Journal of Information and Data Management, 2026, 17:1, doi: 10.5753/jidm.2026.5971

A Conceptual Framework for Building and Exploring Semantic Views of Enterprise Knowledge Graphs

Vania Maria Ponte Vidal [**Universidade Federal do Ceará** | ***vaniapvidal@gmail.com***]
José Renato S. Freitas [**Universidade Federal do Ceará** | ***jrenatosfreitas@gmail.com***]
Tulio Vidal Rolim [**Universidade Federal do Ceará** | ***tulio.xcrtf@gmail.com***]
Narciso Arruda [**Universidade Federal do Ceará** | ***narciso@lia.ufc.br***]
Marco A. Casanova [**Department of Informatics – Pontifical Catholic University of Rio de Janeiro** | ***casanova@inf.puc-rio.br***]
Chiara Renso [**Institute of Information Science and Technology (ISTI-CNR)** | ***chiara.renso@isti.cnr.it***]

Received: 28 April 2025 • **Published:** 13 March 2026

Abstract An Enterprise Knowledge Graph (EKG) provides a powerful foundation for knowledge management, data integration, and analytics within organizations. It achieves this by offering a semantic view that semantically integrates diverse data sources from the organization's data lake. This paper introduces a novel Data Design Pattern for constructing semantic views, referred to as *DDP_SV*, specifically designed to support the creation of semantic views within an EKG. The proposed *DDP_SV* organizes both data and metadata into four hierarchical layers, providing a standardized structure that facilitates the development, maintenance, and reuse of semantic views across different contexts. Building upon this foundation, a second key contribution is a novel incremental methodology for constructing the semantic view of an EKG, grounded in the proposed data design pattern. This methodology adopts a "pay-as-you-go" data integration strategy, allowing organizations to progressively build, refine, and evolve their knowledge graphs while ensuring semantic consistency, scalability, and adaptability throughout the integration process. In addition, the paper presents an interactive graphical interface designed to support context-sensitive navigation of the semantic view. This tool enhances user interaction by enabling intuitive exploration and deeper utilization of resources within the EKG.

Keywords: Data Integration, Enterprise Knowledge Graph, Semantic View, Data Design Pattern.

1 Introduction

An Enterprise Knowledge Graph (EKG) have emerged as a promising solution and a robust foundation for knowledge management, data integration, and advanced analytics across organizations, by offering a semantic view as noted by [Grainger *et al*., 2016; Ehrlinger and Woss, 2016]. In this context, the primary goal of the semantic view is to provide a unified ontological framework emerging from the semantic integration of the data sources from an organization's data lake. This integration establishes a comprehensive and coherent organizational data environment, enabling seamless access and fostering streamlined decision-making processes.

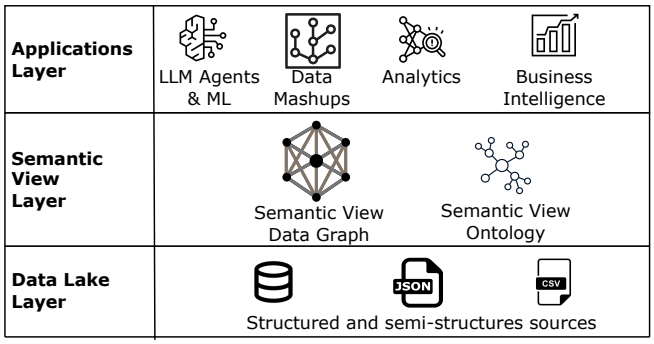

Figure 1. Architecture of EKG Systems.

The general architecture of EKG systems, as proposed by Galkin *et al*. [2017], and illustrated in Figure 1, consists of three distinct layers:

(i) the *Data Lake Layer* serves as the foundational storage environment of the EKG architecture. It provides a centralized and scalable repository capable of accommodating diverse data sources, including unstructured, semi-structured, and structured data. The data lake is not responsible for semantic integration; rather, it preserves data in their native formats and supplies the raw material that the subsequent layers of the EKG will semantically transform, enrich, and integrate into the unified knowledge graph;

(ii) the *Semantic View Layer* is responsible for semantically integrating the data sources from the data lake into a coherent and unified representation. This layer comprises two main components. The Semantic View Ontology (SVO) defines the conceptual and terminological structure of the view, its classes, properties, and axioms, operating at the schema or TBox level. The Data Graph of semantic view contains the corresponding ABox assertions of TBox, representing the factual instances and their relationships that instantiate the model defined by the SVO;

(iii) the *Application Layer* implements domain-specific applications and services that leverage the integrated and semantically enriched data provided by the Semantic View Layer. By consuming the unified semantic view, this layer enables advanced functionalities such as semantic querying, analytics, reasoning, and knowledge discovery.

The construction and maintenance of a semantic view in an EKG system present three major challenges: (i) Extraction and Transformation – Integrating data from heterogeneous sources within a data lake into a unified representation, using

a shared vocabulary defined by the SVO; (ii) Semantic Linking – Establishing connections between semantically equivalent entities across different data sources to enable cross-referencing and semantic alignment; (iii) Data Fusion and Quality Enhancement – Merging multiple representations of the same real-world object into a single, consistent representation while resolving data inconsistencies to improve overall data quality.

Although Enterprise Knowledge Graphs have become central to semantic integration and organizational data management, the state of the art lacks a formally defined and reusable architectural model that structures the semantic view produced during integration. Existing approaches offer processes, workflows, mapping languages, and linking techniques, but they do not specify how the resulting data and metadata should be organized into a canonical representation. This gap leads to heterogeneous implementations, limited reusability, and difficulty supporting incremental extensions across sources and domains.

The absence of a unified and implementation-independent structure for semantic views means that organizations lack a principled way to represent exported data, identity links, unified entities, and fused information in a consistent and reusable manner. Without such a structure, semantic integration efforts remain ad hoc, difficult to reproduce, and challenging to evolve incrementally.

In this perspective, our research question is "*How can we formalize a reusable and implementation-independent architectural pattern that organizes the data and metadata of a semantic view in an EKG into well-defined layers, supporting both consistent semantic integration and incremental construction?"*

To address these challenges, Vidal *et al*. [2024] introduced a data design pattern tailored for the logical organization of data within the semantic view of an EKG. This pattern structures the data and metadata into three hierarchical layers: *exported views*, *linkset* and *unification views*, and *fusion views*. This structured approach addresses the specific challenges of semantic data integration, simplifies maintenance, and enhances the flexibility and depth of exploration of the semantic view across various contexts.

The work of Vidal *et al*. [2024] also detailed an interactive graphical interface developed to support context-based exploration of semantic view resources. Using this interface, users can effectively track the data lineage of a particular resource through its transformational flow before its integration into the knowledge graph of the semantic view. This capability is invaluable for understanding the origin, transformations, and data integrations within the knowledge graph.

This article extends the work reported in Vidal *et al*. [2024] in two directions:
(i) It presents a novel incremental methodology for constructing the semantic view of an EKG, leveraging the data design pattern. In this approach, each data source within the data lake is integrated into the semantic view one at a time, facilitating manageable and systematic incorporation. This methodology supports a *pay-as-you-go* data integration strategy, enabling the incremental and continuous construction of semantic views without requiring a complete specification beforehand to maintain their knowledge graph [Paton *et al*., 2016].
(ii) It implements a domain-specific knowledge graph for music, called *SV_Music*, which integrates music-related data from two sources: *DBpedia* and *MusicBrainz*. *SV_Music* is freely available at [`https://semantic-ekgraphs.github.io/semantic-music/`], and can be explored through either a graphical user interface or a SPARQL endpoint, providing flexible access for both casual users and advanced queries. *SV_Music* serves as a real-world case study to validate the effectiveness of the proposed framework.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces the proposed data design pattern for structuring the semantic view of an EKG. Section 4 formalizes the conceptual specification of each view layer in the data design pattern and defines the semantics for deriving the state of a view from its declarative specification. Section 5 presents a novel incremental methodology for constructing the semantic view. Section 6 describes an interactive graphical interface developed to support the exploration of resources within the semantic view. Section 7 concludes the paper and outlines directions for future research.

