

# A Data Design Pattern for Building and Exploring Semantic Views of Enterprise Knowledge Graphs

Vânia M. P. Vidal<sup>1</sup>, Renato Freitas<sup>1</sup>, Narciso Arruda<sup>1</sup>,  
Marco A. Casanova<sup>2</sup>, Chiara Renso<sup>3</sup>

<sup>1</sup>Department of Computing – Federal University of Ceará  
Fortaleza – CE – Brazil

<sup>2</sup>Department of Informatics – Pontifical Catholic University of Rio de Janeiro  
Rio de Janeiro – RJ – Brazil

<sup>3</sup>ISTI Institute of National Research Council – Pisa – Italy

{vvania,narciso}@lia.ufc.br, renato.freitas@alu.ufc.br  
casanova@inf.puc-rio.br, chiara.renso@isti.cnr.it

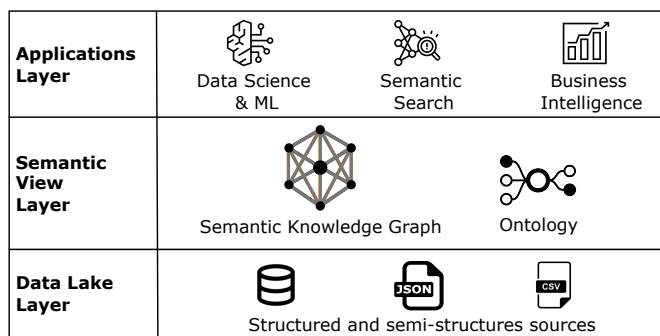
**Abstract.** *An Enterprise Knowledge Graph (EKG) is a robust foundation for knowledge management, data integration, and advanced analytics across organizations. It achieves this by offering a semantic view that semantically integrates various data sources within an organization’s data lake. This paper introduces a novel data design pattern (DDP) aimed at constructing and managing the semantic view of an EKG. The proposed DDP logically organizes data into three hierarchical levels, facilitating the maintenance and the versatile exploration of the semantic view in various contexts. Furthermore, this paper details an interactive graphical interface developed to support context-sensitive navigation of the semantic view, enhancing user interaction and resource utilization.*

## 1. Introduction

In recent years, large-scale data management has become a critical challenge for organizations. The emergence of concepts such as Big Data and the need for complex analyses demand solutions that not only store large volumes of data but also allow for efficient interpretation and analysis [Ruan et al. 2016]. Enterprise Knowledge Graphs emerged as a promising solution for these demands [Ehrlinger and Wöß 2016].

An Enterprise Knowledge Graph (EKG) is a robust foundation for knowledge management, data integration, and advanced analytics across organizations, as noted by [Grainger et al. 2016], by offering a semantic view. 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.

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 a centralized storage system designed to accommodate various data types, including raw, unstructured, and (semi-)structured data; (ii) the *Semantic View Layer* is responsible for semantically integrating data from various sources, aiming to construct a unified and



**Figure 1. Architecture of EKG Systems.**

coherent presentation known as the semantic view. A central element of this layer is the ontology of the semantic view. This ontology is vital for defining a common vocabulary that facilitates the transformation and integration of data from diverse sources; (iii) the *Application Layer* implements specific applications or services that leverage the integrated data derived from the *Semantic View Layer*. Applications within this layer utilize the unified data model to extract insights, facilitate decision-making, and drive business processes.

The primary contribution of this paper is the introduce a novel data design pattern (DDP) specifically crafted for logically organizing data within a semantic view of an EKG. This DDP strategically organizes the data and metadata of the semantic view into three hierarchical levels: exported semantic views, linkset and unification views, and fusion views. This structured approach addresses the specific challenges of semantic data integration and simplifies maintenance and enhances the flexibility and depth of exploring the semantic view across various contexts.

This paper also details an interactive graphical interface developed to support context-based navigation of semantic view. By leveraging 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 origins, transformations, and data integrations within the knowledge graph.

The subsequent sections are structured as follows. Section 2 discusses related works. Section 3 introduces a data design pattern for logically organizing data within a semantic view of an EKG. A case study in Section 4 shows how a semantic view is constructed using the DDP. Section 5 presents the graphical interface designed for exploring the resources of the semantic view. Finally, Section 6 expose the conclusions.

