UNIVERSITÀ DEGLI STUDI DI BRESCIA
Dipartimento di Ingegneria dell'Informazione

DOTTORATO DI RICERCA IN INGEGNERIA DELL'INFORMAZIONE
XXXIII Ciclo

A Semantics-Enabled Approach for Personalised Data Lake Exploration

Dottorando: **Massimiliano Garda**

Tutor e Relatore: **Prof. Valeria De Antonellis**

Co-tutor: **Prof. Devis Bianchini**

Coordinatore del Dottorato: **Prof. Costantino De Angelis**

Settore Scientifico Disciplinare: ING-INF/05

Abstract

Oggigiorno, la crescente disponibilità di dati raccolti negli ecosistemi sociali e produttivi di tutto il mondo ha aperto nuove sfide nella gestione dei dati, che possono essere impiegati per ottenere informazioni utili a fine strategico e direzionale. All'interno di questi contesti, per poter predisporre una infrastruttura idonea alla raccolta dei dati provenienti da diverse tipologie di sorgenti (p.e., sensoristica e Internet of Things, sistemi legacy, Open Data), è necessario tenere in considerazione il fatto che questi dati sono intrinsecamente caratterizzati da elevato volume, velocità e varietà. Queste ultime sono altresì note come le "3V" dei big data: il volume riguarda la quantità di dati generati e memorizzati, la varietà discende dalle diverse tipologie di dati (strutturati, non strutturati, semi-strutturati), mentre la velocità è legata alla capacità dei dispositivi moderni di produrre flussi di dati a frequenze sempre più elevate, che devono necessariamente essere gestiti in maniera appropriata, senza pregiudicare le successive fasi analitiche.
I big data richiedono metodi e tecniche adeguati per inferire conoscenza da una mole di dati grezzi, supportando utenti con ruoli e competenze diverse nelle loro decisioni strategiche. A questo scopo, la moltitudine di dati grezzi viene raffinata, elaborata ed aggregata da esperti analisti, modellando opportuni *indicatori*, analizzabili secondo diverse *dimensioni*. Un indicatore, rispetto a dei dati grezzi, permette di ottenere una visione di insieme del fenomeno osservato, misurando al contempo i risultati conseguiti e le performance operative di un'organizzazione.
Un contesto applicativo significativo, fortemente caratterizzato dalla presenza di big data, sono le *smart cities*. Difatti, l'esplorazione dei dati raccolti all'interno di una Smart City, contestualizzata rispetto ai ruoli e alle competenze dei cittadini, diviene uno strumento vitale per assicurare maggiore efficienza e qualità nei servizi offerti alla cittadinanza. L'*esplorazione dei dati* provenienti da sorgenti eterogenee e la successiva *personalizzazione* dell'informazione, divengono quindi due pilastri fondamentali da tenere in considerazione.
Negli ultimi decenni, i *Data Warehouse* sono stati largamente adottati per memorizzare grandi quantità di dati storici. Essi sono stati principalmente progettati per abilitare e supportare le attività di analisi, centralizzando e consolidando dati provenienti da più origini. Il punto di forza di un Data Warehouse riguarda l'organizzazione dei dati, che avviene secondo un *modello multi-dimensionale*, dove i dati aggregati sono rappresentati dagli indicatori, che possono essere esplorati secondo diverse prospettive di analisi (le dimensioni). Un modello multi-dimensionale consente sia di focalizzare l'esplorazione solamente su un sottoinsieme di dati di interesse, fronteggiando varietà e complessità del dominio analizzato, sia di differenziare i dati in base all'importanza che essi assumono per chi li esplora. Benchè i Data Warehouse abbiano dimostrato incontrovertibili attitudini nel favorire una esplorazione mirata dei

dati, sono vincolati all'impiego di rigide procedure di trasformazione ed elaborazione dei dati (i cosiddetti processi ETL, Extract-Transform-Load), che risultano poco adatte ad essere applicate in un contesto big data come le smart cities, comportando onerosi costi di trasformazione, in termini di tempo e risorse.
Per supplire a questa lacuna, i *Data Lake* sono stati recentemente proposti, assicurando maggiore flessibilità rispetto ai Data Warehouse, abbandonando la tradizionale politica ETL in favore del paradigma ELT (Extract-Load-Transform), caricando massivamente i dati nel loro formato originario, processandoli e trasformandoli solamente in caso di necessità. In quest'ottica, un ambiente Data Lake promuove strategie di integrazione dati maggiormente mirate come il *pay-as-you-go* (a consumo, trasformando ed integrando pochi dati per volta nel tempo) oppure *on-demand* (solo su specifica richiesta, quando i dati sono necessari per alimentare gli strumenti di analisi).
Tuttavia, la flessibilità dei Data Lake nel sostenere l'eterogeneità delle diverse sorgenti dati che lo compongono richiede di adottare opportuni metodi e strumenti per garantire un'esplorazione dei dati mirata, personalizzata secondo le esigenze dell'utente. L'obiettivo primario è scongiurare la formazione di un *Data Swamp* (letteralmente, una "palude di dati") ossia un Data Lake il cui contenuto risulta essere talmente disorganizzato da rendere l'estrazione di conoscenza pressochè impossibile.
Una strategia efficace per evitare l'insorgere di un Data Swamp consiste nell'arricchire il contenuto del Data Lake con dei metadati descrittivi, per agevolarne l'esplorazione, prevedendo l'adozione di architetture stratificate su diversi livelli informativi, concepite per aggiungere conoscenza ai dati contenuti nel Data Lake, favorendone la successiva esplorazione. In quest'ottica, recentemente, il connubio tra Web Semantico e Data Lake ha trovato larga adesione, attingendo agli standard consolidati del W3C ed ai modelli di rappresentazione e condivisione della conoscenza, tendendo in considerazione il significato (semantica) dei dati nel processo esplorativo. Sempre inerente all'esplorazione, la quantità dei dati contenuta in un Data Lake richiede metodi intuitivi per rappresentare le esigenze esplorative degli utenti, limitando lo sforzo cognitivo attraverso l'introduzione di elementi di personalizzazione. A questo scopo, i sistemi basati su preferenze sono stati proposti negli ultimi decenni come strumenti efficaci per descrivere i bisogni analitici degli utenti, suggerendo quali dati esplorare compatibilmente con i loro interessi, riducendo lo sforzo di ricerca e garantendo un' analisi più focalizzata. In aggiunta, un sistema basato su preferenze garantisce di classificare i risultati delle ricerche, assicurando che l'attenzione dell'utente converga verso quei dati che meglio soddisfano i suoi obiettivi esplorativi.
In questo lavoro di tesi viene proposto un modello, organizzato secondo tre livelli, per personalizzare l'esplorazione dei dati in un contesto Smart City. Ogni livello aggiunge progressivamente conoscenza ai dati contenuti in un Data Lake, per favorirne

un'esplorazione personalizzata. Nel primo livello (partendo dal basso), le sorgenti dati vengono arricchite da Modelli Semantici, atti a descrivere il contenuto delle sorgenti in maniera formale, garantendo una visione unificata dei dati attraverso l'adozione di strumenti del Web Semantico, per fronteggiare l'eterogeneità. Il Data Lake arricchito con i Modelli Semantici viene perciò definito come *Semantic Data Lake* (Capitolo 2). Nel secondo livello, i concetti e le relazioni dei Modelli Semantici sono utilizzati come punto di partenza per la creazione di indicatori e dimensioni di analisi, utilizzati dagli utenti per esplorare il contenuto del Data Lake in forma aggregata (Capitolo 3). Nel terzo ed ultimo livello, i profili utente e le preferenze espresse sugli indicatori sono impiegate per personalizzare l'esplorazione, suggerendo agli utenti indicatori compatibili con i loro interessi di ricerca, le attività che essi compiono ed il ruolo ricoperto all'interno della Smart City (Capitolo 4).
L'approccio proposto in questo lavoro di tesi è stato concepito per supportare gli esperti di dominio e gli analisti di dati nella costruzione del modello a tre livelli, assicurando agli utenti finali un'esplorazione personalizzata dei dati. Nell'approccio formalizzato, viene rimarcata l'importanza di una netta distinzione tra i ruoli e le competenze degli attori coinvolti: (i) gli esperti di dominio sono supportati nella definizione del Semantic Data Lake, attraverso un tool pensato per eseguire disambiguazione semantica ed estrazione di concetti, impiegando la conoscenza che gli esperti di dominio possiedono in merito alle sorgenti dati ed al loro contenuto; (ii) gli analisti di dati sono guidati, nella definizione di indicatori e dimensioni, da una Multi-Dimensional Ontology, che contiene i concetti basilari per la definizione di nuovi indicatori e dimensioni; (iii) gli utenti finali, hanno a disposizione un tool per l'esplorazione degli indicatori, in base al loro profilo ed alle loro preferenze, non richiedendo a priori una conoscenza dettagliata dell'architettura e dei livelli sottostanti. Il contributo scientifico principale di questo lavoro di tesi riguarda la proposta di combinare sinergicamente, ed in un'unica architettura, i tre livelli ivi presentati. Per ogni livello, verrà descritto un tool che implementa le funzionalità offerte all'interno del livello stesso, evidenziando il contributo innovativo introdotto rispetto alla letteratura esistente. L'approccio è stato validato all'interno di un contesto Smart City, date le peculiarità del dominio e l'interesse suscitato recentemente dalla comunità scientifica.

Abstract

Nowadays, the increasing availability of data in worldwide organisations has paved the way to new challenges in the management of data, which can be used to address business problems that wouldn't have been possible to tackle so far. Data is collected from many different sources, sensors, devices, IoT systems, Open Data APIs and legacy systems, and is continuously ingested into massive data stores. Collected data is characterised by high volume, velocity and variety (big data). Volume descends from the quantity of generated and stored data, massively collected by organisations from several data sources, whereas variety is inherently due to the different types of data (e.g., structured, unstructured, semi-structured). Instead, velocity is ignited by the proliferation of devices generating data at an unprecedented speed, which must be handled in a timely manner, taking into account the frequency of generation and recording.

These characteristics call for methods and techniques for extracting value and knowledge from data, enabling users with different roles and competencies to capitalise on available information. Data necessitates further elaboration by experts and specialists to obtain useful information for users, deriving proper *indicators*, deemed as renowned tools for exploring in an aggregate form data, according to different *dimensions*. Indeed, indicators are capable of carrying out a comprehensive view of the observed phenomenon, for fulfilling strategic goals.

A widespread scenario where it is worth addressing such challenges regards *smart cities*, where several applications and services designed for citizens could benefit from the exploration of data coming from heterogeneous sources, properly adapted to the specific context (citizens' roles, activities) in which the applications and services are delivered. In this scenario, main concepts are therefore *data exploration*, to extract useful insights from heterogeneous data sources, and *personalisation*, to tailor exploration facilities around the profiles of citizens who explore.

In the last decades, *Data Warehouses* infrastructures have been widely used for storing current and historical data in one single place, with the aim of analysing data fostering *multi-dimensional models*, where indicators are explored according to different analysis dimensions. Multi-dimensional models permit to focus the analysis on a subset of data, facing on the one hand the complexity and the variety of the domain and on the other hand the different importance that data has for whom explores it. Although Data Warehouses are prone to data analysis and exploration, they obey to rigid Extract-Transform-Load (ETL) processes, whereby data is transformed and then processed before being included into the Data Warehouse. Yet, in dynamic and rapidly evolving environments such as smart cities, this is unfeasible to sustain due to the disruptive nature of big data, since it may lead to a huge consumption of computational resources. Therefore, *Data Lakes* have been proposed as ground-

breaking solutions, ensuring a higher degree of flexibility with respect to Data Warehouses, capable of managing different types and formats of data sources in accordance with the Extract-Load-Transform (ELT) approach, where data is loaded "as is", and transformed only when it becomes necessary. In this respect, Data Lakes compel to consider *pay-as-you-go* (integrate data over time as deemed necessary) or *on-demand* policies (transform data only when it is required to be consumed by analytical tools).

The indisputable flexibility of Data Lakes to cope with cumbersome heterogeneity of data sources requires methods and tools to ensure effective exploration of data and personalised access to the Data Lake, in order to avoid to shift towards a *Data Swamp*, that is, a highly disorganised data repository from which knowledge extraction is more difficult and less effective. Adding proper metadata to the Data Lake, thus creating a tiered storage structure that prevents a Data Lake from turning into a Data Swamp, is a key feature that enables people to perform personalised search for data. To this purpose, the synergy between Semantic Web technologies and Data Lakes has been recently advocated. Semantic Web technologies rely upon open standards from the W3C community, and offer practical knowledge representation models, empowering exploration by exploiting the meaning (semantics) of information. Furthermore, when deploying data exploration applications, the vast amount of data retained in Data Lakes requires intuitive instruments for describing users' desires on available data, alleviating exploration difficulty by introducing personalisation aspects in the exploration. To this aim, preference-based systems have been proposed in the literature of the last decades as valuable instruments apt to avoid empty results while exploring data on the one hand, and information flooding on the other. Besides, preferences allow for ranking query results so that the user may first see the data that better meets her exploration goals and demands.

In this thesis, novel methods and tools are proposed to overcome the issues presented so far. Specifically, we propose a model articulated over multiple layers for personalising smart city data exploration. Each layer is conceived to progressively enrich the organisation of data starting from a Data Lake, in order to enable its personalised exploration. Firstly, proper Semantic Models are defined on top of smart city data sources, meant for semantically describe their content, thus ensuring a unified view of data, according to Semantic Web technologies, and overcoming heterogeneity issues. This brings to what has been denoted as *Semantic Data Lake* (Chapter 2). Secondly, concepts and relationships of the Semantic Models are used to semantically define indicators and analysis dimensions leveraged by users to explore Smart City data (Chapter 3). Lastly, users' profiles and preferences on indicators are exploited to personalise the exploration experience, providing users with indicators more related to their search interests, the activities they perform and the role they cover in the smart city (Chapter 4).

The overall approach has been conceived to support domain experts and data analysts during the construction of the multi-layered model, and end-users during personalised data exploration. The approach clearly distinguishes the involvement of all actors: (i) domain experts are supported in the definition of the Semantic Data Lake, using tools for semantic disambiguation of terms and concepts extraction, thus exploiting domain experts' knowledge about data; (ii) data analysts are assisted during the definition of indicators and analysis dimensions, through the exploitation of a Multi-Dimensional Ontology, that contains main pivotal concepts to define new indicators and dimensions; (iii) end-users have at their disposal new tools to explore indicators according to their interests, preferences and profile, without requiring any skill or competencies on the underlying layers. The main novel contribution of the approach relies on the combined engineering of such layers. For each layer, a tool that implements the layer is described and the cutting-edge features with respect to the state of the art are highlighted. The approach is validated in the Smart City scenario, given the proper characteristics of this domain and the widespread interest on it expressed by the research community in recent years.

Acknowledgements

Through these few words, I would like to thank the people who had a fundamental role in coordinating the work behind this thesis, my supervisors Valeria De Antonellis and Devis Bianchini. I thank Systematica-TEC S.r.l. and its owner, Marco Dalla Bona, for funding my Ph.D. I really appreciated that. Last but not least, my gratitude goes to my family and my friends for their constant support and encouragement in the most difficult moments of this experience.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivations

In the landscape of the economic growth of the latest years, data is becoming an intangible asset for worldwide productive and social ecosystems. The increasing availability of data has paved the way to new challenges in the management of data, which can be used to address business problems that wouldn't have been possible to tackle so far. Data is collected from many different sources, sensors, devices, IoT systems, Open Data APIs and legacy systems, and is continuously ingested into massive data stores. Collected data is characterised by high volume, velocity and variety and it is referred to as "big data". Volume descends from the quantity of generated and stored data, massively collected by organisations from several data sources, whereas variety is inherently due to the different types of data (e.g., structured, unstructured, semi-structured). Instead, velocity is ignited by the proliferation of devices generating data at an unprecedented speed, which must be handled in a timely manner, taking into account the frequency of generation and recording.
These characteristics call for methods and techniques for extracting value and knowledge from data, enabling users with different roles and competencies to capitalise on available information. Data necessitates further elaboration by experts and specialists to obtain useful information for users, deriving proper *indicators*, deemed as renowned tools for exploring in an aggregate form data, according to different *dimensions*. Indeed, indicators are capable of carrying out a comprehensive view of the observed phenomenon, for fulfilling strategic goals.
A widespread scenario where it is worth addressing such challenges regards *smart cities*, where several applications and services designed for citizens could benefit from the exploration of data coming from heterogeneous sources, properly adapted to the specific context (citizens' roles, activities) in which the applications and services

are delivered [15]. Collected data contributes to improve the key dimensions upon which a Smart City is grounded, touching different thematic areas [1]: People (easing the engagement of citizens in participatory politics), Economy (boosting productivity and favouring employment), Living (enhancing healthcare services), Mobility (to increase the resilience of transportation infrastructures) and Environment (to achieve sustainable development). In smart cities, the so-called *datafication* phenomenon [10] tends to transform many aspects of everyday life of users into data, with the aim of enhancing the quality and the efficiency of services delivered to users within the thematic areas previously cited, transforming a city into a data-driven ecosystem. In this scenario, main concepts are therefore *data exploration*, to extract useful insights from heterogeneous data sources, and *personalisation*, to tailor exploration facilities around the profiles of citizens who explore. For instance, municipality environmental engineers may monitor pollution levels indicators for planning timely corrective actions in case levels overtake warning thresholds, whereas citizens may check pollution levels to be aware of the air quality. Conversely, building administrators may be interested in monitoring indicators concerning the electrical energy efficiency of their buildings, in order to compare the performance of such buildings with others sharing the same energetic class, to consider the adoption of effective renovation plans, thus upgrading the energetic class of their buildings.

In the last decades, *Data Warehouses* infrastructures have been widely used for storing current and historical data in one single place, with the aim of analysing data fostering *multi-dimensional models*, where indicators are explored according to different analysis dimensions [23, 33]. Multi-dimensional models permit to focus the analysis on a subset of data, facing on the one hand the complexity and the variety of the domain (e.g., aggregated smart meters measures permit to assess the electrical consumption at different granularity levels like apartments, buildings and districts) and on the other hand the different importance that data has for whom explores it (e.g., environmental engineers are interested in consulting the concentration of each pollutant, whereas citizens are more likely to be interested in acquiring a general overview of pollution phenomenon). Although Data Warehouses are prone to data analysis and exploration, they obey to rigid Extract-Transform-Load (ETL) processes, whereby data is transformed and then processed before being included into the Data Warehouse. Yet, in dynamic and rapidly evolving environments such as smart cities, this is unfeasible to sustain due to the disruptive nature of big data, since it may lead to a huge consumption of computational resources.

Therefore, *Data Lakes* have been proposed as ground-breaking solutions, ensuring a higher degree of flexibility with respect to Data Warehouses. The prominent flexibility of Data Lakes descends from the capability of managing different types and formats of data sources, in accordance with the Extract-Load-Transform (ELT) approach, where data is loaded "as is", while transformation and elaboration of data is delayed

until it becomes necessary. Indeed, with respect to legacy Data Warehouses, Data Lakes promote a *schema-on-read* approach, ensuring fast data ingestion due to the fact that data should not follow any predefined internal schema (as for *schema-on-write* approaches, also referred to as *schema first, data later*). Moreover, Data Lakes compel to consider *pay-as-you-go* or *on-demand* policies, transforming data only when it is required to be consumed by analytical and exploration tools [48]. Specifically, a pay-as-you-go approach starts with very few integrated data, which is improved over time as deemed necessary, whereas an on-demand integration approach first finds data sources that contain relevant data for exploration, and then integrates them in a meaningful way.
The indisputable flexibility of Data Lakes to cope with cumbersome heterogeneity of data sources requires methods and tools to ensure effective exploration of data and personalised access to the Data Lake, in order to avoid to shift towards a *Data Swamp*, that is, a highly disorganised data repository from which knowledge extraction is more difficult and less effective. Adding proper metadata to the Data Lake, thus creating a tiered storage structure that prevents a Data Lake from turning into a Data Swamp, is a key feature that enables people to perform personalised search for data, to set up exploration tasks. To this purpose, the synergy between Semantic Web technologies and Data Lakes has been recently advocated. Semantic Web technologies rely upon open standards from the W3C community, and offer practical knowledge representation models; in this respect, data sources content can be formally represented according to semantic concepts and relationships, as a starting point to implement knowledge-based applications for data exploration purposes [63], empowering exploration by exploiting the meaning (semantics) of information.
Furthermore, when deploying data exploration applications, the vast amount of data retained in Data Lakes requires intuitive instruments for describing users' desires on available data, alleviating exploration difficulty by introducing personalisation aspects in the exploration. The underlying idea is that different users may find different things relevant to a search due to different preferences they have, which are usually affected by the context in which the user is exploring data (e.g., the exploration goals and aims of an environmental engineer are clearly distinguished from those of a building administrator). To this aim, preference-based systems have been proposed in the literature as valuable instruments apt to avoid empty results while exploring data on the one hand, and information flooding on the other. Besides, preferences allow for ranking query results so that users may first see the data that better meets their exploration goals and demands, avoiding a trial-and-error exploration strategy, which may hinder the exploration task.

In this thesis, novel methods and tools are proposed to overcome the issues presented so far. Specifically, we propose a model articulated over multiple layers for

personalising smart city data exploration. Each layer is conceived to progressively enrich the organisation of data starting from a Data Lake, in order to enable its personalised exploration. Firstly, proper Semantic Models are defined on top of smart city data sources, meant for semantically describe their content, thus ensuring a unified view of data, according to Semantic Web technologies, and overcoming heterogeneity issues. This brings to what has been denoted as *Semantic Data Lake* (Chapter 2). Secondly, concepts and relationships of the Semantic Models are used to semantically define indicators and analysis dimensions leveraged by users to explore Smart City data (Chapter 3). Lastly, users' profiles and preferences on indicators are exploited to personalise the exploration experience, providing users with indicators more related to their search interests, the activities they perform and the role they cover in the Smart City (Chapter 4).
The overall approach has been conceived to support domain experts and data analysts during the construction of the multi-layered model, and end-users during personalised data exploration. The approach clearly distinguishes the involvement of all actors: (i) domain experts are supported in the definition of the Semantic Data Lake, using tools for semantic disambiguation of terms and concepts extraction, thus exploiting domain experts' knowledge about data; (ii) data analysts are assisted during the definition of indicators and analysis dimensions, through the exploitation of a Multi-Dimensional Ontology, that contains main pivotal concepts to define new indicators and dimensions; (iii) end-users have at their disposal new tools to explore indicators according to their interests, preferences and profile, without requiring any skill or competencies on the underlying layers. The main novel contribution of the approach relies on the combined engineering of such layers. For each layer, a tool that implements the layer is described and the cutting-edge features with respect to the state of the art are highlighted. The approach is validated in the Smart City scenario, given the proper characteristics of this domain and the widespread interest on it expressed by the research community in recent years.

1.2 Contributions

This thesis presents a novel approach to personalise the exploration of Smart City data. Going into details, the contributions of the thesis are the following:

- Formalisation of a methodology for constructing a semantic overlay over a Data Lake (which is referred to as *Semantic* Data Lake), wherein data sources are not represented according to their raw attributes but through a formal representation of their meaning. For each data source, domain experts provide a semantic annotation of data sources attributes, modelling also their mutual relationships, building the Semantic Model associated with the source. Semantic

Models of data sources are in turn linked each other in the semantic overlay, to state the relation between attributes belonging to different data sources.

- Definition, on top of the Semantic Data Lake, of an ontology-based modelling framework for indicators and related dimensions. The framework relies on a reference ontology, named Multi-Dimensional Ontology (MDO), which specifies all the elements needed to describe indicators and analysis dimensions. Indicators and analysis dimensions are therefore defined by data analysts starting from base concepts and relationships contained within the Semantic Models, according to the MDO. Indicators and dimensions form the Smart City Exploration Graph (SCEG). In the SCEG, the definition of indicators is enriched with knowledge related to the domain of indicators, activities performed by users (influenced by one or more indicators) and users' categories, for enabling the exploration of indicators. To access data sources content and retrieve indicators values, data analysts establish proper *mappings*, that is relationships aimed at linking indicators and dimensions in the Smart City Exploration Graph with concepts and relationships in the semantic overlay.

- Formalisation of a methodology to achieve personalised exploration of Smart City indicators, leveraging users' profiles, to suggest data which is more compliant with users' interests. Specifically, profiles contain the definition of both *contexts*, reflecting users' roles and exploration goals in the Smart City environment, and *preferences*, formulated as soft constraints apt to better focus the exploration of indicators.

1.3 Approach Overview

The approach proposed in this thesis relies on a methodology that progressively refines the organisation of knowledge about Smart City starting from data sources. Figure 1.1 shows the approach overview, the involved roles and modules that implement the three phases. Starting from the bottom, the Smart City Data Lake is conceived as a set of heterogeneous data sources (e.g., IoT devices, sensors, energy providers databases) modelled, regardless their structure and content, according to: (i) *attributes* and (ii) *data sets*, representing the content of data sources expressed as a set of attributes values. Attributes in a data source undergo an annotation process, performed by the domain expert, whose aim is to derive a *Semantic Model* for the data source, representing the domain expert's knowledge about the data source. Semantic Models of data sources are combined in the *semantic overlay*, which contains linkage relationships between them, to capture relations between attributes belonging to different data sources. Domain experts are supported in the aforementioned tasks

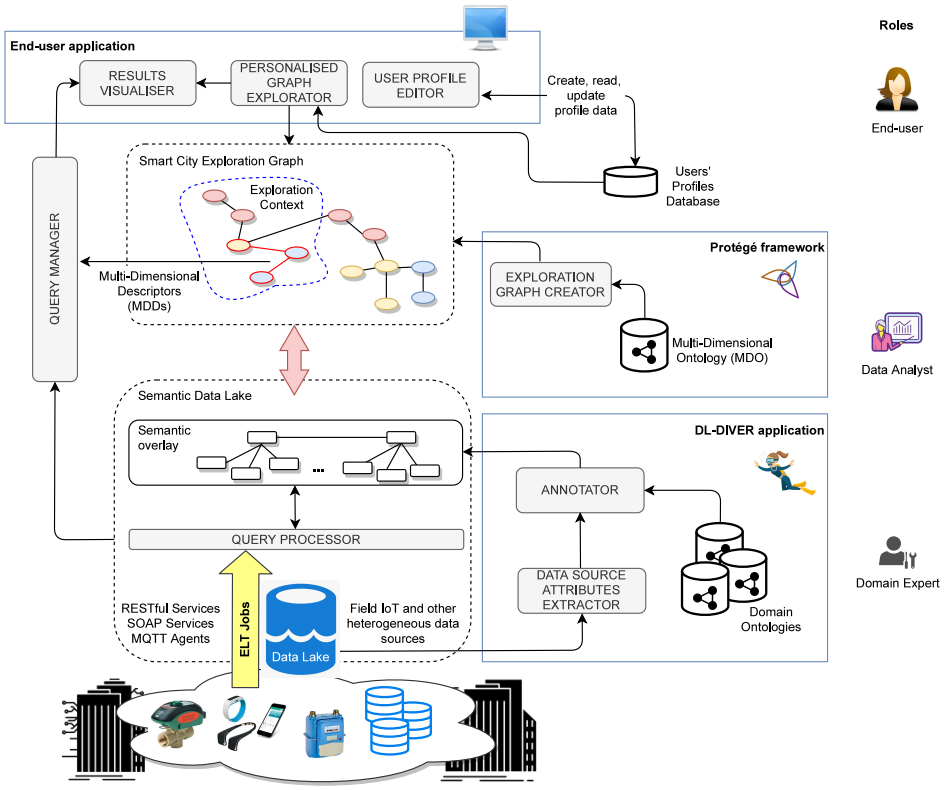


Figure 1.1: Approach overview.

by a Web-based tool, DL-DIVER (acronym for Data Lake Diver), implementing the modules `DATA SOURCE ATTRIBUTES EXTRACTOR` and `ANNOTATOR`. In particular, the `DATA SOURCE ATTRIBUTES EXTRACTOR` module connects to a data source and extracts its attributes and sample data sets instances, whereas the `ANNOTATOR` module curates the lexical enrichment of attributes and their semantic annotation. As a result, the Data Lake equipped with the semantic overlay is referred to as *Semantic* Data Lake. On top of Semantic Models, data analysts design proper *indicators*, as they are renowned tools for providing meaningful data aggregation and exploration, according to different perspectives (*dimensions*). To model indicators, data analysts are supported by the *Multi-Dimensional Ontology* (MDO), which semantically specifies all the elements needed to describe indicators and analysis dimensions. The semantic representation of indicators is further enriched by associating indicators with their domains (e.g., environment, health) and activities (e.g., air pollution monitoring) that are influenced by the knowledge provided by the indicators, which can be performed by specific users' categories within the Smart City (e.g., citizens, environmental engineers). Enriched indicators definition is contained within the *Smart City*

Exploration Graph. Data analysts are supported in this task by the Protégé tool, accomplishing the `EXPLORATION GRAPH CREATOR` module functionalities. The tool is an ontology management editor, allowing to import the MDO and let data analysts model indicators starting from the base concepts and relationships contained in the MDO, thus extending the Smart City Exploration Graph.
Portions of the Smart City Exploration Graph can be dynamically accessed by different categories of users (determining different *Exploration Contexts*), based on users' profiles, thus supporting personalised and interactive exploration of Smart City indicators. To this aim, there are several modules fulfilling different tasks. The `USER PROFILE EDITOR` module allows the registration of users to the platform and the creation of their profile. The `PERSONALISED GRAPH EXPLORATOR` module enables to express preferences on available indicators, in order for users to be suggested with indicators more compliant with their exploration goals. Through the `RESULTS VISUALISER` module, users learn about indicators descriptions in the Smart City Exploration Graph and see actual indicators values.
The exploration of actual values of indicators and related dimensions, selected by a user from the Smart City Exploration Graph, is assured by the presence of *mappings*. Mappings are specific types of relationships meant for query answering purposes, which are established by the data analyst from the concepts of the Smart City Exploration Graph to the concepts of the Semantic Models in the semantic overlay. Exploiting the definition of indicators in the Smart City Exploration Graph, the mappings and the semantic overlay, the `QUERY PROCESSOR` module enables to access Data Lake sources content, retrieve and aggregate data to provide actual values of indicators.

1.4 Thesis outline

This thesis is organised over three chapters, each of them devoted to discuss in details a specific aspect of the approach, summarised in the previous section: (i) the creation of a semantic overlay on top of a Data Lake (Chapter 2); (ii) the creation of a framework for modelling and exploring indicators (Chapter 3); (iii) the adoption of users' contexts and preferences to enable a personalised exploration of Data Lake content (Chapter 4). The approach proposed in this thesis is also referred to as *three-layered*, where each layer (described by a chapter) progressively refines and extracts knowledge from Smart City data. For each chapter, the related literature and the novel contributions will be discussed, along with the partial implementation and experiments, apt to validate the layer.
The remainder of the thesis is organised as follows:

- Chapter 2 explains how to model the Semantic Data Lake, focusing on the

representation of data sources in the semantic overlay.

- Chapter 3 formalises the methodology proposed to support data analysts for the creation of indicators and defines the Smart City Exploration Graph as the entry point for users to explore Smart City data. It also defines the exploration process, and how to query the Data Lake in order to retrieve indicators values.
- Chapter 4 defines the methodology and the procedure for the exploration of indicators, grounded on users' contexts and preferences.
- Chapter 5 closes the thesis, sketching future research directions.

Chapter 2

Semantic Data Lake

2.1 Introduction

This chapter presents the creation of the Semantic Data Lake. In the first part of the chapter, we formalise a Data Lake model for representing data sources, regardless of their structure and content. Then, we describe how domain experts construct a semantic overlay on top of the Data Lake, composed of Semantic Models representing the domain experts' view of Data Lake sources, to ensure interoperability and improve data access. Lastly, we discuss about the implementation and the evaluation of the effectiveness of a Web-based tool (called DL-DIVER, acronym for Data Lake Diver) aimed at supporting domain experts in the creation of the semantic overlay through the lexical enrichment of data sources attributes, their semantic annotation and, finally, the refinement and the creation of semantic relationships.

2.1.1 Related Work

In the literature, Data Lake research efforts focus mainly on two aspects, inherently devoted to address heterogeneity issues: (i) from a higher viewpoint, through the formalisation of models to represent Data Lake sources, in order to abstract from their type and content; (ii) practically, proposing the development of services and platforms to manage Data Lake content. The table in Figure 2.1 summarises the surveyed approaches on (Semantic) Data Lakes.
Concerning the former aspect, in the KAYAK framework [39] a Data Lake is conceived as a collection of datasets composed of attributes, data objects and profiles. Therein, the aim of such modelling strategy is to assess the affinity between datasets, in order to optimise the pipelines of data preparation. Authors in [41] sketch a model for Semantic Data Lake sources, consisting of attributes and entities, gathering similar attributes. Conversely, the work presented in [19] models each data source in the Data Lake

Approach	Data Model	Type of Application Environment	Lexical Enrichment	Semantic Annotation	Semantic Models	Support for domain experts
KAYAK [39]	Datasets and attributes	Data Lake	✗	✗	✗	✓(assessment of joinalibity between datasets)
Squerall [41]	Attributes	Semantic Data Lake	✗	✓	✗	~ (partial support during annotation)
Diamantini et al. [19]	Keywords extracted from data sources	Data Lake	✓	✓	✗	✗
Ben Hamadou et al. [29]	Dataspace	Polystore	✗	✗	✗	✗
Pomp et al. [53]	N/A	Semantic Data Lake	✗	✓	✓	✗
Constance [27]	N/A	Semantic Data Lake	✗	✓	✗	✗
CoreKG [7]	N/A	Data Lake	✓	✗	✗	~ (services and APIs)
DATAMARAN [22]	Record templates	Data Lake	✗	✗	✗	N/A (automatic approach)
Martinez-Prieto et al. [43]	N/A	Hadoop-based Data Lake	✗	✗	✗	✗
Munshi et al. [47]	N/A	Hadoop-based Data Lake	✗	✗	✗	✗(limited to analysis tasks)
Azure Data Lake Store [57]	File-oriented	Data Lake (File service)	✗	✗	✗	N/A
Rangarajan et al. [58]	N/A	Hadoop-based Data Lake	✗	✗	✗	~ (in Security and Analytics layers)
Ours	Data sets defined over attributes	Semantic Data Lake	✓	✓	✓	✓

Figure 2.1: Summary of approaches on (Semantic) Data Lakes.

as a network-based structure (called *Ego Network*) grounded on metadata extracted from the data source. The content of the data source is represented through a set of keywords; nodes in the network represent the keywords whereas arcs between nodes are added according to existing similarities (both lexical and string). The proposal of [29] exploits a *dataspace* as a solution to overtake problems in data integration tasks. In such work, the dataspace enables to answer multi-dimensional queries over a polystore, leveraging the relationships between attributes in order to assemble a query graph, used to drive the query answering process. The DATAMARAN approach [22] aims at automatically extracting a structured representation from log datasets, inferring record templates (regular expressions) from instantiated record and identifying noise data. It is conceived as an automatic approach, enforced with metrics to derive the structure template with best coverage over available logs. A domain-specific conceptual data model for a Data Lake is proposed in [43], whose role is to enable an overview of the business information of the ATM (Air Traffic Management), thus being more oriented towards exploration purposes rather than representing Data Lake content.
Regarding the development of services and platforms for managing Data Lake content, most of the existing approaches focus intensively on data source preparation [39] and on offering proper services for data curation, security and provenance [7]. The

work in [47] presents a smart grid Big Data ecosystem based on the state-of-the-art Lambda architecture, apt to handle various types of smart grid data, and upon which data mining applications can be set up. A commercial implementation of Data Lake services is the Azure Data Lake Store [57], deemed as a scalable and secure file system, designed and optimised for a broad spectrum of Big Data analysis tasks. Grounded on the healthcare domain, the proposal of [58] fosters a Data Lake architecture based on the Hadoop Distributed File System (HDFS), where a dedicated Analytics Layer implements both a clustering algorithm to find the patient clusters with similar health conditions and support vector machine to find the most successful healthcare recommendations for the each cluster. The system proposed in [27] is aimed at discovering, extracting, and summarising the structural metadata from the data sources, performing semantic annotation, to avoid ambiguities, albeit no detail regarding the latter aspect is supplied. Similarly to the previous work, authors in [53] attempt to outline best practices aimed at deriving a semantic abstraction for a data source, but the task is manually performed by domain experts, without relying upon an assisted procedure. The proposal of [41] moves a step forward, drawing concepts and relationships for semantic annotation from a set of foundation ontologies, but also in this case domain experts are not supported throughout the annotation process.

