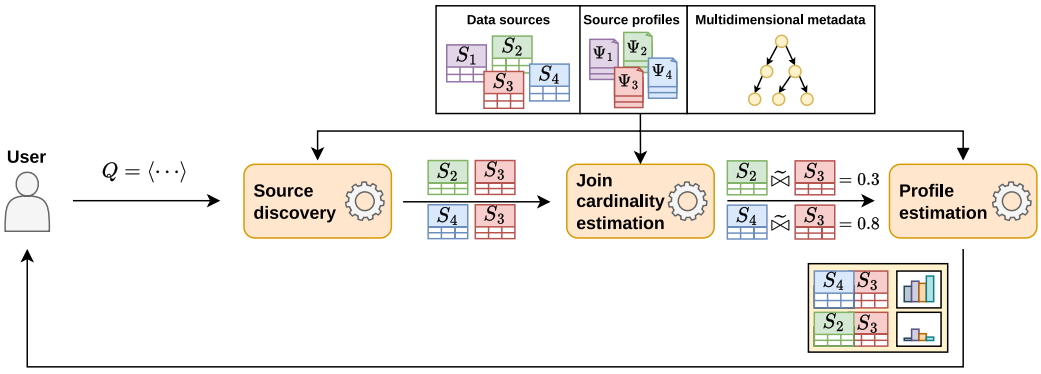


Fig. 3. An overview of the proposed methodology.

of analysis $\mathcal{D}_i$ is defined as a set $\mathcal{D}_i = \{L_1^{\mathcal{D}_i}, \ldots, L_m^{\mathcal{D}_i}\}$, of (dimension) levels $L_j^{\mathcal{D}_i}$. For the sake of simplicity, we will drop superscript $\mathcal{D}_i$ when the dimension of the level is clear from the context.

In the following, we will denote by $\mathbb{D}$ the set of possible dimensions. Levels of $\mathcal{D}_i$ are organized in a hierarchy through a *partial order* relation $\leq_L$ defined over $\mathcal{D}_i$.[3] For instance, CITY and REGION may be levels of a geographical dimension GEO, with CITY $\leq_L$ REGION. The partial order is also typically called *roll-up relation*. Each level $L_j \in \mathcal{D}$ is associated to a set of elements called *members*, denoted by $\delta(L_j) = \{m_1^{L_j}, \ldots, m_n^{L_j}\}$. In the following, we will refer to m_p, instead of $m_p^{L_j}$, whenever the level is clear from the context. In other terms, the set of members defines the *domain* of a level attribute (e.g., *Firenze* and *Siena* are members of the CITY level). A relation between members of two levels connected by a roll-up relation is also defined, called *partOf*. For instance, we can state that both cities *Firenze* and *Siena* are part of the *Tuscany* region. More formally, given a dimension $\mathcal{D} \in \mathbb{D}$, levels $L_i, L_j \in \mathcal{D}$ such that $L_i \leq_L L_j$, $\text{partOf}_{L_i,L_j} = \{\langle m_p, m_q \rangle | m_p \in \delta(L_i), m_q \in \delta(L_j), L_i \leq_L L_j\} \subseteq \delta(L_i) \times \delta(L_j)$.

The partOf relation is binary and non-injective: if $L_i \leq_L L_j$, then each member of L_i is in a partOf relation with only one member of L_j; in turn, each member of L_j is composed by at least one member of L_i. A graphical illustration is provided in Fig. 4.

An indicator ind_i is defined as a quantitative measure aggregated at different levels of the dimensional hierarchy. For instance, considering the total number of circulating vehicles as indicator, we may have $\langle Firenze, 5000 \rangle$, $\langle Siena, 500 \rangle$, at city level. Assuming *Firenze* and *Siena* as the only members of CITY, this leads a value $\langle Tuscany, 5500 \rangle$ at region level. Averages or other kinds of aggregation functions can be calculated as well. The set of all indicators will be denoted by $\mathcal{I}$.

Based on this representation, we now define how data source attributes are semantically aligned to multidimensional concepts.

3.1.2. Attribute mapping

In order to provide a unique semantics to data sources, two mapping functions are defined, for dimensional levels and indicators respectively:

Definition 1 (*Mapping Functions*)**.** Given the set of dimensions $\mathbb{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$, the set of indicators $\mathcal{I}$ and a source $S_k = \{a_1, \ldots, a_m\}$, the following mapping functions are defined:

- $\omega^{lev} : S_k \to \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_n$ is a partial injective function mapping an attribute $a_i \in S_k$ to a level $L_j \in \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_n$;
- $\omega^{ind} : S_k \to \mathcal{I}$ is a partial injective function mapping an attribute $a_i \in S_k$ to an indicator $ind_j \in \mathcal{I}$.

[3] According to the notion of partial order, the relation $\leq_L$ is reflexive, antisymmetric, and transitive.

Injectivity of the mappings guarantees that there not exist two attributes of the same source mapped to the same level or indicator. Furthermore, we must ensure that each attribute a_i can be mapped to either a level or an indicator, not both. Hence, the following properties must hold for mappings:

- $\omega^{lev}(a_i) = L_j \Rightarrow \nexists\, ind_z \in \mathcal{I}, \omega^{ind}(a_i) = ind_z$
- $\omega^{ind}(a_i) = ind_z \Rightarrow \nexists\, L_j \in \mathcal{D}_k, \omega^{lev}(a_i) = L_j$

Fig. 4 graphically shows a mapping of attribute *location* to the level CITY (red arrow).

When an attribute $a_i \in S_k$ is mapped to a level L_j (i.e., $\omega^{lev}(a_i) = L_j$), then its values $\nu(a_i)$ are aligned to members $\delta(L_j)$.

In the following we will denote by $\mathcal{S}_{L_j} = \{S_k \mid \exists\, a_i \in S_k, \omega^{lev}(a_i) = L_j\}$ the set of sources containing attributes mapped to the level L_j, and by $\mathcal{S}_{ind_j} = \{S_k \mid \exists\, a_i \in S_k, \omega^{ind}(a_i) = ind_j\}$ the set of sources containing attributes mapped to ind_j.

We finally introduce lightweight source profile metadata that are exploited in query answering.

3.1.3. Data source profile

We define the data source profile as a set of attribute profiles, one for each attribute. We first define the plain profile (or simply profile) of an attribute a_i, which intuitively summarizes the distribution of values $\nu(a_i)$. We then extend this notion across hierarchy levels through the concept of aggregated profile.

Definition 2 (*Attribute Profile*)**.** Given a source S_k and an attribute $a_i \in S_k$ mapping to level L_j of the dimension $\mathcal{D}$: $\omega^{lev}(a_i) = L_j^{\mathcal{D}}$, the *profile item* of a_i for member $m_p \in \delta(L_j^{\mathcal{D}})$ is defined as $\psi(m_p, a_i) = \langle m_p, y_p \rangle$, where $y_p = |\{v \in \nu(a_i) : v = m_p\}|$. The *profile* of attribute a_i is then defined as the set of profile items: $\Psi(a_i) = \{\psi(m_1, a_i), \ldots, \psi(m_n, a_i), \psi(\perp, a_i)\}$, where the last profile item intuitively collects the number of values that are not members of $L_j^{\mathcal{D}}$: $\psi(\perp, a_i) = \langle \perp, |\nu(a_i)| - \sum_{p=1}^{n} y_p \rangle$. The corresponding set of members $\{m_1, \ldots, m_n\}$ will be denoted by $dom(\Psi(a_i))$.

From the definition, it turns out that $\Psi(a_i)$ is defined only for attributes which can be correctly mapped to dimension levels.

Profiles can be rolled up along dimension hierarchies, allowing queries at different levels of granularity.

Definition 3 (*Aggregated Attribute Profile*)**.** Given the profile $\Psi(a_i) = \{\psi(m_1, a_i), \ldots, \psi(m_n, a_i)\}$ for an attribute a_i such that $\omega^{lev}(a_i) = L_j^{\mathcal{D}}$, and given a level $L_z^{\mathcal{D}}$ such that $L_j^{\mathcal{D}} \leq_L L_z^{\mathcal{D}}$, the *aggregated profile item* of a_i for member $m_q \in \delta(L_z)$ is defined as : $\psi^{\uparrow}(m_q, a_i) = \langle m_q, y_q \rangle$, where:

$$y_q = \sum_{\langle m_p, y_p \rangle \in \Psi(a_i)} y_p \mid \exists\, m_p \in \delta(L_j) \wedge \langle m_p, m_q \rangle \in \text{partOf}_{L_j,L_z}$$

The *aggregated profile* of attribute a_i at level L_z is then defined as the set of aggregated profile items: $\Psi_{L_z}^{\uparrow}(a_i) = \{\psi^{\uparrow}(m_1, a_i), \ldots, \psi^{\uparrow}(m_h, a_i)\}$, $m_i \in \delta(L_z)$.