2 Related Work

The construction of Enterprise Knowledge Graphs (EKGs) from semantic data integration requires structured methodologies, frameworks, and tools. Several studies have addressed this challenge from complementary perspectives, contributing to aspects such as interoperability, modular design, lifecycle management, and user interaction.

Although the classical formulation of *pay-as-you-go* data integration was developed mainly between 2012 and 2017, the underlying principle of incremental and progressive semantic integration remains highly relevant in the context of modern EKGs. Recent works on knowledge graph lifecycle and modular evolution ([Simsek *et al*., 2022], [Angelis *et al*., 2024], [Cimmino and García-Castro, 2024],[Rehman *et al*., 2025]), ontology-driven EKG engineering ([Sequeda and Lassila, 2021]), and LLM-assisted incremental KG construction ([Yang *et al*., 2024], [Tian *et al*., 2025]) demonstrate a renewed interest in incremental integration methodologies. Our approach extends this principle to the construction of semantic views, situating *pay-as-you-go* as a conceptual strategy aligned with the incremental nature of EKG development.

To support the knowledge graph process construction documentation, Azizi [2023] develop an RDF-based knowledge graph to represent the data integration process, using a unified schema to describe a data integration system. Although this facilitates integrated analysis of heterogeneous data sources, the work does not specify mechanisms for linking, unification, or fusion of semantic views, nor does it offer tools to support such activities.

From the perspective of tools to support knowledge graph (KG) interaction and visualization, several tools have been proposed. Schultz *et al*. [2011] propose the Linked Data Integration Framework (LDIF) designed for KG construction

and visualization. LDIF focuses on the backend processes of semantic data management, providing mechanisms for data extraction, transformation, linking, and fusion of heterogeneous RDF sources into a unified knowledge base.

Haase *et al*. [2019] introduce *metaphactory*, a standards-based platform for KG management with customizable user interfaces. De Souza *et al*. [2022] present *TKGEvolViewer* for visualizing KG evolution, while Sellami and Zarour [2022] propose *KeyFSI*, a faceted, keyword-based interface to support exploration.

While Paton *et al*. [2012] conceptualize *pay-as-you-go* as a process strategy for progressively refining mappings and integration rules, they do not propose a unified architectural model that structurally supports incremental integration. In contrast, our approach incorporates *pay-as-you-go* as a structural property of the *DDP_SV*— the four-layer architecture allows each data source to be integrated independently, without reprocessing previously integrated sources, because each view layer is modular, declarative, and incrementally extensible.

This work advances the field by proposing an integrated approach that combines a unified formal model for organizing data and metadata with a methodology based on this model to construct EKG semantic views. These contributions are supported by a formal data design pattern and an exploration tool, the approach enhances both the engineering process and the use of EKGs in multiple dimensions.

3 Data Design Pattern for Constructing a Semantic View

This section introduces a data design pattern, referred to as *DDP_SV*, specifically developed to logically organize both data and metadata within the semantic view of an EKG [Vidal *et al*., 2024]. At the core of the proposed framework, the semantic view is composed of two interconnected knowledge graphs: the *Data Graph* and the *Metadata Graph*, each discussed in the subsections below.

A knowledge graph is commonly represented using the Resource Description Framework (RDF)[1], a W3C standard for modeling data on the Web in the form of RDF triples (s, p, o). Each triple encodes a statement in which the subject (s) and the object (o) denote nodes in the graph, while the predicate (p) represents the directed edge that specifies a relationship or fact connecting them. In this sense, a knowledge graph can be viewed as an RDF graph composed of a set of such triples, collectively capturing entities, their attributes, and the relationships among them [Deb Nath *et al*., 2020].

3.1 Data Graph

The *Data Graph* serves as the informational backbone of the semantic view, representing the instance-level data that embodies the semantics defined by the ontology of the semantic view.

Within the proposed framework, the Data Graph provides a structured and unified representation of the integrated data sources, capturing entities, relationships, and literal values that collectively instantiate the conceptual model defined by the ontology.

To ensure scalability and semantic clarity, the Data Graph is systematically organized into a four-tier hierarchy of views, each capturing and structuring integrated data at distinct levels of abstraction, as detailed below (see Figure 2(a)).
(i) *Exported Views Layer* – This foundational layer consists of RDF views exported from the data sources within a data lake. These *exported views* are generated by systematically mapping raw source data to a common, shared vocabulary defined by the Semantic View Ontology (SVO).

(ii) *Linkset Views Layer* – This layer establishes semantic connections between equivalent entities across different exported views using identity relations such as *owl:sameAs*. These links enable semantic alignment and cross-referencing of distributed data.

(iii) *Unification Views Layer* – Built upon the *linkset views*, this layer consolidates semantically equivalent entities into a unified representation. Unification ensures that resources referring to the same real-world object are represented as a single–unified resource in the semantic view.

(iv) *Fusion Views Layer* – The topmost layer addresses and resolves conflicts that may arise when multiple sources provide inconsistent or conflicting information about the same entity or relationship. This layer applies conflict resolution techniques to improve the accuracy, consistency, and overall quality of the integrated data.

These four layers emerge as a natural generalization of the classical challenges in semantic data integration rather than as an arbitrary design choice. Each layer corresponds directly to a recurring integration problem: structural heterogeneity (exported views), semantic equivalence identification (linkset views), identifier consolidation (unification views), and conflict resolution (fusion views). By aligning the pattern with these well-established challenges, the *DDP_SV* provides a principled and systematic structure for organizing semantic integration results within an EKG, independent of the specific tools or processes used to construct them.

3.2 Metadata Graph

The *Metadata Graph*, depicted in Figure 2(b), serves as the repository for all metadata related to the semantic view. It plays a critical role in describing both the SVO and the view specifications across all levels of the *DDP_SV*.

The view specification metadata provides a declarative description of how each view, in the semantic view, is constructed, including: (i) the input data sources used; (ii) the transformation logic applied; and (iii) the structure of the resulting RDF triples.

This metadata inherently captures data lineage, documenting the entire lifecycle of data from its origin and transformation steps to its final representation in the EKG. Documenting this lineage enables organizations to trace data prove-

[1] https://www.w3.org/TR/rdf12-concepts/

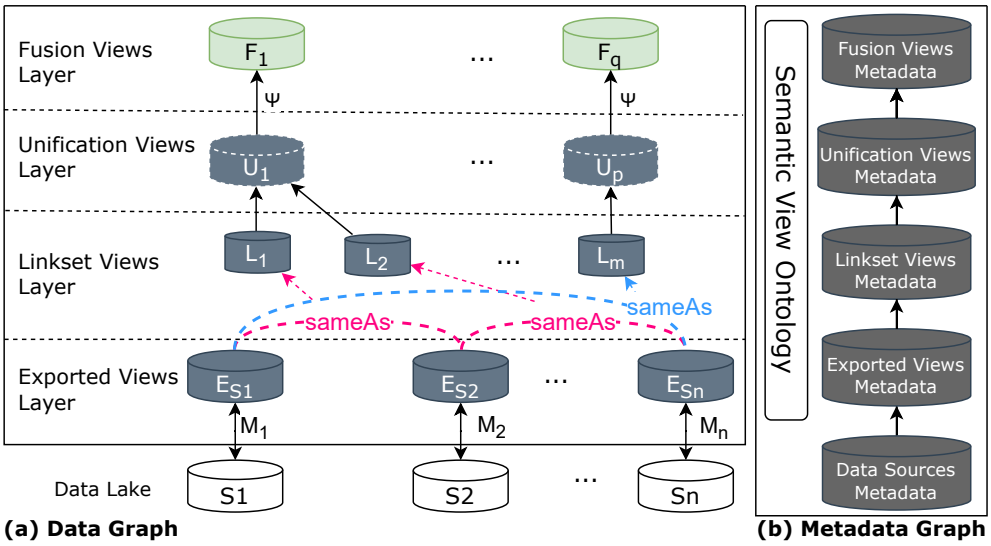

Figure 2. Data Design Pattern *DDP_SV* for EKG's Semantic Views.

nance and transformation history, fostering transparency, accountability, and trust in the semantic view.