2.1.2 Novel Contributions

In this chapter we define the modelling fundamentals at the basis of the construction of a semantic overlay upon a Data Lake (thus referred to as Semantic Data Lake), through a stepwise procedure supervised by the domain expert, who possesses the knowledge to annotate Data Lake sources.
The role and benefits of semantics to provide a unified view of heterogeneous data sources have been extensively examined in the database area during the last decades [21]. Semantic integration has been a major research area for the database community. Methods and tools for the definition of global views of information from multiple and heterogeneous sources, to support querying facilities and navigation among heterogeneous sources have been developed. In particular, a query over the global unified view is translated into queries over the single data sources schemas, leveraging a set of semantic correspondences between the global schema and the local schemas [13, 14].
In our approach, we formalise a model for the Data Lake, which takes into account both attributes of data sources and their content. In the excerpt of the literature analysed in the previous section, Data Lake models (where present) focus mainly on the representation of data sources attributes. Only the KAYAK approach [39] attempts to deal with both data sources attributes and their content, but the modelling strategy is only a preliminary sketch.

Given the Data Lake model, to create the semantic overlay, data sources attributes are lexically enriched, relying upon both a lexical database and abbreviations dictionary. Then, each lexically enriched attribute undergoes an annotation process, whereby concepts used for annotation are drawn from a set of domain ontologies. Semantically annotated attributes (and relationships connecting them, in turn derived from the aforementioned set of domain ontologies) are the output of the annotation process, and they are referred to as the *Semantic Model* for the source. Hence, Semantic Models of data sources compose the semantic overlay upon the Data Lake. The Data Lake proposal in [19] moves towards a semantics-enabled approach, considering lexical enrichment and semantic annotation of attributes, but the assumption behind the annotation process is that there the knowledge graph for achieving annotation is chosen upfront, without giving the possibility of searching within a broader set of domain ontologies as conceived in our proposal.
With respect to the existing approaches grounded on Semantic Data Lakes [41, 53], our approach leverages a semi-automatic semantic annotation and offers a support for domain experts, which rely on a Web-based tool for managing the semantic overlay. Indeed, in the aforementioned works, domain experts are not supported in managing the semantic overlay and they are forced to rely upon their expertise only. In the scope of the related literature, our proposal shares some issues with the dataspace-based approach in [29], where data sources in an heterogeneous context are abstracted considering only their attributes and mutual relationships to ease the subsequent query process, without applying an upfront transformation on data sources content. However, with respect to the former research effort, we move forward by introducing lexical enrichment for data sources attributes and providing a semantic support. On the one hand, lexical enrichment of data sources helps reducing the terminological distance from data source attributes names (often expressed through abbreviations or acronyms) and the names of concepts used for annotation. On the other hand, semantic annotation through domain ontologies ensures a formal representation of the meaning associated with data source attributes, with the goal of favouring sharing, reusability and interoperability, which are amongst the goals of the Semantic Web initiative. Indeed, according to [59, 62], assuring semantic enrichment in a Data Lake plays a key role in data exploitation, as data sources content is summarised so that they are more understandable to the users.
The content of this chapter extends the preliminary overview we gave in our previous works [6, 8], thus detailing Semantic Data Lake modelling and describing the implemented tool for supporting domain experts.

Data source type	Basic element	Attributes in the schema of a data set as a subset of
CSV	-	Columns of the CSV
JSON	-	Flat representation of JSON (see example in Figure 2.3)
Relational database	Table	Schema of each table
Document-oriented store	Collection (set of JSON douments)	Flat representation of each JSON in the collection

Figure 2.2: Data sources types and equivalence between their basic elements and the respective representation as data sets.

2.2 Data Lake model

In our approach, we model a data lake DL as a set of data sources, formalised as follows.

Definition 1 (Data Source) *A data source $\mathcal{S}_i \in DL$ is represented as a triple $\langle \mathcal{A}_i, \mathcal{DS}_i, \mathcal{M}_i \rangle$, where: (i) $\mathcal{A}_i$ is a set of attributes, (ii) $\mathcal{DS}_i$ is a set of data sets, representing the content of the data source regardless its nature (i.e., structured, semi-structured, unstructured) and (iii) $\mathcal{M}_i$ is a set of attribute-value pairs containing metadata apt to access the source (e.g., username, password, URL, name and type of the source), constituting the access metadata for the source.*

Definition 2 (Data Set) *A data set $ds_i^j \in \mathcal{DS}_i$ is defined over a subset of attributes $\mathcal{A}_i^j \subseteq \mathcal{A}_i$ of the source $\mathcal{S}_i$. For ds_i^j, attributes $a_1, a_2, \ldots, a_j$ constitute the schema of ds_i^j, whereas the values associated with such attributes determine the instances of ds_i^j, the latter denoted as $I(ds_i^j)$.*

Figure 2.2 summarises how basic storage objects of main data source types (either relational, NoSQL[1] or file formats like CSV and JSON) can be abstracted through data sets.

Example 1 *Figure 2.3 illustrates three examples of data sources and their representation as attributes and data sets. Going into details, data source $\mathcal{S}_1$ is a Comma Separated Values (CSV) file, containing carbon dioxide concentration levels sensed by environmental sensors. Each concentration measure (`value`) has an identifier (`mID`), a timestamp denoting the instant when the measure has been captured (`ts`),*

[1] `http://nosql-database.org/`

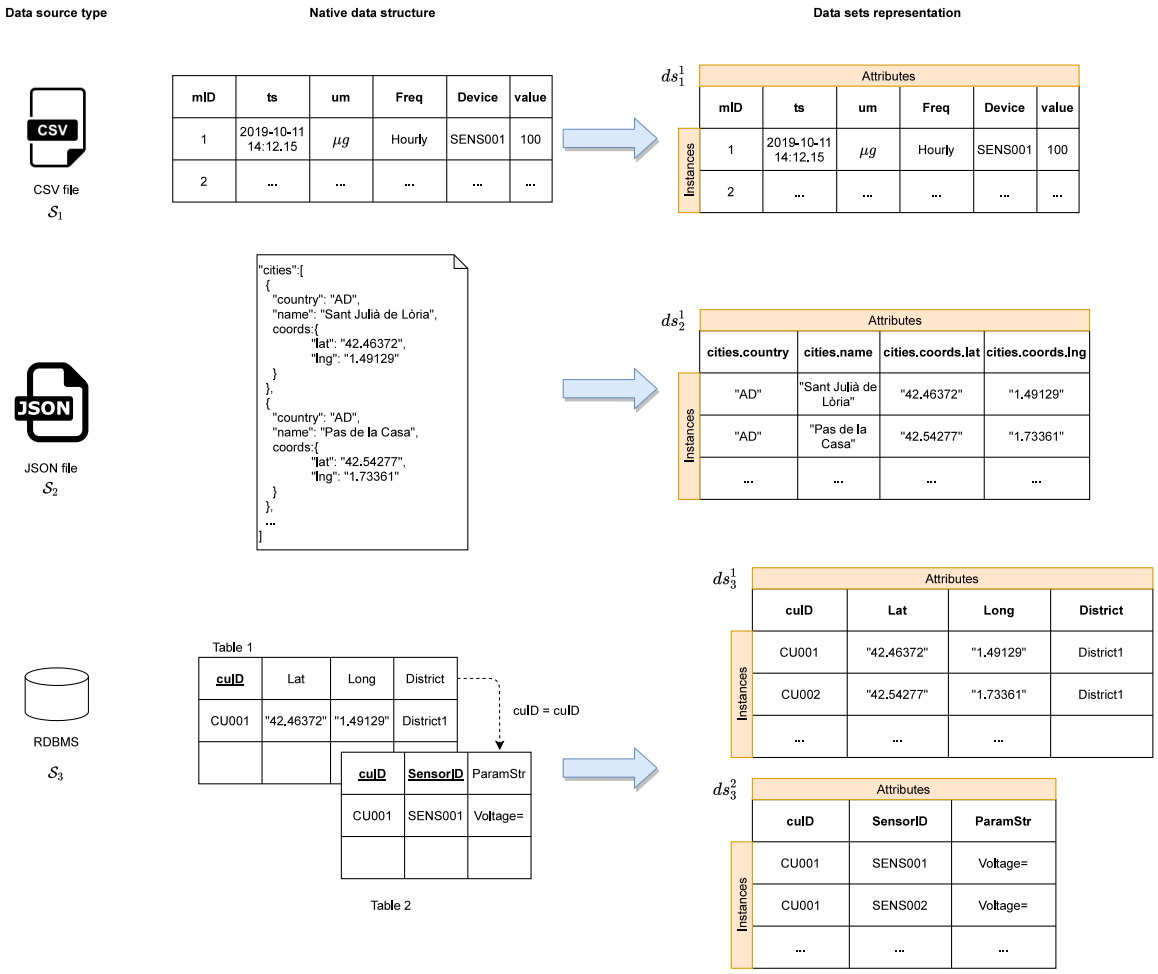


Figure 2.3: Example of data sources and related representation as attributes and data sets (Example 1).

the unit of measure `um` *(in the case of pollutant concentration, µg or ppm), the serial number of the sensor (*`Device`*) and the temporal sampling frequency (*`Freq`*) of the measure (whether hourly, each minute, second etc.) Data source* $\mathcal{S}_2$*, instead, is a JSON file containing geospatial information regarding cities. Each city belongs to a country (represented by the* `country` *identifier) and has name* `name`*. Moreover, for each city, location is expressed through latitude and longitude coordinates (*`lat` *and* `lng` *attributes). Lastly, data source* $\mathcal{S}_3$ *is a relational database containing two tables, retaining information regarding control units and the sensors they host. Specifically, Table 1 records the spatial location of control units. It contains the serial number of the control unit (*`cuID`*) and the geospatial coordinates (expressed in terms of latitude* `Lat` *and longitude* `Long`*). Conversely, Table 2 retains information about sensors,*

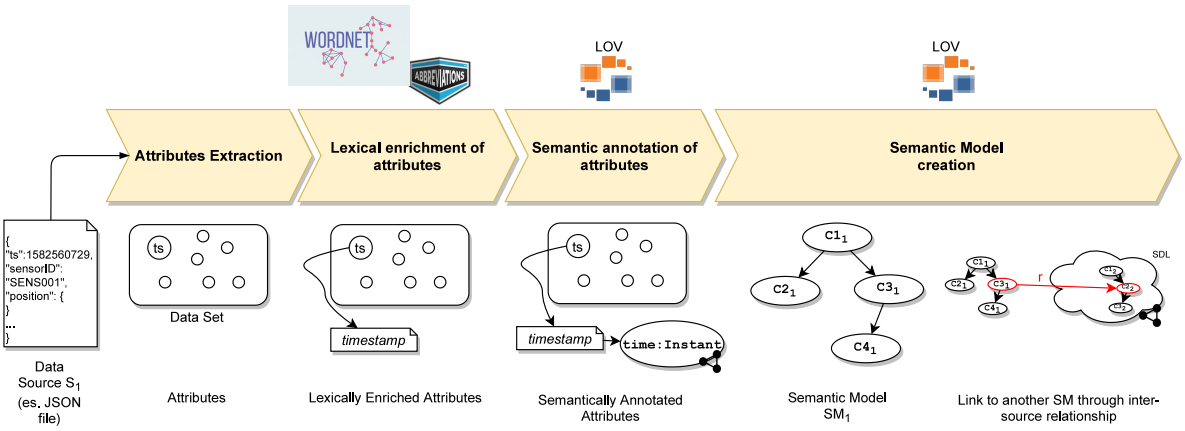


Figure 2.4: Summary of the steps required for the creation of the Semantic Model for a data source.

that is their serial number (`sens`*), the serial number of the control unit hosting the sensor (*`cu`*) and a configuration string (*`paramsString`*), related to sensors working parameters (e.g., the operating mode, status).*

2.3 Semantic overlay

Semantic Models represent the domain experts' view of data sources, relying upon domain ontologies with the aim of weaving a *semantic overlay* on the Data Lake, to ensure interoperability among heterogeneous sources and improve data access. In this respect, the Data Lake equipped with the semantic overlay is referred to as *Semantic Data Lake*.

Definition 3 (Semantic Data Lake) *A Semantic Data Lake is formalised as a pair* $SDL = \langle \mathcal{O}, DL \rangle$*, whereby: (i)* $\mathcal{O}$ *is the semantic overlay and (ii)* DL *is the set of Data Lake sources.*

In the following subsections, the steps of the procedure leading to the creation of a Semantic Model for a data source (illustrated in Figure 2.4) are thoroughly described, paving the way to the formal definition of the semantic overlay content.

2.3.1 Lexical enrichment of data source attributes

For an attribute $a_i^k \in \mathcal{A}_i$, a so-called *Entity Property* can be defined. Specifically, an Entity Property is a label associated with an attribute a_i^k by a domain expert, as a result of a lexical enrichment process. This task is accomplished with the support of

Attributes						
mID	**ts**	**um**	**Freq**	**Device**	**value**	
-	*Timestamp*	*Unit of measure*	*Frequency*	*Device*	*Value*	EP
1	2019-10-11 14:12.15	μg	Hourly	SENS001	100	Instances
2	...	...	...	...	...	

Figure 2.5: Example of Lexically Enriched Attributes.

two external linguistic facilities: (i) an abbreviations API[2], providing a dictionary of acronyms and their expansion, categorised by domain (e.g., medical, government) and (ii) WordNet [46], the widely adopted lexical database[3]. The two linguistic facilities are conceived as support tools to let domain experts define an Entity Property for an attribute not in a totally blind way, augmenting the degree of expressiveness and significance for the label. The output of this step are the *Lexically Enriched Attributes* (in brief, LEA), formalised as follows.

Definition 4 (Lexically Enriched Attribute) *A Lexically Enriched Attribute (LEA) is formalised as* $LEA_i^k = (a_i^k, ep_i^k)$*, where: (i)* a_i^k *is an attribute belonging to the set of attributes* $\mathcal{A}_i \in \mathcal{S}_i$ *and (ii)* ep_i^k *is the Entity Property label associated with attribute* a_i^k *through the lexical enrichment step.*

Example 2 *Figure 2.5 reports a data set with related instances for the sample Smart City data source* $\mathcal{S}_1$*, described in Example 1, containing carbon dioxide concentration levels sensed by environmental sensors. The labels representing the lexical enrichment of the attributes have been chosen by the domain expert relying on both WordNet (e.g., for* `Freq` *attribute) and the abbreviations API (e.g.,* `ts` *and* `um` *attributes). As it can be seen from the figure, LEA assignment is not mandatory (e.g., the* `mID` *attribute has no associated label).*

2.3.2 Semantic annotation of data source attributes

Once lexical enrichment has been performed as shown in the previous step, each defined Entity Property undergoes semantic annotation. Specifically, the domain expert uses the Entity Property of a_i^k to find a suitable concept (namely, *Entity*

[2] `https://www.abbreviations.com/api.php`
[3] `https://wordnet.princeton.edu/`

Concept) that describes the meaning of the attribute. Semantic annotation is based on a set of domain ontologies stored within an open access repository, LOV - Linked Open Vocabularies [68]. The repository is accessed through APIs freely available.

Example 3 *To find a suitable concept for annotating the* `ts` *attribute of the data source $\mathcal{S}_1$, the domain expert will invoke the LOV Search Term API, using the Entity Property associated with the attribute (that is, "timestamp"). The LOV Search Term API allows a domain expert to search over Linked Open Vocabularies ecosystem for a vocabulary term (class, property, datatype or instance). The API invocation syntax for the example is:* `https://lov.linkeddata.es/dataset/lov/api/v2/term/search-?q=timestamp&type=class` *where the parameter* `q` *is bound to the Entity Property and the* `type` *parameter is apt to filter the results according to their type (since the search is focused on concepts, the value is set to "class").*

In the case a proper concept cannot be found in the previous search, the domain expert can define a new concept by specialising it from an existing one (through the `rdfs:subClassOf` semantic relationship) or, in the worst case, by creating the concept from scratch based on his/her domain knowledge. The output of this step is the set of *Semantically Annotated Attributes* (in brief, SAAs) for each data source $\mathcal{S}_i$.

Definition 5 (Semantically Annotated Attribute) *A Semantically Annotated Attribute (SAA) is denoted as $SAA_i^k = (LEA_i^k, c_i^k)$, being: (i) LEA_i^k a Lexically Enriched Attribute and (ii) c_i^k the concept apt to semantically annotate the attribute $LEA_i^k.a_i^k$ belonging to the data source $\mathcal{S}_i$, exploiting its describing label $LEA_i^k.ep_i^k$.*

2.3.3 Semantic Model creation

Two concepts $SAA_i^1.c_i^1$ and $SAA_j^2.c_j^2$ (where c_i^1 and c_j^2 may be either concepts belonging to different sources or to the same source) can be connected by means of a *semantic relationship*. Concepts annotating Lexically Enriched Attributes and semantic relationships between them are the main pillars of *Semantic Models*, abstracting Data Lake sources in the semantic overlay. Given a Data Lake source $\mathcal{S}_m$, the domain expert is in charge of building the related Semantic Model $\mathcal{SM}_m$, modelled as a set of *concepts* $\mathcal{C}_m$ and *semantic relationships* $\mathcal{R}_m$. Formally, a semantic relationship r_h is defined as follows.

Definition 6 (Semantic relationship) *A semantic relationship r_h is a tuple*

$$\langle t_h, c_i^a, c_j^b, u_h, M_h \rangle$$

where: (i) t_h is the type of the relationship (either intra-source or inter-source); (ii) c_i^a and c_j^b are concepts associated with two Semantically Annotated Attributes SAA_i^a and SAA_j^b; (iii) u_h is the URI of the relationship holding as domain c_i^a and as range c_j^b; (iv) M_h is a set of attribute-value pairs containing metadata associated with r_h.

A semantic relationship r_h connecting two concepts $SAA_i^a.c_i^a$ and $SAA_j^b.c_j^b$ belonging to the same Semantic Model (i.e., $i = j$) is referred to as *intra-source*. Instead, r_h is an *inter-source* semantic relationships if c_i^a and c_j^b belong to different Semantic Models (i.e., $i \neq j$). Inter-source semantic relationships are apt to connect Semantic Models in the semantic overlay on top of the Data Lake. In this respect, the semantic overlay content is formalised as follows.

Definition 7 (Semantic overlay) *The semantic overlay $\mathcal{O}$ of the Semantic Data Lake SDL is composed of two sets, that is $\mathcal{O} = \langle MOD, IR \rangle$. $MOD = \{\mathcal{SM}_1, \mathcal{SM}_2, \ldots, \mathcal{SM}_n\}$ is the set of Semantic Models of all sources and IR is the set containing inter-source relationships connecting Semantic Models in the set MOD.*

2.3.3.1 Semantic relationships definition

Concerning intra-source semantic relationships in the Semantic Model $\mathcal{SM}_i$ of a data source $\mathcal{S}_i$, the domain expert is supported in the definition of the set $\mathcal{R}_i$. Specifically, for each concept $c_i^k \in \mathcal{C}_i$ annotating a Semantically Annotated Attribute SAA_i^k, derived as explained in Section 2.3.2, the set of domain ontologies is looked up for candidate relationships to be included in the set $\mathcal{R}_i$.
In particular, a semantic relationship $\hat{r}$ belonging to the set of domain ontologies is deemed as candidate to be included in $\mathcal{R}_i$ if either: (a) $SAA_i^k.c_i^k \subseteq dom(\hat{r})$, where $dom(\hat{r})$ denotes the domain of $\hat{r}$; (b) $SAA_i^k.c_i^k \subseteq range(\hat{r})$, where $range(\hat{r})$ denotes the range of $\hat{r}$. Then, from the candidate relationships, the domain expert may choose to keep only the ones that are effectively needed or, conversely, to add custom relationships between concepts (i.e., relationships not found in the set of domain ontologies). In this respect, the set $\mathcal{R}_i$ is composed not totally in a blind way by the domain expert, who, similarly to what happens for the annotation of data sources attributes, relies on the LOV domain ontologies repository, accessed through the same APIs as explained in Section 2.3.2.

Example 4 *Figure 2.6 illustrates the Semantic Model for the data source $\mathcal{S}_1$. Dashed lines in the figure represent the association between Lexically Enriched Attributes and concepts drawn from the set of domain ontologies. As it can be noticed from the figure, two concepts (*`PhysicalDevice` *and* `PollutantEmission`*) have been obtained as a specialisation of domain ontologies ones. Focusing on the two aforementioned concepts and supposing that the domain expert is willing to define a relationship between them, the set of domain ontologies is looked up to find candidate relationships as explained before. Candidate relationships to connect the aforementioned concepts are retrieved from the domain ontologies repository by looking up relationships having as domain/range* `sosa:Device` *and* `sosa:Observation` *(i.e., the parent concepts of* `PhysicalDevice` *and* `PollutantEmission`*, respectively). Amongst them, the*

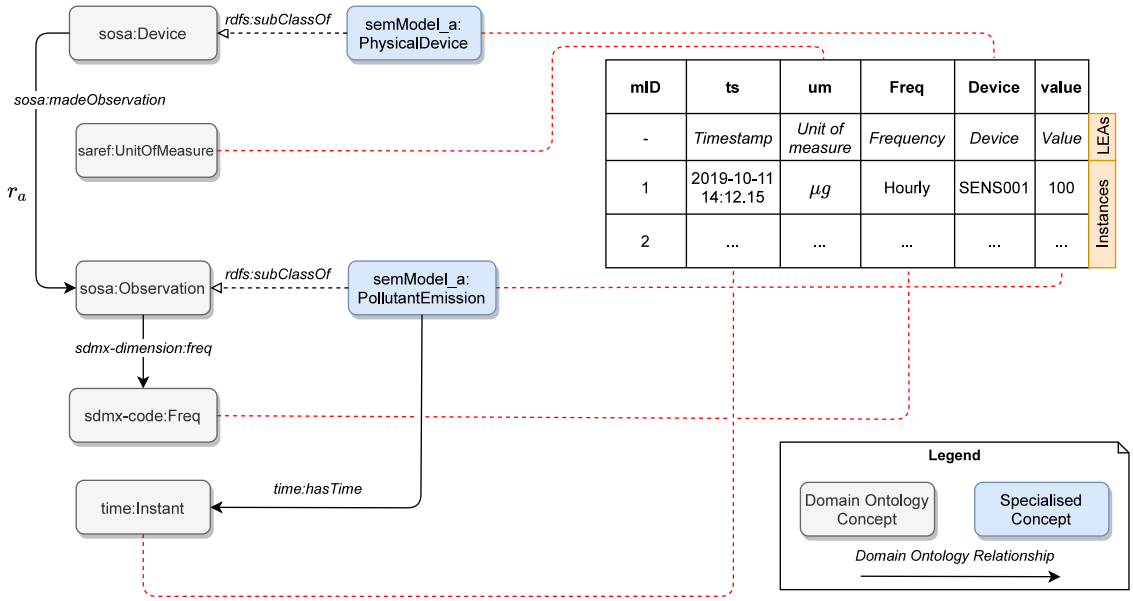


Figure 2.6: Example of Semantic Model.

semantic relationship r_a (i.e., `sosa:madeObservation`*) has been chosen by the domain expert from the set of candidate ones since $dom(r_a) \supseteq$* `PhysicalDevice` $\wedge range(r_a) \supseteq$ `PollutantEmission`*. For readability purposes, only a subset of semantic relationships have been reported in the figure.*

Noteworthy, once a Semantic Model of a data source is derived and refined, it has to be added to the semantic overlay. In this respect, the domain expert can connect the aforementioned Semantic Model with the others, by specifying inter-source relationships. Also for this case, inter-source relationships are either suggested relying upon the set of domain ontologies or manually defined by the domain expert. The automatic retrieval of such relationships is out of the scope of this thesis, as it is a consolidated topic already covered by flagship research efforts [56].

Example 5 *Figure 2.7 illustrates three sample semantic relationships r_1, r_2, r_3. Semantic relationship r_1 connects a concept belonging to $\mathcal{SM}_l$ to a concept belonging to $\mathcal{SM}_m$ (therefore its type is inter-source). Conversely, r_2 and r_3 are intra-source semantic relationships. In the example, semantic relationship r_3 has a non-empty metadata set M_3, which contains the attribute-value pair $\langle isIntraFK,$* `true`$\rangle$*. The domain expert added this additional information to underline the fact that, despite being r_3 an intra-source semantic relationship, it connects two SAAs related to two different data sets schema (e.g., descending from two tables of a relational databases). Noteworthy, the metadata set of a semantic relationship may contain information*

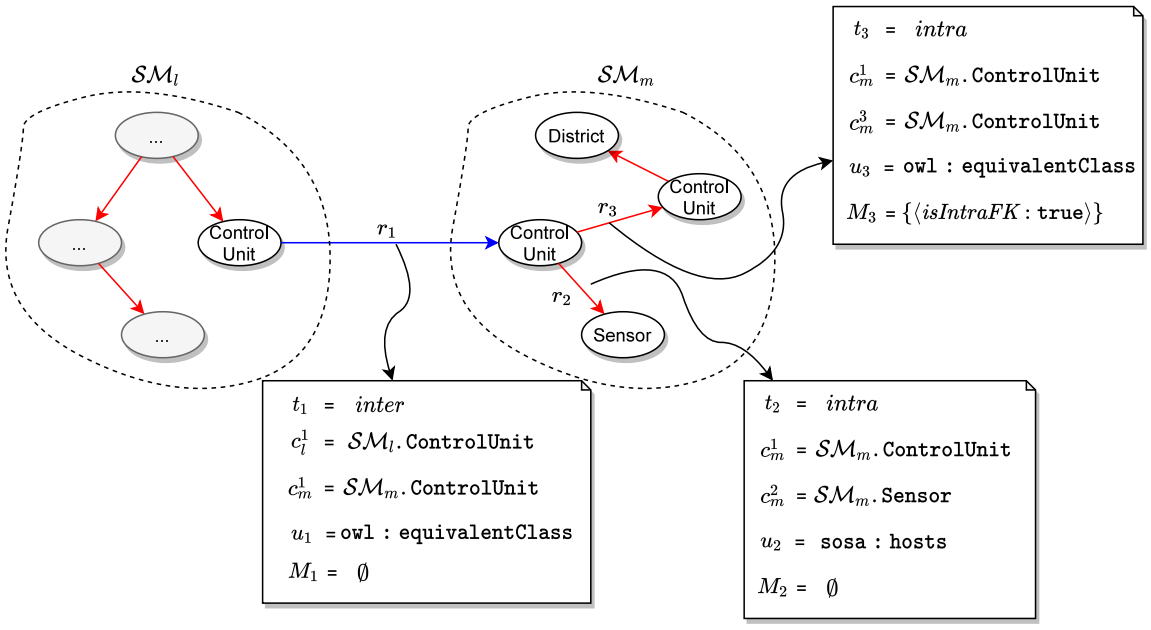


Figure 2.7: Example of intra-soruce and inter-source semantic relationships.

related to transformation operations to be applied on attributes (e.g., for converting string attribute values into integers).

2.4 Implementation

Inspired by the lack of a comprehensive environment to fruitfully assist domain experts in the creation of the semantic overlay on top of a Data Lake, we designed DL-DIVER, a Web-based semi-automatic solution independent of any particular application context. With DL-DIVER, domain experts fulfil the lexical enrichment of data sources attributes, their semantic annotation and, finally, the refinement and the creation of semantic relationships. Generally speaking, concepts and relationships in the overlay can be used for the development of semantics-enriched data exploration applications on the Data Lake.
In the following, we present the main features of DL-DIVER[4]. The application back-end has been developed using PHP 7.3 scripting language and exploits EasyRDF[5] PHP library for the management of Semantic Web data. Application data is persisted in a MySQL database. Concerning the front-end of the application, it has been created relying on the jQuery library and Bootstrap for the design of the style. This

[4]The demonstration video of the tool is available at `https://tinyurl.com/video-tool-sdl`
[5]`https://www.easyrdf.org/`

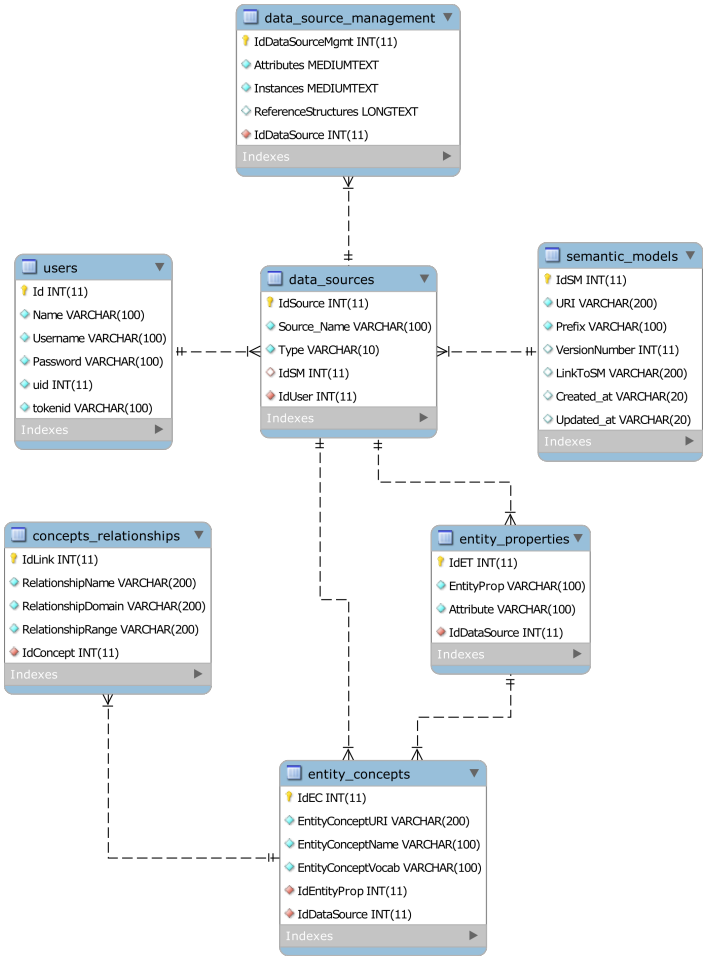


Figure 2.8: Data model for the application database.

application implements the functionalities of `DATA SOURCE ATTRIBUTES EXTRACTOR` and `ANNOTATOR` modules, with reference to the approach overview presented in Section 1.3.

2.4.1 Application database

Figure 2.8 illustrates the data model (tables, attributes and relationships between tables) for the application database, devoted to store metadata information about data sources, Entity Properties, Entity Concepts and Semantic Models. Specifically, it contains 7 tables:

- `users`, containing information related to domain experts having access to the application (that is, profile data of domain experts). Amongst its fields, the `uid`

and `tokenid` are exploited to query and access the abbreviations API.

- `data_sources`, containing information related to data sources (e.g., the name and the type) handled within the application, along with a reference to the domain expert who is responsible for the management of the source and to the associated Semantic Model.
- `data_source_management`, retaining for each data source the information regarding attributes and data sets schema.
- `entity_properties`, storing the Entity Properties for the attributes of the data sources.
- `entity_concepts`, storing for each Entity Property the Entity Concept assigned by the domain expert.
- `concepts_relationships`, storing the relationships belonging to each Entity Concept.
- `semantic_models`, containing information related to the Semantic Models created by domain experts, plus other data concerning the version of a Semantic Model, its URI, creation/update time references and the path representing the physical location of the file.

2.4.2 Data source management

The home page of the tool (*Data Sources Dashboard Page*, Figure 2.9) showcases the data sources managed by the domain expert. From this page, a domain expert can add a new data source (clicking on the "+" button) or see the details of an existing one (*Data Source Overview Page*, Figure 2.11). In the *Add Data Source Page* (Figure 2.10), the domain expert specifies connection parameters for the data source to be added, depending on the type of the source (the tool currently supports MySQL, MongoDB, PostgreSQL, CSV, JSON). On the other hand, from the *Data Sources Dashboard Page* the domain expert can inspect the details of the selected data source, obtained from the data source representation as a set of *data sets*. Specifically, the pages summarises, for each data set: (i) attributes and sample instances for each attribute; (ii) Entity Property and Entity Concept associated with each attribute (if any). Moreover, in the case of a relational data source and of a document-oriented database, an additional column is displayed, containing the reference table or collection, respectively (Figure 2.12 reports an example for a relational database). From the *Data Source Overview Page*, the domain expert can move to the *Entity Property Definition Page* (Figure 2.13) or to the *Entity Concept Definition Page* (Figure 2.14), clicking on the buttons next to each attribute.

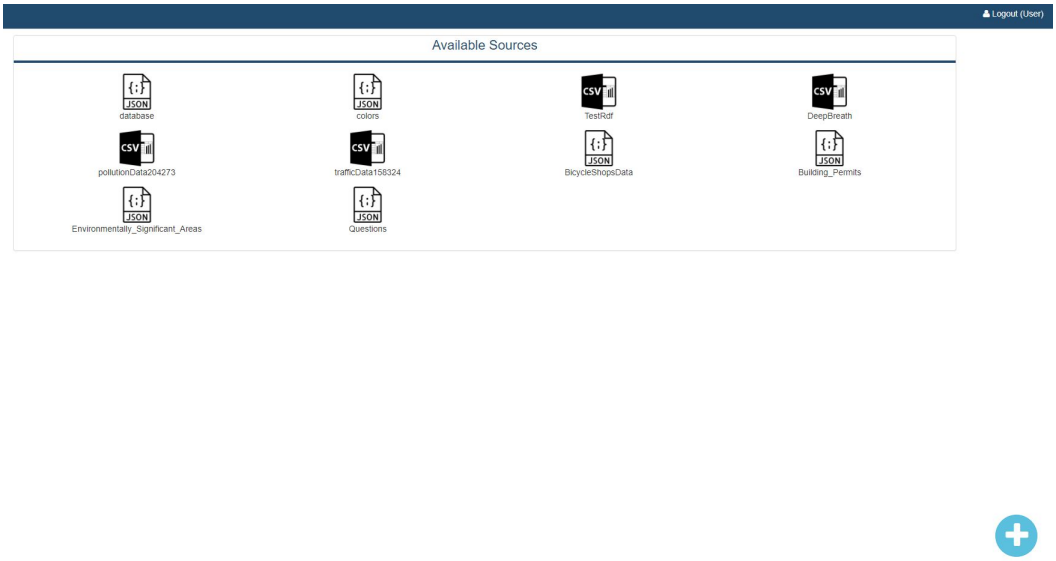


Figure 2.9: Data Sources Dashboard Page.

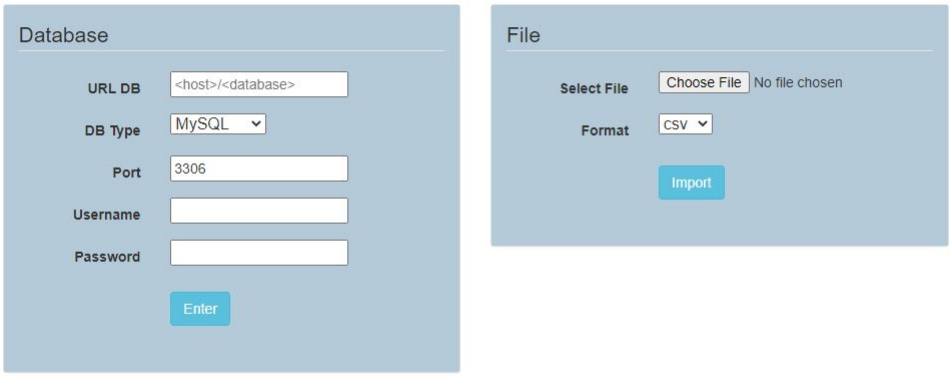


Figure 2.10: Add Data Source Page.

2.4.3 Entity Property definition

Clicking on the "Define Entity Property" button next to an attribute, the domain expert is redirected to the *Entity Property Definition Page* (Figure 2.13). The page contains a form with two fields: (i) a pre-filled textbox with the attribute technical name (as from the data source) and (ii) an editable textbox containing

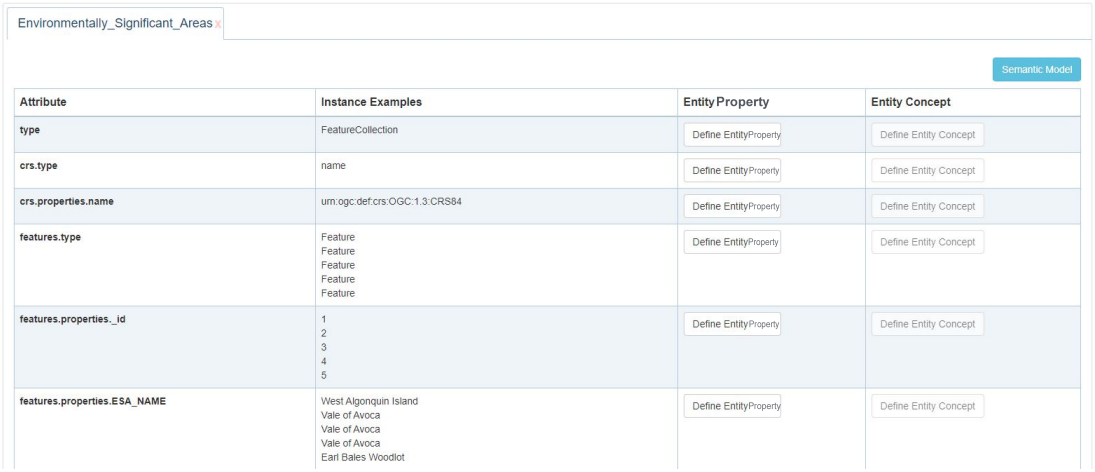


Figure 2.11: Data Sources Overview Page.