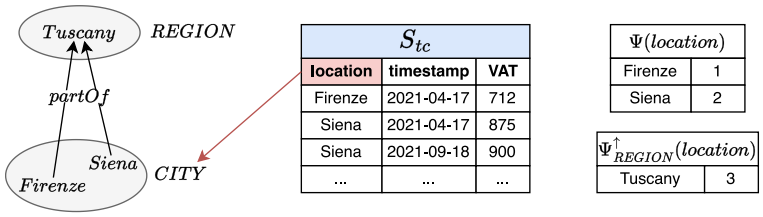


Fig. 4. Profile and aggregated profile of attribute *location* of source S_{tc}. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Example 3 (*Profiles and Aggregated Profiles*)**.** Fig. 4 shows an example of profile and aggregated profile of attribute *location* of the source S_{tc}, considering only the rows explicitly shown. As before, we assume a GEO dimension with levels $CITY \leq_L REGION$, and $\{\langle Firenze, Tuscany\rangle, \langle Siena, Tuscany\rangle\} \subseteq partOf_{CITY,REGION}$. The attribute *location* maps to the CITY level of the GEO dimension.

3.2. Source discovery

Source discovery is the first phase of the methodology. Starting from a declarative analytical query, the objective is to identify combinations of analytic cubes capable of satisfying the requested indicators and dimensions. To formalize this process, we first define the query model and the form of the expected result.

Definition 4 (*Query*)**.** A query is a tuple:

$$Q = \langle \mathcal{I}_Q, \mathcal{L}_Q, \mathcal{F}_Q \rangle,$$

where $\mathcal{I}_Q = \{ind_1, \ldots, ind_{n_Q}\}$, is a set of indicators, $\mathcal{L}_Q = \{L_1, \ldots, L_{m_Q}\}$ is a set of levels representing the dimensional schema at which the result is requested, and $\mathcal{F}_Q = \{f_1, \ldots, f_{g_Q}\}$ is a set of filters. A dimensional schema is well-formed if the levels belong to distinct dimensions: $\forall L_j, L_z \in \mathcal{L}_Q, L_j \neq L_z, L_j \in D_h \wedge L_z \in D_g \implies D_h \neq D_g$.

A filter defines an aggregated condition on the occurrences of a specific value in a data source. Specifically, a filter is a tuple $f_v = \langle m_p, \theta, n_p \rangle$ where:

- $m_p \in \delta(L_j)$ is a member of a certain dimensional level L_j;
- θ is a comparison operator (for example, $\leq, \geq$, or $=$);
- $n_p \in [0, 1]$.

An *evaluation* function $eval(f_v, S_k)$ is designed to evaluate filters exploiting profiles: if the filter can be correctly applied, then the function evaluates to the boolean condition

$$\frac{y_p}{|v(a_i)|} \; \theta \; n_p,$$

where y_p is the value of the member's profile $\langle m_p, y_p \rangle$. Conditions of applicability are that (1) the source has an attribute a_i that is mapped to a level, L_j and either (2) m_p is a member of L_j or (3) m_p is a member of a higher level $L_z \geq L_j$. In this case, the corresponding aggregated profile item is taken for comparison.

Example 4 (*Filter*)**.** The filter $\langle Firenze, \geq, 0.1 \rangle$ can be applied to retrieve sources with an attribute (mapped to the level CITY) that contains at least 10% of values corresponding to *Firenze*. Similarly, a filter $\langle Tuscany, \geq, 0.1 \rangle$ can be applied to the same data source to constrain the occurrences of cities located in the *Tuscany* region.

A possible answer to a query Q is a set of data sources that (i) have the same dimensional schema of Q, (ii) contain the indicators in $\mathcal{I}_Q$, and (iii) satisfy all filters in $\mathcal{F}_Q$. The following definition provides a more formal characterization:

Algorithm 1 Source Discovery

Input: $Q = \langle \mathcal{I}_Q, \mathcal{L}_Q, \mathcal{F}_Q \rangle, \mathcal{I}_Q = \{ind_1, \ldots, ind_{n_Q}\}, \mathcal{L}_Q = \{L_1, \ldots, L_{m_Q}\}$, $\mathcal{F}_Q = \{f_1, \ldots, f_{g_Q}\}$
Output: $\mathcal{S}_Q = \{\mathbb{S}_1, \ldots, \mathbb{S}_r\}$

1: $\mathcal{S} = \bigcap_{i=1}^{m_Q} \mathcal{S}_{L_i}$
2: **for** $S_k \in \mathcal{S}$ **do**
3: **for** $f_v \in \mathcal{F}_Q$ **do**
4: **if not** $eval(f_v, S_k)$ **then**
5: $\mathcal{S} = \mathcal{S} \setminus \{S_k\}$
6: **break**
7: $\mathcal{S}_Q = \{\mathbb{S}_j = \{\hat{S}_1, \ldots, \hat{S}_n\} \mid \hat{S}_i \in (\mathcal{S}_{ind_i} \cap \mathcal{S}), i = 1, \ldots, n_Q\}$
8: **return** $\mathcal{S}_Q$

Definition 5 (*Solution to a Query*)**.** Given a query $Q = \langle \mathcal{I}_Q, \mathcal{L}_Q, \mathcal{F}_Q \rangle$, with $\mathcal{I}_Q = \{ind_1, \ldots, ind_{n_Q}\}$ and $\mathcal{L}_Q = \{L_1, \ldots, L_{m_Q}\}$, a set of sources $\mathbb{S} = \{\hat{S}_1, \ldots, \hat{S}_n\}$ is a solution for Q iif $\forall \hat{S}_k \in \mathbb{S}$: (i) $\hat{S}_k \in \bigcap_{j=1}^{m_Q} \mathcal{S}_{L_j}$, (ii) $\hat{S}_k \in \mathcal{S}_{ind_k}$, and (iii) $\forall f_v \in \mathcal{F}_Q, eval(f_v, \hat{S}_k) = True$.

We can have several alternative solutions to a query. The procedure to identify them is described in Algorithm 1.

First, the set $\mathcal{S}$ of data sources mapping to all levels in $\mathcal{L}_Q$ is identified (line 1), since only data sources matching the dimensional schema of Q are considered. These sources are then checked against all filter conditions in $\mathcal{F}_Q$ (lines 2–6). Finally, the set $\mathcal{S}_Q$ is generated by considering only combinations of sources in $\mathcal{S}$ that collectively provide all the indicators in $\mathcal{I}_Q$. Note that the cardinality of a solution is less than or equal to the number of indicators in $\mathcal{I}_Q$: $| \mathbb{S}_j | = n \leq n_Q$. The equality holds when each data source in $\mathbb{S}_j$ provides a distinct indicator, while $n < n_Q$ if at least one data source $\hat{S}_i \in \mathbb{S}_j$ contains more than one indicator of $\mathcal{I}_Q$. At the limit, if a single source contains all the indicators needed to answer the query $n = 1$. In the following we provide a brief analysis of the complexity of the source discovery phase.

Complexity analysis. Let us consider a query Q specifying n_Q indicators, m_Q levels and g_Q filters. Line 1 calculates the intersection of m_Q sets. Assuming each includes n_{S_L} sources, the cost is $O(m_Q * n_{S_L})$. The cycle of lines 2–6 executes the *eval* function (which we assume to cost $O(1)$) $n_S * g_Q$ times. In line 7, to calculate the sets S_{ind_i} is enough to consider all n_S elements. Finally, the cartesian product requires $O(n_S^{m_Q})$ operations. In total, the cost is $O(m_Q * n_{S_L} + n_S * g_Q + n_S^{m_Q})$. Asymptotically, since $n_{S_L}, m_Q, g_Q \ll n_S$, the cost is dominated by $O(n_S^{m_Q})$.

Example 5 (*Source Discovery*)**.** Suppose that the user wants to analyze AQI and VAT indicators by CITY and DAY, applying a filter which requires cities in the *Tuscany* region to appear in at least 10% of the values. The Algorithm 1 will first find all the data sources $\mathcal{S} = \{S_{cs}, S_{nw}, S_{cn}, S_{ne}, S_{tc}, S_{tn}\}$ containing attributes mapped to CITY and DAY. Then, given that only S_{cs}, S_{cn} and S_{tc} contain cities of *Tuscany* for at least the 10% of the total elements, data sources S_{nw}, S_{ne} and S_{tn} will be filtered out from $\mathcal{S}$ (see Fig. 5). The result $\mathcal{S}_Q$ will contain the combinations $\mathbb{S}_1 = \{S_{cs}, S_{tc}\}$ and $\mathbb{S}_2 = \{S_{cn}, S_{tc}\}$, since S_{cs} and S_{cn}

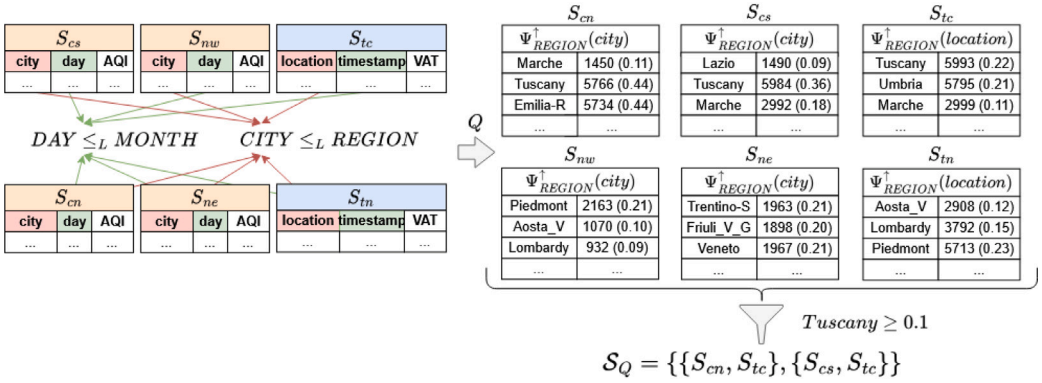


Fig. 5. An example of Source Discovery for the query $Q = \langle\{AQI, VAT\}, \{City, Day\}, \{\langle Tuscany, \geq, 0.1\rangle\}\rangle$. For simplicity, aggregated profiles are explicitly shown with the normalized aggregated values in brackets.

each contain an attribute mapped to the indicator AQI and S_{tc} contains an attribute mapped to the indicator VAT.