## 2. Related Work

The problem of constructing an EKG by combining large-scale data processing with robust semantic technologies has been addressed in several previous works [Vidal et al. 2015] [Ruan et al. 2016] [Galkin et al. 2016] [Song et al. 2017]. Most of these works focus on different challenges related to constructing the EKG’s semantic view.

[Vidal et al. 2015] proposed a framework for specification and incremental maintenance of Linked Data Mashup views. A Linked Data Mashup view is constructed by

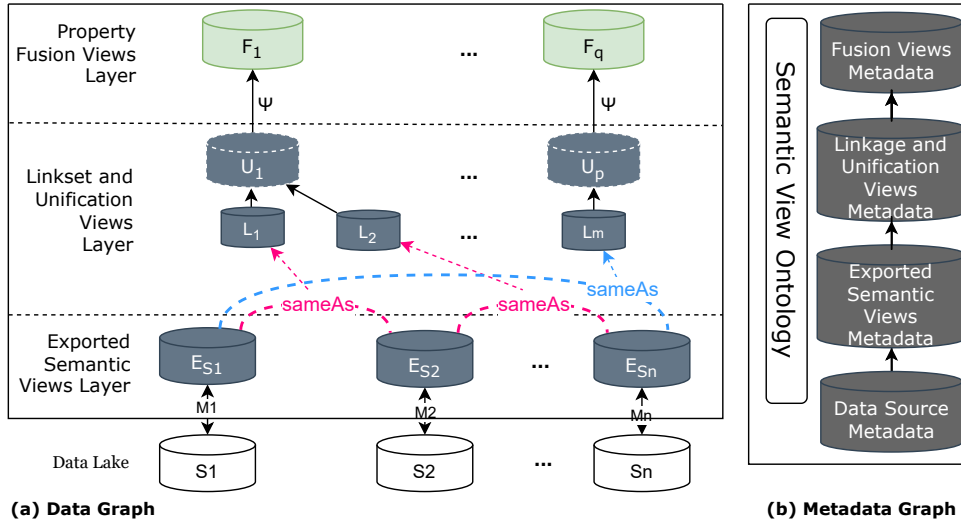


Figure 2. Data Design Pattern for EKG’s Semantic Views.

integrating data available on the Web of Data, tailored to a specific application. This work evolves from the framework in [Vidal et al. 2015], addressing the shift from application-oriented to domain-oriented views. The semantic view differs from a data mashup view in that it requires the development and implementation of a detailed and generalized ontology to ensure consistent data interpretation across the organization. Consequently, the semantic view can be leveraged to construct specialized data mashups for analytical applications.

Regarding previous works in knowledge graph exploration, several contributions have been made. [Haase et al. 2019] introduced “metaphacts”, a platform based on open standards that aims to facilitate the management of knowledge graphs (KGs) and provides customizable interfaces for various usage scenarios. [de Souza et al. 2022] developed the “TKGEvolViewer” tool to visualize the evolution of KGs. [Sellami and Zarour 2022] introduced KeyFSI, a keyword-based, faceted navigation user interface designed to facilitate data exploration and visualization within a KG. Additionally, [Avila and Vidal 2023] presented LiRB, an interactive web interface that allows for exploring KGs. Despite these significant contributions, none of these tools can be applied to explore the different contexts of semantic views in EKG systems.

### 3. Data Design Pattern for Constructing EKG’s Semantic Views

#### 3.1. Overview

This section introduces a Data Design Pattern (DDP) to organize the data and metadata of the semantic view logically. This DDP provides a solid and reliable structure to address the specific challenges of semantic data integration, promoting consistent interpretation and effective use of the semantic view. The semantic view in this architecture comprises two distinct graphs: the Data Graph and the Metadata Graph, as depicted in Figure 2.

The Data Graph, shown in Figure 2(a), represents the actual data of the semantic view. It is formally delineated through three hierarchical levels of views: *Exported Semantic Views*, *Linkset and Unification Views*, and *Fusion Views*. These levels are defined in detail in the following subsections.

The Metadata Graph, depicted in Figure 2(b), acts as the repository for the metadata of the semantic view. It plays a crucial role by providing detailed information about the specifications of views at various levels within the semantic view’s Data Graph. This facilitates several key functions, including: usability, data lineage, and governance. It is important to highlight that the Metadata Graph must be intrinsically linked to the Data Graph. This interconnected approach ensures a comprehensive understanding of the semantic view. It enhances the management and active use of Data and Metadata graphs, leading to better data management, usability, and governance.