In the proposed framework, the SVO is vital for defining a common vocabulary that facilitates the transformation and integration of EKG data sources. The SVO spans all layers of the *DDP_SV* and is built by integrating semantically equivalent elements across exported views. This integration involves two key processes:

(i) *Unification of Semantically Equivalent Properties* – Properties with similar semantic roles across different exported views are identified and unified under a common vocabulary. This standardization ensures a consistent vocabulary for describing attributes across all layers.

(ii) *Establishment of Generalization Classes* – Semantically equivalent classes from various sources are grouped under shared superclasses, forming generalization classes. These provide an abstract semantic foundation for the unification and fusion views, supporting alignment at the conceptual level. This systematic approach enables that the SVO remains adaptable and semantically coherent, even as new data sources are integrated over time.

Importantly, the *Metadata Graph* is intrinsically linked to the *Data Graph*. This tight integration fosters a holistic understanding of the semantic view and strengthens metadata-driven operations such as data discovery, quality assessment, governance, and reuse.

4 Formal Basis for the Conceptual Specification of *DDP_SV* Views

This section presents the formal definitions necessary to conceptually specify the views of each layer in the *DDP_SV*. It also defines the formal semantics for constructing the state of a view based on its declarative specification.

4.1 Exported View

In our *DDP_SV*, every data source within the data lake exports an RDF view, named *exported view*, created through a systematic mapping of the data source to a common and shared vocabulary defined by the SVO.

Definition 4.1 (Exported View Specification) *The specification of an exported view is a tuple $\langle \mathcal{E}, S, \mathcal{O}, M \rangle$, where:*

- *$\mathcal{E}$ is the name of the exported view;*
- *S is the data source in the data lake that exports $\mathcal{E}$;*
- *O is an ontology defined as a tuple $\mathcal{O} = \langle C, P, R \rangle$, where C is the set of classes, P the set of properties, and R a set of semantic constraints operating at the terminological level (TBox).*
- *M is a set of mapping rules that specify how elements of the source schema S (such as tables, columns, hierarchies, or graph patterns) are translated into RDF terms defined by the ontology O. Thus, the mapping rules serve as the formal mechanism that drives the semantic transformation of source data into the RDF triples that populate the ABox of O.*

Given a specification of an exported view $\langle \mathcal{E}, S, \mathcal{O}, M \rangle$ and $S(t)$, the state of the data source S at time t, the data graph of $\mathcal{E}$, at time t, denoted $\mathcal{E}(t)$, is constructed by applying the transformation rules M over $S(t)$, more formally:

$$\mathcal{E}(t) = \{(s, p, o) \mid (s, p, o) \in M(S(t))\}$$

The generation of RDF triples is governed by the structural properties of the underlying data source, such that the mapping rules are defined according to the specific characteristics of each data model. For tabular data sources—most notably relational databases—the mapping rules conform to the R2RML specification (W3C Recommendation) Das *et al*. [2012] in which database tables are mapped to RDF classes and each row is instantiated as a subject whose identifier is commonly derived from the table's primary key. In contrast, for hierarchical or semi-structured sources such as JSON or CSV, the mapping rules follow the RML framework Dimou *et al*. [2014], an extension of R2RML that accommodates nested, heterogeneous structures by generalizing logical sources and enabling flexible transformation pipelines. Consequently, the triples are generated systematically with

the type of data source, ensuring semantic consistency in the resulting RDF graph.

4.2 Linkset View

The linkset views creation process involves defining relationships or "sameAs" links between entities that are equivalent across different exported views. To establish these links, users should first define the "sameAs" linkset view by specifying the classes of the exported views and match function.

Definition 4.2 (Linkset View Specification) *The specification of a linkset view is a tuple $\langle \mathcal{L}, T, W, \mu \rangle$, where:*

- *$\mathcal{L}$ is the name of the linkset view;*
- *T and W are classes of different exported views;*
- *μ is a match function.*

The match function compares the state of two instances, e_1 and e_2, of classes T and W, returning "true", if e_1 and e_2 satisfy the match condition of μ; otherwise, it returns "false".

Given a specification of a linkset view $\langle \mathcal{L}, T, W, \mu \rangle$, and $T(t)$ and $W(t)$, representing the state of classes T and W at time t, respectively, the data graph of $\mathcal{L}$ at time t, denoted as $\mathcal{L}(t)$, is defined as follows:

$$\mathcal{L}(t) = \big\{(r_1, sameAs, r_2) \mid r_1 \in T(t),\ r_2 \in W(t),\ \text{and}\ \mu(r_1, r_2, t) = true\big\}$$

4.3 Unification View

A unification view should be specified for each generalization class in the SVO, defined as follows:

Definition 4.3 (Unification View Specification) *The specification of a unification view is a tuple $\langle \mathcal{U}, G, \eta \rangle$, where:*

- *$\mathcal{U}$ is the name of the unification view;*
- *G is a generalization class of the SVO;*
- *η is a normalization function, which maps all IRIs of the instances of G, to a canonical target IRI in $\mathcal{U}$.*

Given a specification of a unification view $\langle \mathcal{U}, G, \eta \rangle$ and $G(t)$, the state of G at time t, then the data graph of $\mathcal{U}$ at time t, denoted $\mathcal{U}(t)$, is defined as:

$$\mathcal{U}(t) = \big\{(s, p, o) \mid (r, p, o) \in G(t) \text{ and } \eta(r, t) = s\big\}$$

The design of a normalization function for unification view $\langle \mathcal{U}, G, \eta \rangle$ must satisfy the following axiom:

$$\forall x_1 \forall x_2 \in G(t)\ (x_1\ sameAs\ x_2 \Leftrightarrow \eta(x_1, t) = \eta(x_2, t))$$

Therefore, the IRIs x_1 and x_2 are unified to the same canonical IRI iff they are declared equivalent via a sameAs statement of the form "$x_1\ sameAs\ x_2$".

4.4 Fusion View

In our framework, the user is free to decide how to resolve the problem of discrepancy when combining various representations of the same real-world object into a single view (canonical IRI) to specify fusion views. This is done with the help of *"property fusion assertion"*.

Definition 4.4 (Property Fusion Assertion) *A property fusion assertion (PFA) is a quadruple $\langle \mathcal{A}, G, p, \Psi \rangle$, where:*

- *$\mathcal{A}$ is the name of the PFA;*
- *G is a generalization class of the semantic view ontology;*
- *p is a property of G;*
- *Ψ is a conflict resolution function.*

The conflict resolution function Ψ, accepts as input a canonical IRI c and a set of values for p of c, and produces a single value. Definitions 4.5 and 4.6 below define how to solve conflicts based on a PFA of datatype properties and object properties, respectively.

Definition 4.5 (Conflict Resolution of DataType Property) *Let:*

- *$\langle \mathcal{A}, G, p, \Psi \rangle$ be a PFA for property p of G;*
- *$\langle \mathcal{U}, G, \eta \rangle$ be the unification view specification for G;*
- *$\mathcal{U}(t)$ be the state of $\mathcal{U}$ at time t;*
- *c be a canonical IRI in $\mathcal{U}(t)$;*
- *$V = \big\{v \mid (c, p, v) \in \mathcal{U}(t)\big\}$; V contains the set of all values of property p for resource c.*

The result for solving conflicts in property p of resource c, at time t, using the PFA $\mathcal{A}$, denoted $\mathcal{A}(c, t)$, is defined as:

$$\mathcal{A}(c, t) = (c, p, v), \textit{where } v = \Psi(c, V).$$

Definition 4.6 (Conflict Resolution of Object Properties) *Let:*

- *$\langle \mathcal{A}, G, p, \Psi \rangle$ be a PFA for object property p of generalization class G.*
- *$\langle \mathcal{U}, G, \eta \rangle$ be the unification view specification for G.*
- *$\mathcal{U}(t)$ be the state of $\mathcal{U}$ at time t.*
- *c be a canonical IRI in $\mathcal{U}(t)$.*
- *$V = \big\{u \mid (c, p, o) \in \mathcal{U}(t) \text{ and } \eta(o, t) = u\big\}$; V contains the set of all canonical IRIs related to c via object property p.*

The result for solving conflicts in property p of resource c, at time t, using the PFA $\mathcal{A}$, denoted $\mathcal{A}(c, t)$, is defined as:

$$\mathcal{A}(c, t) = (c, p, v), \textit{where } v = \Psi(c, V).$$

In the context of a generalization class G, a PFA should be defined for each property of G where there is potential for conflicting values. To compute the fusion view for a resource c it is necessary to resolve the conflicts of all PFAs specified for G, as follows.