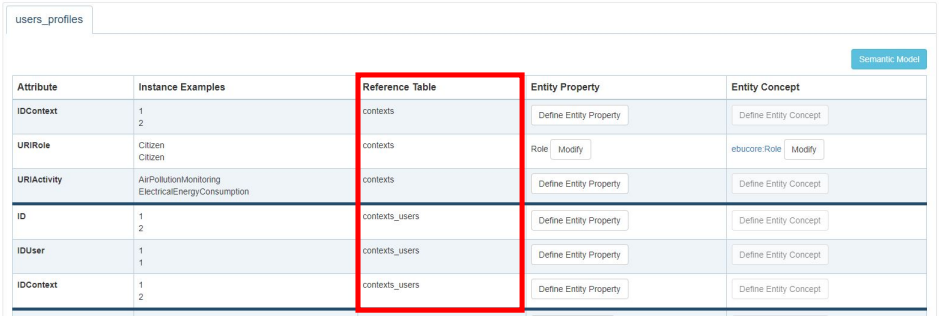


Figure 2.12: Data Sources Overview Page for a relational data source.

the Entity Property to be associated with the attribute (in the case of unassigned Entity Property, it contains by default the attribute technical name). To support domain experts in the choice of a suitable Entity Property for the attribute, the page proposes search results from the Abbreviations API and WordNet, in order to let domain experts define an Entity Property for an attribute not in a totally blind way, augmenting the degree of expressiveness and significance for the label. Roughly speaking, the Abbreviations API returns all the possible expansions for the attribute, considering it either as an abbreviation or an acronym, together with the category the abbreviation belongs to (i.e., the thematic domain, such as government, health). Conversely, WordNet search outcome is represented in a tree-like structure, resembling the corresponding online version[6], containing the list of the synsets for the attribute. Regardless of the search strategy, the domain expert can select the Entity Property

[6] `http://wordnetweb.princeton.edu/perl/webwn`

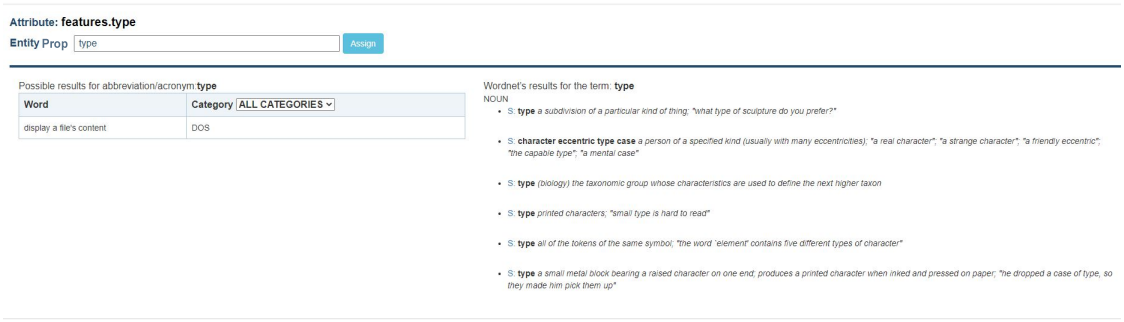


Figure 2.13: Entity Property Definition Page.

which best fits, according to his/her knowledge of the domain, the description of the attribute, that will be inserted in the aforementioned editable textbox at the top of the page. At this point, the search is re-triggered considering as a new searching item the formerly selected Entity Property, thus enabling the domain expert to further refine (if needed) the Entity Property, until the most appropriated has been found.

2.4.4 Entity Concept definition

From the *Data Source Overview Page* (Figure 2.11), the button "Define Entity Concept" leads to the *Entity Concept Definition Page* (Figure 2.14). Through this page, the domain expert can annotate the attribute, assigning a proper Entity Concept, browsing the set of domain ontologies using the related Entity Property as a searching key. The domain expert may either limit the search in a specific ontology (chosen from a dropdown list retaining the last recently used for the data source) or otherwise searching the whole set of domain ontologies (in case this is the first attribute that is being annotated for a given source or searches within specific ontologies produced no results). The list of the obtained concepts is displayed in a table reporting: (a) the name of the concept, (b) the name of the ontology the concept belongs to and (c) the complete URI of the concept. Then, the domain expert selects a specific concept or, by clicking on the "Specialise" button, creates a new concept as a specialisation of the one formerly selected (Figure 2.15).

2.4.5 Semantic Model creation

Each time a concept is added to the Semantic Model, relationships involving such concept are retrieved from the set of the domain ontologies, according to the steps reported in Section 2.3.3. When the domain expert is willing to export the Semantic Model, from the *Data Source Overview Page* (Figure 2.11), he/she can download the Semantic Model of the source by clicking on the "Semantic Model" button.

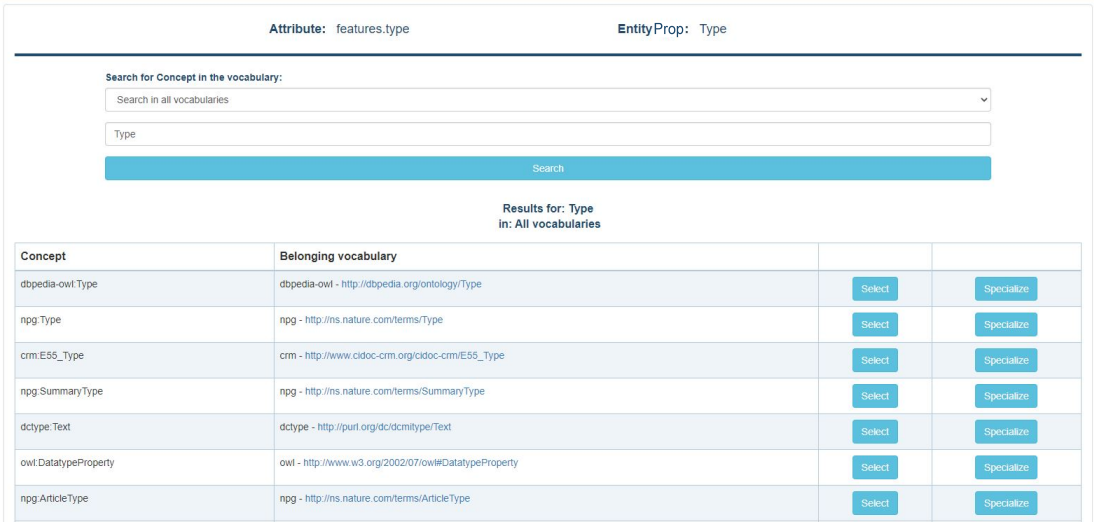


Figure 2.14: Entity Concept Definition Page.

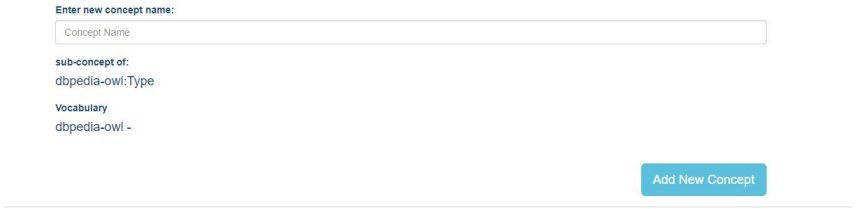


Figure 2.15: Entity Concept Specialisation Page.

The domain expert is redirected to another page (Figure 2.16), where he/she can check whether for the data source a Semantic Model previously created exists (along with the version and the time reference). Indeed, the tool assigns a progressive version number identifier for each created Semantic Model and keeps a reference to its latest version. After clicking on the "Create SM" button, the Semantic Model file is generated (in the RDF-XML format) and it is stored in a triplestore, which contains all the Semantic Models of Data Lake sources, plus inter-source semantic relationships (that is, the semantic overlay). However, before saving it, the domain expert has at his/her disposal the possibility of validating the Semantic Model relying upon an external service[7], invoked directly from the tool, and visualise/export also its graphical representation.

[7] `https://www.w3.org/RDF/Validator/`

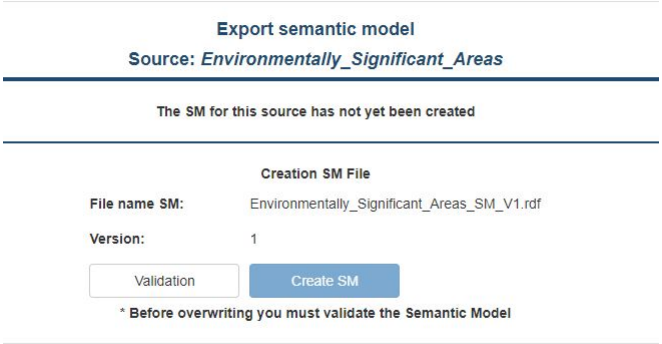


Figure 2.16: Semantic Model Export Page.

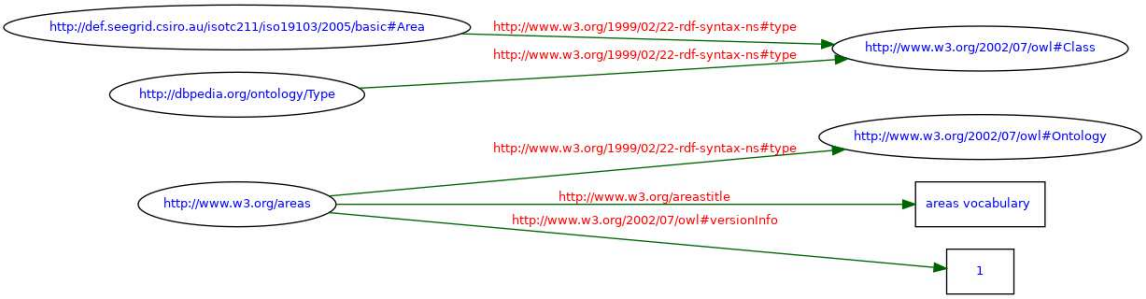


Figure 2.17: Example of Semantic Model graph generated by the validation service.

2.5 Experiments

Experiments have been performed in order to evaluate the effectiveness of the DL-DIVER tool in supporting domain experts for semantic annotation of data source attributes and for generation of Semantic Models. In the following sections, we describe the metrics and the type of experiments conducted, providing also a critical discussion about the results.

2.5.1 Metrics

In the experiments, we focused on both Entity Property (EP) and Entity Concept (EC) searches. In particular, given a search term t (which in the case of EP search is the attribute name, whereas for EC search is the Entity Property label), the well-known metrics *precision* $P(t)$ (i.e., the number of relevant results compared to the total number of returned results, in the context of a search operation) and *recall* $R(t)$ (i.e., the number of relevant terms returned among the search results, compared to the total number of relevant terms) are evaluated for quantifying the effectiveness of

	Abbreviations API		WordNet	
Attribute	$\overline{P(t)}$ **(%)**	$\overline{R(t)}$ **(%)**	$\overline{P(t)}$ **(%)**	$\overline{R(t)}$ **(%)**
Area.Point.x	0	0	2.9	100
retr_ts	0.9	40.0	0	0
Ws_URL	5.5	66.7	100	100
_id$oid	7.8	90.0	100	100
code.hex	30.4	35.0	25.0	20.0
One-timepassword	0	0	0	0
unit_cost	0	0	88.5	11.1
product_name	0	0	92.5	8.3
photo.file	14.3	100	20.0	5.0

Figure 2.18: Average precision $\overline{P(t)}$ and recall $\overline{R(t)}$ values obtained for searches with different types of attributes.

the two kinds of searches. As usual, $P(t)$ and $R(t)$ values belong to the $[0, 1]$ range, where, a precision score of 1.0 means that all search results are relevant (without saying anything about whether all relevant terms were retrieved) whereas a recall score of 1.0 means that all relevant terms were retrieved by the search (but says nothing about how many irrelevant terms were also retrieved). In the experiments, for each search term t, we conducted $N = 10$ different test, thus computing average values of precision $\overline{P(t)}$ and recall $\overline{R(t)}$. Moreover, for both EP and EC searches, we calculated also other summary statistics: (a) *percentage of searches without results*; (b) *percentage of searches with results* and (c) *percentage of searches with relevant results*.

2.5.2 Entity Property searches

We conducted tests on specific attributes (abbreviations and, broadly speaking, attributes with possibly special characters as delimiters in the name) using DL-DIVER to derive Entity Properties for data source attributes. In the following, we report the outcome of experiments conducted upon different kinds of attributes (either *simple* or *compound*). In this respect, a search term can be either a simple attribute or a compound attribute, the latter composed of several tokens.

Remarks on searches results. From a general viewpoint, the combination of the two types of searches (i.e., WordNet and abbreviations) returned good results over the 50% of searches, regardless the type of attribute. On the one hand, for simple attributes, WordNet searches often produced empty results in spite of the abbreviation searches; this is due to the fact that, in the majority of the cases, simple attributes were

Type of search	% of searches with results	% of searches without results	% of searches with relevant results
Abbreviations API	85,71%	14,29%	50%
WordNet	42,86%	57,14%	33,33%

(a)

Type of search	% of searches with results	% of searches without results	% of searches with relevant results
Abbreviations API	100%	0%	55,56%
WordNet	88,89%	11,11%	50%

(b)

Figure 2.19: Average searches results statistics for simple (a) and compound (b) attributes.

more likely to represent abbreviations or acronyms instead of self-explaining words. In this case, the limited number of relevant search results obtained by applying WordNet has been properly balanced using the abbreviations dictionary. On the other hand, for compound attributes, both WordNet and Abbreviations API searches have proven to carry out non-empty results, since a compound attribute is composed of tokens upon which the application and WordNet and Abbreviations API searches is effective, as explained above. From a higher viewpoint, Figure 2.19 reports summary statistics for Entity Property searches for simple and compound attributes. For simple attributes, there is a substantial difference between WordNet and Abbreviations API searches. Indeed, while for Abbreviations API searches the percentage of searches with results is on average 85.71%, for WordNet it is 42.86%. This is due to the fact that searches involving simple attributes have been performed using their raw names, which are more likely to represent abbreviations and thus the Abbreviations API search is more suitable. For compound attributes, performance of WordNet and Abbreviations API searches improve, leading to a percentage of 100% for Abbreviations API and 88.89% for WordNet, since results are obtained for each single token in a compound attribute. With WordNet, only the 11.11% of searches produced an empty result; it happened for compound attributes whose tokens were all acronyms or abbreviations.
Shifting the focus on single term searches, the table in Figure 2.18 contains average precision and recall for a set of simple and compound attributes, undergoing semantic disambiguation. Considering the compound attribute `Area.Point.x`, WordNet search proved to be effective because each composing token (`Area`, `Point` and `x`) is a commonly used term. Conversely, for the compound attribute `retr_ts`, its tokens (`retr` and `ts`) are apt to be disambiguated through the Abbreviations API, as they

Attribute	Assigned Entity Property	$\overline{P(t)}$ (%)	$\overline{R(t)}$ (%)	Specialisation occurred (at least one time)?
Area.Point.x	xpoint	66.7	100	NO
retr_ts	timestamp	42.8	100	NO
Ws_URL	url	6.5	66.7	YES
_id$oid	Identifier	70.3	99.5	NO
code.hex	code	42.1	97.8	YES
One-timepassword	otp	28.6	100	NO
unit_cost	cost	32.3	75	YES
product_name	product	76.7	79.2	NO
photo.file	photo	19.1	81.1	YES

Figure 2.20: Average precision $\overline{P(t)}$ and recall $\overline{R(t)}$ values of search results for Entity Concepts for a sample of attributes (LEA search). The last column specifies whether a specialisation was needed during the annotation process.

represent acronyms/abbreviations. Generally speaking, average precision for certain attributes (e.g., `ws_URL`, `_id`) was very low if the search was performed through the abbreviations dictionary since, for these specific terms, the set of obtained results contained more than one hundred of items (specifically, 182 results for `ws_URL` and 123 for `_id`) and amongst them, only a few were relevant.

2.5.3 Entity Concept searches

Here we report the tests conducted for deriving Entity Concepts. In particular, we focused on two kinds of experiments: (a) search Entity Concepts starting from attributes with no preliminary lexical enrichment (called *plain* search); (b) search Entity Concepts from Lexically Enriched Attributes (hereafter called *LEA* search). Also for this type of search, $N = 10$ searches for each attribute (plain search) and Entity Property (LEA search) have been performed and average values of precision and recall have been computed. Additionally, we reported whether a *specialisation* was performed by the domain expert (in at least one of the searches), during the annotation process. Table in Figure 2.20 summarises experiments results on a group of representative attributes coming from Data Lake sources of the example in Figure 2.3.

Remarks on searches results. From the table in Figure 2.21 we can see that, if a lexical enrichment of data sources attributes has not been previously performed, the search of the Entity Concepts leads to an empty result in the 88.89% of cases, thus demonstrating the limit in using attributes raw labels as searching keys. Conversely,

Type of search	% of searches with results	% of searches without results	% of searches with relevant results	% of searches with specialisation of a relevant result
Plain	11,11%	88,89%	33,33%	5,56%
LEA	98,88%	1,12%	86,35%	25,50%

Figure 2.21: Average Entity Concept searches results statistics.

the introduction of Entity Properties improved searches by delivering always a non-empty set of results, and 86.35% of the searches contained relevant results. Shifting towards concept specialisation issues, in the 25.5% of LEA searches, the domain expert had to specialise a concept (starting from one drawn from the set of relevant ones) in order to achieve the required annotation. This percentage grows with respect to the plain search case, accordingly to the fact that there is a broader set of relevant results to start from (i.e., more alternative concepts). Focusing again on LEA searches, only the 1.33% of the searches demanded for the creation of a new concept from scratch (that is, being inherently a subclass of `owl:Class` concept). Remarkably, the latter case happened when sought concepts represented technical terms or they belonged to a domain not covered (or only partially) by the LOV ontologies (e.g., terms from the environmental domain, like pollutants technical names).
Moving the focus to Entity Concept searches for a single Entity Property, average recall values were above the 50% for all LEA searches, thus demonstrating that the introduction of Entity Properties helped in improving the subsequent retrieval of suitable Entity Concepts, obtaining also those items that were unreachable using only the raw attribute name. Conversely, average precision values were low (less than 50%) for Entity Properties like `code` due to the fact that using these Entity Properties as keywords for searching the LOV ontologies delivered a lot of results (in this example, 670) and amongst them, only a few were relevant (e.g., `code` is used with different meanings in different domains). To cope with this kind of situation, a domain expert had to perform a specialisation in the annotation process.

Chapter 3

Indicators Modelling for Data Lake Exploration

3.1 Introduction

In this chapter, we present the creation of a framework for modelling and exploring indicators. The framework has as a pivotal element the Multi-Dimensional Ontology (MDO), exploited by data analysts to represent indicators and exploration dimensions. The MDO is provided with validation rules, to support data analysts in the design of indicators. The steps of the indicator modelling procedure are discussed leveraging the MDO in the Smart City domain to build the Smart City Exploration Graph, a knowledge structure containing the semantic representation of indicators, and where indicators are related to the activities performed by users and users' categories in the Smart City domain (that is, indicators definition is enriched with *personalisation* concepts). In the last part of the chapter, we explain the execution mechanism for issuing queries over the Data Lake, which is grounded on two features: (i) mappings, defined by data analysts to identify how indicators in the Smart City Exploration Graph can be calculated in terms of data source attributes; (ii) a query procedure, having as input the portion of the semantic overlay tailored by mappings and targeted to create the query plan to retrieve and aggregate indicators values from Data Lake sources.

3.1.1 Related Work

Ontology-based conceptual models for indicators design. In the literature, ontology-based conceptual models have been widely adopted for the design of indicators, due to their shared and easily understandable conceptualisation. For instance, the work presented in [20] proposes an ontology for representing indicators,

Approach	Domain-independent	Presence of rules (Aim)	Concepts to model exploration contexts
Diamantini et al. [20]	✓	NO	YES (Business objectives)
Kritikos et al. [35]	✓	NO	NO
Del et al. [42]	✓	YES (Infer additional knowledge)	NO
Amor et al. [3]	✗	NO	YES (Activities)
Li et al. [37]	✗	NO	YES (Users' roles, goals)
Pasic et al. [51]	✗	NO	NO
Kuster et al. [36]	✗	NO	YES (Actions, Individuals, Profiles)
Mehdi et al. [44]	✗	YES (Infer additional knowledge)	NO
Sanfilippo et al. [61]	✗	YES (Validation)	NO
Ours	✓	YES (Validation)	YES (Users' categories, activities)

Figure 3.1: Summary of approaches on ontology-based conceptual models for indicators design.

developing a data explorer aimed at supporting users in the extraction of indicators values from a shared repository. The innovative aspect of the work lies in the fact that it is one of the first research efforts implementing a solver for automatically deriving indicators from their calculation formulas, starting from the semantic representation of indicators. Differently, authors in [35] strove to propose a flexible ontology for Key Performance Indicators (KPIs) modelling, considering also a meta-model for tracking dependency issues descending from the inherent evolution of the monitored system. Lastly, in [42] an ontology-driven approach is proposed to model KPIs, emphasising the importance of correlation between indicators values, to infer new knowledge. Generally speaking, amongst the three aforementioned works, only [20] envisages to consider additional knowledge to bind indicators to exploration contexts, expressing users' goals while exploring data, albeit they are not related to users' roles. The work in [3] proposes an ontology for designing KPIs with the goal of improving Business Process Management. In the ontology, KPIs are related and grouped according to the activities of the Business Process, and a prototype application is exploited by the user to learn about relationships between KPIs involved in an activity, mined through association rules. In this approach, exploration of KPIs is contextualised according to the chosen activity, but no considerations about the users exploring KPIs are made (e.g., limiting the visibility of activities/KPIs according to the role of users in the Business Process, such as process designers or participating actors). Focused on the Smart City domain, authors in [37] model the

Energy Management KPI (EM-KPI) ontology with the aim, on the one hand, of facing the interoperability problem between cross-domain heterogeneous data and, on the other hand, of enabling the extraction of insightful data for stakeholders, avoiding unnecessary analysis. Despite exploration contexts are taken into account, emphasising stakeholders' roles and related performance goals grouping KPIs, the modellisation of KPIs is domain-dependant and it demands the materialisation of all files related to energy consumption data as Linked Data (RDF), before starting the exploration. Another contribution in the Smart City domain is provided by [36], where a semantic data model (the Urban District Sustainability Assessment ontology, in brief UDSA) is considered to support near real-time urban sustainability assessment and enhance the semantics of sensor network data. This approach reconciles several domain-specific ontologies within one high-level ontology, thus providing a useful resource also for the wider smart cities context, and considers the introduction of personalisation concepts to drive the exploration of indicators. The work in [51] provides a condition monitoring ontology for automation systems, incorporating ISO standards for condition monitoring and KPIs, but neither rules for enforcing inference of additional knowledge nor concepts for exploration contexts modelling are envisaged. Lastly, two research efforts in the manufacturing field (turbine applications [44] and additive manufacturing [61], respectively) employ rules in their KPIs ontologies to automate the selection of appropriate aggregation function to calculate the composite metrics (the former) and validate process templates inserted in the knowledge base, supporting the users with proper error messages arisen by violations of the rules (the latter). Nevertheless, in both the previous works, exploration issues are not taken into account.

Data Lake exploration techniques. In the recent years, different strategies have been devised to query Data Lakes content for exploration purposes. Indeed, there is not a one-size-fits-all technique to foster, since the choice depends mainly on the application domain and on other factors (e.g., number of Data Lake sources to handle and their type). For instance, in [41] the renowned OBDA (Ontology-Based Data Access) paradigm has been employed, whereby heterogeneous data is seamlessly accessed and queried through a semantic layer supported by a reasoning tool which interprets a list of mappings (written in a suitable language by domain experts) to access data sources content, providing a semantic representation of data in RDF format, which can be queried through the SPARQL language. The work presented in [19] extracts, from a network-based representation of data sources, proper *Thematic Views*, concerning one or more topics of interest for the user, obtained by extracting and merging data coming from different sources and used as an entry point to query the Data Lake. In both the two previous work, the leading strategy is to derive an alternative representation of data sources, either entirely based on a semantic

Approach	Abstraction layer over the Data Lake	Query Strategy	Transformation applied before/during data ingestion
Squerall [41]	NO	OBDA	NO (virtual RDF triples)
Diamantini et al. [19]	Ego-Networks	Thematic Views	NO
Ben Hamadou et al. [29]	Dataspace	Query Graph	NO
Lytra et al. [38]	Knowledge Graph	N/A	YES
Walker et al. [69]	Knowledge Graph	N/A	YES
Malysiak-Mrozek et al. [40]	Knowledge Graph	N/A	YES
Hai et al. [28]	Datasets	Logical rules from JSONiq query	NO
Kasrin et al. [31]	Knowledge Graph	Knowledge Coordinators + Upper Level Domain Model	NO
Kondylakis et al. [34]	Data Management Layer	Mappings + Real-time on-line/off-line integration of data	YES
Singh et al. [64]	Data Tiles (word cloud of tags + visualisations)	Probabilistic join of datasets	NO (inverted-index update when a new dataset is uploaded)
Ours	Semantic overlay	Query Graph + Mappings	NO

Figure 3.2: Summary of approaches on Data Lake exploration techniques.

representation of their content (as performed in [41]) or, in a mixed way, based mainly on metadata extracted from data sources and only marginally on their content (as in [19]). Authors in [29] propose an approach applied in a polystore environment, composed of several data sources. Attributes of data sources and mutual relationships are abstracted in a so-called *dataspace*; in this respect, to explore polystore content, queries issued by users are rewritten leveraging the dataspace, creating local plans (to be executed on the single data sources) and a global plan, which to joins and aggregates data coming from different data sources. The proposal of [31] considers a Polyglot Persistence Unit containing a Catalog responsible for providing inventory services and for generating unique identifications used for referencing datasets and accessing them. This identification is added to a knowledge graph to create a link between a metadata store and the data store, orchestrated by one or more Knowledge Coordinators. The approach in [28] aims at set up a query rewriting mechanism for heterogeneous Data Lakes. Specifically, a query parser creates a set of logical rules from the input query (expressed in JSONiq format) which are in turn translated into executable queries over the single involved data sources. Integration and join of partial results is performed through Spark SQL. Authors in [34] devise a Data Management Layer in the frame of a Data Lake infrastructure, setting up an ontology-based integration system where integration can be either at run-time or through

an ETL process. Integrated information is then accessible through proper APIs for subsequent exploration. The iFuse system [64] pays attention on Data Fusion for deriving insights from Data Lake sources. In iFuse, each dataset is represented visually as a *data tile* consisting of a word cloud of tags generated from attribute names or provided by the user and the visualisations associated with the dataset. Moreover, it introduces the notion of probabilistic joining of data, to be employed when datasets do not have a common join key but their fusion may be advocated for analytical reasoning. Conversely, in all of the following approaches, knowledge graphs structures contain the representation of the whole data source content, without capitalising on metadata derived from attributes and relationships between them, which would be useful to avoid a resource consuming translation of data source content into a graph representation. In [38], data sources content is directly linked towards external knowledge bases, creating a graph structure reflecting Data Lake content. In other cases, the process is also empowered and supported by the adoption of probabilistic techniques [40]. Furthermore, in [69] a knowledge graph stores data fragments, represented as different typologies of nodes inside the graph, whose purpose is to abstract data sources content and assure effective data access.

3.1.2 Novel Contributions

In this chapter, we define a *Multi-Dimensional Ontology* (in brief, MDO), containing baseline concepts and relationships for modelling indicators and dimensions. With respect to the ontology-based approaches for indicators design described in the previous section, we conceived the employment of the MDO in scenarios characterised by high heterogeneity, wherein is not feasible to compute indicators values in advance. In fact, in the case of a Semantic Data Lake as formalised in this thesis, data analysts, when designing indicators, rely only on the abstraction of data sources in the semantic overlay, since data is extracted from the Data Lake according to a pay-as-you-go or on-demand approach. Thus, since better focusing the exploration on a subset of indicators becomes of paramount importance when monitoring a phenomenon of interest, we introduced, with respect to the previous modelling frameworks, personalisation aspects in the MDO, to express that an indicator is suitable for one or more user's categories and that influences one or more activities performed by users. The MDO, employed in the Smart City domain, allowed the construction of the so-called *Smart City Exploration Graph* (in brief, SCEG), containing the conceptualisation of indicators and dimensions, enriched with personalisation elements, conceived as the starting point for users to explore Smart City data. The SCEG modelling procedure, performed relying on concepts and semantic relationships of the MDO, is enforced by *validation rules*, to support data analysts in the modelling phase, which is an error-prone task, especially when dealing with an increasing number of indicators.

In the works presented in the former section, validation rules are not present; only the proposal of [42] considers the adoption of rules, but the aim there is to set up a reasoning system for KPI modelling focused on the assessing whether redundant KPIs are present in the ontology, thus not meant to capture errors in indicators modelling. In the last part of the chapter, *mappings* will be exploited as a way to associate the definition of an indicator (in the Smart City Exploration Graph) to concepts and relationships in the semantic overlay of the Data Lake. In other words, mappings enable to specify what are the Semantically Annotated Attributes of data sources needed to define an indicator and calculate its values along the associated dimensions. To obtain actual values of indicators, we adapted the Ben Hamadou's approach proposed in [29] to our three-layered model, in order to create a query plan to retrieve data from Data Lake sources. Despite Ben Hamadou's approach has been conceived for a polystore (that is, a loosely coupled integration of heterogeneous data sources, accessed through their local language), the idea behind its methodology, devoted to build a query plan for heterogeneous data sources, can be adapted to a Semantic Data Lake environment as devised in this thesis, whose underlying data model involves data sources attributes and data sets. Considering the approach in [29], the steps of the exploration procedure of indicators as defined within the SCEG, targeted to retrieve indicators values from Data Lake sources, will be formalised. This chapter is based on our previously published works [6, 8, 9], where we introduced the Multi-Dimensional Ontology and the Smart City Exploration Graph. Here, we move a step forward by formalising the overall exploration procedure, giving also evidence of implementation and experimentation issues.

3.2 The Multi-Dimensional Ontology

In this section we describe the *Multi-Dimensional Ontology* (in brief, MDO), conceived as a modelling framework leveraged by data analysts to semantically represent indicators, their dimensional characterisation and formulas to aggregate them. The design of the MDO has been inspired by an existing research effort [20], where authors proposed an ontology-based model for the formal representation of indicators, to be employed in a data warehouse context. However, we extend the conceptualisation provided in [20], including also: (i) concepts and relationships for exploration personalisation (ii) concepts which are not directly exploited to model indicators, but to verify their correctness (the so-called *validation concepts*). Hereafter, the conceptual areas of the MDO, depicted in Figure 3.3, are described, emphasising the role they fulfil when modelling indicators.

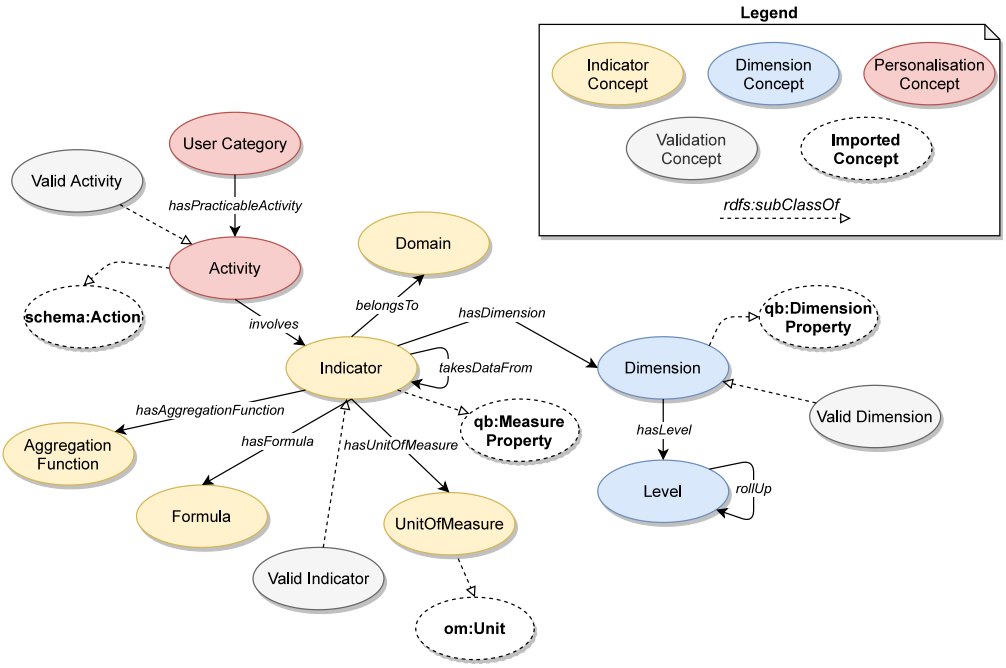


Figure 3.3: TBox of the Multi-Dimensional Ontology (in the figure, the semantic areas are highlighted with different colours).

3.2.1 Indicators concepts

Indicators are designed to aggregate data according to several perspectives and represent the ultimate target for data exploration, as their value may give valuable insights to users. In the ontology, indicators can be modelled as *simple* or *composite*, i.e. calculated from other indicators (through `takesDataFrom` semantic relationship). Being measurable quantities, they are endowed with a unit of measure (abstracted through the `UnitOfMeasure` concept) and have a calculation formula (`Formula` concept). For an indicator, it is also possible to define an aggregation function (`AggregationFunction` semantic concept), adopted as standard operation to be applied whenever switching from a dimensional level to another (through the `rollUp` semantic relationship). Finally, indicators may be associated with domains, such as environment, health, energy and so forth; this is represented inside the ontology through the `belongsTo` semantic relationship, connecting `Indicator` concept with `Domain` concept.

3.2.2 Dimensions concepts

Indicators can be segmented and calculated according to several dimensions (`hasDimension` relationship), which are in turn organised into levels, modelled through the `Level` concept. The `rollUp` semantic relationship enables a hierarchical organisation of levels, thus ensuring to navigate from the lowest to the highest aggregation level, resembling the renowned OLAP *roll-up* operator.

3.2.3 Personalisation concepts

Owing to the wide variety of data that can be explored, data exploration can be personalised taking into account user's category (e.g., building manager, citizen) and activities performed by users (e.g., building monitoring, check air pollution), the former abstracted through the `UserCategory` concept, whereas the latter through the `Activity` concept. An activity may involve one or more indicators to be observed, compatible with its purpose (`involves` relationship). Activities can be performed by users, according to their role inside the Smart City context (`hasPracticableActivity` relationship).

3.2.4 Validation concepts

Modelling indicators can be an error-prone task, especially when data analysts have to deal with an increasing number of indicators. To assist the data analyst in the modelling task while designing a specific SCEG starting from the MDO concepts, *validation rules* have been included in the MDO. These rules are reported in the following and are defined according to Description Logic (DL) syntax [5]; these rules are implemented through *anonymous classes* (i.e., MDO concepts defined through a specific DL expression, used uniquely for validation purposes):

- *ValidActivity* $\iff$ (`involves some Indicator`), asserting that a valid activity involves at least one indicator;
- *ValidDimension* $\iff$ (`hasLevel some Level`), asserting that a valid dimension hierarchy, being associated with an indicator, must gather at least one dimension level;
- *ValidIndicator* $\iff$ (`belongsTo some Domain`) $\wedge$ (`hasDimension some Dimension`) $\wedge$ (`hasUnitOfMeasure only UnitOfMeasure`) $\wedge$ (`involvedIn some Activity`), asserting that a valid indicator belongs to at least one domain, is explorable according to at least one dimension hierarchy, eventually has a unit of measure and is involved in at least one activity.

In particular, the rules are leveraged to check the semantic definition of new activities, dimension hierarchies and indicators. In this respect, the data analyst will run a *reasoner* (a stand-alone software module linked to the modelling tool), in order to check whether the modelled portion of the SCEG was correctly defined. Focusing on indicators, a valid indicator concept will be classified by the reasoner both as a sub-class of MDO `Indicator` and `ValidIndicator` concepts, implementing the namesake validation rule (also for a dimension hierarchy and activities). Therefore, it must hold that an indicator *ind* is valid iff $ind \sqsubseteq \texttt{Indicator} \wedge ind \sqsubseteq \texttt{ValidIndicator}$ and the same for dimension hierarchies and activities, where $\sqsubseteq$ denotes the subsumption binary operator.