3.3. Join cardinality estimation

Multiple source combinations may satisfy the same query. Executing every candidate join to assess its usefulness is generally infeasible in the context of very large repositories. Therefore, we associate each candidate solution with an estimate of its join cardinality, referred to in the following as the *joinability index* (ji), which can be computed efficiently without materializing the join.

The key observation is that joins are performed on the dimensional attributes of candidate sources. Since these attributes represent the key of the sources, the cardinality of the join between two data sources corresponds to the number of common combinations of their dimensional levels, that is, to the cardinality of the intersection of the corresponding sets. This quantity can be efficiently estimated through set containment, which measures the fraction of elements of one set that are contained in another.

Estimating the cardinality of a join involving more than two sources is considerably more challenging, as it requires estimating the cardinality of the intersection among multiple sets. Rather than attempting a direct estimation, the proposed approach decomposes the problem into a sequence of pairwise containment estimations. Specifically, the sources are processed in ascending order of cardinality. At each iteration, the smallest source is selected as the reference,[4] since the cardinality of the final join cannot exceed its size. The containment between this source and each of the remaining sources is then estimated using LSH Ensemble [1], with the minimum estimated value providing an upper bound on the join cardinality. The same procedure is recursively applied to the remaining sources, and the final estimate is obtained as the minimum of the upper bounds computed at each recursive step.

It is noteworthy that this procedure returns an overestimate of the cardinality of the join of k sources. To give an example, given $S_x = \{a, b, c\}$, $S_y = \{a, b, d, e\}$ and $S_z = \{b, c, d, e\}$, the estimate turns out to be 2, while the true cardinality of the join is 1.

The whole procedure is presented in Algorithm 2 where, with some abuse of notation, we denote by S_i the data source restricted to dimensional attributes, and by $|S_i|$ the number of data items of the source. The core of the recursive step is a dichotomous search over the containment threshold τ: at each iteration, the function *LSH_Ensemble* identifies the sources whose estimated containment with S^* is at least τ (line 10). If at least one source does not satisfy the threshold, the upper half of the search interval is discarded (line 11); otherwise, the lower half is discarded and τ becomes a new lower bound for the containment score (lines 13–14). The search terminates when the interval width becomes smaller than the precision parameter λ, yielding an estimate τ^* of the containment score. As expected, smaller values of λ provide more accurate estimates at the expense of longer execution times. Since the sources are processed in ascending order of cardinality, the cardinality of the join can only decrease at each recursive step. Therefore, the upper bound of the search interval is initialized to the containment estimate obtained in the previous recursion (line 7), avoiding unnecessary search iterations. The dichotomous search strategy was introduced as an enhancement of a previous version of the algorithm [5], leading to improved overall computational efficiency.

We provide below a brief analysis of the complexity of the Algorithm.

Complexity analysis. Let us consider an input list of n elements. Lines 1–7 involve initialization and elementary operations, each executing in constant time with a cost of $O(1)$. Lines 8–14 perform a bounded number of iterations k, ranging from 1 to $log_2(1/\tau)$, where τ is a fixed threshold. For each iteration, LSH Ensemble is executed with sublinear complexity, leading to overall asymptotic complexity $O(n^\rho)$, $0 \leq \rho \leq 1$. At line 15, the algorithm computes the minimum of a list of length $n-1$, which requires linear time, $O(n)$. Subsequently, line 16 performs a constant-time comparison and invokes the algorithm recursively on a list of decreasing size. The running time $T(n)$ therefore satisfies the recurrence $T(n) = O(n^\rho) + O(n) + T(n-1), \quad T(1) = O(1)$. Expanding the recurrence yields $T(n) = O(n) + O(n-1) + \cdots + O(1) = O\left(\sum_{k=1}^{n} k\right)$. Since $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$, it follows that asymptotically $T(n) = O(n^2)$.

3.4. Solution profile estimation

Cardinality alone does not explain why a candidate join is useful. To illustrate this, consider the histograms in Fig. 2. It turns out that the solution generated by joining sources S_{cn} and S_{tc} completely rules out data on three of the six regions. It is possible that an alternative solution produces a join of lower cardinality, which however provides a more balanced distribution of data across regions, making it better suited to analytical requirements. Therefore, the final phase of the methodology is devoted to estimating the profile of each candidate solution. Hereby, we rely on a histogram-based approach assuming independence of the dimensional attributes, which is a simple yet efficient approach and one of the most adopted solutions for industrial DBMSs [24]. Accordingly, the profile of a join can be approximated independently for each attribute, which allows us to start directly from the attribute profiles of the participating sources, without executing the join.

Let $J = \bowtie_{k=1}^{n} S_k$ be the join of sources in the solution $\mathbb{S} = \{S_1, \ldots, S_n\}$ of the query Q defined over the dimensional schema $\mathcal{L}_Q = \{L_1, \ldots, L_{m_Q}\}$. We denote by $\Psi^J = \{\Psi^J_1, \ldots, \Psi^J_{m_Q}\}$ the profile of J, where $\Psi^J_i = \{\langle m, y\rangle | m \in \delta(L_i), y \geq 0\}$ is the profile of the attribute

[4] If more than one source exists, one of them is selected arbitrarily.

Algorithm 2 *Join Cardinality Estimation*

Input: $\mathbb{S}_i = \{S_1, ..., S_n\} \in \mathcal{S}_Q, S^* \in \arg\min_{S_j \in \mathbb{S}_i}(|S_j|), V = |S^*|, \lambda \in [0, 1]$
Output: Estimated cardinality of the join $\bowtie_{i=1}^{n} S_i$

1: **function** JoinCardEst($\mathbb{S}_i, S^*, V, \lambda$)
2: $\tau^* = 0$
3: **if** $|\mathbb{S}_i| == 1$ **then**
4: **return** V
5: **else**
6: $first = 0$
7: $last = V/|S^*|$
8: **while** $first <= (last - \lambda)$ **do**
9: $\tau = (first + last)/2$
10: $\Lambda = LSH_Ensemble(S^*, \mathbb{S}_i \setminus \{S^*\}, \tau)$
11: **if** $|\Lambda| < (|\mathbb{S}_i| - 1)$ **then** $last = \tau$
12: **else**
13: $first = \tau$
14: $\tau^* = \tau$
15: select $S_m \in \arg\min_{S_j \in \mathbb{S}_i \setminus \{S^*\}}(|S_j|)$
16: **return** $min(\tau^*|S^*|, \texttt{JoinCardEst}(\mathbb{S}_i \setminus \{S^*\}, S_m, \tau^*|S^*|, \lambda))$

Algorithm 3 *Profile estimation*

Input: $\mathcal{L}_Q = \{L_1, \ldots, L_{m_Q}\}, \mathbb{S} = \{S_1, \ldots, S_n\}$
Output: Ψ^J

1: $\Psi^J_1, \ldots, \Psi^J_{m_Q} = \emptyset$
2: **for each** $L_j \in \mathcal{L}_Q$ **do**
3: $A_j = \{a \in S_i \mid \omega^{lev}(a) = L_j, S_i \in \mathbb{S}\}$
4: $M_j = \bigcap_{a \in A_j} dom(\Psi(a))$
5: **for each** $m_h \in M_j$ **do**
6: $P_h = \{\langle m_h, y_h \rangle \in \Psi(a), a \in A_j\}$
7: $I(m_h) = \min_{P_h}(y_z)$
8: $\Psi^J_j = \Psi^J_j \bigcup \{\langle m_h, I(m_h) \rangle\}$
9: $\Psi^J = \bigcup_{i=0}^{mQ} \Psi^J_i$
10: **return** Ψ^J

corresponding to level L_i in J, and the value y represents the number of occurrences of member $m \in \delta(L_i)$ in J. We estimate y as $min_k(y_k)$, where $\langle m, y_k \rangle$ is the profile item for the member m in the source $S_k \in \mathbb{S}$. Since dimensional attributes form a key for the data sources, this value is an upper bound for y. Indeed similar arguments apply as for the case of joinability index: each occurrence of a member in the join must correspond to an occurrence of that member in every participating source. Consequently, its frequency in the join cannot exceed the minimum frequency across sources.