Definition 4.7 (Fusion View) *Let:*

- *$\langle \mathcal{U}, G, \eta \rangle$ be the specification of a unification view for G;*

- $\mathcal{U}(t)$ *be the state of* $\mathcal{U}$ *at time* t*;*
- $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *be all PFAs of generalization class* G*;*
- $p_1, \ldots, p_n$ *be the properties associated with PFAs* $\mathcal{A}_1, \ldots, \mathcal{A}_n$*, respectively;*
- c *be a canonical IRI in* $\mathcal{U}(t)$*.*

The data graph of fusion view for c in time t, denoted $\mathcal{FV}(c,t)$, is defined by:

$$\mathcal{FV}(c,t) = \bigcup_{i=1}^{n} \mathcal{A}_i(c,t) \ \cup \ \big\{(c,p,v) \mid (c,p,v) \in \mathcal{U}(t),$$
where p is datatype property and $p \neq p_i, 1 \leq i \leq n\big\} \cup$
$\big\{(c,p,u) \mid (c,p,o) \in \mathcal{U}(t)$, where p is an object property,
$p \neq p_i, 1 \leq i \leq n$, and $\eta(o,t) = u\big\}$

5 Incremental Approach for the Construction of Semantic Views

5.1 Overview

This section presents an incremental methodology for constructing the semantic view of an EKG, leveraging the *DDP_SV* introduced in Section 3. The approach comprises 5 primary tasks, as showed in Figure 3. Each data source in the data lake is integrated into the semantic view incrementally, facilitating manageable and systematic incorporation.

Using a *pay-as-you-go* strategy, allowing for the gradual construction and maintenance of the knowledge graph. By integrating one data source at a time, organizations can prioritize resources effectively, ensuring that each addition is aligned with specific business needs and objectives.

For the integration of a new data source (*DS*), the following pipeline is executed (see Figure 3):
(i) *Semantic View Ontology Modeling* — In this phase, SVO evolves progressively as new data sources are integrated, thus, the initial version of the SVO mirrors the classes and properties of the first data source's ontology. When integrating a new *DS*, the SVO is extended to incorporate new classes and properties that accurately represent the semantics of *DS*.
(ii) *Exported View Construction* — Develop both the specification and the corresponding data graph for the RDF view exported by *DS*. This step leverages the vocabulary defined in the extended SVO, resulting from step 1, to construct the exported view vocabulary. The exported view provides a structured representation of *DS*'s data, facilitating its integration into the broader semantic view.
(iii) *Linkset View Construction* — Establish specifications and corresponding data graphs to store sets of links that connect instances from the exported view of *DS* to semantically equivalent instances in other exported views within the semantic view. These linksets enable the identification and alignment of equivalent entities across different data sources, promoting data interoperability.
(iv) *Unification View Construction* — Design new unification views that encompass the new generalization classes introduced into the extended version of the SVO. These views aggregate semantically similar entities, providing a cohesive representation of related concepts. Unification views serve as a foundation for resolving redundancies and inconsistencies across data sources.
(v) *Fusion View Construction* — Implement property fusion views to resolve data conflicts arising from overlapping properties in the new unification views. This involves defining fusion strategies that reconcile discrepancies and consolidate information accurately. Fusion views ensure that the most reliable and relevant data is presented in the semantic view, enhancing data quality and trustworthiness.

The culmination of this process is the generation of both the *Metadata Graph* and the *Data Graph* that constitute the semantic view. By following this approach, organizations can build and maintain their semantic view efficiently, ensuring scalability, flexibility, and alignment with evolving data landscapes.

5.2 Case Study: The "*SV_Music*" Semantic View

To validate the methodology, we constructed a domain-specific knowledge graph for music, called *SV_Music*, which integrates music-related data from two sources: *DBpedia*[2] and *MusicBrainz*[3].

DBpedia constitutes the main resource of Linked Open Data on the Web, containing more than 228 million entities to date. *MusicBrainz* is an open encyclopedia that gathers music metadata, including artists, releases, recordings, works, labels, and their interconnections.

Figure 4 depicts a fragment of the *DBpedia ontology (dbo:)* using the vocabularies *Dublin Core (dc:) [Weibel* et al*., 1998]* and *Friend of a Friend (foaf:) [Brickley, 2005]* for music data representation. *MusicBrainz* ontology has as its core the *Music Ontology (mo:)* and incorporates *dc:* and *foaf:* vocabularies, as illustrated in Figure 5.

The semantic view *SV_Music* is freely available via a semantic portal[4]. This portal provides structured access to both data and metadata graphs of *SV_Music*.

5.3 Semantic View Ontology Modeling

5.3.1 Incremental Process

In the incremental approach to constructing the semantic view ontology (SVO), the process begins by modeling the ontology for a selected data source from the data lake. It's advisable to start with the data source that contains the primary entities of the EKG, as this forms the foundational structure for subsequent integrations.

Therefore, the initial version of the SVO mirrors the classes and properties of this primary data source's ontology. To integrate a new data source (*DS*) into the EKG, the process involves three key activities:

(i) *Modeling the Data Source Ontology (DSO)* — This activity entails creating an ontology that accurately represents the structures, entities, relationships, and attributes inherent

[2] https://www.dbpedia.org/
[3] https://musicbrainz.org/
[4] `https://semantic-ekgraphs.github.io/semantic-music/index.html`

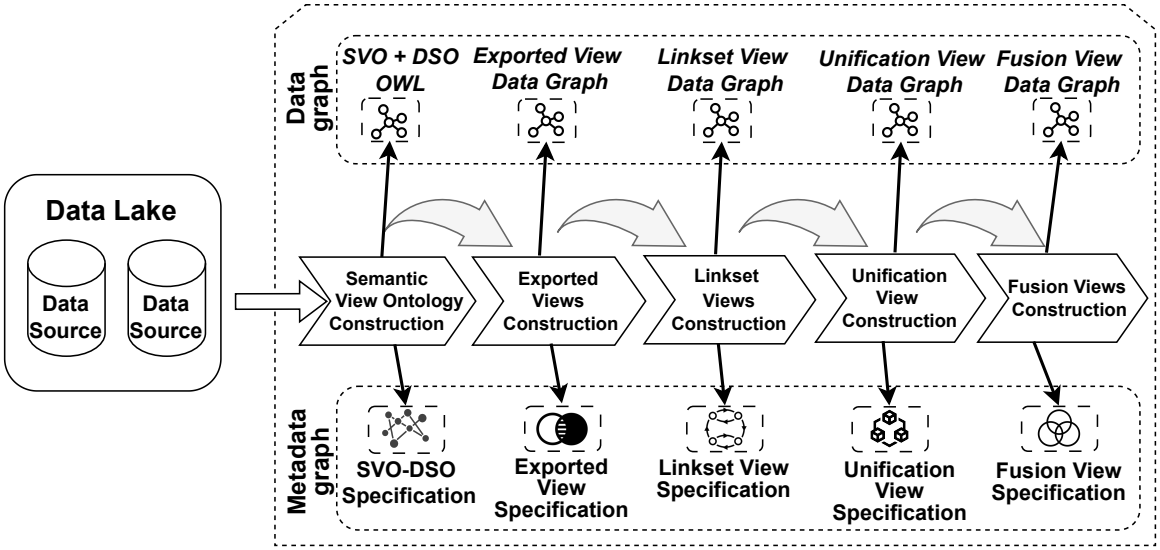

Figure 3. Framework to Construct EKG's Semantic View.