3.2.5 Imported concepts

For the definition of the MDO, pivotal concepts from available foundation ontologies have been exploited to: (i) represent users' activities (Schema.org ontology[1]), (ii) characterise indicators and dimensions as analytical data entities (Data Cube ontology[2]) and (iii) model units of measure for indicators (ontology of units of measures[3]). Indeed, in the context of ontology engineering, reuse of existing concepts and relationships is recommended as a key factor to develop cost effective and high quality ontologies, reusing components that have already been validated [11].

3.3 Indicators modelling procedure

In the following we describe the process leading to the creation of novel indicators, starting from the knowledge provided by the MDO. Practically, data analysts semantically describe indicators through the *specialisation* (denoted through the symbol $\sqsubseteq$) of MDO concepts and relationships, in order to weave the knowledge about indicators and related dimensional hierarchies. This entails the usage of semantic relationships `rdfs:subClassOf` (for sub-classing a concept) and `rdfs:subPropertyOf` (for sub-classing a relationship). In details, to model an indicator, the data analyst performs the steps reported in the list below.

1. **Creation of indicator concept.** This operation may involve the specialisation of the `Indicator` concept of the MDO or one of its sub-concepts (that is, one of the formerly created indicators). In this way, the indicators hierarchy is further extended.

[1] `https://schema.org/` (prefix: `schema`)
[2] `https://www.w3.org/TR/vocab-data-cube/` (prefix: `qb`)
[3] `http://www.ontology-of-units-of-measure.org/resource/om-2/` (prefix: `om`)

2. **Definition of composite indicators.** Indicators whose calculation depends on other fine-grained indicators are called *composite* indicators. They have to be associated with their dependencies, through the `takesDataFrom` semantic relationship. For each component indicator the modelling procedure has to be repeated in a recursive manner.

3. **Creation of indicator surrounding knowledge.** This includes the creation of a `Formula` (i.e., specifying the calculation expression for a composite indicator, in terms of its composing indicators), the `UnitOfMeasure` for the indicator and the `AggregationFunction`.

4. **Link to dimensional hierarchies.** Once an indicator has been modelled, it has to be bound to one or more dimensional hierarchies, representing the analytical perspectives meant for it. The data analyst may reuse previously created hierarchies or define new ones, relying on the pivotal concepts `Dimension` and `Level` from the MDO. To guarantee the navigability across dimensional levels, the data analyst may include `rollUp` relationships.

Example 6 *Figure 3.4 reports an example of semantic representation of indicators that is compliant with the MDO. An indicator (e.g.,* `AirPollutionIndicator`*) is described through a formula and an aggregation function (see* `hasFormula` *and* `hasAggregationFunction` *relationships), a unit of measure (e.g.,* `ppm`*) and it is associated with at least one dimension (e.g.,* `SpatialDimension`*, by means of the* `hasDimension` *relationship). Dimensions are articulated over different granularity levels (for instance, the* `SpatialDimension` *is articulated over the* `Apartment`*,* `Building`*,* `District` *and* `City` *levels). Indicators can be defined in a recursive way, that is, they can be computed starting from other fine-grained indicators. For example, in Figure 3.4 the* `HouseholdCO2` *indicator is computed starting from the* `CO2Heaters`*.*

3.4 The Smart City Exploration Graph

The semantic representation of indicators, their dimensional characterisation and aggregating formulas as applied in the Smart City domain, are contained within the Smart City Exploration Graph $\mathcal{G}$, which is formalised as follows.

Definition 8 (Smart City Exploration Graph) *The Smart City Exploration Graph (SCEG) is a graph data structure defined as* $\mathcal{G} = \langle C, R \rangle$ *where: (i)* C *contains concepts derived from the specialisation MDO concepts, that is* $\forall c_i \in C, c_i \sqsubseteq c_j^{MDO}, i \neq j$ *and (ii)* R *contains relationships derived from the specialisation of MDO relationships, that is* $\forall r_i \in R, r_i \sqsubseteq r_j^{MDO}, i \neq j$*.*

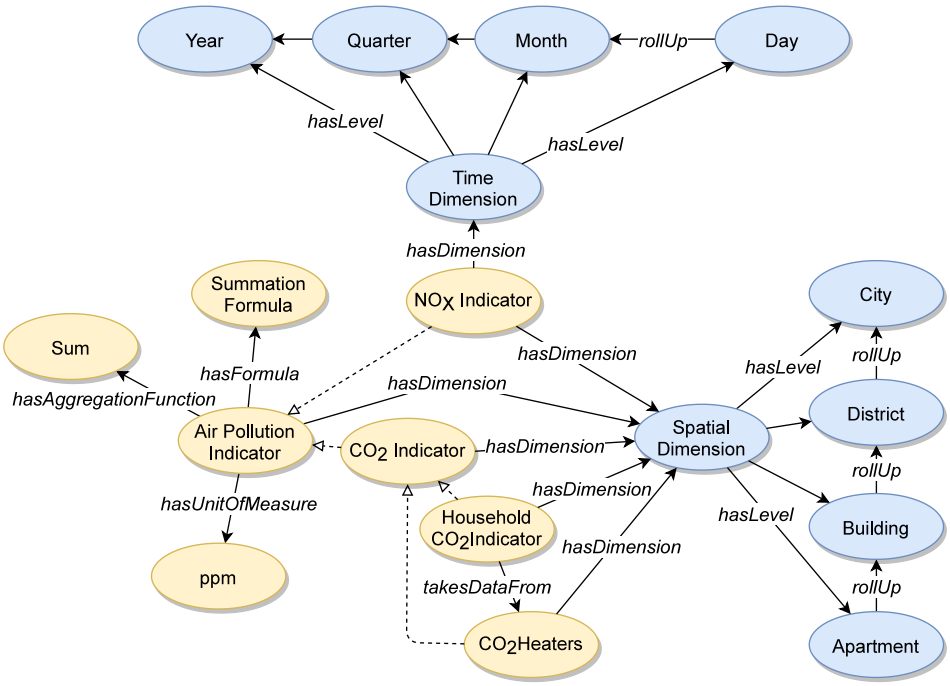


Figure 3.4: Example of semantic representation of indicators.

In the Smart City Exploration Graph, the semantic representation of indicators is further enriched by associating indicators with their domains (e.g., environment, health) through the `belongsTo` semantic relationship, binding an `Indicator` concept to a `Domain` concept. Moreover, concepts and relationships of the Smart City Exploration Graph $\mathcal{G}$ derived from the MDO are added to assert that the knowledge of some indicators influences specific activities, in turn performed by specific users' categories within the Smart City. This is achieved by considering Personalisation Concepts `UserCategory` and `Activity` from the MDO. In particular, the `hasPerformableActivity` relationship binds a `UserCategory` concept (or one of its sub-concepts) to an `Activity` concept (or one of its sub-concepts). Similarly to what happens for indicators and dimensional levels, activities can be organised into a hierarchy. Additionally, the `involves` semantic relationship links an `Activity` concept to one or more `Indicator` concept.

Example 7 *In Figure 3.5, indicators are linked with Personalisation Concepts from the MDO, modelling activities that can be carried out in a Smart City by different categories of users. For example,* `AirPollutionIndicator` *is involved in the* `AirPollutionMonitoring` *activity (*`involves` *semantic relationship), which can be performed both by a* `Citizen` *and a* `BuildingAdministrator` *(*`hasPracticableActivity` *relationship). In this respect, the Smart City Exploration Graph $\mathcal{G}$ is conceived as a*

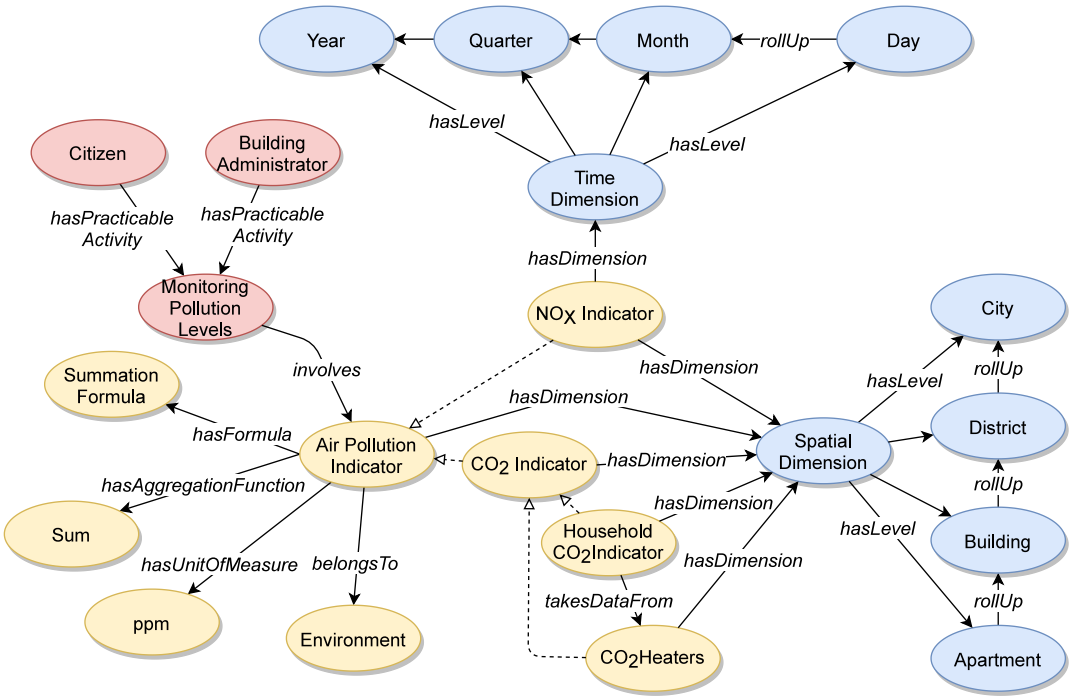


Figure 3.5: Smart City Exploration Graph enriched with personalisation concepts.

starting point for users to explore Smart City data.

3.4.1 Multi-Dimensional Descriptors

Focusing on the Smart City Exploration Graph $\mathcal{G}$, we conceive two possible strategies for exploring it. The first one is an interactive exploration, that is fostering a graphical tool for navigating it by moving across nodes, leveraging their interlinking relationships. This exploration strategy, despite being intuitive, may be unpractical as the number of indicators (and related dimensions) in the Smart City Exploration Graph grows. The second strategy is grounded on the assumption that users, when exploring data, inherently reason according to a multi-dimensional organisation. For instance, an environmental engineer may be interested in inspecting the daily concentration of a pollutant in a specific district of the Smart City. In this respect "day" and "district" are the two exploration dimensions, descending from the exploration needs of the user. Thus, focusing on the second strategy, and for concisely expressing the binding of an indicator to its associated dimensions, we formalise the notion of *Multi-Dimensional Descriptor* (MDD), defined over $\mathcal{G}$. Roughly speaking, a MDD allows to contextualise the values of a Smart City indicator according to one or more dimension, resembling the association of an indicator to its dimensional

Algorithm 1: MDDs Generation

Input : Smart City Exploration Graph $\mathcal{G}$
Output: Set of MDDs $\mathcal{T}$

1 # *Get indicators concepts from* $\mathcal{G}$
2 $\mathcal{I} \leftarrow \{i \mid i \in \mathcal{G} \wedge i \sqsubseteq \texttt{Indicator}\}$;
3 $\mathcal{T} \leftarrow \emptyset$;
4 **foreach** $ind \in \mathcal{I}$ **do**
5 # *Get dimensions concepts from* $\mathcal{G}$ *associated with ind*
6 $D_{ind} \leftarrow \{d \mid d \in \mathcal{G} \wedge ind \texttt{ hasDimension } d\}$;
7 $n \leftarrow |D_{ind}|$;
8 # *Creation of levels vectors through Cartesian product of levels sets*
9 # $Levels(d_i) = \{ALL\} \cup \{l \mid d_i \texttt{ hasLevel } l\}$
10 $\mathcal{L} \leftarrow Levels(d_1) \times Levels(d_2) \times \ldots \times Levels(d_n)$;
11 **for** $j = 1 \ldots |\mathcal{L}|$ **do**
12 $\tau_j \leftarrow \langle ind, L_j \in \mathcal{L} \rangle$;
13 $\mathcal{T} \leftarrow \mathcal{T} \cup \tau_j$
14 **return** $\mathcal{T}$

coordinates of the OLAP paradigm. In the following, the formal definition of MDD is given.

Definition 9 (Multi-Dimensional Descriptor) *A Multi-Dimensional Descriptor, defined over the Smart City Exploration Graph* $\mathcal{G}$*, is denoted as* $\tau_i = \langle ind_i, L_i \rangle$*, where: (i)* ind_i *represents an indicator concept from* $\mathcal{G}$ *and (ii)* L_i *is a vector of dimensional level concepts. Specifically:*

- D_i *is the set of dimensions associated with* ind_i*;*
- L_i *is a vector of dimensional levels represented as* $L_i = (l_i^1, l_i^2, \ldots, l_i^n)$*,* $l_i^j \in Levels(dim_i^j)$*,* $dim_i^j \in D_i$*,* $n = |D_i|$*, where* $Levels(dim_i^j)$ *denotes the set of dimensional levels associated with dimension* dim_i^j *in* $\mathcal{G}$*, comprising the special level denoting the coarsest aggregation (ALL);*

We denote with $\mathcal{T}$ *the overall set of MDDs for the Exploration Graph* $\mathcal{G}$*. For an indicator* $ind_i \in \mathcal{G}$*, the maximum number of possible MDDs is equal to* $\prod_{j=1}^{n} |Levels(dim_i^j)|$.

MDDs generation procedure from the Smart City Exploration Graph $\mathcal{G}$ is summarised in Algorithm 1. The algorithm considers the set of indicators available in $\mathcal{G}$, denoted with $\mathcal{I}$ (line 2) and, for each indicator $ind \in \mathcal{I}$, retrieves the associated dimensions

(lines 6-7). Then, starting from the dimensions, levels vectors are computed by performing the Cartesian product, denoted with $\mathcal{L}$, of the set of levels of each dimension (line 10). Lastly, the MDDs related to *ind* (whose number is equal to $|\mathcal{L}|$) are assembled by constructing pairs in the form $\langle ind, L_j \rangle$, with $L_j \in \mathcal{L}$, and are added to the output set $\mathcal{T}$.

Example 8 *With reference to Figure 3.5, considering a sample MDD* $\tau_1 = \langle$`CO2Indicator`$, ($`City`$,$ `Year`$)\rangle$*, dotted notation is used to access its content, that is* $\tau_1.ind_1 =$ `CO2Indicator` *and* $\tau_1.L_1[$`SpatialDimension`$] =$ `City`*,* $\tau_1.L_1[$`TimeDimension`$] =$ `Year`*. The maximum number of possible MDDs involving* `CO2Indicator`*, obtained combining the levels of the two dimensions, is equal to 25 (*$= 5 \cdot 5$*), including for each of the two dimensions also the special level ALL.*

3.5 Semantic Data Lake exploration

After the explanation of the modelling procedure for indicators and dimensions, and their representation as a set of MDDs, we report in this section the mechanism that allows the execution of queries over the Data Lake. Before delving into the discussion of the query mechanism, we will introduce some preliminary definitions devoted to formalise how the indicators, defined in the Smart City Exploration Graph, can be related to the concepts and relationship in the semantic overlay by means of *mappings*.

3.5.1 Mappings

In our model, mappings enable to identify how indicators can be calculated in terms of Data Lake sources attributes, represented in the overlay by concepts associated with SAAs and relationships between them. To link the definition of an indicator in the Smart City Exploration Graph to its counterpart in terms of concepts and relationships of the semantic overlay, we devise a formalisation inspired by the work in [49]. According to the former work, and adapting the definition to our model, a mapping is formalised as follows.

Definition 10 (Mapping) *A mapping* map_i *is a triple* $\langle id_i, t_i, expr_i \rangle$ *where: (a)* id_i *is the unique identifier of the mapping; (b)* t_i *is the type of the mapping and (c)* $expr_i$ *is an expression associated with the mapping, formalised according to the Description Logic syntax, containing only subsumption (*$\sqsubseteq$*) and property composition (*$\circ$*) binary operators.*

Using the notation in [49], given a mapping map_i, the related expression $expr_i$ can be exploited to:

- map a concept $c_i \in \mathcal{G}$ to a concept in the semantic overlay ($t_i = concept$);
- map a relationship in $r_i \in \mathcal{G}$ to one or more relationships in the semantic overlay ($t_i = relationship$). In this case, the property composition binary operator $\circ$ can be used to create a *property chain*, thus mapping r_i to a composition of properties $r_1 \circ r_2 \circ \ldots \circ r_n$ in the semantic overlay.

Mappings are exploited by data analysts to create the *mappings set* for an MDD τ_i, to ensure subsequent exploration. In other words, the mappings set for τ_i is apt to identify the portion of the semantic overlay containing the definition of the indicator in terms of concepts and relationships of the semantic overlay. In the following, we provide the definition for a *single MDD mappings set*, involving the indicator ind_i and the dimensional levels in the vector L_i.

Definition 11 (Single MDD mappings set) *Given a MDD τ_i, the associated mappings set is denoted with $\Sigma(\tau_i)$ and it is composed of three elements*

$$\Sigma(\tau_i) = \langle map^{ind_i}, M^{L_i}, M^{rel} \rangle$$

where: (i) map^{ind_i} represents a mapping towards semantic overlay concept for ind_i; (ii) M^{L_i} is a set containing mappings for dimensions in L_i; (iii) M^{rel} is a set containing mappings for the relationships connecting ind_i with each dimension in L_i in the Smart City Exploration Graph.

As previously mentioned, the creation of mappings is in charge of data analysts. For a MDD τ_i, the data analyst follows these procedure to create the mappings set $\Sigma(\tau_i)$:

1. starting from the indicator ind_i she creates the respective mapping map^{ind_i} towards a concept in the semantic overlay;
2. for each dimensional level in L_i, she creates the respective mapping $map^{l_i^k}$ towards a concept in the semantic overlay (all these mappings are retained in the set M^{L_i});
3. lastly, she maps each relationship connecting ind_i with a dimensional level in L_i to its counterpart in the semantic overlay, in order to re-create the link between the indicator and the dimensional level present in the Smart City Exploration Graph in terms of the concepts and relationships in the semantic overlay. This assures *answerability* when exploring ind_i values along dimensions in L_i (i.e., there exist paths connecting the indicator to its dimensions).

Henceforth, in the examples, we will use the notation $\texttt{object}^{EG}$ for referencing a concept (or relationship) belonging to the Smart City Exploration Graph, whereas $\texttt{object}^{O}$ for a concept (or relationship) belonging to the semantic overlay.

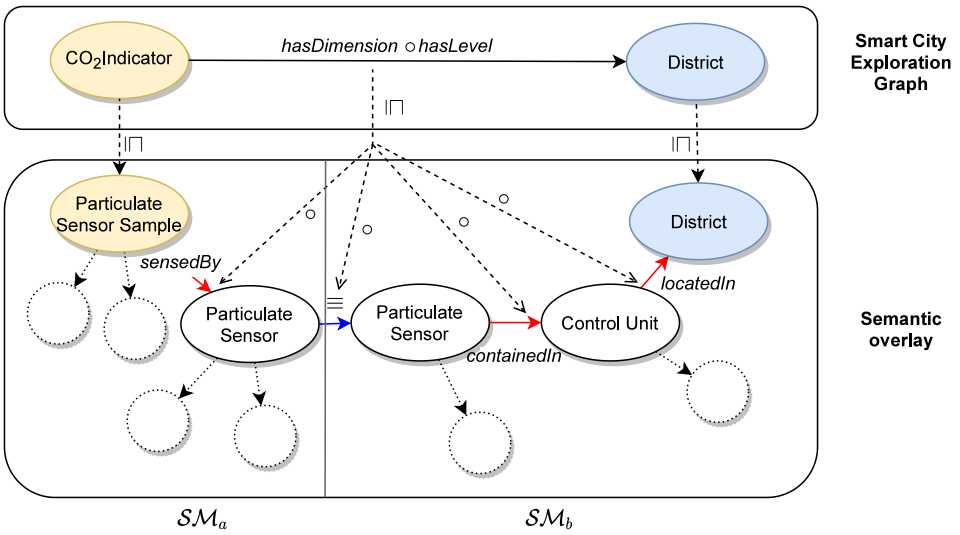


Figure 3.6: Example of mappings for the indicator `CO2Indicator`, the `District` dimensional level and the relationship connecting them in the Smart City Exploration Graph towards the semantic overlay. Concepts and relationships in the semantic overlay are from Semantic Models $\mathcal{SM}_a$ and $\mathcal{SM}_b$.

Example 9 *Let us consider the example reported in Figure 3.6. The first mapping to be created involves the indicator and dimension level concepts (resp.,* $\texttt{CO2Indicator}^{EG}$ *and* $\texttt{District}^{EG}$*), expressing that* $\texttt{CO2Indicator}^{EG}$ *is mapped (using the sub-class relationship* $\sqsubseteq$*) to* $\texttt{ParticulateSensorSample}^{O}$ *and* $\texttt{District}^{EG}$ *to* $\texttt{District}^{O}$*. Secondly, also the relationship connecting the indicator with its dimensional level in the Smart City Exploration Graph has to be mapped. Referring to the example, the relationship* $\texttt{hasDimension}^{EG} \circ \texttt{hasLevel}^{EG}$ *is defined as a sub-property of the composition of four different relationships of the semantic overlay, forming a property chain:* $\texttt{sensedBy}^{O} \circ \equiv^{O} \circ \texttt{containedIn}^{O} \circ \texttt{locatedIn}^{O}$.

Noteworthy, it may happen that, for monitoring a phenomenon of interest, multiple MDDs should be considered for exploration at the same time, in order to obtain a comprehensive view of the observed phenomenon, at different granularity (e.g., pollution levels at district, city and region level). When considering a subset of MDDs $\mathcal{T}_k \subseteq \mathcal{T}$, the data analyst has to provide additional mappings for `rollUp` relationships, which are exploited to relate each other levels belonging to the same dimension. To this aim, we formalise the definition of *multiple MDDs mappings set* as follows.

Definition 12 (Multiple MDDs mappings set) *Given a set of MDDs* $\mathcal{T}_k \subseteq \mathcal{T}$*, the mappings set for* $\mathcal{T}_k$ *is denoted as* $\Sigma(\tau_1, \tau_2, \ldots, \tau_n)$*, with* $n = |\mathcal{T}_k|$*, and it is*

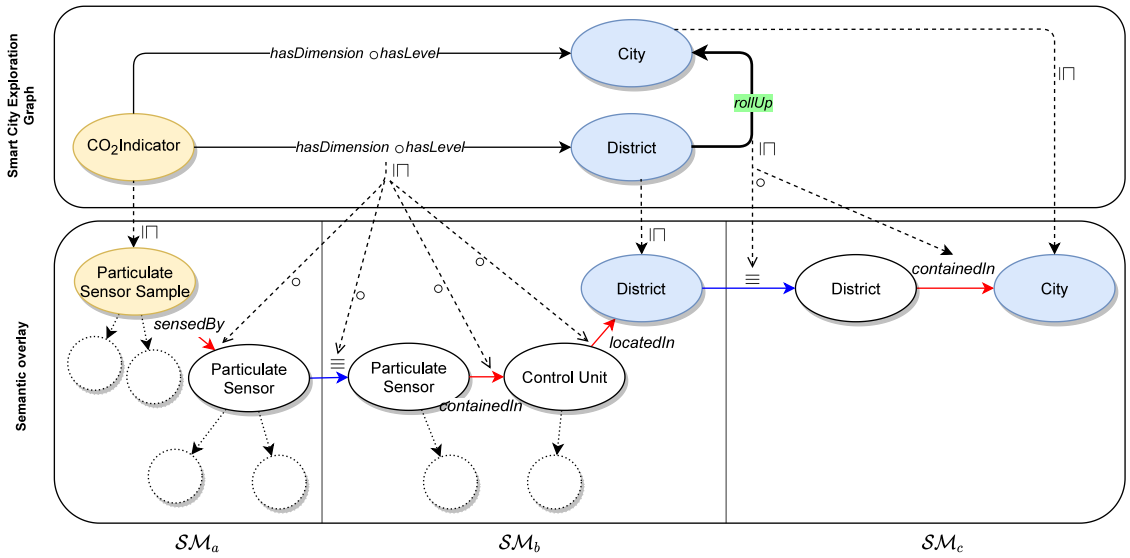


Figure 3.7: Details for the mapping required for the roll-up relationship between $\texttt{City}^{EG}$ and $\texttt{District}^{EG}$ concepts, in order to summarise indicator values along the spatial dimension.

formalised as:

$$\Sigma(\tau_1, \tau_2, \ldots, \tau_n) = \langle \Sigma(\tau_1), \Sigma(\tau_2), \ldots, \Sigma(\tau_n), M^{rollUp} \rangle$$

where: (i) $\Sigma(\tau_1), \Sigma(\tau_2), \ldots, \Sigma(\tau_n)$ *are the mappings sets associated with MDDs in* $\mathcal{T}_k$*; (ii)* M^{rollUp} *is a set containing mappings for roll-up relationships between dimensional levels concepts.*

Example 10 *In Figure 3.7, assuming to focus only on the* `Spatial` *dimension, there are two MDDs referred to the same indicator,* $\tau_1 = \langle \texttt{CO2Indicator}, (\texttt{City}) \rangle$ *and* $\tau_2 = \langle \texttt{CO2Indicator}, (\texttt{District}) \rangle$*. In the figure, it is reported also the mapping involving the* `rollUp` *relationship between* `District` *and* `City`*. As can be seen from the figure, the mapping process involves three distinct Semantic Models, namely* $\mathcal{SM}_a$*,* $\mathcal{SM}_b$ *and* $\mathcal{SM}_c$*.*

Given a MDD τ_c, where $\tau_c.ind_c$ is a *composite* indicator, Definition 12 can be also exploited to express mappings for τ_c. Indeed, denoting with $\mathcal{T}_c = \{\tau_k | ind_c \texttt{ takesDataFrom } ind_k \ \wedge \ L_k \equiv L_c\}$ the set containing the MDDs related to the composing indicators of $\tau_c.ind_c$, sharing with $\tau_c.ind_c$ the same dimensional levels vectors, the mappings set for τ_c will correspond to the multiple MDDs mapping set $\Sigma(\tau_{c_1}, \ldots, \tau_{c_n})$, where $\tau_{c_1}, \ldots, \tau_{c_n} \in \mathcal{T}_c$. Remarkably, for $\Sigma(\tau_{c_1}, \ldots, \tau_{c_n})$, $M^{rollUp} = \emptyset$, since dimensional levels vectors in $\mathcal{T}_c$ are the same.

3.5.2 Exploration procedure overview

Roughly speaking, the data layers of our model (in top-down order, the Smart City Exploration Graph, the semantic overlay and Data Lake sources) needs to be somehow interconnected for processing complex analysis queries, targeted to the exploration of indicators values. The exploration procedure at the basis of our model is articulated over three macro-steps, briefly summarised in the following.

1) The user selects one MDD τ_i from the set $\mathcal{T}$ of available ones, to start the exploration.

2) In the selected τ_i, ind_i concept is the indicator whose values must be calculated according to the dimensions in the vector L_i, representing the analysis dimensions for ind_i. We adopt the notation Q_{τ_i} to stress the fact that τ_i is used to issue a *query* over the Data Lake.

   Q_{τ_i} circumscribes a portion $\mathcal{O}(Q_{\tau_i})$ of the semantic overlay $\mathcal{O}$ (Definition 7), containing the concepts associated with Semantically Annotated Attributes (SAAs) and relationships between them, required to calculate ind_i according to the dimensions in L_i. The aforementioned portion of the semantic overlay can be identified by exploiting *mappings*, created by the data analyst between indicator/dimensions concepts in τ_i and concepts and relationships in the semantic overlay, constituting the *mappings set* $\Sigma(\tau_i)$.

3) Over $\mathcal{O}(Q_{\tau_i})$, the methodology proposed in [29] by Ben Hamadou et al. is applied, in order to derive the query plan associated with Q_{τ_i}. The creation of the query plan for Q_{τ_i} is divided into two steps: (i) creation of single *local plans* (meant for retrieving data sets of single Data Lake sources); (ii) creation of the *global plan* (to join data sets coming from different data sources).

In the next sections we will describe the query model, based on a MDD τ_i (Step 2), and the creation of the global query plan to provide actual indicator values for Q_{τ_i} (Step 3). We will not discuss here on how the user can be suggested with the most promising MDDs to start the exploration from, as this aspect, related to Step 1, will be deeply covered by Chapter 4.

3.5.3 Query model

In the previous section, we introduced the formalism apt to create mappings from the Smart City Exploration Graph (represented as a set of MDDs) and concepts and relationships in the semantic overlay, constituting the mappings sets for the MDDs. According to the exploration procedure overview, the exploration of the

Semantic Data Lake starts from the selection of a MDD τ_i, which is employed to issue a query Q_{τ_i} over the Semantic Data Lake. From a higher viewpoint, the query model being presented in this section is adapted from the proposal of [29]. It focuses on the answering of specific types of queries, the so-called GPSJ queries [26] (that is, Generalised Projection, Selection and Join), which are the most common class of queries employed to perform OLAP-like analysis and, thus, are suitable also for the MDDs.

Definition 13 (Query) *A query over the Semantic Data Lake achieved through a MDD τ_i, denoted as Q_{τ_i}, is defined as:*

$$Q_{\tau_i} = \langle q_\pi, \gamma, q_\sigma, f \rangle$$

where: (i) q_π is a set containing the projection objects of the query; (ii) γ is an aggregation function to be applied to the results of the query; (iii) q_σ is a set containing the selection objects of the query and (iv) f is a formula to be applied on the indicator(s) in q_π.

For a query Q_{τ_i}, where ind_i is a simple indicator, these assumptions hold:

- The set $q_\pi = \{ind_i\} \cup \{l_i^{j=1,\ldots,|L_i|}\}$ contains the indicator concept ind_i of τ_i and the dimensional level concepts in the vector L_i. Moreover, there exists the mappings set $\Sigma(\tau_i)$ for τ_i. The content of q_π (limited to the dimensional levels) corresponds to the *group by* set of Q_{τ_i}.
- The aggregation function γ for the indicator ind_i is obtained from the Smart City Exploration Graph $\mathcal{G}$, that is ind_i `hasAggregationFunction` γ. Specifically, this function is the default operation to apply when aggregating ind_i values according to the dimensional levels in q_π.
- The set q_σ contains filtering predicates for the dimensional levels in the set q_π, which can be eventually specified at query time. Here, we assume that this set is implicitly populated with information retained in the profile of the user issuing the query, aimed at restricting the access to Smart City data (for instance, a building administrator is allowed to inspect indicators for the administered buildings only). At a high level of abstraction, selection predicates in q_σ are triplets in the form $\langle l_i^j, op, val \rangle$, where: (a) l_i^j is a dimensional level from q_π; (b) *op* is a binary comparison operator (e.g., $=$, $>$, $<$) and (c) *val* is a value.
- The formula f is applied on the indicator ind_i in q_π. Specifically, such formula (if defined by the data analyst) is contained in the Smart City Exploration Graph, that is ind_i `hasFormula` f in $\mathcal{G}$.

The former assumptions are still valid in the case ind_i is a composite indicator, but the formula f will express how its composing indicators (asserted in $\mathcal{G}$ through the `takesDataFrom` semantic relationship) have to be composed together. Moreover, the set q_π of Q_{τ_i} will contain all the composing indicators of ind_i, and the query answering process will leverage the mappings sets defined for each composing indicator of ind_i.

Example 11 *Let us consider this sample MDD*

$$\tau_1 = \langle \texttt{CO2Indicator}, (\texttt{District}) \rangle$$

and the related query

$$Q_{\tau_1} = \langle \{\texttt{CO2Indicator}, \texttt{District}\}, \texttt{Sum}, \emptyset, \texttt{CO2Indicator}/1000 \rangle$$

Resorting to the structure of the sample portion of $\mathcal{G}$ depicted in Figure 3.4, `CO2Indicator` *is a simple indicator (i.e., with no dependencies). The aggregation function γ (*`Sum`*) is obtained by navigating the semantic relationship* `hasAggregationFunction` *in $\mathcal{G}$. Moreover, the formula f_1 for the indicator is* `CO2Indicator`$/1000$*, to be applied before the aggregation function γ, is contained in $\mathcal{G}$. Concerning the set $q_\pi = \{\texttt{CO2Indicator}, \texttt{District}\}$ of Q_{τ_1}, we assume in this example that it exists the mappings set $\Sigma(\tau_1)$ involving the indicator and the dimensional level in q_π.*

Example 12 *Let us consider another MDD, regarding a composite indicator*

$$\tau_2 = \langle \texttt{ParticulateMatterIndicator}, (\texttt{District}) \rangle$$

and the related query

$$Q_{\tau_2} = \langle \{\texttt{ParticulateMatterIndicator}, \texttt{District}\}, \texttt{Avg}, \emptyset, f_2 \rangle$$

The formula f_2 contains the expression

$$\texttt{PartMatter10Indicator} + \texttt{PartMatter25Indicator} + \texttt{UFPIndicator}$$

With respect to the previous example, the formula f_2 states that `ParticulateMatterIndicator` *can be obtained by summing the values of three simple indicators, containing the values of PM_{10}, $PM_{2.5}$ and ultra-fine particulate (UFP). Also in this case, the aggregation function γ (*`Avg`*) will be applied on the summation results. Instead, the set q_π of Q_{τ_2} will be* {`PartMatter10Indicator`, `PartMatter25Indicator`, `UFPIndicator`, `District`}.
The mappings set for the query Q_{τ_2} corresponds to the multiple MDDs mappings set of the three simple indicators in f_2, denoted with $\Sigma(\tau_{2_{PM10}}, \tau_{2_{PM25}}, \tau_{2_{UFP}})$, as defined in the end of Section 3.5.1.

3.5.4 Query execution plan

3.5.4.1 Ben Hamadou's approach in a nutshell

In our approach, we resort to the methodology proposed by Ben Hamadou et al. in [29] to create a query plan aimed at answering queries when dealing with heterogeneous data sources. In [29], data sources in an heterogeneous context are explored through the employment of a *dataspace*. The aim of a dataspace is to set up a query answering mechanism to query heterogeneous data sources by defining an *execution plan*, which spans different data sources of a polystore infrastructure. The key idea is to leverage the abstraction of data sources attributes and mutual relationships in the *dataspace* as a way to access data, facing heterogeneity issues. In this respect, the execution plan for a query is composed of single *local plans*, which can be executed directly on a single data source, and of a *global plan*, joining data coming from different data sources. The information necessary for the creation of the execution plans is contained within a supporting data structure called *datagraph*, whose nodes represent data source attributes whereas edges indicate relationships between attributes (i.e., if they are either sibling attributes, nested attributes, foreign keys). Roughly speaking, the datagraph is exploited to find the connections between data sources attributes and to obtain the execution plan for a given query. A query issued over the dataspace tailors a portion of the datagraph (referred to as *query graph*), which is the starting point for deriving local plans and the global plan to be issued over the data sources. In the context of this thesis, we do not foster a dataspace as conceived by [29], but we adapt the methodology proposed by Ben Hamadou et al. to our three-layered model, as it will be explained in the next sections. As already remarked in the introduction of this chapter, although the approach in [29] has been deployed in a polystore architecture, where data sources are accessed through their local language, its principles and the methodology aimed at deriving the query graph and the execution plan for a given query Q_{τ_i} can be adapted to the Semantic Data Lake context of this thesis, in order to retrieve and join data sets of different data sources. The steps of the methodology of [29] are graphically summarised in Figure 3.8. In the figure, there are also highlighted the steps which had to be adapted in order to foster the aforementioned methodology in our model.

3.5.4.2 Application of Ben Hamadou's approach to our three-layered model

After having provided a brief overview of the approach proposed in [29], we explain in this section its application to our three-layered model, in order to retrieve from the Semantic Data Lake actual values of indicator ind_i, aggregated according to dimensions in L_i of Q_{τ_i}.