Note that aggregated profiles of the solution can be easily derived from Ψ^J and presented to the user.

Operationally, the procedure to generate the estimated profile is described in Algorithm 3. As a first step, for each level $L_j \in \mathcal{L}_Q$, the attributes of sources in $\mathbb{S}$ that are mapped to L_j are identified and stored in the set A_j (line 3). Then, members that are common to all the sources are stored in the set M_j (line 4). For each member m_h in M_j, the set of its profile items across all the sources is collected (line 6), and the minimum value of y_h is added to the estimated profile of the level at hand (lines 7–8). In the following we provide a brief analysis of the complexity of profile estimation.

Complexity analysis. Let us consider an input list of n sources for a solution and m_Q levels in the query. The cycle in lines 2–8 is executed m_Q times. In line 3, all attributes in the n sources mapped to L_j are retrieved (at most 1 attribute per source), which requires to iterate over all sources, i.e., n steps. Using a hash-based intersection algorithm, the intersection in line 4 can be done in time linear in the sum of the set sizes, i.e., $|A_j| * |dom(\Psi(a))|$. Considering an average number n_{attr} of members in the profile of an attribute, the cost is $|A_j| * n_{attr}$, where $|A_j| = n$ since each source has one attribute mapped to L_j. In lines 5–8, there are $|M|$ iterations, where $|M|$ is the average of the $|M_j|$, in which all A_j are processed to find the minimum (n times, line 6–7) and finally a union is performed, for a total of $|M| * m_Q * (n + 1)$ operations. The total number of operations is therefore $T = O(m_Q * n) + O(m_Q * n * n_{attr}) + O(m_Q * |M| * n)$. Since m_Q and n are much smaller than $|M|$ and do not grow with $|M|$, and given that $|M| < n_{attr}$, the cost simplifies to $T = O(n_{attr}) + O(|M|) = O(n_{attr})$.

4. Implementation

In this section, we report an implementation of the discussed approach on top of a Data Lake architecture which includes a Knowledge Graph featuring a *knowledge layer*, representing dimensions and indicators of interest according to a reference ontology, and a *metadata layer*, including metadata on the sources, and mappings to the knowledge layer. The knowledge layer provides a representation of the domain knowledge in terms of definitions of indicators, dimension hierarchies and dimension members. We refer to KPIOnto[5] as the ontological schema, which provides the terminology for the formal definition of the concepts described in Section 3.1. We refer the interested reader to [25] for a general description and to [26] for a comparison among alternative models, including standard RDF Schema vocabularies for aggregate statistics such as the RDF Data Cube [27]. Similarly, the metadata layer relies on a graph representation, following the approach proposed in [23] to uniformly represent attributes of both structured and semi-structured sources, while the representation of the profiles is based on the vocabulary proposed in [28]. Mappings between source attributes and dimensional levels are identified by a semi-automatic process, as discussed in [5], which relies on the evaluation of the similarity between attribute values and members of the dimensional levels, based on an efficient estimation of set containment.

Fig. 6 illustrates an excerpt of the Knowledge Graph related to dimensions and indicators of Example 2. The schema for a data source is shown, which includes three attributes: 'VAT', 'timestamp', and 'location'. The 'VAT' attribute is mapped to the VAT indicator through the property `dl:mapToIndicator`, while 'location' and 'timestamp' are mapped to the CITY and the DAY level (`dl:mapTo`), respectively.

5. Evaluation

In this section, we illustrate the efficiency and effectiveness of the proposed methodology. Section 5.1 is dedicated to present an evaluation on synthetic datasets, allowing to study the role of various dataset features on performances, and is followed by an evaluation on real-world datasets in Section 5.2. In Section 5.3 we evaluate the effectiveness of the proposed approach on the case study introduced in Section 1. Finally, a conclusive discussion summarizes the main insights in Section 5.4.

Experiments were carried out on a virtual machine running Ubuntu 22.04, with an 8-core Intel(R) Xeon(R) Gold 6238R CPU and 64 GB of RAM.[6]

5.1. Synthetic datasets

In the following, we present some evaluations of the approach on synthetic datasets. In 5.1.1 an analysis of the computation times of the source discovery algorithm (Algorithm 1) is presented. In 5.1.2, the join cardinality estimation algorithm (Algorithm 2) is evaluated in terms of (i) efficiency, by comparing it to join computation times,

[5] http://w3id.org/kpionto.
[6] Please note that the specifications of the virtual machine are lower than those typically used for Data Lake environments.

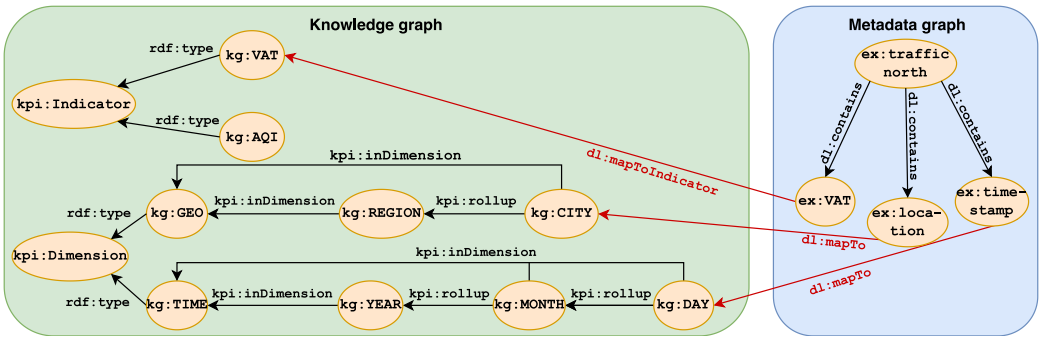


Fig. 6. An excerpt of the Knowledge Graph describing the schema for a data source.

and (ii) efficacy, by considering the error between estimated and actual join cardinalities, while in 5.1.3 we assess the quality of the estimated rankings.

To support the experiments, we defined a synthetic Knowledge Graph consisting of a number of 10 indicators and 4 dimensions, each including 5 levels. For each dimension, partial order relations between its levels are defined as follows: $L_0 \preceq_L L_1$, $L_1 \preceq_L L_2$, $L_2 \preceq_L L_3$, and $L_3 \preceq_L L_4$. For the generation of members, we set the growing factor to 1 and the initial population to 10. The growth factor determines how the population size increases from one level to the next in the hierarchy. As such, the level L_k of a dimension will include 10^k distinct members, where k is the level's index in the hierarchy defined by the partial order relations. Thus, L_0 has 10 members, L_1 has 10^2 members, and so on.

5.1.1. Source discovery

With reference to Algorithm 1, efficiency is evaluated based on the time required to identify all combinations of data sources that can respond to the query. The algorithm operates on source metadata, specifically on the set $\mathcal{S}_{L_i}$ of sources mapping to level L_i and the set $\mathcal{S}_{ind_j}$ of sources mapping to the indicator ind_j. These two sets are computed only once at the time the source is added to the Data Lake.

Settings. To evaluate the execution time of the algorithm, we consider the following scenario: (i) all sources contain attributes that map to all levels in $\mathcal{L}_Q = \{L_1, \ldots, L_{m_Q}\}$; (ii) each of the indicators in $\mathcal{I}_Q = \{ind_1, \ldots, ind_{n_Q}\}$ appears in a data source with a 30% probability; (iii) no filters are defined (i.e., $\mathcal{F}_Q = \emptyset$). This scenario describes an unfavorable condition for the execution of the algorithm. Indeed, from (i) it turns out that no source can be filtered out (line 1 of Algorithm 1). The same applies for (iii), since filters typically reduce the number of sources to be considered. If each source contained all n_Q indicators in $\mathcal{I}_Q$, Algorithm 1 would generate k^{n_Q} combinations, where k is the number of sources in the Data Lake. To make the experiment more realistic, we introduced a 30% probability that an indicator belongs to a source. We think that this value would be considered high in a real-world context, where this probability is expected to be inversely proportional to the number of available sources. Algorithm 1 has been executed 10 times for each pair k and n_Q, where $k \in \{10, 20, 50, 100\}$ and $n_Q \in \{2, 3, 4, 5\}$.

Results. The average number of combinations returned by the algorithm and the average execution time (in seconds) are reported in Table 2. The results show that execution times, as expected, increase linearly with the number of combinations identified. It is worth highlighting that, even under the unfavorable scenario designed for this experiment, the execution times remain very low and suitable for practical use.

5.1.2. Estimation of the join cardinality

The estimation of join cardinality performed by Algorithm 2 is first evaluated in terms of efficiency by comparing its computation time to that of an actual join. Next, we assess its effectiveness by measuring the error between the estimated and true join cardinalities.