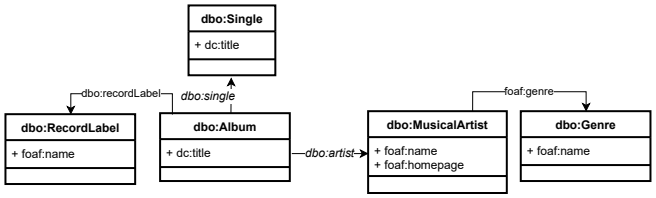

Figure 4. *DBpedia* Ontology.

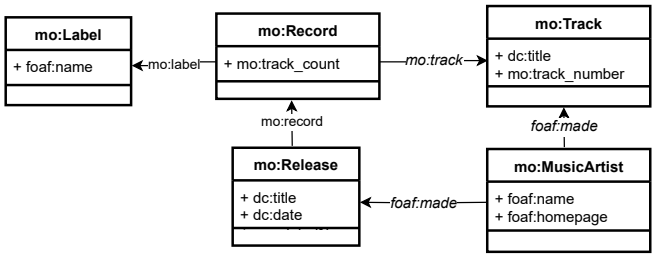

Figure 5. *MusiBrainz* Ontology.

to the new data source. A thorough analysis is required to capture the schema and semantics of the data source effectively;

(ii) *Semantic Annotation of the DSO* — Align the classes and properties of the DSO with the existing SVO and other relevant ontologies, such as those in the Linked Open Data (LOD) cloud[5]. This alignment identifies semantic correspondences, ensuring consistent representation of similar concepts across different data sources;

(iii) *Incorporation of Classes and Properties into the SVO* — Extend the SVO by integrating new classes and properties from the DSO, promoting ontological harmonization Noy *et al*. [2001]. Unify semantically equivalent properties under a shared semantic definition to maintain consistency. Additionally, define generalization classes as superclasses that group semantically equivalent classes from various exported views, providing broader, more abstract concepts that facilitate integration.

This structured and incremental approach ensures that the SVO evolves systematically, promoting semantic consistency and interoperability between integrated data sources.

5.3.2 Ontology of *SV_Music*

In the context of the case study, the initial version of *SV_Music* ontology mirrors the classes and properties of the *MusicBrainz* ontology as shown in Figure 5. Subsequently, the *SV_Music* ontology has to be extended by integrating classes and properties from the *DBpedia* ontology (see Figure 4). To achieve this, it was first necessary to align the *DBpedia* and *MusicBrainz* ontologies. The following class correspondences were identified:

- $CCA_1 \rightarrow$ *mo:MusicArtist* $\equiv$ *dbo:MusicalArtist*;
- $CCA_2 \rightarrow$ *mo:Record* $\equiv$ *dbo:Album*;
- $CCA_3 \rightarrow$ *mo:Track* $\equiv$ *dbo:Single*;

For the integration of these *DBpedia* concepts into the *SV_Music* ontology, it was necessary to introduce new generalization classes – *mo:MusicArtist*, *mo:Record*, and *mo:Track* – which serve as superclasses grouping semantically equivalent classes, as defined by the correspondence assertions CCA_1, CCA_2, and CCA_3.

When naming these generalization classes, preference was given to reusing the class names from the *Music Ontology* (*mo:*), as these classes are already well-established within the music domain and are appropriate for aggregating instances from different data sources.

It is important to note that a subclass of a generalization class should not have the same name as its superclass. If a naming conflict occurs, the specific classes must be renamed by adding a predefined prefix associated with their respective data source.

In the case study, the prefix "dbp:" is used for classes specific to *DBpedia*, and the prefix "mbz:" is used for classes specific to *MusicBrainz*. Figure 6 illustrates the resulting class hierarchy adopted for the case study.

The *SV_Music* ontology is available through this link[6] in

[5] https://lod-cloud.net/

[6] `https://semantic-ekgraphs.github.io/semantic-music/ontologia_visao_semantica.html`

the Semantic Portal, where users can explore its structure and semantics in detail.

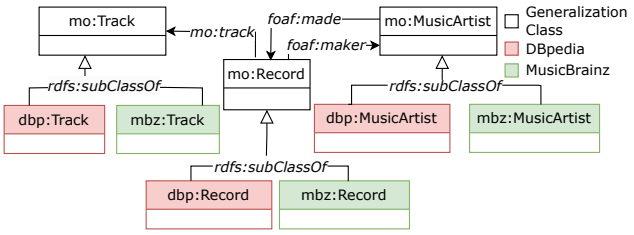

Figure 6. Main Classes and Relationships of *SV_Music* ontology.

5.4 Exported View Construction

In this phase, the objective is to construct an exported view for the new data source (*DS*) by leveraging the existing semantic view ontology (SVO) and predefined matching assertions between *DS* and SVO.

5.4.1 Development Process

The development of an exported view for a new data source comprises two primary steps:

1. *Generation and Validation of Mappings from DS to SVO* – The goal of this step is to create operational mappings from *DS* to SVO, utilizing the established matching assertions [Neto *et al*., 2013]. To ensure the quality and accuracy of these mappings, it's crucial to assess them using established frameworks. Tools like *Luzzu* [Debattista *et al*., 2016] provide a comprehensive methodology for Linked Data Quality Assessment, offering extensible interfaces for defining quality metrics and generating detailed quality reports. Incorporating feedback mechanisms and iterative testing further refines the mappings, ensuring they accurately represent the underlying data and adhere to desired quality standards.

2. *Implementation of the Data Graph* – Upon validating the mappings, the next step involves implementing them to construct the *Data Graph* for the exported view. This *Data Graph* can be realized in two ways:

(i) **Materialized Approach:** In this method, RDF triples are generated and persistently stored within a named graph, enabling efficient querying and management. Named graphs serve as containers for RDF triples, each identified by a unique URI, allowing for modularization and provenance tracking of datasets.

(ii) **Virtual Approach:** Alternatively, virtualization techniques can be employed to create a *Data Graph* that queries the underlying data source in real-time without materializing the RDF triples. This approach ensures up-to-date data retrieval and reduces storage overhead.

5.4.2 Exported Views of *SV_Music*

Within the semantic view *SV_Music*, two exported views are defined: *EV_DBpedia* and *EV_MusicBrainz*. Each exported view is materialized in a distinct named graph, promoting modular data management and enabling accurate provenance tracking.

Figure 7 presents fragments of *EV_DBpedia* and *EV_MusicBrainz*. For instance, the resources *dbr:r1* and *mbr:r2* are instances of the classes *dbp:Record* and *mbz:Record*, respectively.

It is noteworthy that the set of classes and properties associated with each exported view constitutes a subset of the classes and properties defined in the overall semantic view, as determined by the corresponding mapping specifications.

The complete metadata describing each exported view of *SV_Music*— including its ontology, mapping definitions, data source provenance, named graph, and related artifacts — is available through the *SV_Music* Semantic Portal[7].

5.5 Linkset Views Construction

This phase focuses on creating linkset views to connect instances from the new exported view to semantically equivalent instances in other exported views within semantic view.

5.5.1 Development Process

The development of linkset views from the new exported view is carried out in three structured steps:

(i) *Selection of Source and Target Classes* – To define a linkset view, it is essential to identify pairs of classes, one from the new exported view and another from existing exported views, that are semantically related. In the proposed framework, two classes are considered semantically equivalent if and only if, they are both subclasses of the same generalization class defined in the SVO. This criterion ensures that only conceptually aligned classes are linked, streamlining the identification process and promoting semantic coherence and interoperability across data sources.

(ii) *Definition of Matching Criteria* – The match function defines the conditions under which an instance from the source class is considered equivalent to an instance from the target class. It compares the state (i.e., selected property values) of the two instances and returns true if the similarity criteria are satisfied. These criteria may involve exact string matches, similarity thresholds, or more complex functions involving multiple attributes such as names, dates, identifiers or links.

(iii) *Creation of Named Graphs for Linkset Views* – For each linkset view, a dedicated graph is created to store the generated links. This modular structure allows for the separate management, querying, and provenance tracking of link assertions, facilitating incremental updates and traceable alignment operations between data sources.