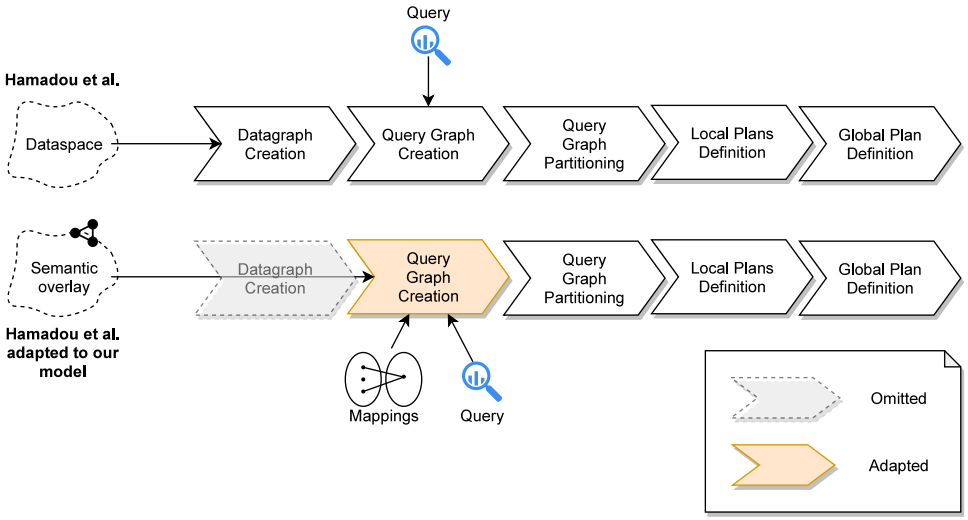


Figure 3.8: Stepwise representation of Ben Hamadou's approach. The lower part of the image depicts how the steps of [29] are adapted to be employed in our three-layered model.

Query graph creation. In [29], the query graph containing the required information for creating the execution plan is obtained from a datagraph, in turn derived from the dataspace. In our three-layered model, we start from the semantic overlay $\mathcal{O}$, containing concepts associated with Semantically Annotated Attributes and relationships between them. Due to the fact that the semantic overlay may contain a lot of concepts and relationships (indeed, as long as novel data sources are added to the Data Lake, the number of Semantic Models in the semantic overlay increases accordingly) we focus directly on the creation of the query graph.
The query graph creation process for Q_{τ_i} is detailed in Algorithm 2. The algorithm requires as input the portion of the semantic overlay, denoted as $\mathcal{O}(Q_{\tau_i})$, identified by the mappings contained in the mappings set $\Sigma(\tau_i)$. The output of the algorithm is the query graph $G_{Q_{\tau_i}}$, composed of a set of nodes N and edges E. Nodes in N are the attributes of data sources, whose annotating concepts belong to $\mathcal{O}(Q_{\tau_i})$ (line 4). Conversely, each edge $e_h \in E$ is a triple $\langle src, dest, type \rangle$ derived from the corresponding semantic relationship r_h in $\mathcal{O}(Q_{\tau_i})$. Specifically, $e_h.src$ is the source attribute of the edge, derived from the domain of r_h (line 9) and $e_h.dest$ is the target attribute of the edge, derived from the range of r_h (line 10). To determine the type of the edge $e_h.type$, the type of the relationship r_h is inspected. If r_h is an inter-source relationship (i.e., connecting SAAs of different data sources) the edge type is set to *fk* (lines 12-13). Otherwise, if r_h is an intra-source relationship, there are two possible cases. If the metadata set of r_h (denoted with M_h) contains the pair $\langle isIntraFK, \texttt{True} \rangle$, the edge type is marked as *intra fk* (i.e., the edge links two

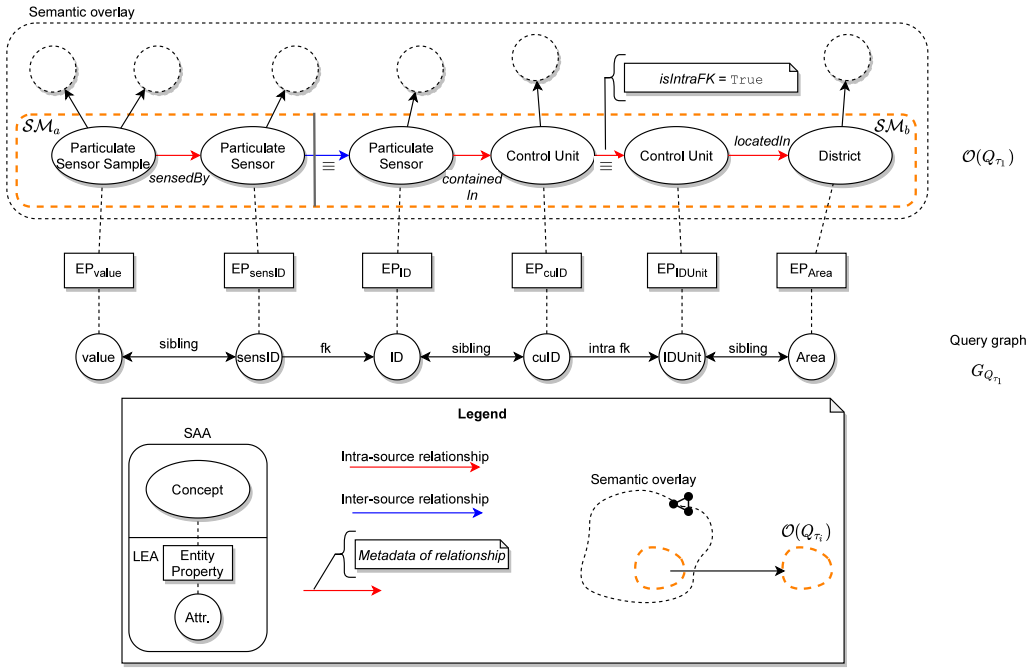


Figure 3.9: Representation of the portion $\mathcal{O}(Q_{\tau_1})$ obtained considering a query Q_{τ_1} and related query graph $G_{Q_{\tau_1}}$.

attributes belonging two different data sets in the same data source, as it happens for data sets corresponding to tables in a relational database) or *sibling*, that is attributes belonging to the same data set (lines 15-18). Lines 21-22 ensure that *sibling* edges are treated as bi-directional. With respect to the edge types in [29], we introduced the edge type *intra fk* (in Ben Hamadou's approach, they are simply referred to as *fk*, making no distinction).

Example 13 *Considering the MDD τ_1 in Example 11, Figure 3.9 shows the portion of the semantic overlay (along with the Semantic Models involved, that is $\mathcal{SM}_a$ and $\mathcal{SM}_b$) for the query Q_{τ_i} (denoted as $\mathcal{O}(Q_{\tau_i})$), identified by considering the mappings set $\Sigma(\tau_1)$ (defined upfront by the data analyst). Each attribute annotated by a concept in $\mathcal{O}(Q_{\tau_i})$, becomes a node in the query graph $G_{Q_{\tau_i}}$, whereas edges between nodes are constructed according to Algorithm 2, starting from semantic relationships in $\mathcal{O}(Q_{\tau_i})$.*

Once the query graph $G_{Q_{\tau_i}}$ has been derived as previously explained, it is translated into a NRA (Nested Relational Algebra) execution plan. Such translation mechanism is articulated over three pivotal phases: (i) partition of $G_{Q_{\tau_i}}$ into several sub-graphs, each of them corresponding to a local query plan targeted to a specific data source in the Data Lake; (ii) definition of each local plan (i.e., identifying data sets within

Algorithm 2: Query graph creation

Input : Portion of the semantic overlay identified by the selected MDD $\mathcal{O}(Q_{\tau_i})$

Output: Query graph $G_{Q_{\tau_i}}$

```
# Initialisation
G_{Q_τi} ← (N ← ∅, E ← ∅);
# Nodes of G_{Q_τi} are attributes of data sources
N ← {SAA_j^k.LEA_j^k.a_j^k | SAA_j^k.c_j^k ∈ O(Q_τi)};
# Edges in G_{Q_τi} are derived from relationships in the portion of the semantic overlay
foreach r_h ∈ O(Q_τi) do
    # Edges in E are triples ⟨src, dest, type⟩
    e_h ← ⟨NULL, NULL, NULL⟩ ;
    e_h.src ← SAA_j^k.LEA_j^k.a_j^k | SAA_j^k.c_j^k ≡ domain(r_h);
    e_h.dest ← SAA_j^k.LEA_j^k.a_j^k | SAA_j^k.c_j^k ≡ range(r_h);
    # Type assignment according to the type of r_h
    if t_h = inter then
        e_h.type ← fk;
    else
        if ⟨isIntraFK, True⟩ ∈ M_h then
            e_h.type ← intrafk;
        else
            e_h.type ← sibling;
    E ← E ∪ e_h;
    # Sibling edges are bi-directional
    if e_h.type = sibling then
        E ← E ∪ ⟨e_h.dest, e_h.src, e_h.type⟩;
return G_{Q_τi};
```

data sources to be accessed, organising access, selection, projection and so forth); (iii) creation of the global plan joining local plans. As a final remark, in the case ind_i in τ_i refers to a composite indicator, the portion of $\mathcal{O}$ input of Algorithm 2 will be tailored by taking into account the mappings sets associated with all the composing indicators of ind_i. Similarly, when considering a subset $\mathcal{T}_k \subseteq \mathcal{T}$ of MDDs, the input of the algorithm will be the portion of $\mathcal{O}$ identified through the multiple MDDs mappings set $\Sigma(\tau_1, \tau_2, \ldots, \tau_n),\ n = |\mathcal{T}_k|$.

Query graph partitioning. To obtain local query plans, the query graph $G_{Q_{\tau_i}}$ undergoes partitioning based on the edges of type *fk* (i.e., linking attributes belonging to different data sources). Specifically, the number of local plans (and inherently of partitions) is equal to $|E_{fk}| + 1$, where E_{fk} denotes the set of edges in $G_{Q_{\tau_i}}$ whose type is *fk*. Noteworthy, as remarked in [29], if two edges in E_{fk} refer to the same data source, this will determine the creation of two local plans to access the data sets of the data source. With respect to [29], in this thesis we adopt the following naming convention: (i) *macro-partitions* are portions of $G_{Q_{\tau_i}}$ induced by *fk* edges; (ii) within macro-partitions, *intra fk* edges (if present) induce *sub-partitions*.

Example 14 *With reference to Figure 3.10, it can be noticed that the query graph $G_{Q_{\tau_i}}$ for the sample MDD τ_1 has been partitioned into two macro-partitions G_a and G_b, due to the fact there exists only an fk edge. Since G_b contains an intra fk edge, other two sub-partitions are identified, namely $G_{b,1}$ and $G_{b,2}$. Roughly speaking, graph partitions reflect the number of accesses to data sources to retrieve data sets. In the example, the number of data sources access will be three, one for $\mathcal{S}_a$ and two for $\mathcal{S}_b$, that is $|E_{fk}| + |E_{ifk}| + 1 = 3$ access are required.*

Local plans and global plan definition. After partitioning of $G_{Q_{\tau_i}}$ has been performed, for each partition of the query graph the corresponding local plan has to be created. Roughly speaking, each local plan consists of an expression (which can be graphically represented as a tree, as in Figure 3.11) in Nested Relational Algebra (NRA), containing the operators data sets access (DS), projection (π), selection (σ) and join ($\bowtie$). In Ben Hamadou's approach, NRA is fostered as it is apt to query both relational databases and document-oriented stores, being in the context of a polystore system, where each data source is accessed through its local language. In this respect, it is still compatible with the abstraction of Data Lake sources content as data sets. Indeed, NRA is a super-set of relational algebra (since it considers also nest and unnest operators). Then, each local plan expression is combined in order to obtain the global plan expression. The creation of local and global plans adheres to the translation steps formalised in [29], starting from the query graph partitions and leveraging the former operators.
With respect to [29], we introduce the data sets access operator DS (instead of the collection access operator since, in Ben Hamadou's approach, the minimal access unit is the so-called collection) and we do not consider the unnest operator, descending from the adoption of *nested* edges in the query graph. In particular, with reference to our case and regarding the overall NRA expression combining the NRA expressions of single local plans: (i) the number of DS operators depends on the number of partitions of the query graph, induced by the presence of both *fk* and *intra fk* edges (i.e., recalling the previous step, the number of DS operators is equal to $|E_{fk}| + |E_{ifk}| + 1$); (ii)

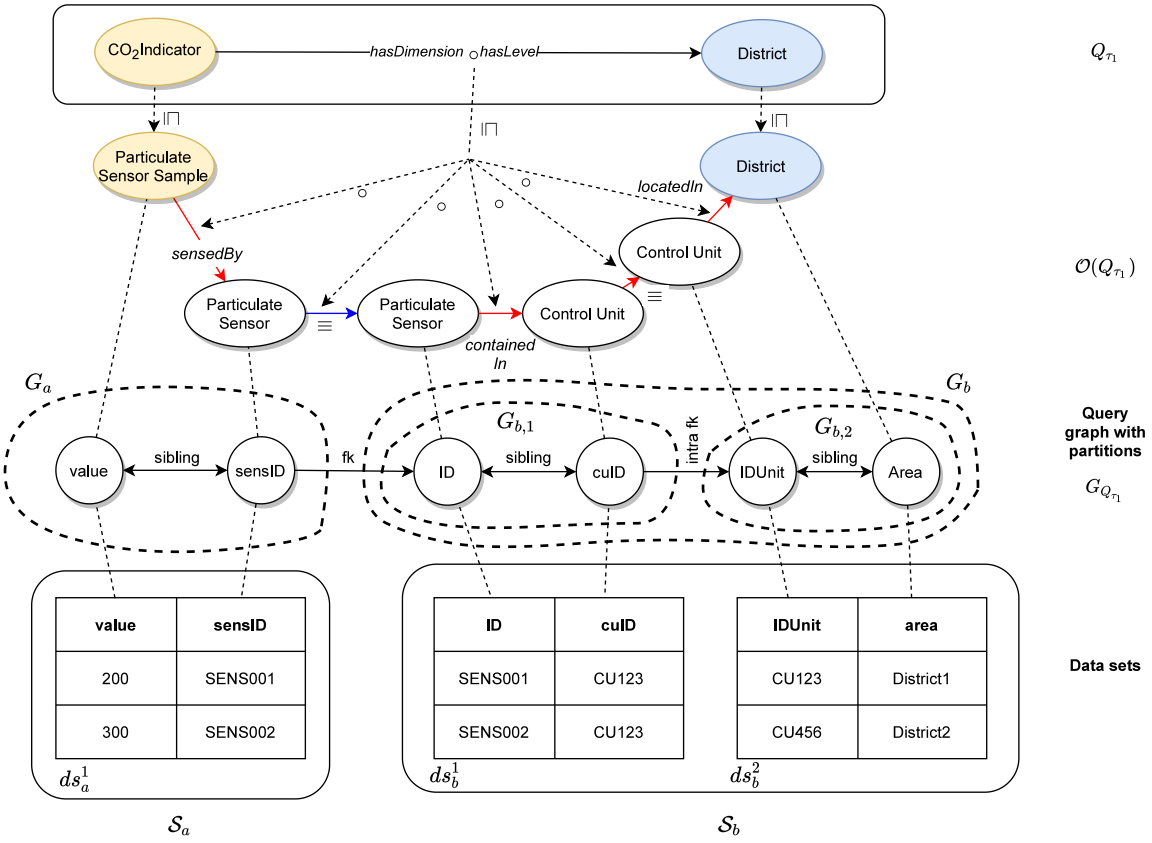


Figure 3.10: Overview of the exploration procedure for the example query Q_{τ_1}.

projections (π) involve both attributes required for join operations (due to *fk* and *intra fk* edges) plus the ones of the q_π set of Q_{τ_i}; (iii) if the set q_σ of Q_{τ_i} is not empty, selection operators (σ) are added according to the predicates contained in q_σ. We remark that joins ($\bowtie$) are added for each *intra fk* edge (within local plans) and for each *fk* edge (for assembling the global plan, combining local plans). Only as a last step, the attributes involved in the projection set q_π of the query Q_{τ_i}, the formula f and the aggregation function γ are applied on the attributes. In the global NRA expression, if the metadata sets of relationships generating *fk* and *intra fk* edges contain transformation functions, then they are applied on join clauses. At this point, the NRA expression can be translated into an SQL-like statement to be executed within a query environment handling heterogeneous data sources.

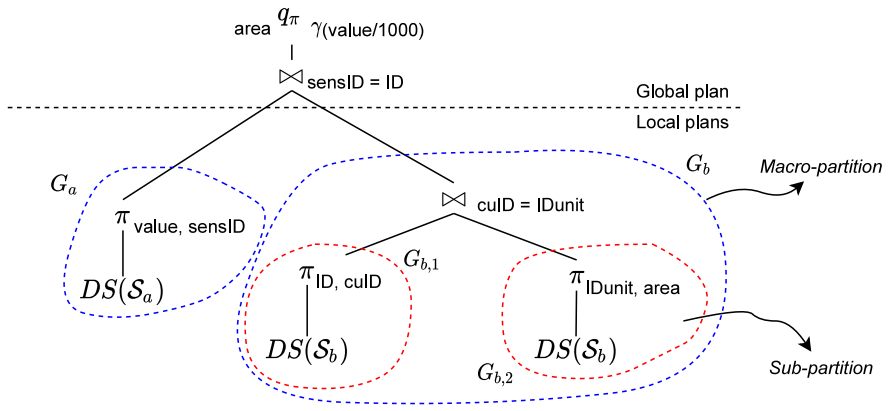


Figure 3.11: NRA tree for the example. In the figure, local plans and global plan are distinguished. Dashed circles delimit local plans associated with specific sub-graphs of $G_{Q_{\tau_1}}$.

3.6 Implementation

In this section we give a brief description about the modelling environment employed by data analysts to create indicators and, from a general viewpoint, maintain the Smart City Exploration Graph, being supported by the validation rules outlined in Section 3.2. Moreover, we will also present the prototype implementation of the query answering process implementing the functionalities of the `QUERY PROCESSOR` and `QUERY MANAGER` modules of Figure 1.1, devoted to query Data Lake sources.

3.6.1 Indicators modelling

We focus here on the indicators modelling task, performed by data analysts, being supported by the validation rules described in Section 3.2.4, thus easing both the creation of novel indicators and the modification of the structure of the Smart City Exploration Graph. Specifically, a data analyst may rely on the validation support according to two different strategies: (a) *during* the modelling of indicators or (b) at the *end* of the modelling task. Moreover, the classification and statistics about errors committed in the modelling process with/without validation rules will be provided.
The indicators design methodology is entirely carried out by exploiting the renowned ontology framework Protégé[4], containing a fully-fledged editor, also available as a Web-based application [67], offering collaboration functionalities, such as the tracking of the changes made to files. Specifically, the design of indicators is tackled by the steps detailed in the following.

[4] `https://protege.stanford.edu/`

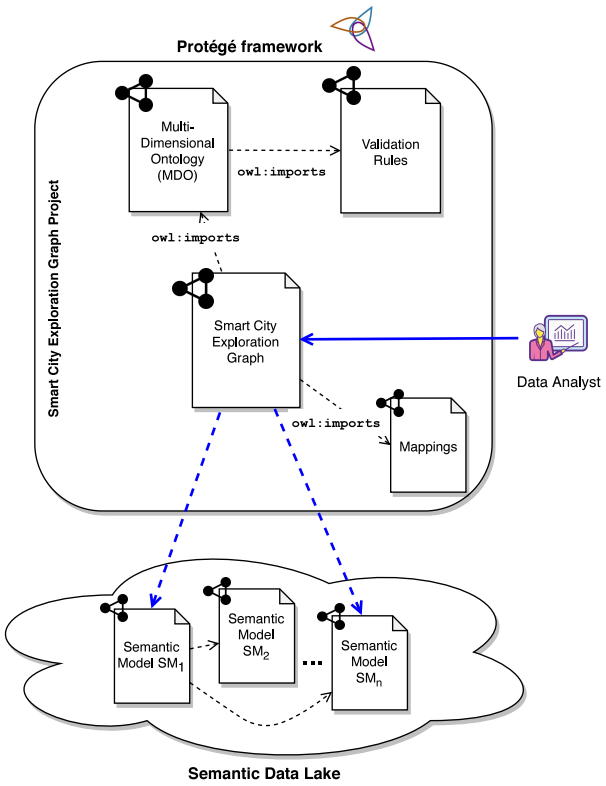


Figure 3.12: Structure of the OWL files used within the Protégé framework.

Access to the Smart City Exploration Graph project. The first step performed by the data analyst is accessing the OWL file containing the Smart City Exploration Graph. This file references the OWL file (through the `owl:imports` clause) of the Multi-Dimensional Ontology, as it contains the baseline semantic concepts and relationships concerning the modelling of Smart City indicators. In turn, the latter refers the file containing the knowledge apt to assist the data analyst throughout the design process (that is, the definition of validation concepts, implementing the validation rules described in Section 3.2.4, which are imported as a separate file, in order to assure the highest level of flexibility, as reported in Figure 3.12).

Design of Smart City indicators. In this step, the data analyst operatively creates concepts and relationships to model indicators and their background knowledge, as subclasses of MDO concepts. For concepts, this is achieved by clicking on the hierarchical tree under the tab "Classes" and then selecting "Add sub-class", whereas for relationships entering the tab "Object Properties" and then selecting "Add sub-

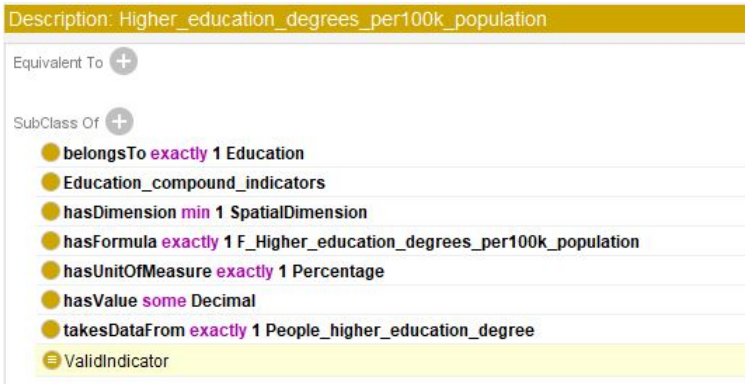


Figure 3.13: Example of valid indicator classified as such by the reasoner.

property". To check the soundness of created indicators, the data analyst runs the reasoner included in Protégé, from the dedicated drop-down menu "Reasoner - Start Reasoner". For instance, focusing on indicators concepts, valid indicators will be classified by the reasoner also under the `ValidIndicator` class (and analogously for dimensional hierarchies and activities). This information is highlighted within the editor, and marked as inferred (derived) knowledge, as illustrated in Figure 3.13. After these checkpoint operations, the data analyst commits the changes made to the overall OWL file.

Creation of mappings for Smart City indicators. Lastly, for the portion of designed knowledge (in particular, for indicators, dimensional levels and relationships between them) the data analyst has to create the mappings to contextualise the indicators, thus creating the mappings sets of indicators, specifying the mapping towards concepts and relationships of the semantic overlay. The creation of the mappings adheres to the methodology sketched in Section 3.5.1. Indeed, mappings are assertions involving concepts and relationships of the semantic overlay, referenced through their namespace, and the creation of property chains (i.e., composition of semantic relationships) is made leveraging the "Sub-property of (chain)" tab in the "Object Properties" view (Figure 3.14). In this respect, it is worth remarking that in Protégé, due to its configuration, to create the mapping for the relationship connecting an indicator to one of its dimensional levels, there is the need of creating upfront a placeholder relationship r_p defined as $r_p \sqsubseteq$ `hasDimension` $\circ$ `hasLevel`. Figure 3.15 provides an example of a generic placeholder relationship named `hasDimensionalLevel`, which is defined as `hasDimension` $\circ$ `hasLevel`. Lastly, complying with the design principle of modularisation, assertions related to the creation of mappings are maintained in a separate OWL file.

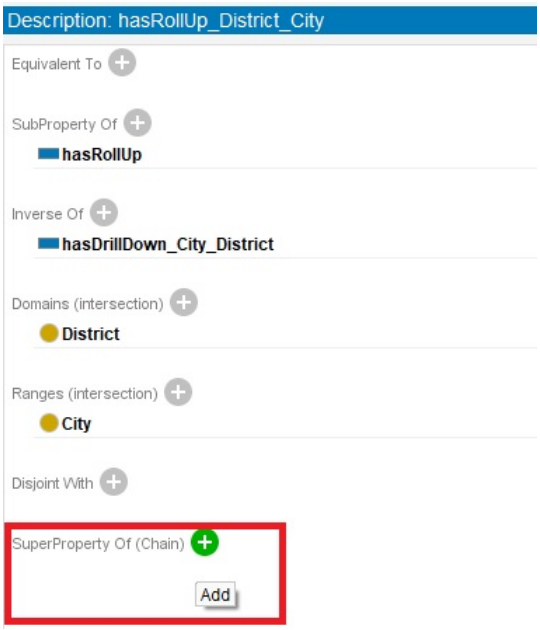


Figure 3.14: Detail of property panel of Protégé for property chain specification.

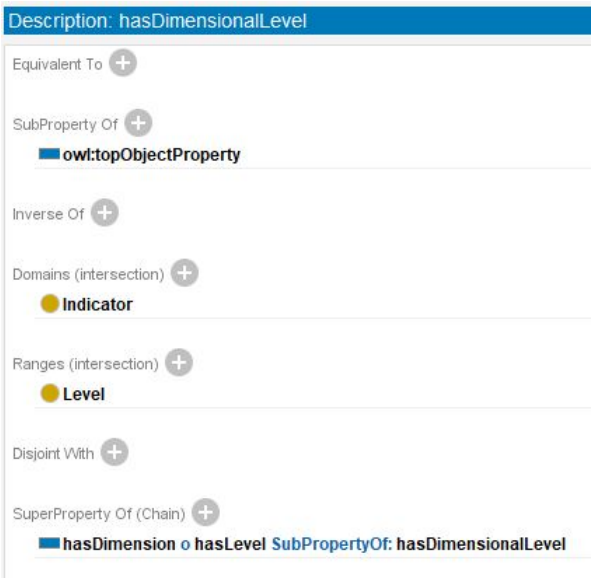


Figure 3.15: Generic example of property placeholder for the relationship between an indicator and one of its dimensional levels.

3.6.2 Query management

In this section, we detail the implementation of the `QUERY PROCESSOR` and `QUERY MANAGER` modules of the reference model depicted in Figure 1.1, devoted to accomplish the query answering process formalised in the previous sections. The `QUERY MANAGER` module is implemented in Java and it is grounded on the Apache Spark SQL[5] module functionalities. It is in charge of steering the overall query answering process, considering one or more MDDs chosen by the user, the mappings from the Smart City Exploration Graph towards the semantic overlay, and the Semantically Annotated Attributes (SAAs) built upon Data Lake sources. On the other hand, the `QUERY PROCESSOR` offers common and seamless primitives to access to Data Lake sources and it is here implemented through the Apache Spark[6] framework. The choice of adopting Apache Spark (and, inherently, of the Apache Spark SQL module) descends from the fact that it has been effectively employed in contexts with heterogeneous data sources and, additionally, Spark offers easy-to-use APIs for operating on large datasets, assuring an unified engine where libraries can be seamlessly combined to create complex workflows.

3.6.2.1 Architecture overview for the query process

Figure 3.16 represents the reference Data Lake infrastructure exploited for the experiments regarding the query process, leading to the calculation of indicators. Numbers in the figure give the order of the interaction flow between the modules. We assume here that data sources ingestion process has been performed into an Hadoop File System (HDFS), and that each relational table or file ingested in HDFS has its own specific path, stored as metadata in the DL-DIVER back-end database. After the selection of an MDD τ_i by the user (1), the `QUERY MANAGER` module is in charge of assembling the respective query Q_{τ_i}. To this aim, the mappings set $\Sigma(\tau_i)$, apt to identify the portion of the semantic overlay containing the Semantically Annotated Attributes required for answering Q_{τ_i}, is retrieved by inspecting the DL-DIVER back-end database (2). Therein, proper tables and views contain information about the mappings towards semantic overlay objects (materialised through the OWL API Module [30], implementing functions for manipulating the OWL file of mappings, which has been described in Section 3.6.1). Moreover, the identification of Semantically Annotated Attributes enables to inherently recognise their belonging data sources. The path associated with each involved data source is leveraged by the `QUERY MANAGER` module to access HDFS (3) and register a temporary table, corresponding to a Spark DataFrame object (4). Each table corresponds to a data set, whose schema is built reading the metadata related to data sets, contained in the

[5] `https://spark.apache.org/sql/`
[6] `https://spark.apache.org/`

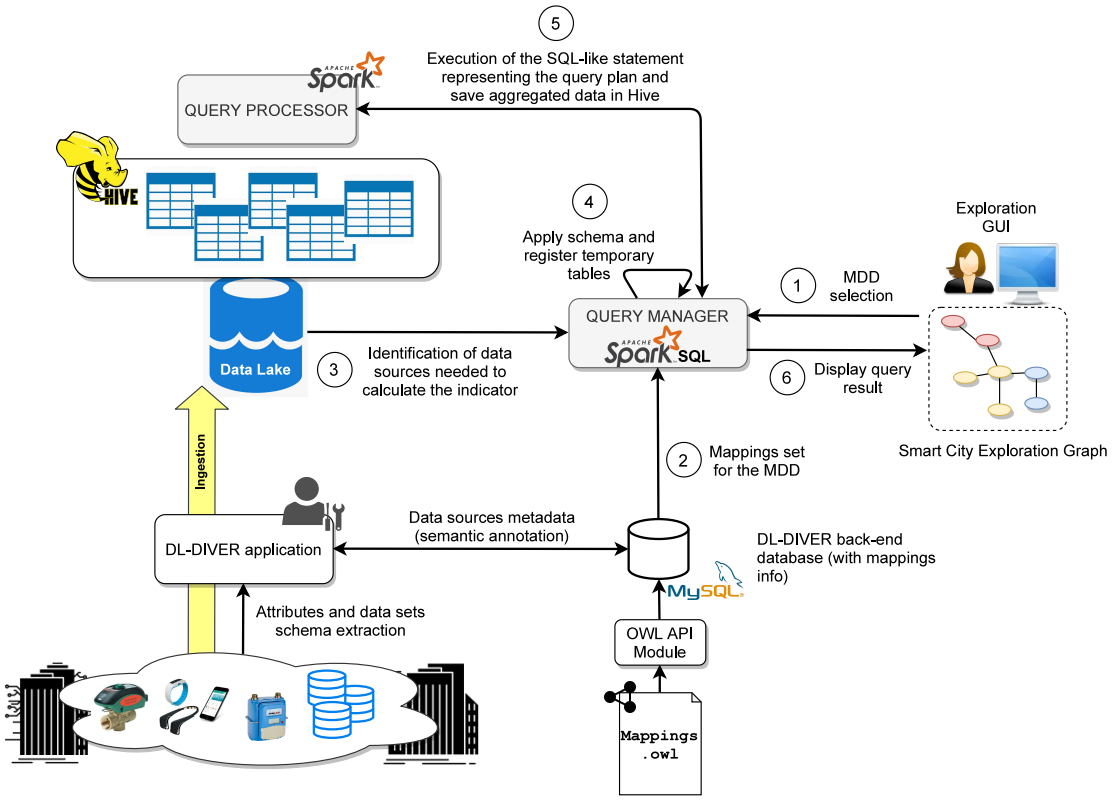


Figure 3.16: Architecture overview for the query process (simplified version).

DL-DIVER back-end database. The SQL-like translation of the query plan associated with Q_{τ_i}, derived with the methodology exposed in Section 3.5.4, is submitted to the `QUERY PROCESSOR` module, which performs projections, selections and joins on data sets, in order to produce as a final output the aggregated values for the indicator ind_i in τ_i, which are stored into Hive (5) in order to enable subsequent exploration by the user (6).

3.6.2.2 Query Processor module

The `QUERY PROCESSOR` module is grounded on Apache Spark environment. Apache Spark is a general-purpose cluster computing engine with APIs for streaming processing, graph processing and machine learning. Spark offers a functional programming API, where users manipulate distributed collections called Resilient Distributed Datasets (RDDs) [70]. Each RDD is an immutable collection partitioned across distributed computation nodes, that can be rebuilt if a partition is lost. When RDD is set to be cached or persisted in memory, each node caches its respective slices from local computation and reuses them in other operations on that RDD. This

lets Spark achieve much higher performance than disk-based Map-Reduce. RDD abstraction supports two kinds of operations: *transformations*, which form a new dataset from a base dataset by using functions such as map, and *actions*, which return the final results after running a series of operations on the dataset. The Spark computational model consists of masters and slaves nodes. Masters lifetime can span over several queries. The slaves are long-lived processes that can store dataset partitions in memory across operations. When the user runs a driver program, it starts with a master, which defines RDDs for the slaves and invokes operations on them. Spark programming model is well suited for bulk iterative algorithms, since RDDs are cached in memory and the data-flow is created lazily, which means the computation takes place only when RDDs are actually needed.

3.6.2.3 Query Manager module

As mentioned in the beginning of Section 3.6.2, the `QUERY MANAGER` module is deployed in Java. Specifically, it relies on the Spark SQL module libraries. Spark SQL is a Spark module for structured data processing [4]. It provides Spark with more information about the structure of both the data and the computation being performed. Spark SQL runs as a library on top of Spark, exposing SQL interfaces, which can be accessed through JDBC/ODBC or through a command-line console, as well as the *DataFrame API*, integrated into Spark's supported programming languages. Specifically, the DataFrame API lets users mix procedural and relational code, allowing the invocation of advanced user-defined functions to be exploited, for instance, within Business Intelligence tools. The main abstraction in Spark SQL's API is a *DataFrame*, a distributed collection of rows with the same schema. A DataFrame is equivalent to a table in a relational database, and can also be manipulated in similar ways to the native distributed collections in Spark (RDDs). Unlike RDDs, DataFrames keep track of their schema and support various relational operations that lead to more optimised execution. DataFrames can be constructed from tables in a system catalog (based on external data sources) or from existing RDDs of native Java/Python objects. The key feature of DataFrames is that they are subject to a lazy evaluation. Indeed, each DataFrame object represents a logical plan to compute a dataset, but no execution occurs until a special output operation (such as `save`). This enables optimisation across all operations that were used to build the DataFrame (e.g., delaying filtering the values of a specific DataFrame column until an output operation such as an aggregation function like `count` is called).

Given the aforementioned premises, and without a loss of generality, a data set instance as conceived in our approach (Section 2.2) is equivalent to a single row of a DataFrame data structure, with columns representing the schema of a data set.

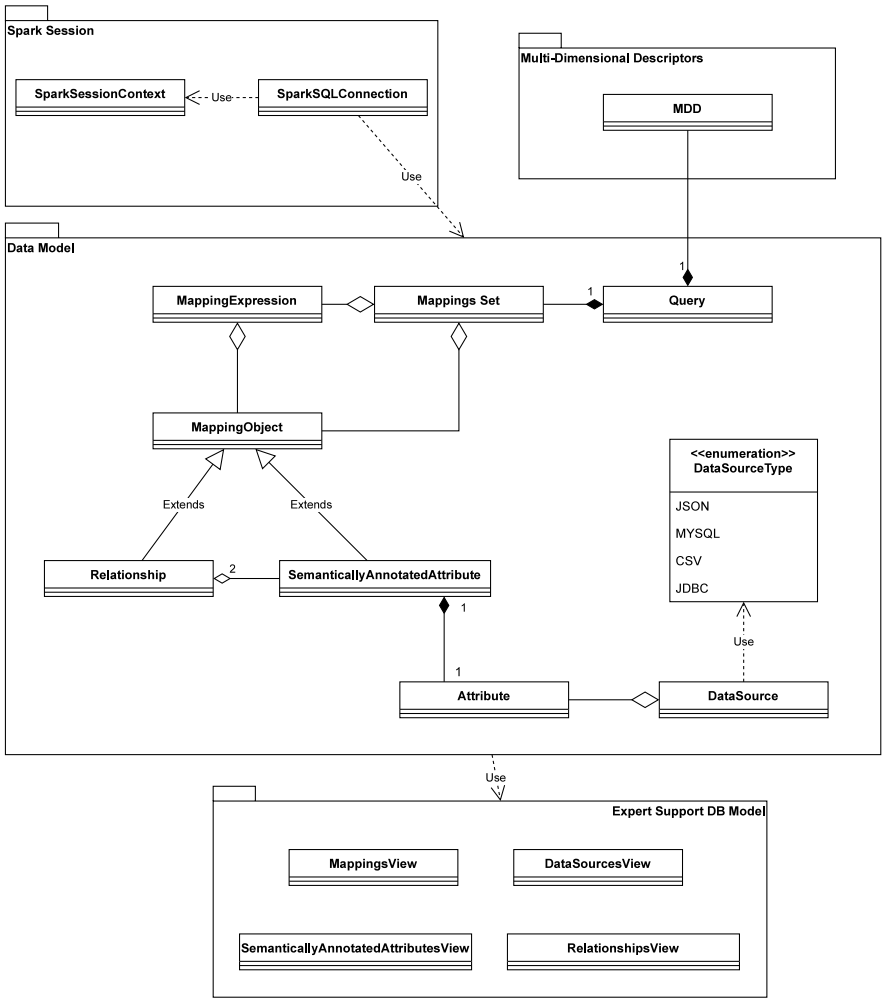


Figure 3.17: UML package diagram for Java classes exploited by `QUERY MANAGER` module.