### 3.2. Exported Semantic Views

In the proposed architecture, every data source within the Data Lake exports an RDF view, named *Exported Semantic View* (ESV), created through a systematic mapping of the source data to a common and shared vocabulary defined by the semantic view ontology.

**Definition 3.1 (Specification of Exported Semantic View).** *The specification of an exported semantic view is a tuple  $\langle \mathcal{E}, S, \mathcal{O}, M \rangle$ , where:*

- $\mathcal{E}$  is the name of the exported semantic view.
- $S$  is the data source in the Data Lake that exports  $\mathcal{E}$ .
- $\mathcal{O}$  is the ontology of the semantic view.
- $M$  is a set of mapping rules between  $S$  and  $\mathcal{O}$ .

Given a specification of an ESV  $\langle \mathcal{E}, S, \mathcal{O}, M \rangle$  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,

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

### 3.3. Linkset Views

The linkset views creation process involves defining relationships or “sameAs” links between entities that are equivalent across different exported views. These “sameAs Links” are inferred by matching property values of resources within these exported views, and are essential for aligning and connecting resources from various data sources. To establish these links, users should first define the sameAs linkset view by specifying the classes of the exported views and the match function.

**Definition 3.2 (Specification of Linkset View).** *The specification of a linkset view is a quadruple  $\langle \mathcal{L}, T, W, \mu \rangle$ , where:*

- $\mathcal{L}$  is the name of the linkset view.
- $T$  and  $W$  are classes of different exported semantic views.
- $\mu$  is a “match function”.

The match function is designed to compare 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  $\mu$ ; 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:

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

### 3.4. Generalization Classes and Unification Views

In the design of the EKG’s semantic view ontology, the two classes,  $T$  and  $W$ , mentioned in the specification of a linkset view, are defined as semantically equivalent and are grouped within the same *Equivalence Class Hierarchy* (ECH). Furthermore, a generalization class is established to represent a broader concept encompassing all other classes within the same ECH. Consequently, every class within a given ECH is a subclass of its corresponding generalization class.

The use of generalization classes contributes to a more organized and understandable structure of the ontology of the semantic view. Besides, generalization classes play a crucial role in defining unification views that bring together resources from diverse exported views linked by the “sameAs” relationship. A unification view should be specified for each Generalization class in the semantic view, defined as follows:

**Definition 3.3 (Specification of Unification View).** *The specification of a unification view for a generalization class  $G$  is a triple  $\langle \mathcal{U}, G, \eta \rangle$ , where:*

- $\mathcal{U}$  is the name of the unification view.
- $G$  is a generalization class.
- $\eta$  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:

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

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

$$\forall x_1 \forall x_2 \in G(t) (x_1 \text{ sameAs } x_2 \Leftrightarrow \eta(x_1, t) = \eta(x_2, t)).$$

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 “ $x_1 \text{ sameAs } x_2$ ”. The unification view aids helps forming a unified depiction of entities by gathering attributes and relationships from multiple ESVs. This facilitates the visualization and querying of resources within a unified context, enabling a more comprehensive understanding and analysis of interconnected data, as described in Section 5.2.

### 3.5. Fusion Views

The goal of a fusion view is to resolve conflicts that may arise when different sources provide divergent information about the same entity or relationship, aiming to improve the information’s quality, accuracy, and reliability in the semantic view. From the unification view of a resource, it is possible to detect conflicting information about the same entity or relationship. The solution for this type of inconsistency requires implementing conflict resolution mechanisms to identify and resolve discrepancies.

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). This is specified with the help of “*Property Fusion Assertion*”.

**Definition 3.4 (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  $G$ .
- $\Psi$  is a “conflict resolution function”.

The conflict resolution function  $\Psi$ , accepts as input a canonical IRI  $c$  and a set of values for  $p$  of  $c$ , and produces a single value. Definitions 3.5 e 3.6 below define how to solve conflicts based on a PFA of data type properties and object properties, respectively.