Tools such as *Silk* [Volz *et al*., 2009], and *Open Refine* [Ham, 2013] support the implementation and execution of linkage rules, offering functionalities such as string similarity computation, data normalization, and value merging algorithms to facilitate the comparison of data across different sources.

[7] `https://semantic-ekgraphs.github.io/semantic-music/localgraphs.html`

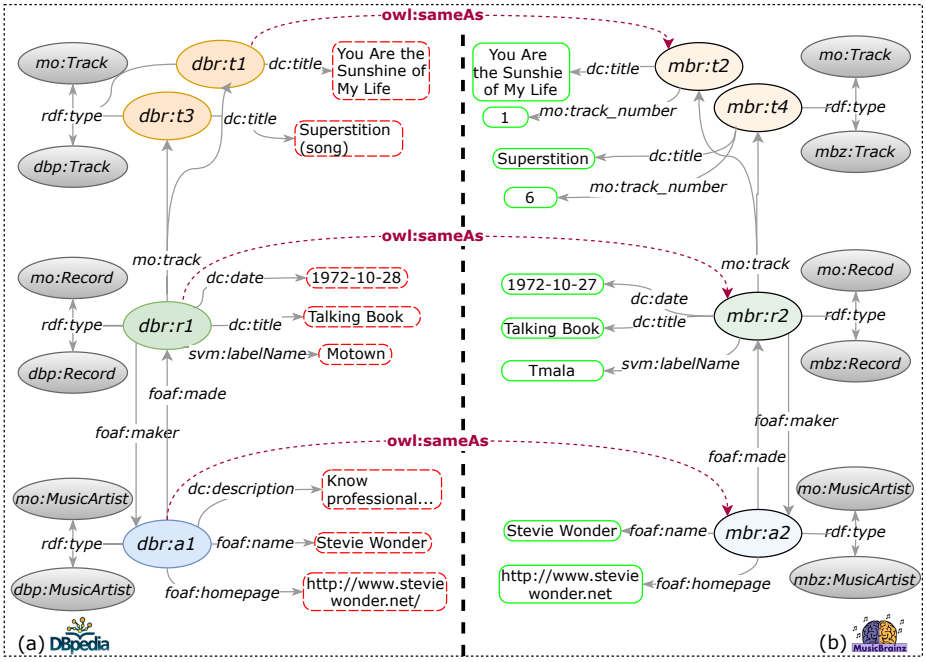

Figure 7. (a) Fragment of *EV_DBpedia*; (b) Fragment of *EV_MusicBrainz*.

5.5.2 Linkset Views for *SV_Music*

Considering the hierarchy of classes of *SV_Music* ontology in Figure 6, three linkset views are established to connect semantically equivalent resources between *EV_DBpedia* and *EV_MusicBrainz*.

LV_Record: This linkset view connects instances of *dbp:Record* (from *EV_DBpedia*) and *mbz:Record* (from *EV_MusicBrainz*). The matching criteria for establishing *owl:sameAs* links include the attributes similarity, such as *dc:title* and *dc:date*. Records with identical or highly similar titles and release dates are considered equivalent.

LV_Track: This view links instances of *dbp:Track* and *mbz:Track*. The equivalence is determined based on matching track titles and associated metadata. For example, tracks sharing the same name and duration across both datasets are linked.

LV_MusicArtist: This linkset connects instances of *dbp:MusicArtist* and *mbz:MusicArtist*. Matching is based on attributes like *foaf:name* and *foaf:homepage*. Artists with identical names and official websites are considered the same entity.

Figure 7 shows examples of sameAs links for each linkset view, represented by the dotted red arrow, such as the RDF triple *(dbr:a1)-owl:sameAs→(mbr:a2)* from *LV_MusicArtist*.

The metadata describing each linkset view of *SV_Music*—including its source and target classes, matching function, named graph and related artifacts — is available through the *SV_Music* Semantic Portal[8].

5.6 Unification View Construction

In the proposed framework, a unification view must be defined for each generalization class present in the SVO. Consequently, when new generalization classes are introduced as part of the integration of a data source, (as described in Step 5.3.2), corresponding unification views should be designed and added to the semantic view.

5.6.1 Development Process

The construction of a unification view for a generalization class requires the definition of a **"normalization function"**. This function is responsible for remapping all IRIs that refer to semantically equivalent entities — declared as such via *owl:sameAs* links — into a single canonical IRI. The purpose is to unify multiple representations of the same real-world entity across different exported views.

A normalization function in this context is an algorithm or rule set that standardizes identifiers by selecting a canonical form from a set of equivalent IRIs. The selection of a canonical IRI can be based on various criteria, such as: the authority of the source, the frequency of use, or predefined organizational standards.

The implementation of the *Data Graph* for a unification view is virtual. That is, the unification view is computed dynamically at query time rather than materialized and stored in advance. During query execution, all properties and relationships associated with semantically equivalent IRIs are consolidated and presented under the canonical IRI defined by the normalization function. This runtime consolidation enables a unified and consistent view of entities across multiple data sources, without the overhead of materialization.

5.6.2 Unification View of *SV_Music*

Based on integration of *DBpedia* concepts into *SV_Music* ontology, new generalization classes: *mo:MusicArtist*, *mo:Record* and *mo:Track* have been introduced. Consequently, three corresponding unification views are constructed: *UV_MusicArtist*, *UV_Record* and *UV_Track*.

[8] https://semantic-ekgraphs.github.io/semantic-music/semantic_link_sets.html

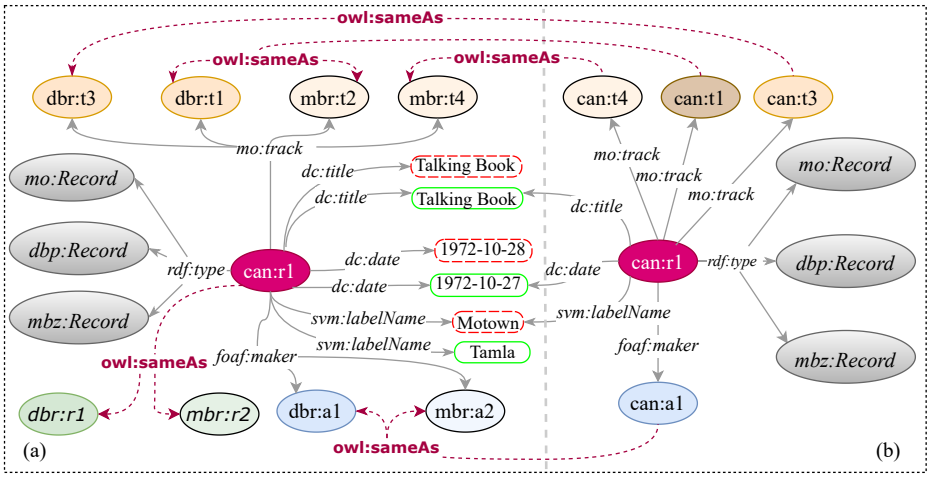

Figure 8. (a) Unification View for Canonical IRI *can:r1*. (b) Fusion View for Canonical IRI *can:r1*.

Each unification view consolidates semantically equivalent instances from different data sources into a single, unified representation. For example, the unification view *UV_Record* merges instances of *dbp:Record* from *EV_DBpedia* and *mbz:Record* from *EV_MusicBrainz*.

Figure 8 illustrates an instance of *UV_Record*, where resources *dbr:r1* (from *DBpedia*) and *mbr:r2* (from *MusicBrainz*) are identified as semantically equivalent. These resources are normalized to a canonical IRI, *can:r1*, which aggregates all classes, attributes, and relationships from both *dbr:r1* and *mbr:r2*.

The canonical IRI *can:r1* is computed by concatenating the local names (i.e., the last segments) of the original IRIs *dbr:r1* and *mbr:r2*. This approach ensures a unique and reproducible identifier for the unified entity, facilitating consistent referencing across the knowledge graph.

The metadata describing each unification view of *SV_Music*— including its generalization class, normalization function, and related artifacts — is available in[9].