Table 2
Source Discovery algorithm: average and standard deviation of the number of returned combinations and execution time (seconds). Standard deviation is not calculable (N/C) when all times fall below the 0.001 s minimum detection threshold.

k	n_Q	Combinations		Time [s]	
		avg	std	avg	std
10	2	7.2	3.74	<0.001	N/C
	3	21	13.2	<0.001	N/C
	4	99.2	121	<0.001	N/C
	5	224	200	<0.001	N/C
20	2	47	29	<0.001	N/C
	3	290	129	<0.001	N/C
	4	1'687	1'830	0.002	0.002
	5	8'781	5'077	0.012	0.007
50	2	251	121	<0.001	N/C
	3	2'437	462	0.002	N/C
	4	41'971	18'271	0.048	0.020
	5	824'699	337'354	1.134	0.471
100	2	886	90	<0.001	N/C
	3	25'481	5'894	0.024	0.006
	4	780'995	164'746	0.916	0.197
	5	25'312'877	4'776'318	35.625	6.749

Settings. Since Algorithm 2 takes a combination of sources as input, we generated several combinations of synthetic sources by varying the number of sources (from 2 to 5) as well as the source cardinalities, resulting in datasets containing 10^5, 10^6, and 10^7 rows. For each combination, the first source was generated with 4 attributes, each of which was mapped to a dimensional level. The values of these attributes were randomly picked from members of the mapping levels. Then, the remaining sources in the combination were generated by copying the first source and replacing a portion of the rows with noise, namely random values that do not map to the members of the levels. The noise was introduced because, in a real world scenario, data in a Data Lake might be sparse and dirty, meaning not all stored data is useful. Hence, the result is a set of combinations $\mathbb{S}_r = \{S_1, \ldots, S_k\}$ of data sources, where $k \in \{2, 3, 4, 5\}$, $r \in \{0\%, 10\%, 20\%, 30\%, 40\%, 50\%\}$ is the percentage of noise added to sources, and $|S_i| \in \{10^5, 10^6, 10^7\}, i = 1, \ldots, k$. In the generated sources, we included only attributes that map to levels, as additional attributes do not impact the execution of Algorithm 2. Indeed, the join among data sources is made only over dimensional attributes, which define a key for each data source. The estimation proposed in Algorithm 2 is based on the LSH Ensemble computed over the concatenation of dimensional levels. Hence, the number of attributes mapping to dimensional levels also does not affect the execution time of Algorithm 2. The experiments have been performed with the parameter $\lambda = 0.049$, which is small enough to provide a good precision. For each combination of sources, Algorithm 2 and the actual join among sources are executed 10 times.

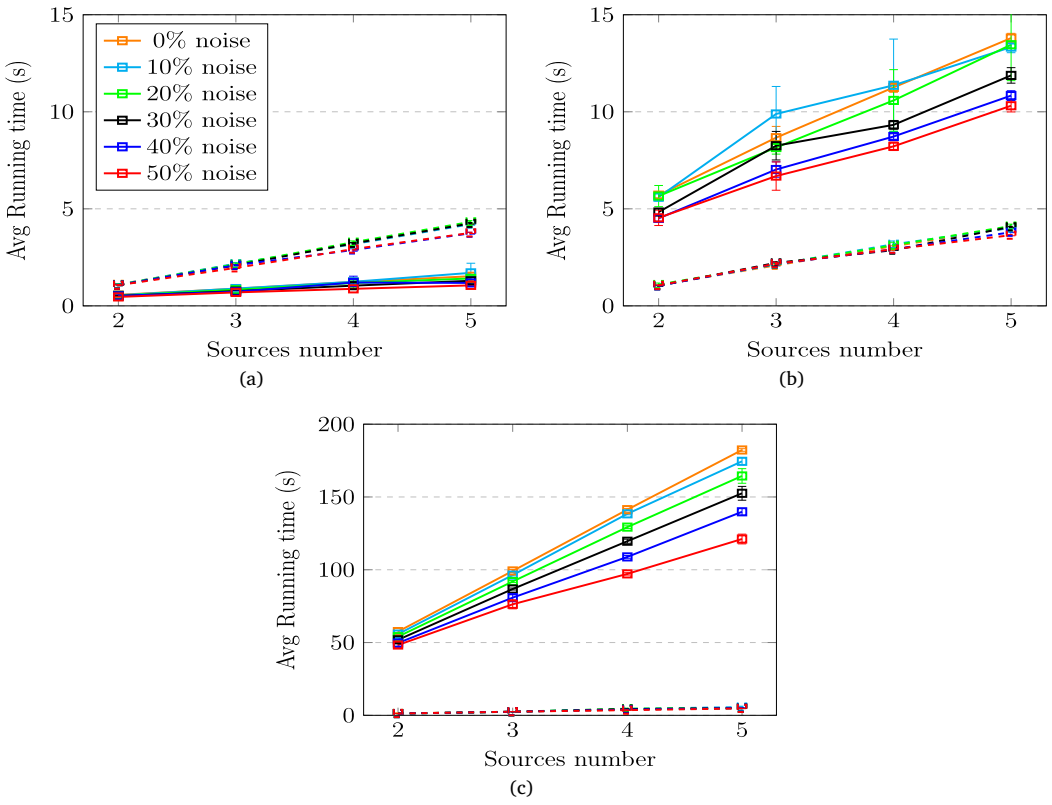


Fig. 7. Average and standard deviation (vertical lines) of execution time of the proposed join cardinality estimation algorithm (dotted lines), with respect to the execution time of the actual join (solid lines). (a) $|S_i| = 10^5$, (b) $|S_i| = 10^6$, (c) $|S_i| = 10^7$.

Results. Fig. 7 compares the execution times of the proposed approach for estimating the join cardinality with those of the actual join computation used to obtain the true cardinality. Specifically, the figure reports the average execution times and standard deviations (over 10 runs) as functions of the number of data sources involved in the join, the source size, and the percentage of added noise. The differences among the three graphs highlight that the benefits of using the joinability index increase with the size of the data sources. In fact, when dealing with relatively small sources ($|Si| = 10^5$), the time required to compute the joinability index exceeds the time needed to perform the actual join. However, as the size of the data sources increases, the proposed methodology yields a performance gain up to 96%. This occurs because the LSH Ensemble is computed over fixed-size MinHash signatures, which makes its execution time independent of dataset size.

Regarding the evaluation of effectiveness, we assessed the quality of the joinability index using the Mean Absolute Percentage Error (MAPE), which measures the deviation between the true join cardinalities and those estimated by the joinability index. The graphs in Fig. 8 show the average MAPE value over 10 runs and its standard deviation as a function of the number of sources in each combination $\mathbb{S}_r$ and the percentage of added noise. Each figure corresponds to a different source size. It can be observed that the highest quality results in cardinality estimation are observed for join involving two datasets. When more than two data sources are considered, and the level of noise increases, the accuracy of the joinability index estimation declines.

5.1.3. Ranking comparison

The quality of the estimated ranking of solutions is evaluated by analyzing its similarity with respect to the ranking based on actual cardinalities computed after the execution of the join. In case the two rankings are highly correlated, it suggests that, even if the estimation lacks high accuracy in absolute terms, the user can still identify the best combinations based on available information.

Settings. The ranking based on estimated cardinality and the ranking based on actual join cardinalities have been compared through the *Kendall* τ [29] and *Rank Biased Overlap (RBO)* [30] metrics. Kendall τ is a rank correlation coefficient which is based on the number of concordant and discordant pairs between two rankings. Its value τ is in $[-1, 1]$ range, where $\tau = 1$ indicates a perfect agreement, while $\tau = -1$ indicates a strong disagreement, i.e., the two rankings are completely different. On the other hand, RBO compares two rankings by giving more importance, thus higher weights, to the first positions. Its value is in the $[0, 1]$ range, meaning that if the value is near to 1 the two rankings are highly similar. The rankings are derived from results obtained in 5.1.2, by fixing both the number of sources k in a combination and number of rows $n = |S_i|$ per source. Thus, a ranking $\mathcal{R}_{k,n}$ is the ordered set of 6 combinations $\mathbb{S}_r$, each for every percentage of noise r. For example, the ranking $\mathcal{R}_{2,10^5}$ orders the combinations of sources $\mathbb{S}_{0\%} = \{S_1, S_2\}$, $\mathbb{S}_{10\%} = \{S_3, S_4\}$, $\mathbb{S}_{20\%} = \{S_5, S_6\}$, $\mathbb{S}_{30\%} = \{S_7, S_8\}$, $\mathbb{S}_{40\%} = \{S_9, S_{10}\}$, $\mathbb{S}_{50\%} = \{S_{11}, S_{12}\}$ where each source has 10^5 rows.