Hereafter, we describe the Java packages and related classes (whose high-level UML diagram is reported in Figure 3.17) exploited by the `QUERY MANAGER` to implement the exploration procedure detailed in Section 3.5. As a preliminary step to implement our prototype, we resorted to the OWL API libraries[7] to read the OWL mappings file (illustrated in Figure 3.12) and save its information in dedicated relational tables of the DL-DIVER back-end database (whose enhanced version is reported in Figure 3.18 and the tables and supporting views are highlighted).

[7] `http://owlapi.sourceforge.net/`

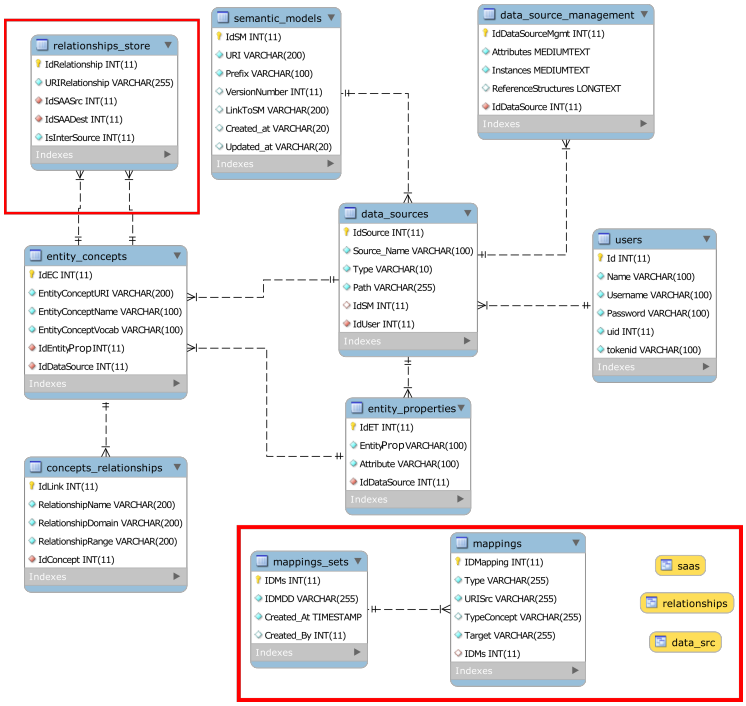


Figure 3.18: DL-DIVER back-end database model enhanced with tables holding the information of the OWL file containing mappings.

Expert Support DB package. This package contains classes devoted to read data from tables and views, stored in the DL-DIVER database, containing the definition of data sources, Semantically Annotated Attributes and mappings (containing relationships between SAAs). The content of the tables/views is accessed through the libraries of Hibernate[8], an object-relational mapping tool for the Java programming language, whose primary feature is mapping from Java classes to database tables/views, and mapping from Java data types to SQL data types. Hibernate also provides data query and retrieval facilities, generating SQL calls and relieving from the manual handling and object conversion of the result set.

Data Model package. This package contains the classes devoted to model data sources, their attributes (along with the related semantic annotation) and mappings, in order to handle mappings sets for indicators and achieve queries answering. Data Model classes objects are constructed starting from the objects of classes Expert Support DB package, the latter representing instances of relational tables in the DL-DIVER back-end database. The `Query` class contains a reference to a Multi-

[8] `http://hibernate.org/`

Dimensional Descriptor τ_i (`MDD` class) and a reference to the associated mappings set (`MappingsSet` class). In turn, a `MappingsSet` object contains a reference to $|L_i|$ `MappingExpression`, each of them representing the link between the indicator ind_i and a dimensional level $l_i^j \in L_i$ in terms of `MappingObject` entities (which can be either `SemanticallyAnnotatedAttribute` or `Relationship`).

Spark Session package. This package contains the `SparkSessionContext` class, leveraged to access (or create) the `SparkSession` instance, in order to communicate with the `QUERY PROCESSOR`. The `SparkSQLConnection` class, instead, contains overloaded methods apt to access data sources of the underlying Data Lake infrastructure, get the SQLContext object to register temporary tables and execute queries, and create Hive tables.

3.7 Experiments

3.7.1 ISO 37120 and ISO 37122 standards

To validate the methodology for the definition of indicators, as explained in this paper, and to test the effectiveness of validation rules to properly support the methodology application, we considered two International Standards proposed for Smart City indicators. The aim of the use of standards is also to share their definitions of indicators as represented in a more formal way, in terms of a Smart City Exploration Graph. In the following, we describe the ISO 37120 ("Sustainable development of communities - Indicators for city services and quality of life") and ISO 37122 standards. Generally speaking, these standards contain the definition (in natural language, thus not being constrained to a specific modelling language) of city indicators, providing definitions for a set of city indicators to steer and measure delivery of city services and quality of life. The ISO 37122 standard is meant as a complement of the older ISO 37120 standard, in the sense that provides new categories of indicators and broadens existing ones.

Indicators and dimensions. For each indicator, a description in the standard specification is given through four parts, each of them covering a specific modelling aspect: (i) general description of the indicator and of its meaning (*general*); (ii) description of elements required for building the indicator (*requirements*); (iii) how the data related to the indicator should be interpreted (*data interpretation*, optional); (iv) data sources from which the indicator values are calculated (*data sources*, optional). Beyond the description, indicators can be classified into three typologies:

- *core indicators*, required to demonstrate performance in the delivery of city services and quality of life;
- *supporting indicators*, recommended to demonstrate performance in the delivery of city services and quality of life;
- *profile indicators*, providing basic statistics and background information to help cities identify other cities that are of interest for peer comparisons.

Regarding the dimension, the ones mainly associated with indicators are *space* and *time*. Indeed, such indicators are related to the spatial organisation of the city (e.g., districts, buildings) and, for some of them, it is also specified the period when data is collected (year, month and day).

Domains. ISO 37120 defines indicators divided into 17 domains, that is: economy, education, energy, environment, finance, fire and emergency response, governance, health, recreation, safety, shelter, solid waste, telecommunication and innovation, transportation, urban planning, wastewater, water and sanitation. In the more recent ISO 37122, instead, the domains are 19: economy, education, energy, environment and climate change, finance, governance, health, housing, population and social conditions, recreation, safety, solid waste, sport and culture, telecommunication and innovation, transportation, urban/local agriculture and food security, urban planning, wastewater, water (and sanitation). As it can be noticed, some domains are still the same between the two standards, some of them have been introduced or renamed.

3.7.1.1 Indicator modelling example

The sulphur dioxide (SO_2) concentration indicator is a supporting indicator, whose description from the ISO 37120 standard is reported in Figure 3.19.
Noteworthy, from the textual description of the indicator a data analyst may define three distinct indicators: (a) `SO2_total_mass_collected`, gathering all the SO_2 produced in a hour; (b) `SO2_day_concentration`, providing the daily concentration by averaging the hourly concentrations throughout a 24 hour period and (c) `SO2_year_concentration`, obtained as the sum of daily concentrations for the whole year divided by 365 days. Amongst these three indicators, (b) and (c) are composite indicators. Indeed, `SO2_year_concentration` depends for its calculation from `SO2_day_concentration` whilst `SO2_day_concentration` depends in turn on `SO2_total_mass_collected` (since it provides the hourly concentration). For clarity purposes, Figure 3.20 illustrates the semantic definition of the `SO2_day_concentration` indicator using the Protégé tool. In the figure, the red boxes emphasise that the indicator is a composite indicator (modelled

SO_2 concentration shall be calculated as the sum of daily concentrations for the whole year (numerator) divided by 365 days. The result shall be expressed as the annual average for daily SO_2 concentration in µg/m^3. The daily concentration shall be determined by averaging the hourly concentrations throughout a 24 hour period from all monitoring stations within the city.

Users of this standard should also note the frequency of SO_2 exposures. Peak exposure is determined by calculating the number of times the 10 minute mean exceeded 500 µg/m^3 of SO2 in a calendar year. Long-term exposure is determined by calculating the number of times the daily mean exceeded 20 µg/m^3 of SO_2 in a calendar year.

NOTE If the local air quality monitoring stations measure SO_2 in parts per billion the following conversion ratio to µg/m3: 1 ppb = 2.62 µg/m3 shall be used. The conversion assumes an ambient pressure of 1 atmosphere and a temperature of 25 degrees Celsius. The general equation is µg/m3 = (ppb)*(12.187)*(M) / (273.15 + °C) where M is the molecular weight of the gaseous pollutant. An atmospheric pressure of 1 atmosphere is assumed.

Figure 3.19: SO_2 requirements description from the ISO 37120 standard. Highlighted boxes represent salient elements exploited to derive the semantic representation of the indicator(s) related to SO_2 pollutant.

through the `takesDataFrom` semantic relationship) due to the fact that it depends on the `SO2_day_concentration`. The calculation expression (`hasFormula` semantic relationship) is made clear in Figure 3.21, where the formula concept `F_SO2_day_concentration` is defined (through `rdfs:isDefinedBy` annotation property) as `SO2_total_mass_collected`/24.

3.7.1.2 Validation rules

To prove the benefits of the introduction of validation rules in the modelling process, we conducted an experiment over a group of 15 users possessing data analysis skills and familiarity with the Protégé framework. Specifically, users were asked to start from the description of a set of ten indicators as of the ISO 37120 standard and to model the related background knowledge, using the MDO. Modelling has been performed in two different modalities: without any validation rule and with validation rules. For both the aforementioned types, the table in Figure 3.22 reports the average values (in percentage) related to the part of knowledge which have been neglected, focusing on representative ones (units of measures, domains, dimensions, link to activity concepts).
As shown in the table, without rules enabled, the majority of errors regarded the

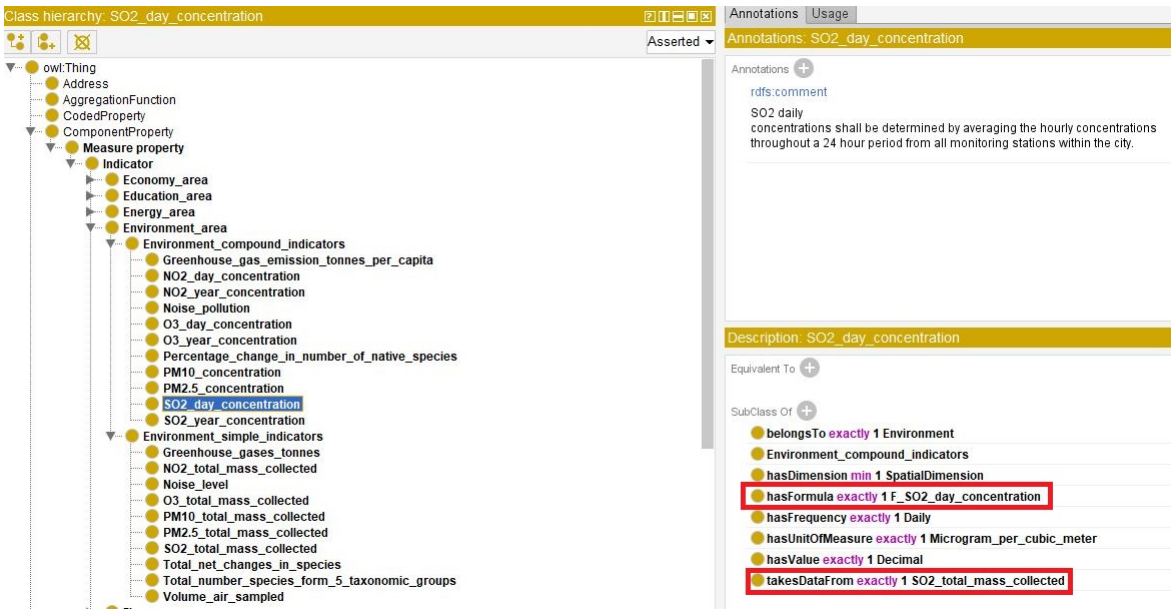


Figure 3.20: SO_2 daily concentration modelled with the MDO.

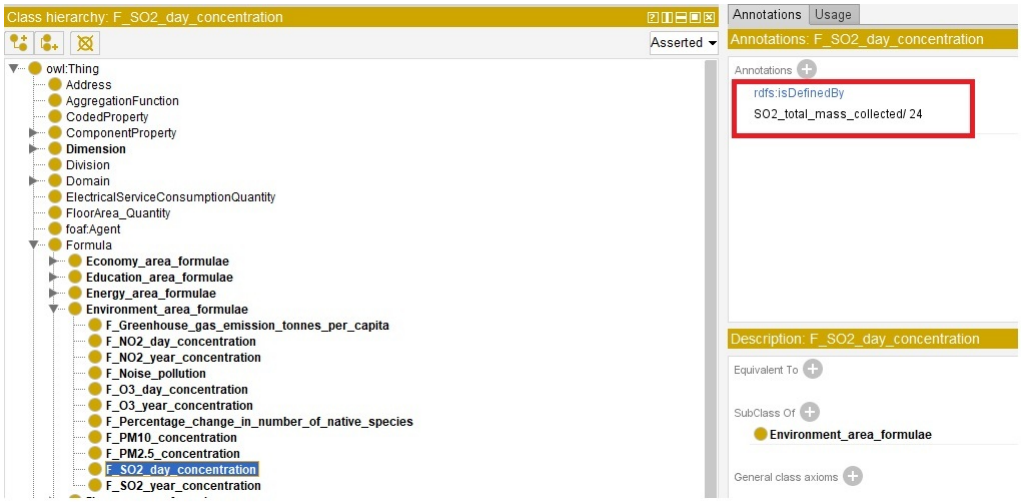


Figure 3.21: Details of the calculation formula for the SO_2 daily concentration.

missing linkage of indicators to the related activity (through the `involves` semantic relationship) and the creation of links between a composite indicator towards its dependencies (by means of the `takesDataFrom` relationship). On the other hand, a lower error rate has been detected for unit of measures and dimensions; this may be due to the fact that when modelling indicators, these features can be inherently extracted from the description of the indicators in natural language and thus they are

	Concepts			Relationships			
	Missing unit of measure	Missing domain	Others	Missing linkage composite indicators	Missing dimension linkage	Missing activity linkage	Others
Without rules	5%	13%	4%	23%	7%	25%	18%
With rules	-	-	5%	21%	-	-	21%

Figure 3.22: Average percentages of occurrence of errors while modelling indicators.

less prone to be subjected to errors with respect to the aforementioned links towards activities and indicators dependencies, which are peculiar of our modelling framework. Broadly speaking, the "Others" column gathers information about errors which are not strictly related to the presence of the devised validation rules. For instance, errors occurring while sub-classing concepts and missing domain/range assignments are not captured by the validation rules, thus they are present regardless of the type of modelling strategy.

3.7.2 Query answering

In this section, we report the evaluation of the query answering approach devoted to retrieve indicators values from the Data Lake, by exploiting the mapping techniques illustrated in this chapter to relate the definition of indicators as of the Smart City Exploration Graph with the concepts and relationships of the semantic overlay.

3.7.2.1 Dataset and experimental setup

We considered four sample data sources, whose content has been synthetically generated through a semi-automatic Web-based tool[9] and subsequently saved according to different storage technologies:

- $\hat{\mathcal{S}}_1$ is a CSV file containing the concentration of six pollutants (ozone, particulate matter 10, particulate matter 2.5, carbon monoxide, sulphur dioxide, nitrogen dioxide) a timestamp and geographical coordinates values to give a spatial and temporal reference to the measures (hereafter referred to as data source "pollutants");

[9] `https://www.generatedata.com/`

- $\hat{\mathcal{S}}_2$ is a JSON file containing information related to the cities corresponding to the coordinates in $\hat{\mathcal{S}}_1$ (hereafter referred to as data source "cities");
- $\hat{\mathcal{S}}_3$ is a MySQL database containing, amongst the others, a table describing the cities, associating them with the countries and the regions they are located in (hereafter referred to as data source "geodata");
- $\hat{\mathcal{S}}_4$ is a CSV file containing the concentration of ultra-fine particulate (in brief, UFP) along with the timestamp and latitude/longitude values to give a spatial and temporal reference to the measures (hereafter referred to as data source "ufp").

Concerning the query workload, two types of queries have been considered: (a) queries involving only simple indicators and (b) queries with composite indicators. In each query, the maximum number of dimensions has been set to two (i.e., time and space). Experiments have been conducted on a Windows PC equipped with an Intel Core i5-3210M processor, CPU 2.50 GHz, 4 cores, 8 logical cores, RAM 8GB. For the Spark environment configuration, one master node and one worker node have been deployed on the same Windows machine.

3.7.2.2 Metrics

In our experiments, we considered two different metrics for each query Q_{τ_i} issued over the Spark query environment, inspired by the ones proposed in the reference approach [29]: (a) the *execution time* which, for each local plan execution, considers both the time required to access and load the data sets for the data source; (b) the *aggregation time*, needed to join partial results of local plans, apply the aggregation function associated with the indicator in Q_{τ_i}, and projecting the attributes in q_π. In particular, the aggregation time calculation is assessed only after all the local plans for Q_{τ_i} have been executed in parallel, thus its calculation is applicable only for the global query result. Overall, the number of *records* for each local plan and for the global plan, due to the evaluation of Q_{τ_i}, have been calculated. In the experiments, we did not consider the ingestion time of data sources into the Data Lake, since the aim here is to assess the feasibility of the query answering process for the exploration of indicators.

3.7.2.3 Simple and composite indicators

We report in the following the summary statistics about two meaningful examples. The first regards the calculation of the average concentration of `COIndicator` (simple indicator) aggregated by: (i) region (spatial dimension) and (ii) year and region (temporal and spatial dimension). In this respect, recalling the notation of

MDDs, and considering as previously mentioned only time and space dimensions, we have: $\tau_1 = \langle \texttt{COIndicator}, (\texttt{Region}, \texttt{ALL}) \rangle$ and $\tau_2 = \langle \texttt{COIndicator}, (\texttt{Region}, \texttt{Year}) \rangle$. The second, instead, involves the calculation of average values for the composite indicator `ParticulateMatter`, denoted by a formula gathering three simple indicators `PartMatter10` + `PartMatter2.5` + `UFP`. Also for `ParticulateMatter`, the same aggregations as for `COIndicator` have been applied, resulting in the MDDs τ_3 and τ_4.

Simple indicator. Figure 3.23 reports the results in terms of number of records, execution time and aggregation time for the queries Q_{τ_1} and Q_{τ_2}, along with the translation of the query plan in SQL. For Q_{τ_i}, the statistics related to the data sources involved in the query process are reported; each query has been issued ten times and average results have been computed. In the two queries, only the number of records in the global result changes, as it depends on the aggregation made and thus on the projected attributes. Moreover, execution times of the single queries in Q_{τ_i} are on average the same, even when the group-by set becomes larger when calculating Q_{τ_2}. This is due to the fact that temporal information for performing aggregation over the `Year` dimension is contained in the same data source `COIndicator` values are contained in (i.e., $\hat{\mathcal{S}}_1$), thus delivering only a slight increment on the aggregation time (20 ms).

Composite indicator. Concerning `ParticulateMatter` composite indicator, the same group-by sets as for the simple indicator case have been adopted (i.e., region, year and region). In particular, Figure 3.24 provides the details related to two different scenarios, that is the calculation of the indicator *before* and *after* the addition of the `UFP` addend, whose data is stored in a different data source ($\hat{\mathcal{S}}_4$) with respect to `PartMatter10` and `PartMatter2.5` addends (retained in $\hat{\mathcal{S}}_1$). It can be noticed that, even though the number of records in the query result decreases after the inclusion of the `UFP` addend (since it has a lower number of values – 50k – with respect to `PartMatter10` and `PartMatter2.5` having both 100k records) the time demanded for aggregation increases, in compliance with the fact that the indicator formula becomes more complex.

3.7.2.4 Remarks on aggregation time

Apart from the examples formerly reported, Figure 3.25 focuses on the aggregation time required for the calculation of a composite indicator in terms of n simple indicators (in the figure, the maximum value of n has been set to five). This kind of experiments have been conducted considering again synthetically generated data sources as previously done. Aggregation time corresponds to the time required to combine (join) the results coming from the execution of local plans and applying, in

$\tau_1 = \langle$COIndicator,(Region,ALL)$\rangle$

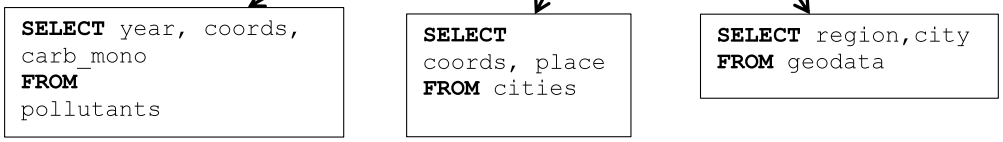
```
SELECT geodata.region, AVG(pollutants.carb_mono)
FROM pollutants,cities,geodata
WHERE pollutants.coords=cities.coords AND cities.place=geodata.city
GROUP BY geodata.region
```

	$\widehat{S_1}$	$\widehat{S_2}$	$\widehat{S_3}$	Q_{τ_1}
Records [#]	100k	100	100k	128
Execution time* [s]	3.722	0.327	4.628	
Aggregation time [s]				0.739

$\tau_2 = \langle$COIndicator,(Region,Year)$\rangle$

```
SELECT geodata.region, pollutants.year, AVG(pollutants.carb_mono)
FROM pollutants,cities,geodata
WHERE pollutants.coords=cities.coords AND cities.place=geodata.city
GROUP BY pollutants.year,geodata.region
```

	$\widehat{S_1}$	$\widehat{S_2}$	$\widehat{S_3}$	Q_{τ_2}
Records [#]	100k	100	100k	483
Execution time* [s]	3.689	0.316	4.662	
Aggregation time [s]				0.751



```
SELECT year, coords,
carb_mono
FROM
pollutants
```

```
SELECT
coords, place
FROM cities
```

```
SELECT region,city
FROM geodata
```

Figure 3.23: Example of queries and related SQL-like statements involving `COIndicator`, with values aggregated according to different perspectives and with caching of intermediate results.

the end, the aggregation function over indicator values in the scope of the group-by set of Q_{τ_i}. We run experiments according to two different strategies, with and without *caching* of intermediate results. In Spark SQL, caching is a common technique for reusing some computation. It has the potential to speed-up other queries that are using the same data. In Figure 3.25, it can be seen that when caching is applied to intermediate results, the aggregation time to carry out the global query plan execution is lower and when the formula has its maximum number of indicators, the difference between aggregation times is $\approx 656ms$ (specifically, $1861ms$ without intermediate caching vs $1205ms$ with intermediate caching). Lastly, in Figure 3.25 we marked two points (A and B), denoting when the i-th indicator values are obtained from a new data source with respect to the $i-1$ one. As expected, the aggregation time increases

τ_4 = ⟨ParticulateMatter, (Region, Year)⟩

Part_Matter (PM) = Part_Matter_10 (PM_{10}) + Part_Matter_2.5 (PM_{25}) + UFP (Ultra-fine particulate)

```
SELECT geodata.region, pollutants.year,
AVG(pollutants.part_matter_10+pollutants.part_matter_25+ufp.ufp)
FROM pollutants,cities,geodata,ufp
WHERE pollutants.coords=cities.coords AND cities.place=geodata.city AND
ufp.coords=cities.coords AND ufp.year=pollutants.year
GROUP BY pollutants.year,geodata.region
```

	$\widehat{S_1}$	$\widehat{S_2}$	$\widehat{S_3}$	Q_{τ_4}
Records [#]	100k	100	100k	7038
Execution time* [s]	3.786	0.285	4.796	
Aggregation time [s]				0.829

	$\widehat{S_1}$	$\widehat{S_2}$	$\widehat{S_3}$	$\widehat{S_4}$	Q_{τ_4}
Records [#]	100k	100	100k	50k	1078
Execution time* [s]	3.904	0.284	4.681	0.340	
Aggregation time [s]					1.011

Figure 3.24: Example of queries involving `ParticulateMatter`, with the same aggregation and with caching of intermediate results. The figure highlights the differences before and after the `UFPIndicator` addend is considered for the calculation of the composite indicator.

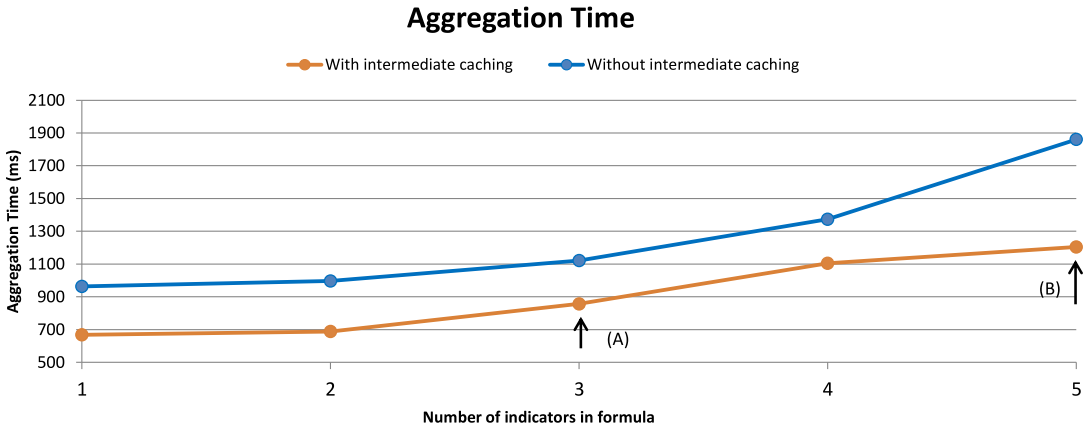


Figure 3.25: Aggregation time for an increasing number of indicators in a calculation formula.

both as the calculation formula becomes more complex and, at the same time, also when the novel indicator included in the formula comes from a new data source, since

the complexity of the global query plan increases.

Chapter 4

Preference-based Personalised Data Exploration

4.1 Introduction

This chapter focuses on the adoption of users' contexts and preferences to enable a personalised exploration of Data Lake content. Firstly, we give the formal definition of users' contexts, determined by users' categories and activities performed by users, and define the preference constructors which can be employed to express preferences on Smart City indicators. Since in our approach preferences are tightly coupled with exploration contexts, modelling choices and decisions of users in their everyday life, they are referred to as *contextual preferences*. In the second part of the chapter, we present the procedure for indicators exploration, aimed at proposing to the user the best indicators according to his/her preferences to start the exploration from. In the last part of the chapter, preference evaluation algorithms from the literature and a prototype Web application are fostered to assess the feasibility of the approach.

4.1.1 Related Work

Preference-based systems have been proposed in the literature of the last decades as valuable instruments apt to avoid empty results while exploring data on the one hand, and information flooding on the other. Besides, preferences allow for ranking query results so that the user may first see the data that better meets her exploration goals and demands. From a higher viewpoint, preferences have been studied under a two-fold perspective, *quantitative* [25, 45, 52], relying upon a scoring function exploited to determine a total order of results, and *qualitative* [2, 24, 25, 60, 66], expressed using binary relations to deliver a strict partial order of results. In the scope of this thesis, we focused on qualitative preferences, yielding a higher expressiveness with

Approach	Applied to	Type of preferences	Own preference constructors	Contextual preferences	Preferences expressed on
Polo et al. [52]	Linked Data	Quantitative	✗	NO	Linked Data resources
Golfarelli et al. [24]	Data Warehouse	Qualitative	✓	NO	Data Marts content
Troumpoukis et al. [66]	Linked Data	Qualitative	✗	NO	Linked Data resources
De et al. [2]	Relational Database	Qualitative	✗	NO	Tuples
PrefSPARQL [25]	Linked Data	Qualitative and Quantitative	✗	NO	Linked Data resources
PREMINE [45]	Relational Database	Quantitative	✓	YES	Tuples
Rosati et al. [60]	Linked Data	Qualitative	~ (ontology for representing preferences)	YES	Linked Data resources
Ours	Semantic Data Lake	Qualitative	✓	YES	Indicators on top of the Semantic Data Lake

Figure 4.1: Summary of approaches on preference-based frameworks.

respect to quantitative ones (since the latter can only capture those user preferences that can be translated into numbers, through a scoring function). In [24] a preference-based framework is implemented in a Data Warehouse environment, where preferences are formulated over aggregation levels of facts. The innovative proposal of the approach is two-fold. On the one hand, it envisages the adoption of custom preference constructors, applied on both dimensions and values of indicators, whereas on the other hand it devises an algorithm for preference evaluation optimised for data marts with millions of records. Conversely, the work presented in [2] suggests the adoption of conditional preferences, to extend the classic SQL query language with soft constraints specified as if-then rules. The former approaches handle multi-dimensional data, preferences are defined over data marts content and context-aware personalisation aspects have not been taken into account. The work in [52] formalises preferences as constraints on semantic resources, mixing both quantitative and qualitative aspects, whereas [25, 66] devise an extension of SPARQL query language for expressing preferences on semantic data. Despite introducing semantics, these approaches focus on the extensions made to the query language, and context-aware personalisation aspects are not considered.

Beyond the qualitative and quantitative aspects, a pivotal position has been fulfilled by *contextual preferences*, adapting their behaviour according to the different situations users are involved in. The work in [45] proposes methodology where data mining is adopted to infer contextual preferences from the past interaction of the user with contextual views over a relational database. Therein, the preference model is grounded on database schema and attributes, with score value to capture users' interest. Authors in [60] introduce a reference model to encode and reason with preferences, an ontological vocabulary to represent preferences and a tool able to

handle and manage preferences as well as to encode them in a set of SPARQL queries. A preliminary effort for context-aware personalisation in a preference-based system has been presented in [18], where the propagation of preferences across exploration contexts has been investigated, considering also effects of context changes. Nevertheless, the former approaches emphasise context formalisation, coarsely treating other exploration and personalisation issues.

4.1.2 Novel Contributions

Considering the ontology-based indicators modelling strategy defined in Chapter 3, we provide here the formalisation of a methodological approach to support personalised exploration of Smart City indicators built on top of the Semantic Data Lake, defined in Chapter 2. Specifically, the approach is grounded on two main pillars. The first are *users' profiles*, composed of *preferences*, expressed through constructors exploiting the structure of the Smart City Exploration Graph presented in the previous chapter. Preferences are organised according to *contexts*, gathering information characterising the situation under which the user explores indicators, influenced by both his/her roles and goals. The second is a four-steps procedure to achieve *personalised exploration* of Smart City indicators, leveraging the profiles as previously defined. With respect to research efforts proposing preference-based frameworks for exploring multi-dimensional data [24], in this thesis we tackle the exploration task from a different perspective. In our approach, preferences are fostered as a way to rank Smart City indicators best fitting users' exploration demands, captured by users' profiles. The ranking of indicators is based on the semantic definition they have in the Smart City Exploration Graph and not on their actual values, as conceived by [24]. Indeed, complying with the on-demand strategy of Data Lake environments, preferences allow to identify in advance one or more indicators of interests for the user, and only at a later time inspect their values, thus avoiding a trial-and-error approach when exploring data, saving cost and resources required to query Data Lake sources. Moreover, the surveyed approaches employed in the Semantic Web field [52, 66] foster preferences expressed on an RDF-based representation of data; therefore, they are not appropriate to be applied on the Smart City Exploration Graph, containing only semantic concepts and not instances (individuals) of such concepts. The content of this chapter illustrates our published research efforts [8, 9], where we introduced the preference model and the preference-based indicators exploration procedure. Herein, we provide additional details about the aforementioned preference model and describe implementation and experimentation issues fostering two renowned preference evaluation algorithms from the literature.

4.2 Users' profiles

Being $\mathcal{U}$ the set of Smart City users, the *profile* $p(u)$ of a user $u \in \mathcal{U}$ is composed of *hard constraints* and *soft constraints*. The former have been recalled in Section 3.5.3 and are exploited at query evaluation time (for instance, a building administrator is allowed to inspect indicators for the administered buildings only). Soft constraints are modelled by means of *preferences*, expressed through constructors exploiting the structure of the Smart City Exploration Graph. Preferences are organised according to *contexts* $\mathcal{CT}_u$, gathering information characterising the situation under which the user explores indicators, influenced by both his/her roles and goals. In the following sections, we provide the formal definition of contexts (detailing also their creation procedure) and of the preference-based framework for exploring Smart City indicators.

4.2.1 Contexts

Users' choices and decision in everyday life impact on users' exploration preferences. For instance, regarding the Smart City domain, it is well known that the preferences of users are strongly influenced by a plethora of aspects regarding the context in which they lie. Indeed, a citizen, while checking air pollution levels, may prefer to inspect each pollutant concentration in detail (e.g., CO_2, SO_x, NO_x), as knowledgeable of environmental issues. However, such preference holds in the former specific context, but it is not applicable for a building administrator, performing building monitoring activity.
In our approach, user's context is modelled by considering the *category* of the user (e.g., citizen, building administrator) and the *activity* that the user is currently performing (e.g., monitoring electrical consumption, checking air pollution levels).

Definition 14 (Context) *A context* ctx^i *belonging to* $\mathcal{CT}_u$ *is a triple* $\langle cat^i, a^i, CP^i \rangle$ *where: (i)* cat^i *represents a category of the user* $u \in \mathcal{U}$*, (ii)* a^i *is an activity performed by the user and (iii)* CP^i *denotes the set of preferences valid for the context (contextual preferences).*

Specifically, contextual preferences can be either: (a) *short-term preferences*, expressed by the user at exploration time, representing an imminent exploration need to be satisfied; (b) *long-term preferences*, stored in user's profile, which are presumed to be static or change slowly over time. The context ctx^i is used to delimit a portion of the Smart City Exploration Graph $\mathcal{G}$, explorable by the user within ctx^i. We will denote such portion as $\mathcal{G}^i$. The portion $\mathcal{G}^i$ is identified starting from user's category and activity. In fact, as can be seen in Figure 4.2, an activity conceived for a user category is linked, through the `involves` semantic relationship, to one or more indicators. Contextual preferences are further used to provide a ranking of indicators, as illustrated in Section 4.4.

4.2.1.1 Procedure for contexts creation

The creation of exploration contexts is in charge of data analysts, who leverage the structure of $\mathcal{G}$. Contexts creation is a preliminary, semi-automatic phase, performed by data analysts, who possess the required knowledge about users' categories of the Smart City domain. To create a new context $\hat{ctx}$, modelling an exploration goal (e.g., "citizens performing air pollution levels monitoring"), the data analyst:

- Identifies in $\mathcal{G}$ the `UserCategory` concept related to the role the prospective context is meant for (e.g., $\hat{cat}$ = `Citizen`).
- Following the `hasPracticableActivity` semantic relationship, chooses in $\mathcal{G}$ an activity concept available for $\hat{cat}$, suitable for semantically describing the exploration goal. If a proper activity cannot be found, the data analyst creates a new activity concept specialising an existing one (e.g., if the activity concept `AirPollutionMonitoring` does not exist in $\mathcal{G}$, it is created as a subclass of the `EnvironmentalMonitoring` activity, supposing that the latter is present in $\mathcal{G}$ and available for citizens).
- When a new activity has been created, he/she links the activity concept to one or more indicator concepts, using the `involves` semantic relationship. Noteworthy, given the activity hierarchy, `AirPollutionMonitoring` inherits all the indicators available for `EnvironmentalMonitoring`.
- Prepares a set of preferences for the novel context (this task is not mandatory).

Once these steps have been completed, the context $\hat{ctx}$ can be bound to one or more users' profiles, to be used for exploration sessions. Broadly speaking, a Smart City user can manage her profile by: (a) selecting/changing the contexts of interest, choosing them from the list of contexts available for her role(s), thus refining the content of $\mathcal{CT}_u$; (b) creating additional long-term preferences for each context in $\mathcal{CT}_u$, joining the ones prepared by the data analyst. Concerning the former point, only contexts compliant with her role(s) will be proposed for inclusion in user's profile.

4.2.2 Preferences

Contextual preferences within a context ctx^i are expressed through *constructors*, having as target objects the Multi-Dimensional Descriptors (MDDs) derived from the Smart City Exploration Graph. In our approach, we focus only on *qualitative* preferences, as they bring a higher expressiveness with respect to the quantitative ones [16, 32]. Preferences are formally defined as follows.