5.7 Fusion Views Construction

In this last phase, building a fusion view for each unification view is crucial to resolve data conflicts and enhance the quality, accuracy, and reliability of the semantic view.

5.7.1 Development Process

To construct a fusion view for a unification view U involves three main tasks:

(i) *Identification of Properties with Conflicts Information* – Analysing the resource of U, it is possible to detect the properties that have discrepancy when combining various representations of the same real-world object into a single view (canonical IRI);

(ii) *Creation of Property Fusion Views* – For each identified conflicting property, define a Property Fusion View (PFV) that specifies a conflict resolution function. This function takes as input the set of conflicting values for a property and outputs a single, resolved value. Common conflict resolution strategies include: *KeepSingleValueByReputation(), KeepMostFrequentValue(), KeepMostRecentValue()*. Each PFV is materialized and stored in a dedicated named graph, promoting modular management and provenance tracking of triples;

(iii) *Implementation of the Fusion View* – The fusion view itself is implemented on-demand using SPARQL queries dynamically. Thus, for a given resource r in U, the fusion view state is computed using SPARQL queries with CONSTRUCT or INSERT, according to the preference of the integration approach. This approach ensures that the fusion view remains up-to-date with underlying data changes. Section 6.3 presents an example.

5.7.2 Fusion View of *SV_Music*

In the semantic view *SV_Music*, three fusion views are constructed corresponding to the previously defined unification views: *FV_MusicArtist*, *FV_Record*, and *FV_Track*.

For *FV_Record*, discrepancies were identified in two properties: *svm:labelName* and *dc:date*. To address these conflicts, property fusion views (PFVs) are defined for each property, employing the conflict resolution function *KeepSingleValueByReputation()*. This function selects the value from the most reputable source among the conflicting ones, thereby enhancing data quality and trustworthiness.

Figure 8 illustrates the fusion view for the canonical resource *can:r1*, which is generated by merging the records *dbr:r1* (from *DBpedia*) and *mbr:r2* (from *MusicBrainz*). The *KeepSingleValueByReputation()* function resolves conflicts in the *svm:labelName* and *dc:date* properties by selecting the most reputable values.

In the fusion view, object properties reference the canonical IRIs of related entities. For example, the *foaf:maker* property of *can:r1* points to the canonical IRI *can:a1*, which represents the unified artist entity derived from *mbr:a2* and *dbr:a1*.

This approach ensures that the fusion views in *SV_Music* provide a consistent and accurate representation of musical records, artists, and tracks by effectively resolving data conflicts and unifying information from

[9] https://semantic-ekgraphs.github.io/semantic-music/visoes_unificacao.html

multiple sources.

The metadata describing each fusion view of *SV_Music*—including its set of property fusion view assertions, alog with their associated named graphs, and related artifacts — is available through the *SV_Music* Semantic Portal[10].

6 A Tool to Explore Resources into Semantic View Using Various Context

This section introduces *ContextEKG_Explorer*, an interactive graphical tool designed to explore the EKG's semantic view, constructed using the *DDP_SV* presented in Section 3.

In the context of an EKG, resource visualization is typically entity-centric, where entities such as "Artist" and "Record" are the central focus. This visualization highlights the properties of these entities and their connections to other entities within the graph. This approach facilitates a deeper understanding of patterns, relationships, and insights within the EKG, making it easier to identify key information.

The key distinction of *ContextEKG_Explorer* is its ability to allow users to explore semantic view entities in multiple contexts. Different contexts provide varied perspectives on the same data. By exploring entities across these contexts, users can uncover insights that might remain hidden in a single, unified view, thus facilitating better decision-making. Additionally, context-based exploration aids in identifying and resolving inconsistencies or gaps in the data, further enhancing the quality and reliability of the information.

ContextEKG_Explorer allows users to explore and visualize a resource in the semantic view across three different contexts: the exported view context, the unification view context, and the fusion view context, as detailed in the following subsections.

6.1 Visualization in Context of Exported Views

To visualize a resource in the exported view (EV) context, the user should follow these steps: first, select an EV; next, choose a class within that EV; and finally, select a resource from the chosen class. For example, consider the resource *dbr:r1* shown in Figure 7(a), which is an instance of the class *svm:Record_DB* from the exported view *EV_DBpedia*. Figure 9 illustrates the resource exploration screen, displaying information about *dbr:r1*, labeled as "Talking Book".

Figure 9. Resource Exploration Screen of *dbr:r1* in Exported View Context.

[10] https://semantic-ekgraphs.github.io/semantic-music/visoes_fusao.html

Figure 10. Resource exploration screen in the Unification View context.

On the left side of the resource exploration screen in Figure 9, the current state of *dbr:r1* is displayed, with information retrieved from the data graph of the *EV_DBpedia*. The state of a resource refers to the current set of properties, values, and relationships associated with that resource. For object properties, such as "maker", the tool allows interactive navigation through the IRI of the referenced object. For instance, the user can navigate to the artist "Stevie Wonder" (*dbr:a1*), who made the record *dbr:r1*. This action displays the state of the artist "Stevie Wonder" in the context of *EV_DBpedia*.

On the right side of this same screen, the context menu of *dbr:r1* is displayed. This menu lists all contexts in which *dbr:r1* can be explored, with the currently active context highlighted in yellow. The user can pick a new context from this menu, allowing them to see the entity in a different way.

For example, the user can switch to the context of the *EV_MusicBrainz*. This action displays the resource exploration screen of the *mbr:a2*, the instance of class *svm:Record_MB* that has a "sameAs" link with *dbr:a1*. If the user chooses to switch to the unification or fusion view of a resource r, the respective state of the canonical resource of r is displayed, as discussed in Sections 6.2 and 6.3.

6.2 Visualization in Context of Unification Views

To visualize a resource in the context of a unification view, the user should first select a generalization class G, and then choose an instance of the unification view of G.

Consider, for example, *can:r1*, an instance in unification view of generalization class *mo:Record*, shown in Figure 8(a). Figure 10 illustrates the resource exploration screen, displaying the unification view of resource *can:r1*, which is the canonical entity for resources *dbr:r1* and *mbr:r2*.

The unification view is virtual; therefore, the unification view of *can:a1* is dynamically computed and generated at the exploration time, as specified in Definition 4.3. As shown in Figure 10, during the construction of the unification view of *can:a1*, the tool aggregates all properties from resources *dbr:r1* and *mbr:r2*, presenting the current state of *can:a1*, and detects discrepancies between the grouped values of the properties. It also captures the provenance of each value.

It is also possible to change context when exploring a resource r in the unification view context. The user can switch

Figure 11. Resource exploration screen in Fusion View context.

to the context of an exported view that includes a resource for which r is the canonical entity. Additionally, the user can switch to the fusion view context of the resource r to display the state of the fusion view of r, as discussed in Section 6.3.

6.3 Visualization in Context of Fusion Views

To visualize a resource in the context of a fusion view, the user should first select a generalization class G, and then choose a canonical resource from the fusion view of G. Consider the resource *can:r1* shown in Figure 8(b). Figure 11 illustrates the resource exploration screen, displaying information for the fusion view of *can:r1*. The fusion view is virtual; therefore, the state of *can:a1* is dynamically computed as specified in Definition 4.7.

While exploring a resource in the fusion view context, the user can choose to switch to one of the other contexts listed in the resource's context menu. The *ContextEKG_Explorer* tool also offers the functionality to browse external websites, such as those connected via the homepage property. This feature enhances the user's ability to access further information beyond the immediate scope of the EKG. Moreover, it facilitates integration with visual RDF browsers, enabling users to be redirected for visual exploration in tools like *GraphDB*[11].

7 Conclusions

This paper introduced an innovative data design pattern (*DDP_SV*) specifically developed for the construction and management of the semantic view of an Enterprise Knowledge Graph (EKG). The proposed architecture structures both data and metadata into a four-level hierarchy of views, offering a conceptual framework for semantic data management and enhanced utilization within EKG systems.