Results. Table 3 shows the averages and standard deviations of Kendall τ and RBO metrics, respectively. From the resulting values, it can be noted that rankings of combinations with $k = 4$ or $k = 5$ are the best most of the time. This may seem counterintuitive because, on the contrary, the estimation error (MAPE) increases as k. However, this is due to the fact that real join cardinalities for pairs of combinations that are near in the ranking are quite similar. For example, the second combination $\mathbb{S}_{10\%}$ in $\mathcal{R}_{2,n}$ has a real join cardinality which differs by 10% from that of $\mathbb{S}_{0\%}$. In the same way, the others in the same ranking differs by a similar percentage from the previous combination, and for such quantities the joinability index can give the same estimates of cardinality, thus leading to incorrect estimated rankings. On the contrary, combinations in rankings for $k = 4$ or $k = 5$, have real join cardinalities that differ from the previous ones in the rankings by 30

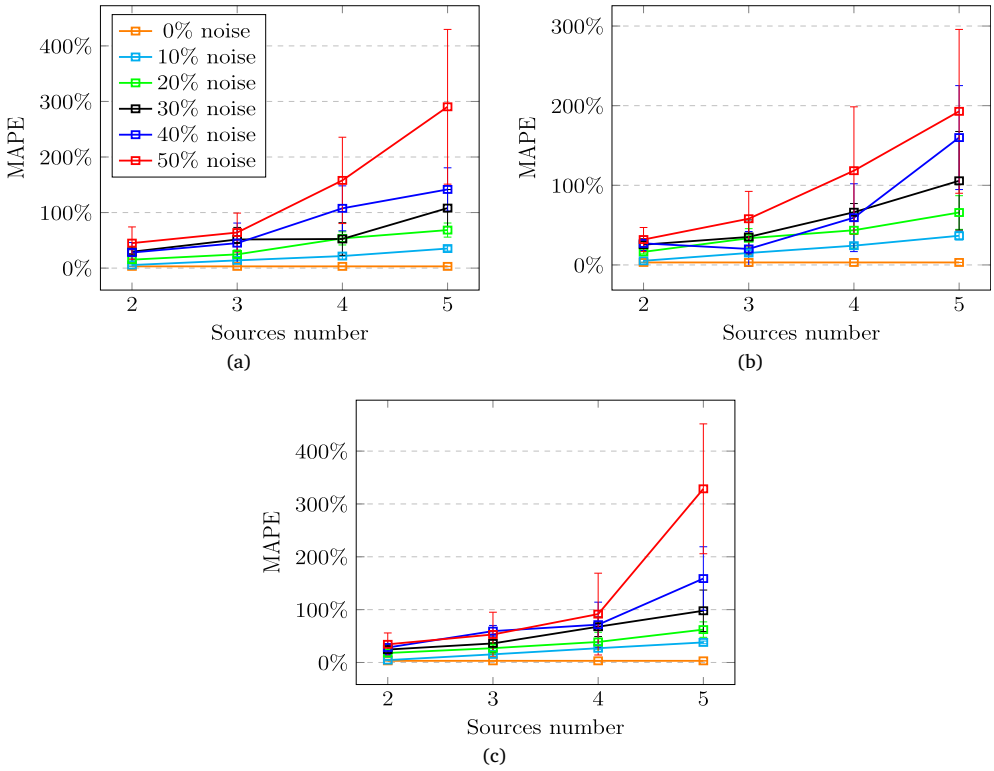


Fig. 8. Average and standard deviation (vertical lines) of the Mean Absolute Percentage Error (a) $|S_i| = 10^5$, (b) $|S_i| = 10^6$, (c) $|S_i| = 10^7$.

Table 3
Average Kendall τ and RBO scores (standard deviations in parentheses).

	$\|S_i\| = 10^5$		$\|S_i\| = 10^6$		$\|S_i\| = 10^7$	
k	Kendall τ	RBO	Kendall τ	RBO	Kendall τ	RBO
2	0.85 (0.104)	0.90 (0.087)	0.93 (0.057)	0.92 (0.065)	0.89 (0.064)	0.92 (0.073)
3	0.92 (0.098)	0.95 (0.058)	0.92 (0.066)	0.92 (0.073)	0.87 (0.124)	0.90 (0.109)
4	0.92 (0.088)	0.94 (0.082)	0.94 (0.063)	0.94 (0.072)	0.97 (0.053)	0.99 (0.028)
5	0.96 (0.061)	0.96 (0.066)	0.95 (0.058)	0.98 (0.031)	0.96 (0.06)	0.95 (0.076)

to 50 percent. This will result in rankings with more widely spaced values, and thus better results. In any case, it can be concluded that the rankings generated by the approach closely resemble the actual rankings. This provides the user with reliable support for assessing the amount of information contained in the resulting combinations.

5.2. Real-world datasets

In this subsection, we evaluate the effectiveness of the join cardinality estimation and the quality of the rankings for two real-world datasets.

5.2.1. Eurostat

In this subsection, we present an evaluation of the joinability index on real-world datasets from Eurostat.[7] Two groups of data sources of varying sizes were selected, each characterized by a different indicator measured along specific levels. The first group comprises eight sources $(S_1, \ldots, S_8)$ that store data at the ⟨Country, Year⟩ levels, with sizes ranging from 360 to 3944 (and an average size of 1000.5). The second group comprises four sources $(S_9, \ldots, S_{12})$ aggregated at the ⟨Region, Year⟩ levels, featuring higher cardinalities that range from 2559 to 11 096 (6088.75 on average).

[7] https://ec.europa.eu/eurostat/data/database.

Table 4
Average MAPE (standard deviation in brackets), Kendall τ and RBO.

Dataset	k	MAPE	Kendall τ	RBO
Eurostat (country, year)	2	27.61 (29.25)	0.61	0.78
	3	38.31 (30.49)	0.57	0.74
	4	47.32 (30.62)	0.61	0.78
	5	55.75 (29.35)	0.69	0.84
Eurostat (region, year)	2	26.24 (13.82)	0.47	0.84
	3	19.94 (17.56)	0.33	0.79
	4	21.47 (0)	N/A	N/A
USITC	2	8.06 (5.30)	0.96	0.99
	3	8.39 (5.63)	0.88	0.97
	4	7.59 (4.52)	0.79	0.93

We evaluated join and joinability index for every possible combination of sources $\mathbb{S}_k$, with k from 2 to a maximum of 5. Therefore, $\sum_{k=2}^{5} \binom{8}{k} = 154$ combinations of sources $S_1 - S_8$ and $\sum_{k=2}^{4} \binom{4}{k} = 11$ combinations of sources $S_9 - S_{12}$ were considered.

Table 4 shows the accuracy of both the join cardinality estimation and the ranking in terms of MAPE, Kendall τ and RBO.

The results demonstrate good estimation accuracy, as indicated by MAPE. More importantly, the index retains its effectiveness in defining the ranking of alternatives, particularly for the top-ranked combinations as evidenced by the RBO. The lower values obtained for Kendall's τ score are due to the presence of numerous ties within each ranking.

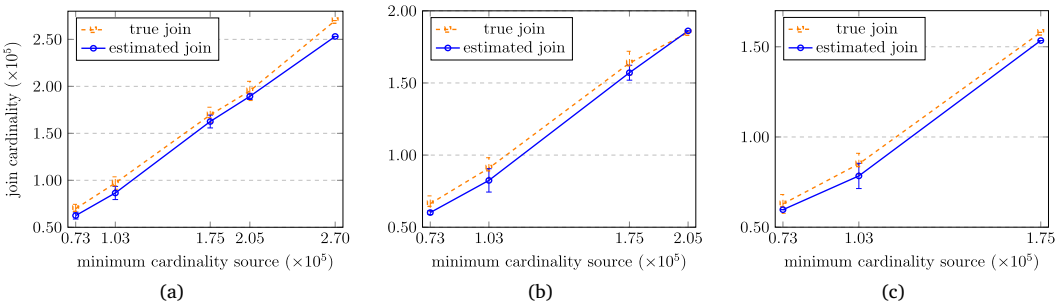


Fig. 9. Average and standard deviation (vertical lines) of the true and estimated join cardinality. Combinations with (a) 2, (b) 3 and (c) 4 sources.

Overall, the results are consistent with those obtained on the synthetic datasets.

5.2.2. USITC dataset

The USITC DataWeb[8] provides public access to the official U.S. import and export statistics compiled by the U.S. Department of Commerce. Datasets can be downloaded for various indicators (e.g., FAS value, quantity) related to domestic and foreign exports. The available dimensions include date, country, district, and commodity. Several data views were extracted through the web tools, covering different types of exports and a range of years. The experiment included six sources, each with six columns, and with the number of rows equal to 300 000, 270 000, 205 000, 173 000, 103 000, and 73 000, respectively. Candidate solutions are generated by combining 2, 3, or 4 sources, resulting in 15, 20 and 15 solutions, respectively.

Fig. 9 shows the average and standard deviation of the true and the estimated join cardinalities. These are plotted against the size of the smallest source, a factor that directly affects the precision of the estimate returned by Algorithm 2. It can be observed that the estimate obtained using the joinability index closely follows the true cardinality of the join. As shown in Table 4, the considered ranking evaluation metrics returns high values. This is especially evident for RBO, confirming the effectiveness of the proposed methodology in identifying source combinations that yield the join with the highest cardinality.

5.3. Case study: evaluation

In this subsection, we present an evaluation of the developed framework, focusing on the case study introduced earlier. As previously mentioned, the data sources from the two organizations overlap, making it particularly interesting to analyze the correlation between AQI and VAT. This requires identifying data sources containing these two indicators, measured by city and day. The input query is defined as $Q = \langle \{AQI, VAT\}, \{CITY, DAY\} \rangle$.