Definition 15 (Preference) *A preference* $\mathbf{P}$ *is a couple* $(<_{\mathbf{P}}, \cong_{\mathbf{P}})$ *where* $<_{\mathbf{P}} \subseteq \mathcal{T} \times \mathcal{T}$ *is binary relation expressing a strict partial order over the set of MDDs* $\mathcal{T}$ *and* $\cong_{\mathbf{P}} \subseteq \mathcal{T} \times \mathcal{T}$ *is a binary relation for expressing substitutability. Given two MDDs* τ_1 *and* τ_2*, the semantics of* $\tau_1 <_{\mathbf{P}} \tau_2$ *is that MDD* τ_2 *is preferred to* τ_1*, whereas the semantics of* $\tau_1 \cong_{\mathbf{P}} \tau_2$ *is that* τ_1 *is substitutable to* τ_2*.*

Preferences base constructors rely on indicators and dimensional hierarchies, granularity levels and indicators domains, modelled by the data analyst in the Smart City Exploration Graph. The use of custom preference constructors for exploring multidimensional data has been already treated in previous research efforts, such as [24], where authors proposed an algebra for preference constructors working on attributes, values of measures and dimension hierarchies. The four main base constructors of our approach are explained in the following.

- IND(i) constructor denotes that τ_2 is preferred to τ_1 ($\tau_1 <_{\text{IND}(i)} \tau_2$) iff the distance of indicator ind_2, contained in τ_2, from indicator i (in the hierarchy induced by `subClassOf` relationships) is less than the distance between ind_1, contained in τ_1, and i. This constructor relies on the function $dist_I(i_h, i_k)$, computing the distance between indicators concepts i_h and i_k within the hierarchy induced by the `subClassOf` relationships.

$$\tau_1 <_{\text{IND}(i)} \tau_2 \iff dist_I(\tau_1.ind_1, i) > dist_I(\tau_2.ind_2, i)$$
$$\tau_1 \cong_{\text{IND}(i)} \tau_2 \iff dist_I(\tau_1.ind_1, i) = dist_I(\tau_2.ind_2, i)$$

$$dist_I(i_h, i_k) = \begin{cases} 0 & i_h \equiv i_k \\ 1 + dist_I(i_h, i_w | i_w \sqsupseteq i_k) & i_k \texttt{ subClassOf } i_h \\ 1 + dist_I(i_k, i_w | i_w \sqsupseteq i_h) & i_h \texttt{ subClassOf } i_k \end{cases}$$

- LEV(dim, l) constructor denotes that τ_2 is preferred to τ_1 ($\tau_1 <_{\text{LEV}(dim,l)} \tau_2$) iff the distance of level $l_2 \in Levels(dim)$, contained in τ_2, from l (in the hierarchy induced by `rollUp` relationships) is less than the distance between $l_1 \in Levels(dim)$, contained in τ_1, and l. This constructor relies on the function $dist_L(l_i, l_j)$, computing the distance between two levels l_i and l_j within the hierarchy dim, induced by `rollUp` relationships ($l_b \succ l_a$ denotes that dimensional level l_b is a successor of level l_a in the `rollUp` hierarchy, at any level)

$$\tau_1 <_{\text{LEV}(dim,l)} \tau_2 \iff dist_L(\tau_1.L_1[dim], l) > dist_L(\tau_2.L_2[dim], l)$$
$$\tau_1 \cong_{\text{LEV}(dim,l)} \tau_2 \iff dist_L(\tau_1.L_1[dim], l) = dist_L(\tau_2.L_2[dim], l)$$

$$dist_L(l_i, l_j) = \begin{cases} 0 & l_i \equiv l_j \\ 1 + dist_L(l_i, l_k | l_j \texttt{ rollUp } l_k) & l_j \succ l_i \\ 1 + dist_L(l_j, l_k | l_i \texttt{ rollUp } l_k) & l_i \succ l_j \end{cases}$$

- DEP(i) constructor denotes that τ_2 is preferred to τ_1 ($\tau_1 <_{\text{DEP}(i)} \tau_2$) iff the formula of indicator ind_2, contained in τ_2, is calculated starting from the indicator i (modelled through the `takesDataFrom` relationship in $\mathcal{G}$, see Figure 3.4), while ind_1, contained in τ_1, does not. This constructor exploits the boolean function $tdf(i_a, i_b)$ returning *True* if i_a `takesDataFrom` i_b in $\mathcal{G}$, *False* otherwise

$$\begin{aligned} \tau_1 <_{\text{DEP}(i)} \tau_2 &\iff \neg\, tdf(\tau_1.ind_1, i) \ \wedge\ tdf(\tau_2.ind_2, i) \\ \tau_1 \cong_{\text{DEP}(i)} \tau_2 &\iff (tdf(\tau_1.ind_1, i) \ \wedge\ tdf(\tau_2.ind_2, i)) \ \vee \\ &\quad (\neg\, tdf(\tau_1.ind_1, i) \ \wedge\ \neg\, tdf(\tau_2.ind_2, i)) \end{aligned}$$

- DOM(dom) constructor denotes that τ_2 is preferred to τ_1 ($\tau_1 <_{\text{DOM}(dom)} \tau_2$) iff the indicator ind_2, contained in τ_2, belongs to domain dom, while ind_1, contained in τ_1, does not. This constructor exploits the boolean function $domain(i, dom)$ returning *True* if i `belongsTo` dom in $\mathcal{G}$, *False* otherwise.

$$\begin{aligned} \tau_1 <_{\text{DOM}(dom)} \tau_2 &\iff \neg\, domain(\tau_1.ind_1, dom) \ \wedge\ domain(\tau_2.ind_2, dom) \\ \tau_1 \cong_{\text{DOM}(dom)} \tau_2 &\iff (domain(\tau_1.ind_1, dom) \ \wedge\ domain(\tau_2.ind_2, dom)) \ \vee \\ &\quad (\neg\, domain(\tau_1.ind_1, dom) \ \wedge\ \neg\, domain(\tau_2.ind_2, dom)) \end{aligned}$$

Example 15 *Given the two sample MDDs* $\tau_a = \langle$`CO2Indicator`, (`District`)$\rangle$ *and* $\tau_b = \langle$`HouseholdCO2Indicator`, (`Apartment`)$\rangle$ *and with reference to Figure 4.2* IND(`AirPollutionIndicator`) *constructor selects as preferred* τ_a*. This descends from the fact that* $dist_I$(`AirPollutionIndicator`, $\tau_a.ind_a$) $= 1$ *is less than* $dist_I$(`AirPollutionIndicator`, $\tau_b.ind_b$) $= 2$*. For the same sample MDDs,* LEV(`SpatialDimension`, `City`) *selects again as preferred* τ_a*, since* $dist_L$(`City`, `District`) $= 1$ *is less than* $dist_L$(`City`, `Apartment`) $= 3$*.*

With respect to the work in [24], grounded in a Data Warehouse context where preferences are expressed directly over data marts contents, in this thesis exploration of Smart City data is achieved through indicators as modelled in the Smart City Exploration Graph, whose definition relies in turn on the semantic representation of Data Lake sources in the semantic overlay. Thus, indicators values are not upfront materialised but retrieved only when the user selects one or more MDDs. Therefore, our four preference constructors (IND, LEV, DEP and DOM) leverage the semantic relationships held in the Smart City Exploration Graph, further increasing the expressiveness of preferences. The base constructors can be in turn combined using two composition operators from the literature, namely the *Pareto* ($\otimes$), composing two preferences with equal priority, and the *Prioritised* ($\triangleright$) composition [32]. In particular, the two binary composition operators are defined as follows.

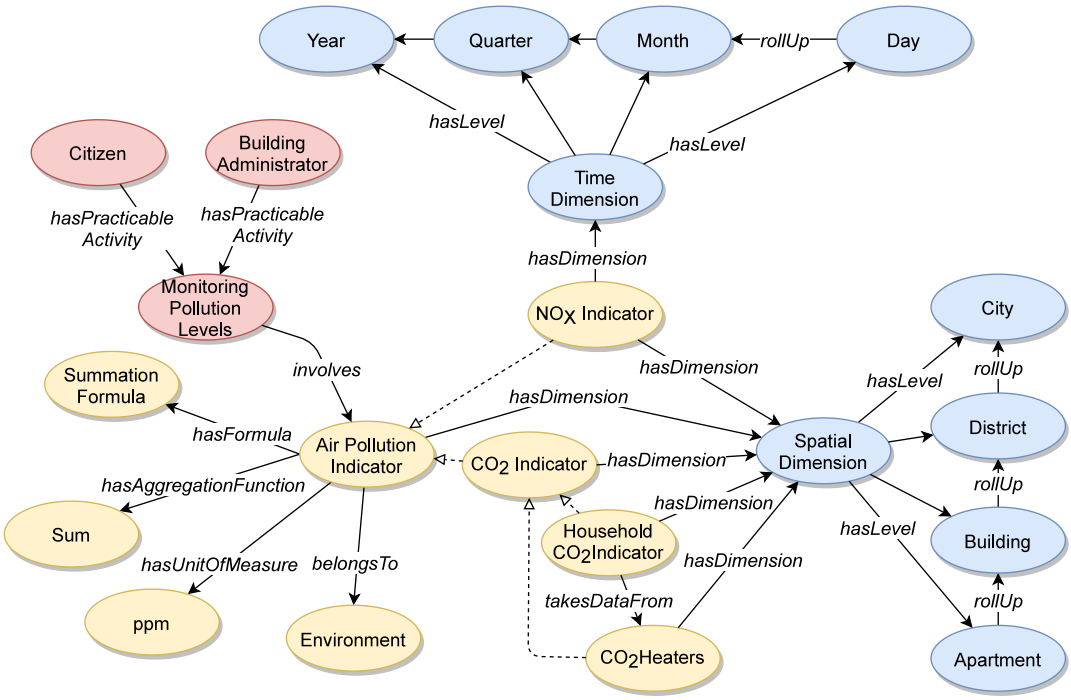


Figure 4.2: Excerpt of the Smart City Exploration Graph containing the semantic representation of indicators in Example 15.

- Pareto composition ($\mathbf{P_1} \otimes \mathbf{P_2}$) – an MDD is better than another if it is better according to one preference and better or substitutable according to the other (the composed preferences are considered equally important):

$$\tau_1 <_{\mathbf{P_1} \otimes \mathbf{P_2}} \tau_2 \iff (\tau_1 <_{\mathbf{P_1}} \tau_2 \wedge (\tau_1 <_{\mathbf{P_2}} \tau_2 \vee \tau_1 \cong_{\mathbf{P_2}} \tau_2)) \vee (\tau_1 <_{\mathbf{P_2}} \tau_2 \wedge (\tau_1 <_{\mathbf{P_1}} \tau_2 \vee \tau_1 \cong_{\mathbf{P_1}} \tau_2))$$

$$\tau_1 \cong_{\mathbf{P_1} \otimes \mathbf{P_2}} \tau_2 \iff \tau_1 \cong_{\mathbf{P_1}} \tau_2 \wedge \tau_1 \cong_{\mathbf{P_2}} \tau_2$$

- Prioritisation ($\mathbf{P_1} \triangleright \mathbf{P_2}$) – an MDD is better than another if it is better according to the first preference or, in case of substitutability, according to the second one (the composed preferences are considered of decreasing importance):

$$\tau_1 <_{\mathbf{P_1} \triangleright \mathbf{P_2}} \tau_2 \iff \tau_1 <_{\mathbf{P_1}} \tau_2 \vee (\tau_1 \cong_{\mathbf{P_1}} \tau_2 \wedge \tau_1 <_{\mathbf{P_2}} \tau_2)$$

$$\tau_1 \cong_{\mathbf{P_1} \triangleright \mathbf{P_2}} \tau_2 \iff \tau_1 \cong_{\mathbf{P_1}} \tau_2 \wedge \tau_1 \cong_{\mathbf{P_2}} \tau_2$$

4.2.2.1 Better-Than Graph (BTG)

A way to visualise the effect of the evaluation of a preference expression $\mathbf{P}$ over a subset of MDDs is to adopt a Better-Than Graph [55] (in brief, BTG). Formally, a BTG is a graph-like and level-wise structure, apt to visualise the preference order, and the following considerations hold in our approach:

- Nodes in a BTG are MDDs. A directed edge from a node τ_a to a node τ_b means that τ_b is dominated by (i.e., is worse than) τ_a. As domination is *transitive*, τ_a also dominates all MDDs dominated by τ_b.
- For every node τ_i, a *level* value can be defined $level(\tau_i)$ and it is the length of the longest path, from the root node, leading to it.
- Given a preference $\mathbf{P} = (<_{\mathbf{P}}, \cong_{\mathbf{P}})$ and the level function of its associated BTG, regarding the strict partial order relation $<_{\mathbf{P}}$, it holds that $\tau_a <_{\mathbf{P}} \tau_b \Rightarrow level(\tau_a) > level(\tau_b)$ and for the substitutability relation $\tau_a \cong_{\mathbf{P}} \tau_b \Rightarrow level(\tau_a) = level(\tau_b)$.

Focusing on levels, the first level of the graph contains the first-best MDDs according to $\mathbf{P}$, the second level will contain the second-best MDDs and so forth.

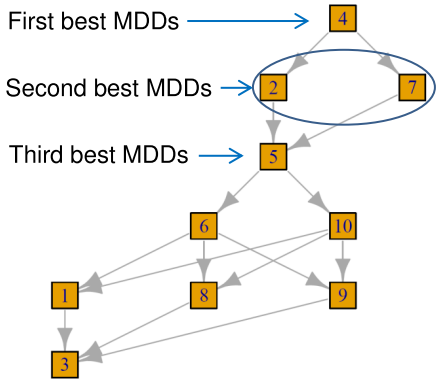
Example 16 *Figure 4.3 reports an example of preference evaluation over a sample subset of MDDs, whose description (in terms of indicator, spatial dimension and domain concepts) is reported in the table. The evaluated preference expression* $\mathbf{P}_1$ *is*

$$\text{IND}(\texttt{CO2Indicator}) \triangleright (\text{LEV}(\texttt{SpatialDimension}, \texttt{District}) \otimes \text{DOM}(\texttt{Environment}))$$

The BTG of $\mathbf{P}_1$ *over the subset of MDDs* $\{\tau_1, \ldots, \tau_{10}\}$ *has as root node* τ_4*, since it is the best MDD in the subset (it refers to* `CO2Indicator`*, belonging to the* `Environment` *domain, and has* `District` *as spatial dimension). The second best MDDs are* τ_2 *and* τ_7 *due to the fact that, despite their indicators belong to the* `Environment` *domain,* `Building` *dimensional level distance from* `District` *is equal to one. The third best MDD is* τ_5*, whose indicator is still in the* `Environment` *domain, but* `Apartment` *dimensional level distance from* `District` *is equal to two.*

4.2.2.2 Preference-based ranking

We will assume that the evaluation of a preference expression $\mathbf{P}$ over a set of MDDs returns the set of MDDs not worse than others, that is the *most preferred* ones. According to the preference literature, the set of best items according to a preference $\mathbf{P}$ can be obtained by applying the so-called *winnow* operator [16] to a (sub)set of the MDDs $\mathcal{T}$, which is formalised as follows:



	ind	*SpatialDim*	*Domain*
τ_1	ElectricalEnergyConsumption	Building	Energy
τ_2	NOXIndicator	Building	Environment
τ_4	CO2Indicator	District	Environment
τ_5	SOXIndicator	Apartment	Environment
τ_6	ElectricalEnergyConsumption	District	Energy
τ_7	SOXIndicator	Building	Environment

Figure 4.3: Representation of preference evaluation over a subset of MDDs through a Better-Than Graph (BTG). The table in the figure reports an excerpt of the description of the MDDs (spatial dimension and domain of indicator).

$$W_{\mathbf{P}}(\mathcal{T}) = \{\tau_i \in \mathcal{T} | \neg\, \exists\, \tau_j \in \mathcal{T} \text{ s.t. } \tau_i <_{\mathbf{P}} \tau_j\}, i \neq j$$

Noteworthy, as demonstrated in [16], the application of the winnow operator returns always a non-empty set of best MDDs (for the details, refer to the demonstration provided in the article). The application of the winnow operator over the representation of (a portion of) the Smart City Exploration Graph as a set of MDDs will provide the *first best* MDDs (located in the topmost level of the graph depicted in Figure 4.3). Nevertheless, the iterated application of the winnow operator can lead to the identification of the *second best* MDDs, the *third best* MDDs and so forth (the second and the third level of the BTG in Figure 4.3). Generally speaking, given a preference $\mathbf{P}$, the n-th iteration of the winnow operator applied to the set of MDDs $\mathcal{T}$ is defined as:

$$W^1_{\mathbf{P}}(\mathcal{T}) = W_{\mathbf{P}}(\mathcal{T})$$
$$W^{(n+1)}_{\mathbf{P}}(\mathcal{T}) = W_{\mathbf{P}}(\mathcal{T} \setminus \bigcup_{1 \leq i \leq n} W^i_{\mathbf{P}}(\mathcal{T}))$$

For instance, $W^{(2)}_{\mathbf{P}}(\mathcal{T})$ returns the second best MDDs from $\mathcal{T}$. In this respect, the

iterated application of the winnow operator to the set $\mathcal{T}$ induces a *rank* over the MDDs.

4.3 Preference evaluation algorithms

In literature, preference evaluation is also referred to as Skyline Problem [12] and is aimed at filtering the input, thus keeping only those objects that are not worse than any other. In this section, we describe to two widely-adopted algorithms for preference evaluation. In the scope of this thesis, we will leverage such algorithms to estimate preference selectivity and complexity, thus assessing the feasibility of our approach when considering different preference expressions, as it will be discussed in Section 4.6. In our approach, we feed a preference evaluation algorithm with the set of MDDs available in the exploration context selected by the user ($\mathcal{T}^r$, being ctx^r the active exploration context), in order to identify a subset of $\mathcal{T}^r$, containing only the interesting MDDs. A MDD is interesting if it is not dominated by any other MDD.

4.3.1 Block-Nested-Loops (BNL)

To compute the Skyline, authors in [12] proposed an algorithm that, rather than considering a data point (MDD) at a time, produces a block of Skyline MDDs in every iteration. It is called *block-nested-loops algorithm* (BNL, Algorithm 3) and it can be applied to any query (i.e., regardless of its complexity in terms of composing constructors). As demonstrated in [12], in the best case, the complexity of the algorithm is of the order $O(n)$, being n the number of MDDs in the input whereas in the worst case, the complexity is of the order of $O(n^2)$. Roughly speaking, the block-nested-loops algorithm repeatedly reads the set of MDDs, keeping track of a set S of incomparable tuples (i.e., not better and not worse than others). When an MDD τ_i is read from the input set, it is compared to all tuples in S and, based on this comparison, τ_i is either eliminated or placed into S. Three cases can occur:

- τ_i is dominated by a MDD within S. In this case, τ_i is discarded and will not be considered in future iterations.
- τ_i dominates one or more MDDs in S. In this case, these MDDs are eliminated from S and will not be considered in future iterations; τ_i is inserted into S.
- τ_i is incomparable with all MDDs in S. In this case, τ_i is inserted into S.

Algorithm 3: Block-Nested-Loops (BNL) algorithm

Input : Set of MDDs $\mathcal{T}^r \subseteq \mathcal{T}$ for the context ctx^r, preference expression $\overline{\mathbf{P}}$
Output: Skyline MDDs S

```
# Initialise set of best MDDs according to P̄
S ← ∅;
τ_1 ← first(T^r);
# Initialise skyline with first MDD
S ← S ∪ τ_1;
# Loop through the rest of MDDs
foreach τ_i ∈ T^r \ {τ_1} do
    to_drop ← ∅;
    dominated ← False;
    foreach τ_j ∈ S do
        # Check whether τ_i is preferred (dominated) with respect to τ_j
          according to P̄
        if τ_i <_P̄ τ_j then
            dominated ← True;
            break;
        if τ_j <_P̄ τ_i then
            to_drop ← to_drop ∪ τ_j;
    if dominated = True then
        continue;
    S ← S \ to_drop;
    S ← S ∪ τ_i;
return S;
```

4.3.2 Sort-Filter-Skyline (SFS)

To overcome certain demonstrated limitations and inefficiencies of BNL (e.g., no optimisations applicable, no optimal number of passes, no ordering of objects which is potentially useful for driving the query process), the Sort-Filter-Skyline (in brief, SFS) algorithm was introduced. Authors in [17] demonstrated that any total order of the input (in our case, the MDDs available for the exploration context) with respect to any monotone scoring function (ordered from highest to lowest score) is a topological sort with respect to the skyline dominance partial relation. This guarantees that, being $\mathcal{F}$ a scoring function and τ_i and τ_j two MDDs, $\mathcal{F}(\tau_i) \geq \mathcal{F}(\tau_j)$ implies that τ_j does not dominate over τ_i ($\tau_j \not\succ \tau_i$) The name of the algorithm denotes that,

before applying BNL on the set of MDDs, they are sorted according to a topological order. Amongst the advantages presented in [17] (i.e., optimal number of passes of the algorithm over the input) the *pre-sorting* of the input ensures that points in the skyline can be progressively returned, without having to wait for all the input to be read. Nevertheless, this algorithm has three main drawbacks: (i) it can be employed only for preference expressions containing Pareto composition; (ii) it is suited for numerical data with high cardinality domains; (iii) the choice of the monotonic sorting function. Regarding the latter point, a sorting function should limit the number of MDDs to be read; examples of sorting functions reported in [17] regard the entropy of the input, the volume of the points (multiplication of values of attributes) and the sum of the points.
Noteworthy, in our approach preferences are applied on MDDs (specifically, on the Smart City Exploration Graph concepts), whose components have a categorical domain (that is, preferences are applied only on a finite and limited set of values, such as the levels of a dimension hierarchy). In the case of a compound Pareto preference expression $\mathbf{P}_1 \otimes \mathbf{P}_2 \otimes \ldots \otimes \mathbf{P}_n$, the overall score is computed by summing the single score induced by each preference $\mathbf{P}_i$, the latter belonging to the range $[0, 1]$. Considering a single preference $\mathbf{P}_i$, the descending score depends on the type of base preference constructor in $\mathbf{P}_i$. Specifically, given a MDD τ_i and a base preference constructor $\mathbf{P}_i$:

- for DOM(d) preference the score will be 1 if the domain of indicator $\tau_i.ind_i$ is equal to d, 0 otherwise;
- for DEP(ind_d) preference the score will be 1 if the indicator $\tau_i.ind_i$ depends on ind_d, 0 otherwise;
- for LEV(dim, lev_d) preference the score is $\frac{(n-dist)}{n}$ being $n = Levels(dim) - 1$ and $dist$ the distance of $\tau_i.L_i[dim]$ from lev_d (absolute value);
- for IND(ind) preferences, the score is calculated similarly to LEV but considering the indicators hierarchy in the Smart City Exploration Graph.

4.4 Procedure for indicators exploration

Personalised exploration of Smart City indicators, based on users' contexts previously described, is articulated over four main steps, as shown in Figure 4.4. To present the four steps, let us consider Alice, a citizen living in a Smart City, interested in monitoring air pollution levels to decide whether or not to practise outdoor activities, since pollution has effect on this kind of activities. Alice is just one of the users, belonging to different categories, who are interested in exploring urban data for

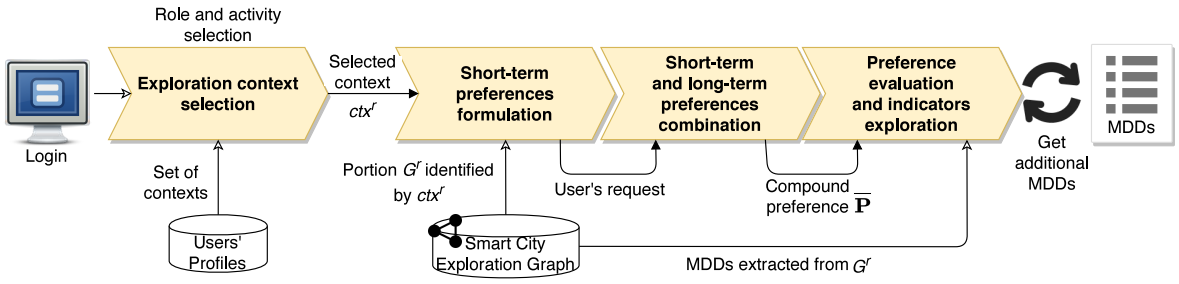


Figure 4.4: Summary of the steps of the indicators exploration procedure.

different purposes. Figure 4.5 shows the front-end Web application[1] that supports Alice during the exploration.

4.4.1 Exploration context selection

The input of this step is the set of contexts associated with the user's profile $p(u)$, stored in a Users' Profiles Database. After log in, the exploration platform proposes the user to select one of the roles available in her profile. Role selection inherently enables only the contexts meant for the chosen role. Lastly, the selection of the activity the user is willing to perform completes the identification of the exploration context ctx^r. Figure 4.5(a) shows the selection of the `AirPollutionMonitoring` activity by Alice, whose profile is associated with the citizen role only. This enables the platform to filter out activities that are not designed for Alice, such as planning timely corrective actions in the case air pollution levels overtake warning thresholds (which is in charge of a different category of users, e.g., the municipality environmental engineers). Moreover, this in turn will focus the exploration on indicators associated with the Alice's context only, identifying a portion of the Smart City Exploration Graph, denoted as $\mathcal{G}^r$, and avoiding Alice to get lost in the exploration of the whole graph $\mathcal{G}$.

4.4.2 Short-term preferences formulation

This step takes as input the context ctx^r previously selected by the user and the portion $\mathcal{G}^r$ of the Smart City Exploration Graph. A list of target indicators and analysis dimensions is proposed to the user for selection. For instance, only indicators and analysis dimensions available for citizens performing air pollution monitoring are proposed to Alice. When Alice chooses the desired indicators, domains and dimensional levels, the concepts are mapped by the platform to base constructors.

[1]The demonstration video of the tool is available at `https://tinyurl.com/pref-app-demo`

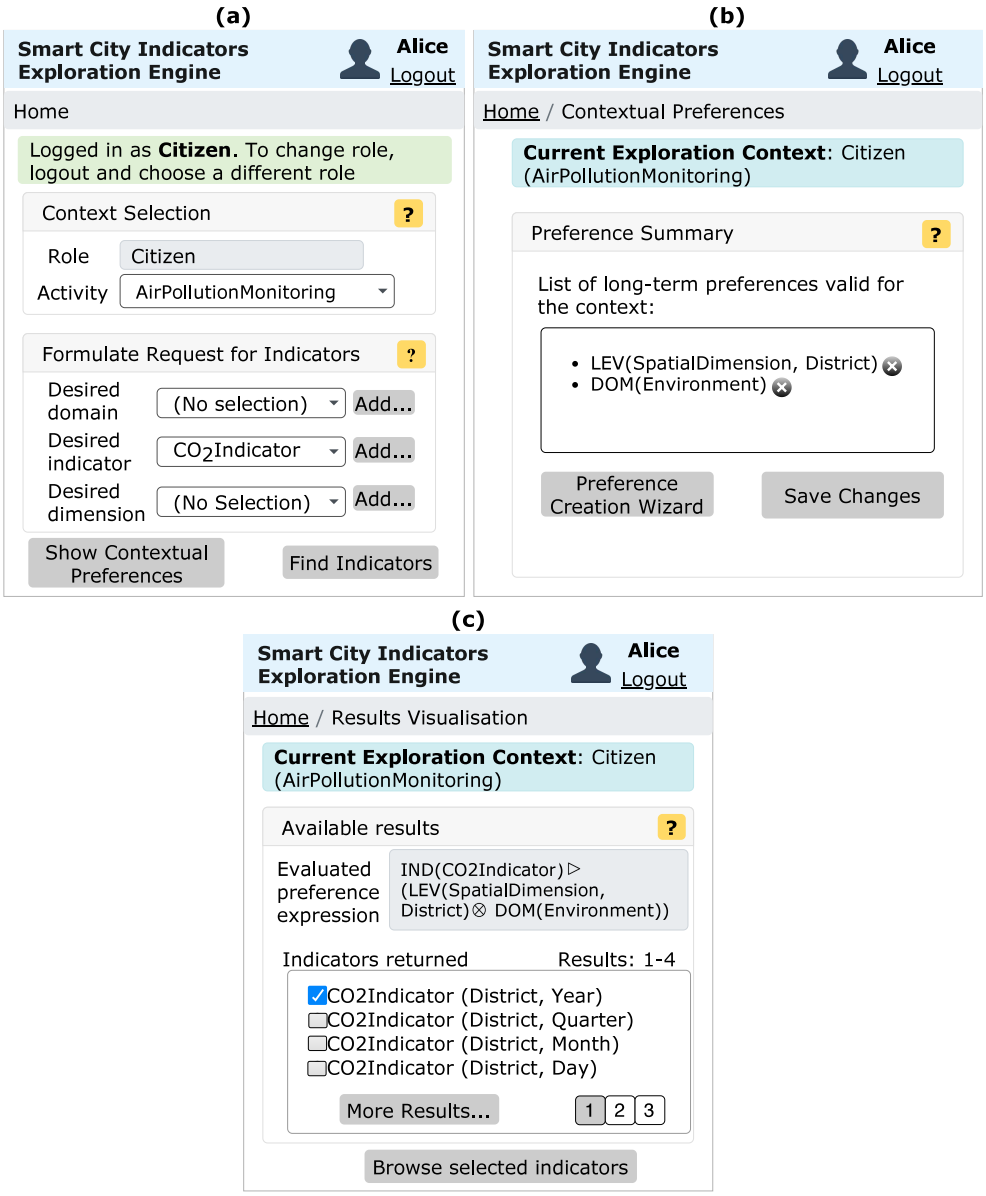


Figure 4.5: GUI for personalised indicators exploration.

In particular, indicator concepts are mapped to IND constructors, domain concepts to DOM constructors and dimensional level concepts to LEV constructors. For example, in Figure 4.5(a), when Alice selects the `CO2Indicator`, the corresponding preference constructor IND(`CO2Indicator`) will be automatically included in the request, thus alleviating Alice from the burden of manually specifying the preference constructor. The obtained constructors in Alice's request constitute the short-term

preferences, i.e., concerning to the current request only, for the current exploration context.

4.4.3 Short-term and long-term preferences combination

The input of this phase is the context ctx^r and the request from the former step, containing the above mentioned short-term preferences. Short-term preferences in the request are combined with long-term preferences in the profile $p(u)$ of the user. The result is the compound preference $\overline{\mathbf{P}}$. For example, during Alice's exploration, long-term preferences DOM(`Environment`) and LEV(`SpatialDimension`, `District`) are considered, since they are valid for the target context ctx^r. Long-term preferences are automatically combined using the Pareto composition operator, since they all assume an equal importance for Alice. Long-term preferences are combined with short-term ones from the previous step, according to the prioritization operator, as they address an immediate need. Long-term preferences are summarised as reported in Figure 4.5(b) and can be accessed by clicking on the "Show Contextual Preferences" button. Further long-term preferences can be added by Alice to her profile through the "Preference Creation Wizard" button. Broadly speaking, long-term preferences in $p(u)$ can be inferred and mined by analysing exploration habits of users, collected over time through the interactions with the platform. However, the automatic retrieval of long-term preferences is out of the scope of this thesis, since the goal here is on the exploration procedure. We will assume that this type of preferences are somehow included in the profile of the user, either manually or not. Lastly, after the request formulation has been finalised, Alice confirms her choices by clicking the "Find Indicators" button.

4.4.4 Preference evaluation and indicators exploration

This step takes as input the compound preference $\overline{\mathbf{P}}$ and the representation of the portion of Smart City Exploration Graph $\mathcal{G}^r$ as a set of MDDs $\mathcal{T}^r$. The preference $\overline{\mathbf{P}}$ undergoes an evaluation process, that identifies a subset of MDDs not worse than others, which in the literature is defined as the *Best Match Only* result [24] (in brief, BMO). In particular, BMO corresponds to the set of best (optimal) MDDs according to the preference $\overline{\mathbf{P}}$ (recalling the winnow operator introduced in Section 4.2.2.2, this is equivalent to calculate $W^1_{\overline{\mathbf{P}}}(\mathcal{T}^r)$). The MDDs in the BMO set are proposed to the user, who can select any of them to explore indicator values. For example, Alice's preference evaluation result is displayed in the first page of the list in Figure 4.5(c) and contains MDDs with `CO2Indicator`, because of IND(`CO2Indicator`) and DOM(`Environment`) preferences, and `District` level, because of the LEV(`SpatialDimension`, `District`) long-term preference. However,

Alice may request for additional (sub-optimal) MDDs (e.g., to receive suggestions of other indicators that are compliant with her exploration goals), which are ranked lower by the evaluation procedure. In fact, by clicking on the "More Results" button, MDDs with `CO2Indicator` and `City/Building` level will be listed and so forth. Therefore, the evaluation of preference $\overline{\mathbf{P}}$ induces an order over the MDDs in $\mathcal{T}^r$ and each MDD earns an implicit rank, quantifying its distance from the optimum (that is, the BMO set contains the first-best MDDs). For example, MDDs associated with `City/Building` levels are characterised by an increasing distance from the `District` level. In this way, Alice may inspect also other MDDs, having increasing distance from the optimum (obtained by calculating $W_{\overline{\mathbf{P}}}^{(i+1)}(\mathcal{T}^r)$). This is particularly useful in case the obtained first-best MDDs are few and Alice wants to explore additional results. Finally, Alice selects one or more MDDs and after clicking on the "Browse selected indicators" button, the multi-dimensional query apt to retrieve indicators values will be issued over the underlying Semantic Data Lake.

4.5 Implementation

This section presents the services (namely, *User Profile* and *Personalisation* services) supporting personalised indicators exploration; they are invoked by the front-end Web application introduced in Section 4.4. The *User Profile* services have been developed using PHP 7.3 scripting language and profile data is retained in a MySQL database. Moreover, the ecosystem of services implementing personalised data exploration functionalities (constituting the *Personalisation* services) is deployed under the Java Apache Tomcat application server. Remarkably, the front-end for personalised indicators exploration (Figure 4.5) and the services described in this section cover the functionalities meant for the *End-user application modules* at the topmost level of the three-layered model presented Section 1.3.

4.5.1 Overview and interaction flow

Figure 4.6 details the structure of modules and services in the topmost layer of the informative model introduced in Section 1.3. Numbers in Figure 4.6 give the order of the interaction flow. The *Preference Composer Service* is in charge of elaborating the request issued by the user (1), combining short-term preferences from the request with long-term ones, valid for the current context (2). The combined preference expression is sent to the *Preference Evaluation Service* (3), which retrieves, through the MDDs Generator module, the MDDs from the Smart City Exploration Graph Database (4). In particular, the *Preference Evaluation Service* implements the preference evaluation algorithm relying on the BMO paradigm discussed in the previous section, involving the MDDs defined over the Smart City Exploration Graph. The Smart

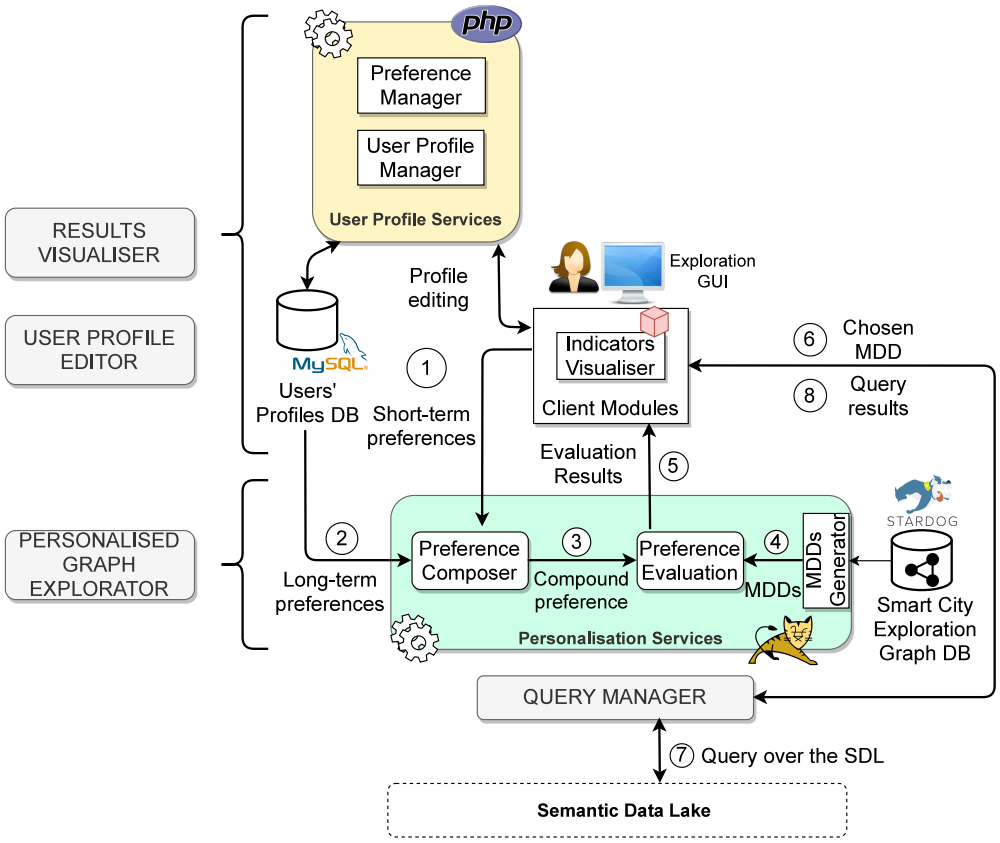


Figure 4.6: Overview of the services for personalised indicators exploration and their interaction with the modules of the three-layered model.