Another key contribution of this work is a novel incremental methodology to construct the semantic view, grounded in *DDP_SV*. This methodology uses a *pay-as-you-go* strategy, allowing organizations to incrementally build and maintain their knowledge graph.

To validate the methodology, we constructed an EKG for the music domain, referred to as *SV_Music*. This knowledge graph integrates music-related data from two open data sources, with both its data and metadata graphs made freely available in semantic portal `https://semantic-ekgraphs.github.io/semantic-music/`.

[11] https://graphdb.ontotext.com/documentation/10.6/

In addition, we develop an interactive graphical interface to facilitate the exploration of semantic resources across multiple contextual views. This tool enables users to trace the data lineage of individual resources, offering visibility into their origins, transformations, and integration processes. By elucidating these transformational flows, the interface enhances users' ability to manage and interpret semantic data within the EKG framework.

As future work, we intend to further explore the use of this framework for constructing semantic vision, exploring new approaches and techniques, seeking to:

- A practical implementation of the proposed framework is still required, as well as an empirical validation involving real users;
- We plan to propose a multi-agent approach with large language models, based in our recently preliminary study [Rolim *et al.*, 2025], aligned with the conceptual framework proposed in this paper, to assist in the incremental and interactive construction of semantic views for EKGs.
- Furthermore, we plan to explore how the semantic view can support the construction of domain-specific data mashups for analytical applications, using the semantic view as a navigational and operational guide across complex enterprise data ecosystems.

Funding

This research was funded by Funcap (Grant #BMD-0008-00739.01.10/24).

References

Angelis, S., Moraitou, E., Caridakis, G., and Kotis, K. (2024). Chekg: a collaborative and hybrid methodology for engineering modular and fair domain-specific knowledge graphs. *Knowledge and Information Systems*. DOI: 10.1007/s10115-024-02110-w.

Azizi, S. (2023). Documenting data integration using knowledge graphs. Mestrado em ciência da computação, Gottfried Wilhelm Leibniz Universität, Hannover.

Brickley, D. (2005). Foaf vocabulary specification.

Cimmino, A. and García-Castro, R. (2024). Helio: a framework for implementing the life cycle of knowledge graphs. *Semantic Web*, 15(1):223–249. DOI: 10.3233/SW-233224.

Das, S., Sundara, S., and Cyganiak, R. (2012). R2rml: Rdb to rdf mapping language. w3c recommendation.

De Souza, E. M. F., Rossanez, A., dos Reis, J. C., and da Silva Torres, R. (2022). Visualização interativa da evolução de grafos de conhecimento. In *Anais do XXXVII SBBD*, pages 343–354. SBC. DOI: 10.5753/sbbd.2022.224301.

Deb Nath, R. P., Hose, K., Pedersen, T. B., Romero, O., and Bhattacharjee, A. (2020). Setlbi: An integrated platform for semantic business intelligence. In *Companion Proceedings of the Web Conference 2020*, pages 167–171. DOI: 10.1145/3366424.3383533.

Debattista, J., Auer, S., and Lange, C. (2016). Luzzu—a methodology and framework for linked data quality assessment. *Journal of Data and Information Quality (JDIQ)*, 8(1):1–32. DOI: 10.1145/2992786.

Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., and Van de Walle, R. (2014). Rml: A generic language for integrated rdf mappings of heterogeneous data.

Ehrlinger, L. and Woss, W. (2016). Towards a definition of knowledge graphs.

Galkin, M., Auer, S., Vidal, M. E., and Scerri, S. (2017). Enterprise knowledge graphs: A semantic approach for knowledge management in the next generation of enterprise information systems. In *International Conference on Enterprise Information Systems*, volume 2, pages 88–98. DOI: 10.5220/0006325200880098.

Grainger, T., AlJadda, K., Korayem, M., and Smith, A. (2016). The semantic knowledge graph: A compact, auto-generated model for real-time traversal and ranking of any relationship within a domain. In *2016 ieee international conference on data science and advanced analytics (dsaa)*, pages 420–429. IEEE. DOI: 10.1109/DSAA.2016.51.

Haase, P., Herzig, D. M., Kozlov, A., Nikolov, A., and Trame, J. (2019). metaphactory: A platform for knowledge graph management. *Semantic Web*, 10:1109–1125. DOI: 10.3233/sw-190360.

Ham, K. (2013). Openrefine (version 2.5). http://openrefine.org. free, open-source tool for cleaning and transforming data. *Journal of the Medical Library Association: JMLA*, 101(3):233. DOI: 10.3163/1536-5050.101.3.020.

Neto, L. E. T., Vidal, V. M. P., Casanova, M. A., and Monteiro, J. M. (2013). R2rml by assertion: A semi-automatic tool for generating customised r2rml mappings. In *The Semantic Web: ESWC 2013 Satellite Events: ESWC 2013*, pages 248–252. Springer. DOI: 10.1007/978-3-642-41242-433.

Noy, N. F., McGuinness, D. L., *et al*. (2001). Ontology development 101: A guide to creating your first ontology.

Paton, N. W., Belhajjame, K., Embury, S. M., Fernandes, A. A., and Maskat, R. (2016). Pay-as-you-go data integration: Experiences and recurring themes. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 81–92. Springer. DOI: 10.1007/978-3-662-49192-87.

Paton, N. W., Christodoulou, K., Fernandes, A. A., Parsia, B., and Hedeler, C. (2012). Pay-as-you-go data integration for linked data: opportunities, challenges and architectures. In *Proceedings of the 4th International Workshop on Semantic Web Information Management*, pages 1–8. DOI: 10.1145/2237867.2237870.

Rehman, H. U. *et al*. (2025). Intelligent configuration management in modular production systems: Integrating operational semantics with knowledge graphs. *Journal of Manufacturing Systems*, 80:610–625. DOI: 10.1016/j.jmsy.2025.03.017.

Rolim, T. V., Freitas, J. R., and Vidal, V. (2025). Metadata-driven construction of semantic views in enterprise knowledge graphs with llm agents. In *Anais Estendidos do XL Simpósio Brasileiro de Bancos de Dados*, pages 468–478, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/sbbd_estendido.2025.248167.

Schultz, A., Matteini, A., Isele, R., Bizer, C., and Becker, C. (2011). Ldif-linked data integration framework. In *Proceedings of the Second International Conference on Consuming Linked Data-Volume 782*, pages 125–130. DOI: 10.5555/2887352.2887364.

Sellami, S. and Zarour, N. E. (2022). Keyword-based faceted search interface for knowledge graph construction and exploration. *International Journal of Web Information Systems*, 18(5/6):453–486. DOI: 10.1108/IJWIS-02-2022-0037.

Sequeda, J. and Lassila, O. (2021). Building enterprise knowledge graphs. In *Designing and Building Enterprise Knowledge Graphs*, pages 97–128. Springer. DOI: 10.2200/s01105ed1v01y202105dsk020.

Simsek, U., Kärle, E., Angele, K., Huaman, E., Opdenplatz, J., Sommer, D., Umbrich, J., and Fensel, D. (2022). A knowledge graph perspective on knowledge engineering. *SN Computer Science*, 4(1):16. DOI: 10.1007/s42979-022-01429-x.

Tian, X. *et al*. (2025). Construction and application of a multi-modal knowledge graph integrated with large language models in the field of manufacturing processes. In *2025 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 288–292. IEEE. DOI: 10.1109/ICAIIC64266.2025.10920784.

Vidal, V., Freitas, R., Arruda, N., Casanova, M. A., and Renso, C. (2024). A data design pattern for building and exploring semantic views of enterprise knowledge graphs. In *Anais do XXXIX SBBD*, pages 1–13, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/sbbd.2024.241024.

Volz, J., Bizer, C., Gaedke, M., and Kobilarov, G. (2009). Silk-a link discovery framework for the web of data.

Weibel, S., Kunze, J., Lagoze, C., and Wolf, M. (1998). Dublin core metadata for resource discovery. Technical report, RFC.

Yang, H. *et al*. (2024). An LLM supported approach to ontology and knowledge graph construction. In *2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 5240–5246. IEEE. DOI: 10.1109/BIBM62325.2024.10822222.