From the case study (see Section 1), the sets of sources including indicators AQI and VAT are, respectively, $S_{AQI} = \{S_{cs}, S_{nw}, S_{cn}, S_{ne}\}$ and $S_{VAT} = \{S_{tc}, S_{tn}\}$. Algorithm 1 retrieves $S_Q = \{\mathbb{S}_1, \ldots, \mathbb{S}_8\}$ where each $\mathbb{S}_i = \{S_j, S_k\}$ is a combination of data sources, such that $S_j \in S_{AQI}$ and $S_k \in S_{VAT}$.

Based on the joinability index computed by Algorithm 2, the user can begin to identify the most promising solutions in terms of amount of information. For instance, the user may select the top three combinations in the ranking, i.e., $\{S_{cs}, S_{tc}\}$, $\{S_{nw}, S_{tn}\}$, and $\{S_{cn}, S_{tc}\}$ (Fig. 5). These three combinations also rank first when considering the actual join.

However, without additional support, the user does not know the informative content of the solutions in advance. For example, combinations with the highest estimated cardinalities might not align with the user's interests. From an exploratory perspective, the approximation of data distribution of the join provided by Algorithm 3 would enable users to better assess the information potential of the combination. Consequently, they could not only prioritize which results to target for real join, but also combine different solutions to enrich the final result.

As an example, let us assume that the user is interested in the northern regions of Italy. Figs. 10, 11, and 12 show, for both geographical and temporal dimensions, the actual and approximate join distributions for $\{S_{cs}, S_{tc}\}$, $\{S_{nw}, S_{tn}\}$, and $\{S_{cn}, S_{tc}\}$ data combinations. Although $\{S_{cs}, S_{tc}\}$ is the solution with the highest estimated cardinality, Fig. 10 clearly indicates that the resulting join does not contain any northern Italian regions. Consequently, the user is likely to discard this solution. In contrast, the estimated data distribution of $\{S_{nw}, S_{tn}\}$ (Fig. 11) only concerns northern regions, and since it is ranked second in the estimated ranking, the user may regard it as a promising solution. The third combination in the estimated ranking, $\{S_{cn}, S_{tc}\}$, contains substantial data on the Emilia-Romagna region (Fig. 12), making it potentially interesting. Additionally, the user might also consider to combine different solutions in order to enrich the result or to compensate for missing values. For example, temporal data distribution of $\{S_{nw}, S_{tn}\}$ shows less data on January, April, May and September, while that of $\{S_{cn}, S_{tc}\}$ appears to have more data on these months. Combinations $\{S_{nw}, S_{tn}\}$ and $\{S_{cn}, S_{tc}\}$ could be unified to fill these gaps and to enrich the final results.

5.4. Discussion

Estimating joinability across multiple multidimensional data sources inevitably involves trade-offs between computational efficiency, estimation accuracy, and practical usability. The approach addresses these challenges by leveraging fixed-size MinHash signatures, ensuring that computation time remains constant regardless of dataset size and making it scalable to large and complex datasets. The use of LSH Ensemble makes the method particularly efficient for larger tables, offering a computationally convenient alternative to directly performing full joins on large datasets.

Nonetheless, certain limitations emerge as the number of data sources increases or as data noise intensifies. Under these conditions, the accuracy of the joinability index tends to decrease, as reflected in a higher MAPE. These deviations originate from both the intrinsic approximation error of the LSH Ensemble and the discretization parameter (λ) in Algorithm 2, which governs the resolution of threshold selection. Despite these sources of error, the method is still able to effectively rank candidate solutions, in particular for the top position (RBO ranges from 0.74 to 0.99 for real-world experiments). Ranking performance is strongest when solutions have well-separated estimated join scores (synthetic and USITC) and weaker when estimates are very similar (e.g., in the Eurostat dataset). In practice, solutions with similar join estimates are often functionally interchangeable, mitigating the practical impact of reduced ranking precision. Users can further exploit profiles or metadata to inform final selection among closely ranked

[8] https://dataweb.usitc.gov/.

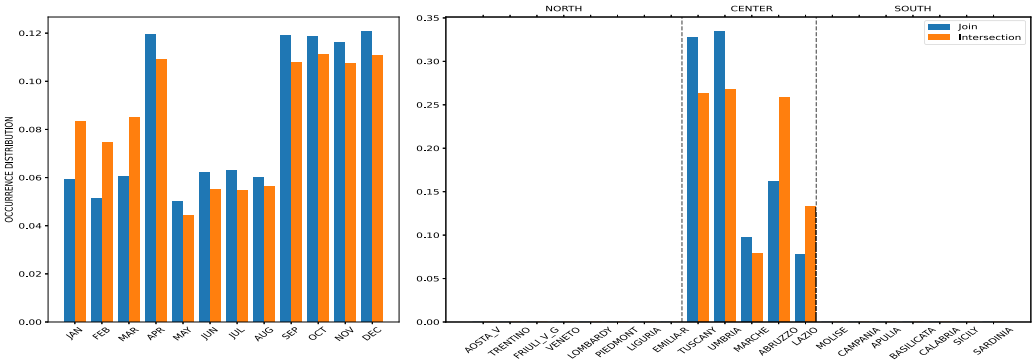


Fig. 10. Join (blue bars) and intersection (orange bars) distributions for $\{S_{cs}, S_{tc}\}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

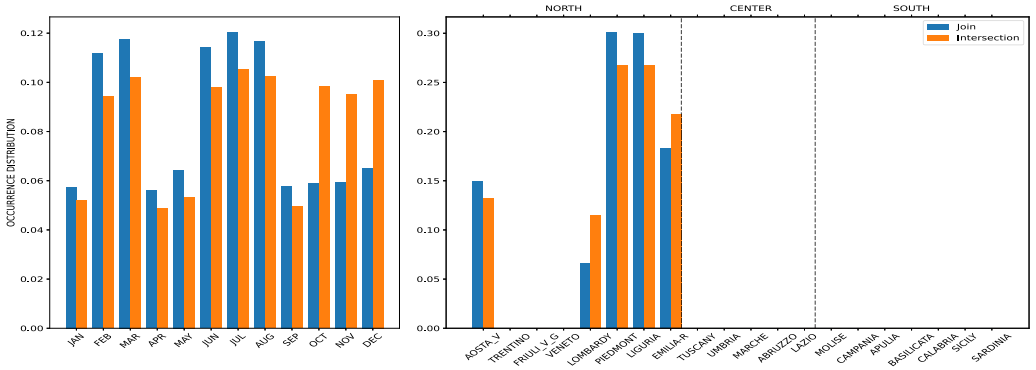


Fig. 11. Join (blue bars) and intersection (orange bars) distributions for $\{S_{nw}, S_{in}\}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

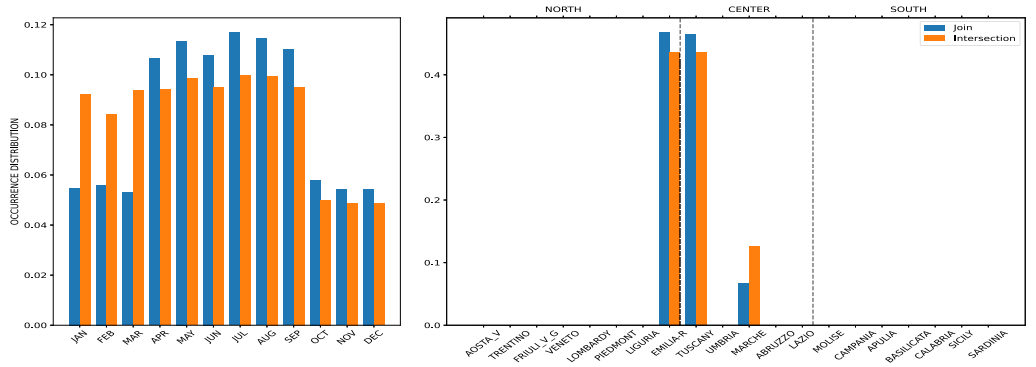


Fig. 12. Join (blue bars) and intersection (orange bars) distributions for $\{S_{cn}, S_{tc}\}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

alternatives. Recent approaches on table integration with noisy data (e.g., abbreviations, typos, and synonyms) can be exploited to address these limitations. While our approach focuses on large-scale dataset discovery, deliberately trading precision for sub-linear performance, fuzzy methods such as [31] resolve data heterogeneity by utilizing LLM embeddings and bipartite matching to reconcile variant cell values into single representative tokens. Such solutions could be paired to our approach in different ways: (i) at source ingestion time, to clean data sources, thereby reducing noise in the data and preventing downstream estimation errors from the start, and (2) after the ranking, to improve the joinability estimation for the top-k solutions.

A final consideration regards structural stability. Although dimensional schemas of data sources are generally stable over time, schema drifts require updating the metadata layer and recomputing the associated metadata and profiles. In practice, such modifications can be regarded as the introduction of a new source into the Data Lake.