City Exploration Graph Database is a Stardog[2] Triplestore, containing the Smart City Exploration Graph definition in OWL language. Preference evaluation results are then pushed to the front-end and presented to the user (5) who can select the indicator she wants to visualise. The chosen MDD is sent to the *Query Manager* (6) which composes the multi-dimensional query (7) apt to retrieve the values of the indicator. At this point, indicator values can be visualised and navigated invoking external Business Intelligence platforms services, with the support of the *Indicators Visualiser* module (8).

4.5.2 User Profile services

User Profile services (i.e., Preference Manager and User Profile Manager) are in charge of managing preferences contained in the profile (long-term preferences), which are

[2] `https://www.stardog.com/`

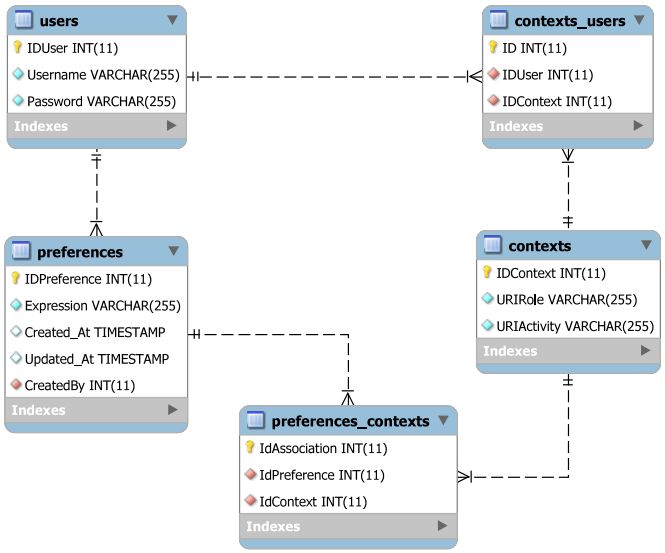


Figure 4.7: Data model for users' profiles database.

stored into a MySQL database, retaining also other information related to users' profiles. Both services are implemented in PHP, where proper classes are devoted to connect to the database and to perform CRUD operations (invoked by the front-end application whenever it is necessary to access profiles' content).
Exploration contexts and preferences are stored in profiles, which in turn are persisted in the *Users' Profiles DB*, whose data model is depicted in Figure 4.7. It is composed of 5 tables:

- `users`, containing information to access the exploration platform.
- `contexts`, containing information about exploration contexts, in terms of roles and activities, represented by the URI associated with the concept in the Smart City Exploration Graph.
- `preferences`, containing the preferences expressions, along with the reference to the users who created them through an assisted procedure (called by the "Preference Creation Wizard" button in Figure 4.5(b)). The latter reference may be empty in the case of preferences not defined by a user, but prepared by data analysts (see the details of contexts creation in Section 4.2.1).
- `contexts_users`, expressing the available exploration contexts for a certain user.
- `preferences_contexts`, expressing the applicability of preferences for exploration contexts. Indeed, in our approach we deal with contextual preferences, valid only within specific contexts.

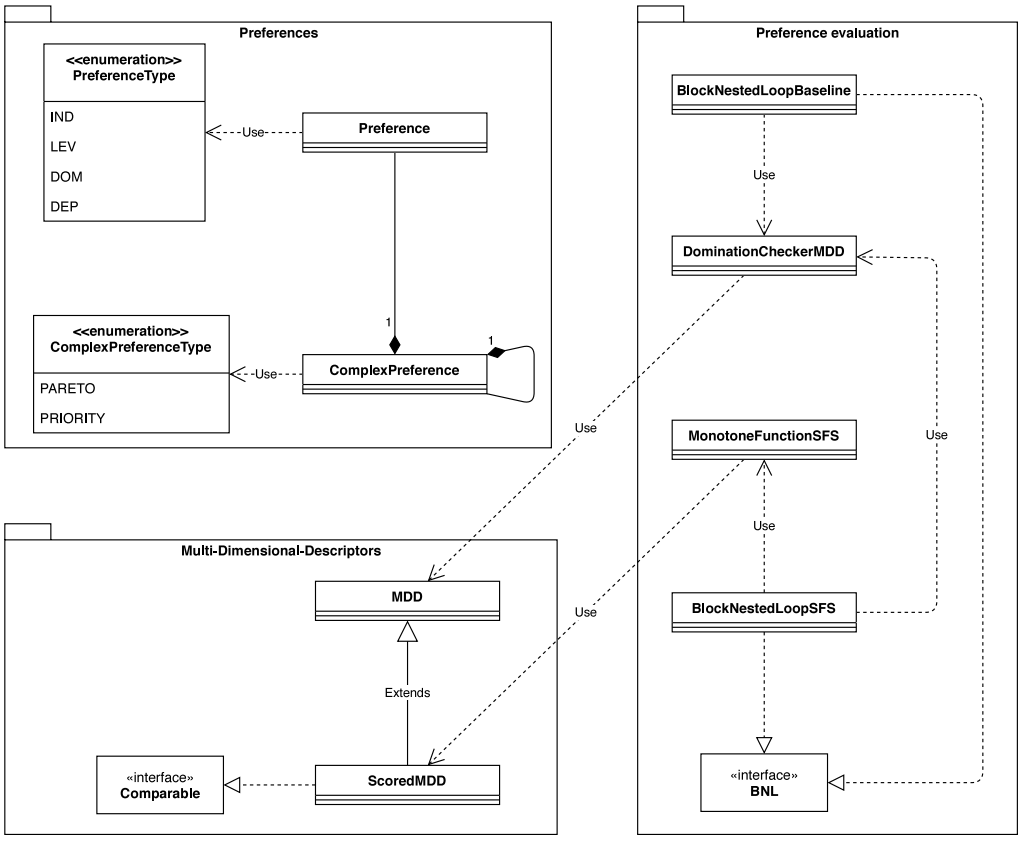


Figure 4.8: UML package diagram for Java classes that implement Personalisation Services.

4.5.3 Personalisation services

Personalisation services (i.e., Preference Composer and Preference Evaluation) receive as input (if specified) the desired indicators, domains and dimensions levels specified by the user during request formulation (that is, the short-term preferences). Indicators are mapped to IND constructors, domains to DOM constructors and dimensional levels to LEV constructors. As mentioned in the beginning of Section 4.5, these services are implemented in Java; hereafter, we are going to describe the Java classes (whose high-level UML diagram is reported in Figure 4.8) exploited by such services.

Preferences. This package contains two main classes: `Preference` and `ComplexPreference`, which are apt to create preference objects. A `Preference` object contains as attributes the *type* of preference (specified through the enum `PreferenceType`, defining four constants for each of the base preference constructors of our approach) and a vector of *arguments*, depending on the type of constructor (apart

from LEV which requires two arguments, the other constructors demand for only one argument). Likewise, a `ComplexPreference` object has a *type* (either Pareto or prioritisation binary operator, specified through the enum `ComplexPreferenceType`) and a vector of two objects representing the arguments of the complex preference (defined as `Object`, as they could be base preferences or, in turn, complex preferences).

Multi-Dimensional Descriptors. This package contains two classes. The main class is `MDD`, abstracting the structure of a MDD from the Smart City Exploration Graph and containing as attributes the URI of the indicator and a vector of URIs, one for each level of the involved dimensions. Conversely, the `ScoredMDD` class extends the `MDD` class by providing also a score value, which is exploited by the pre-sorting preference evaluation algorithm SFS.

Preference Evaluation. This package contains a set of classes devoted to implement the preference evaluation algorithms described in Section 4.3. The class `DominationCheckerMDD` contains the methods apt to assess whether, given two MDDs τ_a and τ_b and a (complex) preference $\overline{\mathbf{P}}$, τ_a dominates τ_b according to $\overline{\mathbf{P}}$ (denoted as $\tau_a \succ_{\overline{\mathbf{P}}} \tau_b$). Moreover, the `BlockNestedLoopBaseline` retains the methods implementing the BNL algorithm whose pseudo-code is reported in Algorithm 3. On the other hand, the classes `MonotonicSumFunctionSFS` and `BlockNestedLoopSFS` are meant for the implementation of the SFS algorithm. Specifically, the former class contains the methods needed to calculate the score to assign to a MDD (according to the input preference). Scores will be exploited in the latter class as a way to ensure the pre-sorting of the MDDs, before the BNL preference evaluation algorithm is executed.

4.6 Experiments

In this section, we report the evaluation of our approach whose aim is two-fold: (i) demonstrating the effectiveness of preferences in pruning the search results, thus increasing the selectivity of best matches using preferences while, at the same time, avoiding empty results; (ii) assessing preference complexity, when dealing with different preference expressions.

4.6.1 Dataset and experimental setup

To validate our approach, we used the Smart City Exploration Graph (denoted as $\mathcal{G}_{ISO}$) derived from the description of indicators according to the ISO 37120 standard[3]

[3]The TBox of $\mathcal{G}_{ISO}$ can be found at `https://tinyurl.com/sceg-iso` (a free Web Protégé account is required).

described in Section 3.7.1. $\mathcal{G}_{ISO}$ contains the semantic description of 223 indicators and, amongst them, 100 are composite. In this respect, the set of MDDs obtained from the graph $\mathcal{G}_{ISO}$ (denoted as MDD_{ISO}) contains $\approx$ 3000 items, and will be the exploited for experiments regarding both preference selectivity and complexity. Experiments have been conducted on a Windows PC equipped with an Intel Core i5-3210M processor, CPU 2.50 GHz, 4 cores, 8 logical cores, RAM 8GB. Noticeably, for the experimentation, the set MDD_{ISO} has been materialised and average generation time for the set was $2s$.

4.6.2 Metrics

As previously mentioned, in the experiments, we focused on preference selectivity and preference complexity. Specifically, to assess the behaviour of the preference evaluation algorithm and with reference to preference expressions as defined in Section 4.2 and the structure of the Exploration Graph, the *preference selectivity* σ has been computed, defined as the ratio between the number of MDDs not worse than others (the BMO result) and the cardinality of the whole set MDD_{ISO}. The lower is the σ value, the better is the behaviour of the algorithm. Concerning preference complexity, the *processing time* t has been calculated, which takes into account the time required to process the queries, in turn depending on the number of MDDs to be accessed, in order to find the BMO.

4.6.3 Experiments on preference selectivity

The workload for selectivity tests consisted of a set of 50 preference expressions prepared using different combinations and numbers of base constructors (from 1 to 5). Specifically, we focused on three representative constructors of our approach, that is, DOM, IND and LEV, combined using both Pareto and Prioritised composition operators. Indeed, these three constructors are the same that can be used to formulate short-term preferences in the request for indicators, from the front-end application described in Section 4.4 The preference queries obtained can be classified into seven groups, depending on the type of base constructors they include. The detailed summary of preference queries is reported in the table of Figure 4.9, along with the average preference selectivity (denoted with $\overline{\sigma}$). According to the results, expressions containing only DOM constructors led to an average selectivity of 41%, expressions with only LEV to 29.2%, whereas in the case of expressions with both DOM and LEV the average selectivity was 8%, thus demonstrating that the two constructors are less selective when applied alone. Noteworthy, the average selectivity of expressions with only IND reaches 1.3%, since the size of the BMO is very restricted, thus leading to high selectivity degrees. The latter generally holds for queries containing IND

	DOM	IND	LEV	$\bar{\sigma}$
Q_D	1÷5	0	0	41%
Q_I	0	1÷5	0	1.3%
Q_L	0	0	1÷2	29.2%
Q_{DI}	1÷3	1÷2	0	11.8%
Q_{IL}	0	1÷3	1÷2	0.2%
Q_{DL}	1÷3	0	1÷2	8%
Q_{DIL}	1÷2	1	1÷2	1.6%

Figure 4.9: Preference selectivity average values ($\overline{\sigma}$) for preference expressions (queries) containing DOM, IND and LEV constructors.

constructors, as can be evinced from the table in Figure 4.9. The high percentages of average selectivity of DOM and LEV are explained by the fact that in $\mathcal{G}_{ISO}$ each domain gathers several indicators and that dimensional hierarchies may be shared by multiple indicators. Hence, DOM and LEV will always find a larger BMO with respect to IND, which focuses on a specific indicator.

4.6.4 Experiments on preference complexity

This kind of tests aims at assessing the relationship between the performance of the preference evaluation algorithm and the complexity of preference expressions. In this case, a set of 30 preference expressions composed of the three aforementioned preference constructors has been prepared. Specifically, for each constructor, an expression has been built by including from one to five times the same constructor, composed through Pareto and Prioritised composition operators, respectively. Figure 4.10 shows the average response time required to process the queries, for an increasing number of base preference constructors, distinguishing between the two different types of composition operators. Broadly speaking, the performance of a preference query (regardless of the constructors involved) depends on the number of MDDs to be accessed, in order to find the BMO. In the worst case, the time complexity of the preference evaluation algorithm is $O(n^2)$, being $n = |\text{MDD}_{ISO}|$, which, for the analysed domain, is acceptable (from the tests, the maximum time spent for evaluating a preference expression was ≈ 4568 ms, in a Pareto composition involving five IND constructors). Overall, it can be noticed from the charts in Figure 4.10 that the execution time when using Prioritised composition tends to be lower than Pareto composition, due to the fact that the cardinality of the BMO set is necessarily smaller. IND processing times are the highest among the three considered constructors, as they comprise the time required to inspect and navigate the indicators hierarchy in

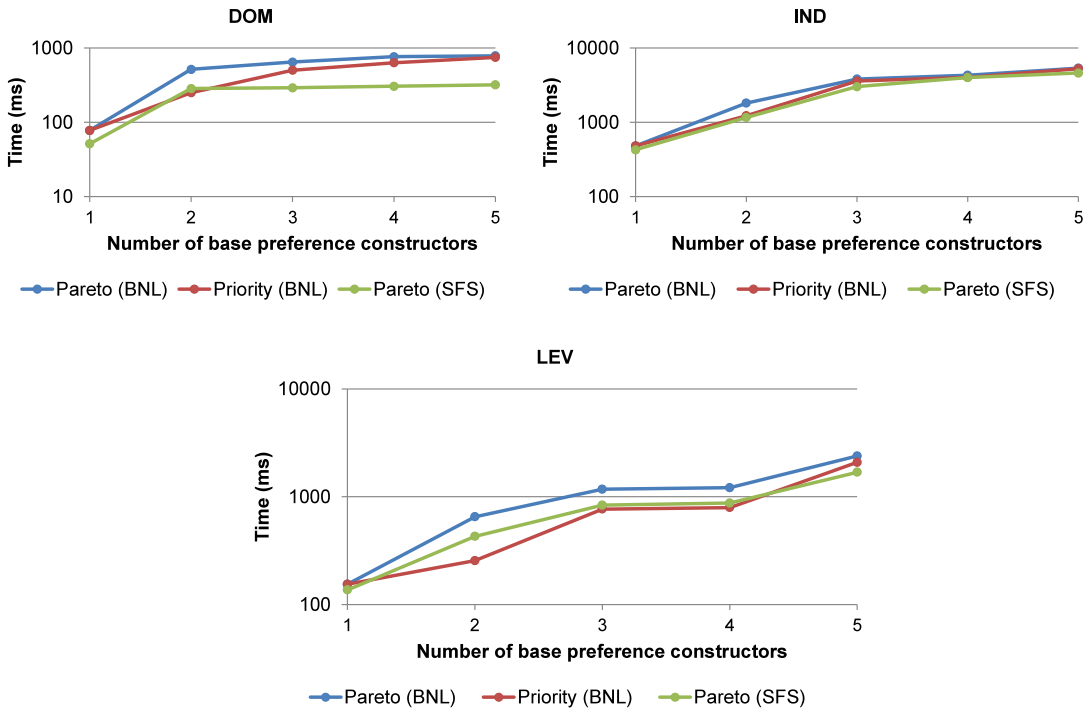


Figure 4.10: Average processing times (log scale) for increasing preference complexity for DOM, IND and LEV constructors.

$\mathcal{G}_{ISO}$. Regardless of the type of constructor, applying the pre-sorting on MDDs (i.e., when using SFS algorithm) in $\mathcal{G}_{ISO}$ ensures a reduction in evaluation times for Pareto composition (blue vs green lines in the charts of Figure 4.10). The time required to sort the MDDs in $\mathcal{G}_{ISO}$ was on average ≈ 8 ms.

Chapter 5

Conclusions

In this thesis, we proposed a semantics-enabled approach to support personalised Data Lakes exploration. Specifically, we developed a model articulated over three layers for personalising Smart City data exploration. Each layer has been conceived to progressively enrich the organisation of data starting from a Data Lake, in order to enable its personalised exploration. Firstly, proper Semantic Models are defined on top of smart city data sources, meant for semantically describe their content, thus ensuring a unified view of data, according to Semantic Web technologies, and overcoming heterogeneity issues. This brings to what has been denoted as *Semantic Data Lake* (Chapter 2). Secondly, concepts and relationships of the Semantic Models are used to semantically define indicators and analysis dimensions leveraged by users to explore smart city data (Chapter 3). Lastly, users' profiles and preferences on indicators are exploited to personalise the exploration experience, providing users with indicators more related to their search interests, the activities they perform and the role they cover in the Smart City (Chapter 4). Hereafter, we summarise the original contributions of this thesis, by providing directions for future works.

In **Chapter 2** we proposed our Data Lake model and the fundamentals at the basis of the construction of a semantic overlay upon it. The semantic overlay is constructed by domain experts starting from data sources attributes, which are lexically enriched. Then, each lexically enriched attribute undergoes an annotation process, whereby concepts used for annotation are drawn from a set of domain ontologies. With respect to existing efforts aimed at modelling heterogeneous data sources in terms of their attributes and mutual relationships, to ease the subsequent query process, we move forward by introducing lexical enrichment for data sources attributes and providing a semantic support. On the one hand, lexical enrichment of data sources helps reducing the terminological distance from data source attributes names (often expressed through abbreviations or acronyms) and the names of concepts used for annotation. On the other hand, semantic annotation through domain ontologies

ensures a formal representation of the meaning associated with data source attributes, with the goal of favouring sharing, reusability and interoperability, which are the goals of the Semantic Web initiative. As a future work, we plan to improve the semi-automatic tool we devised for supporting domain experts in the management of Data Lake sources (DL-DIVER). A particular concern will be given to deliver more interactivity in the procedure devoted to the construction of the Semantic Models, especially focusing on the choice of candidate relationships to be added to the Semantic Model. To fulfil this goal, a notion of quality (based on one or more metrics) will be introduced to give domain experts feedbacks while editing Semantic Models, for instance to suggest corrective actions to improve their design (e.g., remove unused relationships or other common pitfalls regarding relationships as the ones reported in [54]). Moreover, a compelling future research direction is constituted by the possibility of enriching the semantic overlay with metadata related to provenance. This has been effectively proposed in Data Lake environments [65], as a way of tracking information about the activities, entities and people involved in producing a data product stored in the Data Lake, improving of the ability to trace errors back to the root cause when performing data analytics processes.

In **Chapter 3** we defined the *Multi-Dimensional Ontology* (MDO), containing baseline concepts and relationships for modelling indicators, dimensions and roles and activities of Smart City users (the latter referred to as personalisation concepts). Data analysts leverage MDO concepts and semantic relationship to create indicators and analysis dimensions, following a procedure enforced by *validation rules*, since modelling is an error-prone task, especially when dealing with an increasing number of indicators. The conceptualisation of indicators and dimensions, enriched with personalisation elements, constitutes the *Smart City Exploration Graph* (SCEG) on top of the Semantic Models, whose representation as a set of Multi-Dimensional Descriptors (concisely representing indicators together with available dimensions) becomes the starting point for users to explore Smart City data. In the last part of the chapter, the steps of the exploration procedure, targeted to retrieve indicators values from Data Lake sources (*querying*), have been formalised. Actual indicators values can be obtained through *mappings sets*, defined by the data analysts creating *mappings* to associate the definition of an indicator (as of in the Smart City Exploration Graph) to concepts and relationships in the semantic overlay. As a future work, we plan to provide a semi-automatic support for the definition of mappings for indicators, supporting data analysts during this process. In fact, in the current version of the prototype system, there is little support for data analysts while defining mappings from concepts and relationships in the Smart City Exploration Graph to those in the semantic overlay. The aforementioned tool would foster proper metrics to assess the strength of a mapping, for instance resorting to the calculation of its aggregation power [50] (which can be interpreted analogously to the notion of functionality

between facts and dimensions in traditional Data Warehouse environments). In this respect, the formal definition of mapping could be enhanced by introducing a confidence value, thus highlighting the ones with the lowest confidence values, which therefore are candidate to be discarded.

In **Chapter 4** we provided the formalisation of a semi-automatic methodological approach to support personalised exploration of Smart City indicators, defined within the Smart City Exploration Graph. Specifically, this part is grounded on two main pillars. The first one is constituted by the *users' profiles*, composed of *preferences*, expressed through constructors exploiting the structure of the Smart City Exploration Graph. Preferences are organised according to *contexts*, gathering information characterising the situation under which the user explores indicators, influenced by both his/her roles and goals. The second pillar is a four-steps procedure to achieve *personalised exploration* of Smart City indicators, leveraging the profiles in order to derive a set of indicators to be proposed to the user. As a future research direction, we will enhance the preference model by considering the propagation of preferences across exploration contexts, as proposed by [18]. This entails to consider the formalisation of a propagation operation, to establish how preferences holding in a more generic exploration context are propagated to a more specific context. Such propagation operation should take into account the type of preference constructor and its arguments and would leverage the contexts hierarchy induced by the presence of is-a relationships between activities concepts in the Smart City Exploration Graph. Moreover, we will deepen the integration of self-service tools (e.g., Power BI[1]) with the Apache Spark environment (implementing indicators calculation from Data Lake sources), providing interactive visualisations and Business Intelligence capabilities, with an interface simple enough for users to create their own reports and dashboards for exploring Smart City indicators.

[1] `https://powerbi.microsoft.com/en-us/`

References

[1] V. Albino, U. Berardi, and R. M. Dangelico. "Smart cities: Definitions, dimensions, performance, and initiatives". In: *Journal of urban technology* 22.1 (2015), pp. 3–21.

[2] S. de Amo and M. R. Ribeiro. "CPref-SQL: A query language supporting conditional preferences". In: *Proceedings of the 2009 ACM symposium on Applied Computing (SAC 2009)*. 2009, pp. 1573–1577.

[3] E. A. E. H. Amor and S. A. Ghannouchi. "Toward an ontology-based model of key performance indicators for business process improvement". In: *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*. 2017, pp. 148–153.

[4] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, et al. "Spark SQL: Relational data processing in spark". In: *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 2015, pp. 1383–1394.

[5] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, D. Nardi, et al. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[6] A. Bagozi, D. Bianchini, V. De Antonellis, M. Garda, and M. Melchiori. "Personalised Exploration Graphs on Semantic Data Lakes". In: *27th Int. Conference on Cooperative Information Systems (CoopIS 2019)*. Rhodes, Greece, 2019, pp. 22–39.

[7] A. Beheshti, B. Benatallah, R. Nouri, and A. Tabebordbar. "CoreKG: a Knowledge Lake Service". In: *PVLDB* 11.12 (2018), pp. 1942–1945.

[8] D. Bianchini, V. De Antonellis, M. Garda, and M. Melchiori. "A Methodological Approach for enabling Personalised Smart City Data Exploration". In: *2020 IEEE International Smart Cities Conference (ISC2 2020)*. 2020, pp. 106–111.

[9] D. Bianchini, V. De Antonellis, M. Garda, and M. Melchiori. "Contextual Preferences to Personalise Semantic Data Lake Exploration". In: *31st International Conference on Database and Expert Systems Applications (DEXA 2020)*. Bratislava, Slovakia, 2020.

[10] S. E. Bibri. "The anatomy of the data-driven smart sustainable city: instrumentation, datafication, computerization and related applications". In: *Journal of Big Data* 6.1 (2019), p. 59.

[11] E. P. Bontas, M. Mochol, and R. Tolksdorf. "Case studies on ontology reuse". In: *Proceedings of the IKNOW05 International Conference on Knowledge Management*. Vol. 74. 2005, p. 345.

[12] S. Borzsony, D. Kossmann, and K. Stocker. "The skyline operator". In: *Proceedings 17th International Conference on Data Engineering (ICDE)*. 2001, pp. 421–430.

[13] S. Castano and V. De Antonellis. "A schema analysis and reconciliation tool environment for heterogeneous databases". In: *Proceedings. IDEAS'99. International Database Engineering and Applications Symposium*. 1999, pp. 53–62.

[14] S. Castano and V. De Antonellis. "Global viewing of heterogeneous data sources". In: *IEEE Transactions on Knowledge and Data Engineering* 13.2 (2001), pp. 277–297.

[15] S. Chauhan, N. Agarwal, and A. Kar. "Addressing big data challenges in smart cities: a systematic literature review". In: *Info* 18.4 (2016), pp. 73–90.

[16] J. Chomicki. "Preference formulas in relational queries". In: *ACM Transactions on Database Systems (TODS)* 28.4 (2003), pp. 427–466.

[17] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. "Skyline with Presorting: Theory and Optimizations". In: *Intelligent Information Processing and Web Mining*. 2005, pp. 595–604.

[18] P. Ciaccia, D. Martinenghi, and R. Torlone. "Foundations of Context-aware Preference Propagation". In: *Journal of the ACM (JACM)* 67.1 (2020), pp. 1–43.

[19] C. Diamantini, P. Lo Giudice, D. Potena, E. Storti, and D. Ursino. "An Approach to Extracting Topic-guided Views from the Sources of a Data Lake". In: *Information Systems Frontiers* 23 (2021), pp. 243–262.

[20] C. Diamantini, D. Potena, E. Storti, and H. Zhang. "An ontology-based data exploration tool for key performance indicators". In: *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*. 2014, pp. 727–744.

[21] A. Doan and A. Y. Halevy. "Semantic integration research in the database community: A brief survey". In: *AI magazine* 26.1 (2005), pp. 83–83.

[22] Y. Gao, S. Huang, and A. Parameswaran. "Navigating the Data Lake with DATAMARAN: Automatically Extracting Structure from Log Datasets". In: *Proceedings of the 2018 International Conference on Management of Data*. 2018, pp. 943–958.

[23] M. Golfarelli and S. Rizzi. *Data Warehouse Design: Modern Principles and Methodologies*. McGraw-Hill, 2009.

[24] M. Golfarelli, S. Rizzi, and P. Biondi. "myOLAP: An approach to express and evaluate OLAP preferences". In: *IEEE Transactions on Knowledge and Data Engineering* 23.7 (2010), pp. 1050–1064.

[25] M. Gueroussova, A. Polleres, and S. A. McIlraith. "SPARQL with Qualitative and Quantitative Preferences". In: *OrdRing@ ISWC*. 2013, pp. 2–8.

[26] A. Gupta, V. Harinarayan, and D. Quass. "Aggregate-query processing in data warehousing environments". In: *Proceedings of the 21th International Conference on Very Large Databases (VLDB 1995)*. 1995, pp. 358–369.

[27] R. Hai, S. Geisler, and C. Quix. "Constance: An Intelligent Data Lake System". In: *Proc. of the 2016 International Conference on Management of Data (SIGMOD/PODS'16)*. San Francisco, California, 2016, pp. 2097–2100.

[28] R. Hai, C. Quix, and C. Zhou. "Query rewriting for heterogeneous data lakes". In: *European Conference on Advances in Databases and Information Systems (ADBIS)*. 2018, pp. 35–49.

[29] H. B. Hamadou, E. Gallinucci, and M. Golfarelli. "Answering GPSJ Queries in a Polystore: A Dataspace-Based Approach". In: *International Conference on Conceptual Modeling (ER 2019)*. 2019, pp. 189–203.

[30] M. Horridge and S. Bechhofer. "The OWL API: A Java API for OWL Ontologies". In: *Semantic web* 2.1 (2011), pp. 11–21.

[31] N. Kasrin, M. Qureshi, S. Steuer, and D. Nicklas. "Semantic Data Management for Experimental Manufacturing Technologies". In: *Datenbank-Spektrum* 18.1 (2018), pp. 27–37.

[32] W. Kießling. "Foundations of preferences in database systems". In: *Proceedings of the 28th International Conference on Very Large Databases (VLDB 2002)*. Hong Kong, China, 2002, pp. 311–322.

[33] R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, and B. Becker. *The data warehouse lifecycle toolkit*. John Wiley & Sons, 2008.

[34] H. Kondylakis, L. Koumakis, M. Tsiknakis, and K. Marias. "Implementing a data management infrastructure for big healthcare data". In: *Proc. of IEEE EMBS Int. Conf. Biomedical Health Informatics (BHI)*. 2018, pp. 361–364.

[35] K. Kritikos, D. Plexousakis, and R. Woitch. "A Flexible Semantic KPI Measurement System". In: *International Conference on Cloud Computing and Services Science*. 2017, pp. 237–261.

[36] C. Kuster, J.-L. Hippolyte, and Y. Rezgui. "The UDSA ontology: An ontology to support real time urban sustainability assessment". In: *Advances in Engineering Software* 140 (2020), p. 102731.

[37] Y. Li, R. García-Castro, N. Mihindukulasooriya, J. O'Donnell, and S. Vega-Sánchez. "Enhancing energy management at district and building levels via an EM-KPI ontology". In: *Automation in Construction* 99 (2019), pp. 152–167.

[38] I. Lytra, M. Vidal, F. Orlandi, and J. Attard. "A big data architecture for managing oceans of data and maritime applications". In: *Proc. of International Conference on Engineering, Technology and Innovation (ICE/ITMC 2017)*. Madeira, Portugal, 2017, pp. 1216–1226.

[39] A. Maccioni and R. Torlone. "KAYAK: A Framework for Just-in-Time Data Preparation in a Data Lake". In: *Proc. of 30th Int. Conference on Advanced Information Systems Engineering (CAiSE 2018)*. Tallinn, Estonia, 2018, pp. 474–489.

[40] B. Malysiak-Mrozek, M. Stabla, and D. Mrozek. "Soft and Declarative Fishing of Information in Big Data Lake". In: *IEEE Transactions on Fuzzy Systems* (2018), p. 1.

[41] M. N. Mami, D. Graux, S. Scerri, H. Jabeen, S. Auer, and J. Lehmann. "Squerall: Virtual Ontology-Based Access to Heterogeneous and Large Data Sources". In: *Proc. of 18th International Semantic Web Conference (ISWC 2019)*. Auckland, New Zealand, 2019.

[42] M. del Mar Roldán-García, J. García-Nieto, A. Maté, J. Trujillo, and J. F Aldana-Montes. "Ontology-driven approach for KPI meta-modelling, selection and reasoning". In: *International Journal of Information Management* (2019), p. 102018.

[43] M. A. Martínez-Prieto, A. Bregon, I. García-Miranda, P. C. Álvarez-Esteban, F. Díaz, and D. Scarlatti. "Integrating flight-related information into a (Big) data lake". In: *Proc. IEEE/AIAA 36th Digital Avionics Systems Conf. (DASC)*. 2017, pp. 1–10.

[44] G. Mehdi, T. Runkler, M. Roshchin, S. Suresh, and N. Quang. "Ontology-based integration of performance related data and models: An application to industrial turbine analytics". In: *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. 2017, pp. 251–256.

[45] A. Miele, E. Quintarelli, E. Rabosio, and L. Tanca. "A data-mining approach to preference-based data ranking founded on contextual information". In: *Information Systems* 38.4 (2013), pp. 524–544.

[46] G. A. Miller. "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[47] A. A. Munshi and Y. A.-R. I. Mohamed. "Data lake lambda architecture for smart grids big data analytics". In: *IEEE Access* 6 (2018), pp. 40463–40471.

[48] F. Nargesian, E. Zhu, R. J Miller, K. Q. Pu, and P. C. Arocena. "Data Lake Management: Challenges and Opportunities". In: *Proceedings of the VLDB Endowment (VLDB 2019)* 12.12 (2019), pp. 1986–1989.

[49] V. Nebot, R. Berlanga, J. M. Pérez, M. J. Aramburu, and T. B. Pedersen. "Multidimensional integrated ontologies: A framework for designing semantic data warehouses". In: *Journal on Data Semantics XIII*. 2009, pp. 1–36.

[50] V. Nebot and R. Berlanga Llavori. "Towards Analytical MD Stars from Linked Data." In: *KDIR*. 2014, pp. 117–125.

[51] F. Pasic, B. Wohlers, and M. Becker. "Towards a KPI-based ontology for condition monitoring of automation systems". In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2019, pp. 1282–1285.

[52] L. Polo, I. Mínguez, D. Berrueta, C. Ruiz, and J. M. Gómez. "User Preferences in the Web of Data". In: *Semantic Web* 5.1 (2014), pp. 67–75.

[53] A. Pomp, A. Paulus, A. Kirmse, V. Kraus, and T. Meisen. "Applying semantics to reduce the time to analytics within complex heterogeneous infrastructures". In: *Technologies* 6.3 (2018), p. 86.

[54] M. Poveda, M. C. Suárez-Figueroa, and A. Gómez-Pérez. "Common pitfalls in ontology development". In: *Conference of the Spanish Association for Artificial Intelligence*. 2009, pp. 91–100.

[55] T. Preisinger, W. Kießling, and M. Endres. "The BNL++ Algorithm for Evaluating Pareto Preference Queries". In: *Proc. PREFERENCE, Riva del Garda, Italy* (2006).

[56] E. Rahm and P. A Bernstein. "A survey of approaches to automatic schema matching". In: *the VLDB Journal* 10.4 (2001), pp. 334–350.

[57] R. Ramakrishnan et al. "Azure data lake store: a hyperscale distributed file service for big data analytics". In: *Proceedings of the 2017 ACM International Conference on Management of Data*. 2017, pp. 51–63.

[58] S. Rangarajan, H. Liu, H. Wang, and C.-L. Wang. "Scalable architecture for personalized healthcare service recommendation using big data lake". In: *Service research and innovation*. 2015, pp. 65–79.

[59] F. Ravat and Y. Zhao. "Data Lakes: Trends and Perspectives". In: *30th International Conference on Database and Expert Systems Applications (DEXA 2019)*. Linz, Austria, 2019, pp. 304–313.

[60] J. Rosati, T. Di Noia, T. Lukasiewicz, R. De Leone, and A. Maurino. "Preference queries with ceteris paribus semantics for linked data". In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. 2015, pp. 423–442.

[61] E. M. Sanfilippo, F. Belkadi, and A. Bernard. "Ontology-based knowledge representation for additive manufacturing". In: *Computers in Industry* 109 (2019), pp. 182–194.

[62] P. N Sawadogo, E. Scholly, C. Favre, E. Ferey, S. Loudcher, and J. Darmont. "Metadata Systems for Data Lakes: Models and Features". In: *European Conference on Advances in Databases and Information Systems (ADBIS)*. 2019, pp. 440–451.

[63] A. P. Sheth and C. Ramakrishnan. "Semantic (Web) technology in action: Ontology driven information systems for search, integration, and analysis". In: *IEEE Data Engineering Bulletin* 26.4 (2003), p. 40.

[64] K. Singh, K. Paneri, A. Pandey, G. Gupta, G. Sharma, P. Agarwal, and G. Shroff. "Visual bayesian fusion to navigate a data lake". In: *2016 19th International Conference on Information Fusion (FUSION)*. 2016, pp. 987–994.

[65] I. Suriarachchi and B. Plale. "Crossing analytics systems: a case for integrated provenance in data lakes". In: *2016 IEEE 12th International Conference on e-Science (e-Science)*. 2016, pp. 349–354.

[66] A. Troumpoukis, S. Konstantopoulos, and A. Charalambidis. "An extension of SPARQL for expressing qualitative preferences". In: *International Semantic Web Conference (ISWC 2017)*. Vienna, Austria, 2017, pp. 711–727.

[67] T. Tudorache, C. Nyulas, N. F Noy, and M. A Musen. "WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web". In: *Semantic web* 4.1 (2013), pp. 89–99.

[68] P.-Y. Vandenbussche, G. A Atemezing, M. Poveda-Villalón, and B. Vatant. "Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web". In: *Semantic Web* 8.3 (2017), pp. 437–452.

[69] C. Walker and H. Alrehamy. "Personal Data Lake with Data Gravity Pull". In: *Proc. of 2015 IEEE Fifth International Conference on Big Data and Cloud Computing (BDCLOUD '15)*. Dalian, China, 2015, pp. 160–167.

[70] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica. "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing". In: *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*. 2012, pp. 15–28.