6. Conclusion

This research addresses a critical challenge in join discovery within environments characterized by vast, heterogeneous, and often noisy data sources representing analytic cubes. By leveraging profiles of data sources, which represent the value distribution, and metadata at both the schema and content levels, this method enables a more expressive and efficient exploration of joinable analytic cubes. This eliminates the need for full join executions during discovery, significantly reducing computational costs and improving the overall user experience. One of the major contributions of this work is its emphasis on explainable analytics. Users are not only provided with a list of

joinable datasets but also receive detailed explanations and rankings of the alternatives based on estimated joinability and estimated profiles. This helps increasing both precision in discovery and enhanced result interpretability, allowing analysts to make informed decisions. The experimental validation demonstrates that this method is efficient, scalable, and effectively supports decision-making in large-scale data environments.

The approach opens several promising avenues for future investigations. Profiles can be exploited to evaluate content-based similarity among data sources, to provide support for related issues, such as join discovery based on source content, or topic-guided exploration in which the search can be based also on value distribution and similarity among sources besides traditional metadata. Other directions of research will be aimed to further automate, personalize, and streamline the join discovery process, e.g. through Machine Learning and Large Language Models (LLM). Machine Learning models, trained on historical join discovery attempts, metadata similarities, and data distribution, can predict and recommend the most relevant data sources for a given query. LLMs and Retrieval-Augmented Generation approaches can enable natural language exploration by translating user requests into valid queries, expanding semantic query with related concepts, and summarizing discovery output to further enhance explanation, metadata interpretation, and automated documentation.

CRediT authorship contribution statement

Claudia Diamantini: Writing – original draft, Methodology, Investigation. **Alessandro Mele:** Writing – original draft, Visualization, Software. **Domenico Potena:** Writing – review & editing, Methodology, Investigation. **Cristina Rossetti:** Writing – original draft, Software. **Emanuele Storti:** Writing – original draft, Methodology, Investigation.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work, the author(s) used Gemini and ChatGPT to check grammar, typos and style. The author(s) reviewed and edited the output as needed and take full responsibility for the content of the published article.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Cristina Rossetti reports financial support was provided by Ministero dell'Università e della Ricerca (Italy). If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

Cristina Rossetti has received funding from the MUR – DM 118/2023 as part of the project PNRR-NGEU.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.is.2026.102791.

Data availability

Data will be made available on request.

References

[1] E. Zhu, F. Nargesian, K.Q. Pu, R.J. Miller, LSH ensemble: Internet-scale domain search, Proc. VLDB Endow. 9 (12) (2016) 1185–1196.
[2] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, Michael Stonebraker, Aurum: A data discovery system, in: 2018 IEEE 34th International Conference on Data Engineering, ICDE, 2018, pp. 1001–1012.
[3] Yuyang Dong, Kunihiro Takeoka, Chuan Xiao, Masafumi Oyamada, Efficient joinable table discovery in data lakes: A high-dimensional similarity-based approach, in: 2021 IEEE 37th International Conference on Data Engineering, ICDE, 2021, pp. 456–467.
[4] Erkang Zhu, Dong Deng, Fatemeh Nargesian, Renée J Miller, Josie: Overlap set similarity search for finding joinable tables in data lakes, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 847–864.
[5] Claudia Diamantini, Domenico Potena, Emanuele Storti, Analytic processing in data lakes: A semantic query-driven discovery approach, Inf. Syst. Front. (2024) 1–19.
[6] Norman W. Paton, Jiaoyan Chen, Zhenyu Wu, Dataset discovery and exploration: A survey, ACM Comput. Surv. 56 (4) (2023) 1–37.
[7] Dan Brickley, Matthew Burgess, Natasha Noy, Google dataset search: Building a search engine for datasets in an open web ecosystem, in: The World Wide Web Conference, 2019, pp. 1365–1375.
[8] Shuo Zhang, Krisztian Balog, Ad hoc table retrieval using semantic similarity, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, International World Wide Web Conferences Steering Committee, 2018, pp. 1553–1562.
[9] Alex Bogatu, Alvaro AA Fernandes, Norman W Paton, Nikolaos Konstantinou, Dataset discovery in data lakes, in: 2020 Ieee 36th International Conference on Data Engineering, ICDE, IEEE, 2020, pp. 709–720.
[10] Aécio Santos, Aline Bessa, Christopher Musco, Juliana Freire, A sketch-based index for correlated dataset search, in: 2022 IEEE 38th International Conference on Data Engineering, ICDE, IEEE, 2022, pp. 2928–2941.
[11] Sonia Castelo, Rémi Rampin, Aécio Santos, Aline Bessa, Fernando Chirigati, Juliana Freire, Auctus: A dataset search engine for data discovery and augmentation, Proc. VLDB Endow. 14 (12) (2021) 2791–2794.
[12] I. Made Putrama, Peter Martinek, Self-supervised data lakes discovery through unsupervised metadata-driven weighted similarity, Inform. Sci. 662 (2024) 120242.
[13] Xuming Hu, Chuan Lei, Xiao Qin, Asterios Katsifodimos, Christos Faloutsos, Huzefa Rangwala, PolyJoin: Semantic multi-key joinable table search in data lakes, in: Luis Chiruzzo, Alan Ritter, Lu Wang (Eds.), Findings of the Association for Computational Linguistics: NAACL 2025, 2025, pp. 384–395.
[14] Rihan Hai, Sandra Geisler, Christoph Quix, Constance: An intelligent data lake system, in: Proceedings of the 2016 International Conference on Management of Data, 2016, pp. 2097–2100.
[15] Sayed Hoseini, Ahmed Ali, Haron Shaker, Christoph Quix, SEDAR: A semantic data reservoir for heterogeneous datasets, in: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 2023, pp. 5056–5060.
[16] Ahmed Helal, Mossad Helali, Khaled Ammar, Essam Mansour, A demonstration of kglac: A data discovery and enrichment platform for data science, Proc. VLDB Endow. 14 (12) (2021) 2675–2678.
[17] Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, Paul Groth, Dataset search: A survey, VLDB J. 29 (1) (2020) 251–272.
[18] Fatemeh Nargesian, Erkang Zhu, Ken Q Pu, Renée J Miller, Table union search on open data, Proc. VLDB Endow. 11 (7) (2018) 813–825.
[19] Anshumali Shrivastava, Ping Li, Asymmetric minwise hashing for indexing binary inner products and set containment, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 981–991.
[20] Raul Castro Fernandez, Jisoo Min, Demitri Nava, Samuel Madden, Lazo: A cardinality-based method for coupled estimation of jaccard similarity and containment, in: 2019 IEEE 35th International Conference on Data Engineering, ICDE, IEEE, 2019, pp. 1190–1201.
[21] Yuxing Han, Ziniu Wu, Peizhi Wu, Rong Zhu, Jingyi Yang, Liang Wei Tan, et al., Cardinality estimation in DBMS: A comprehensive benchmark evaluation, 2021, arXiv preprint arXiv:2109.05877.
[22] P Griffiths Selinger, Morton M Astrahan, Donald D Chamberlin, Raymond A Lorie, Thomas G Price, Access path selection in a relational database management system, in: Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, 1979, pp. 23–34.
[23] Claudia Diamantini, Paolo Lo Giudice, Domenico Potena, Emanuele Storti, Domenico Ursino, An approach to extracting topic-guided views from the sources of a data lake, Inf. Syst. Front. 23 (2021) 243–262.
[24] Viktor Leis, Bernhard Radke, Andrey Gubichev, Alfons Kemper, Thomas Neumann, Cardinality estimation done right: Index-based join sampling, in: Cidr, 2017.
[25] Claudia Diamantini, Domenico Potena, Emanuele Storti, SemPI: A semantic framework for the collaborative construction and maintenance of a shared dictionary of performance indicators, Future Gener. Comput. Syst. 54 (2016) 352–365.

[26] Claudia Diamantini, Tarique Khan, Domenico Potena, Emanuele Storti, Semantic models of performance indicators: A systematic survey, ACM Comput. Surv. 57 (8) (2025).
[27] Richard Cyganiak, Dave Reynolds, The RDF data cube vocabulary, 2014, W3C Recommendation.
[28] Claudia Diamantini, Alessandro Mele, Domenico Potena, Cristina Rossetti, Emanuele Storti, A metadata model for profiling multidimensional sources in data ecosystems, 2025, arXiv preprint arXiv:2503.15951.
[29] Maurice G. Kendall, The treatment of ties in ranking problems, Biometrika 33 (3) (1945) 239–251.
[30] William Webber, Alistair Moffat, Justin Zobel, A similarity measure for indefinite rankings, ACM Trans. Inf. Syst. 28 (4) (2010).
[31] Aamod Khatiwada, Roee Shraga, Renée J. Miller, Fuzzy integration of data lake tables, in: Wolfgang Lehner, Vanessa Braganholo, Kostas Stefanidis, Zheying Zhang, Alexander Krause, João Felipe Nicolaci Pimentel (Eds.), Proceedings of the 29th International Conference on Extending Database Technology, EDBT 2026, OpenProceedings.org, Tampere, Finland, 2026, pp. 96–102.