**Definition 3.5 (Conflict Resolution based on PFA of Data Type Properties).** Let:

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

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

**Definition 3.6 (Conflict Resolution based on PFA 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 specification of a unification view for  $G$ .
- $\mathcal{U}(t)$  be the state of  $\mathcal{U}$  at time  $t$ .
- $c$  be a canonical IRI in  $\mathcal{U}(t)$ .
- $V = \{u \mid (c, p, o) \in \mathcal{U}(t) \text{ and } \eta(o, t) = u\}$ ;  $V$  contains the set of all canonical IRIs related with  $c$  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:

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

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 3.7 (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  $t$ .
- $\mathcal{A}_1, \dots, \mathcal{A}_n$  be all PFAs of generalization class  $G$ .
- $p_1, \dots, p_n$  be the properties associated with PFAs  $\mathcal{A}_1, \dots, \mathcal{A}_n$ , 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:

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

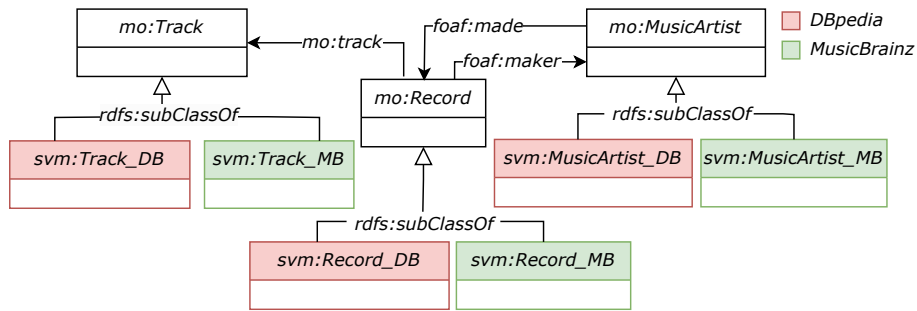


Figure 3. Main Classes and Relationships of *SV\_Music.OWL*.

#### 4. Case Study: Building the Semantic View ‘*SV\_Music*’

This section discusses the use of the DDP presented in Section 3 to construct a semantic view, called *SV\_Music*, which integrates music related data from two data sources: *DBpedia*<sup>1</sup> and *MusicBrainz*<sup>2</sup>. The main components of *SV\_Music* are described in the following.

##### 4.1. Ontology of ‘*SV\_Music*’ and Its Exported View Ontologies

The ontology of the semantic view *SV\_Music*, named *SV\_Music.OWL*, is constructed by uniting the vocabularies of exported semantic view ontologies: *ESV\_DBpedia* and *ESV\_MusicBrainz*, derived, respectively, from data sources *DBpedia* and *MusicBrainz*. Figure 3 depicts, in UML, the main classes of the *SV\_Music.OWL*. It reuses terms from three well-known vocabularies: Dublin Core (DC), Friend of a Friend (FOAF), and Music Ontology (MO). For the new terms, we use the prefix “svm:”.

*SV\_Music.OWL* has the generalization classes *mo:Track*, *mo:Record*, and *mo:MusicArtist* to represent broader concepts encompassing their subclasses. For example, the generalization class *mo:Track*, has two subclasses *svm:Track\_DB* and *svm:Track\_MB*, which are defined exclusively for annotating the instances of classes *svm:Track\_DB* and *svm:Track\_MB*, respectively. This allows for tracking the provenance of the resources and visualizing the resource in different contexts. The class *svm:Track\_DB*, for instance, is exclusively for the ontology of the *ESV\_DBpedia*, meaning that instances of *svm:Track\_DB* are originated from the *DBpedia* data source.

Figure 4 shows a fragment of the *ESV\_DBpedia* and *ESV\_MusicBrainz*. The ontology of the ESV are a subset of the *SV\_Music.OWL*, and they should share the same vocabulary for common properties of subclasses of the same generalization class. For example, the resources *dbr:r1* and *mbr:r2* are instances of generalization class *mo:Track*. These instances share the properties: *foaf:homepage*, *svm:labelName*, *dc:title*, *dc:date*, *foaf:name*, *dc:description*, *mo:track*, *foaf:maker*, and *foaf:made*.

##### 4.2. Linkset Views of ‘*SV\_Music*’

In addition to the exported semantic view, *SV\_Music* has three linkset views for establishing “sameAs” links connecting resources of the exported semantic views *ESV\_DBpedia* and *ESV\_MusicBrainz*:

<sup>1</sup><https://www.dbpedia.org/>

<sup>2</sup><https://musicbrainz.org/>

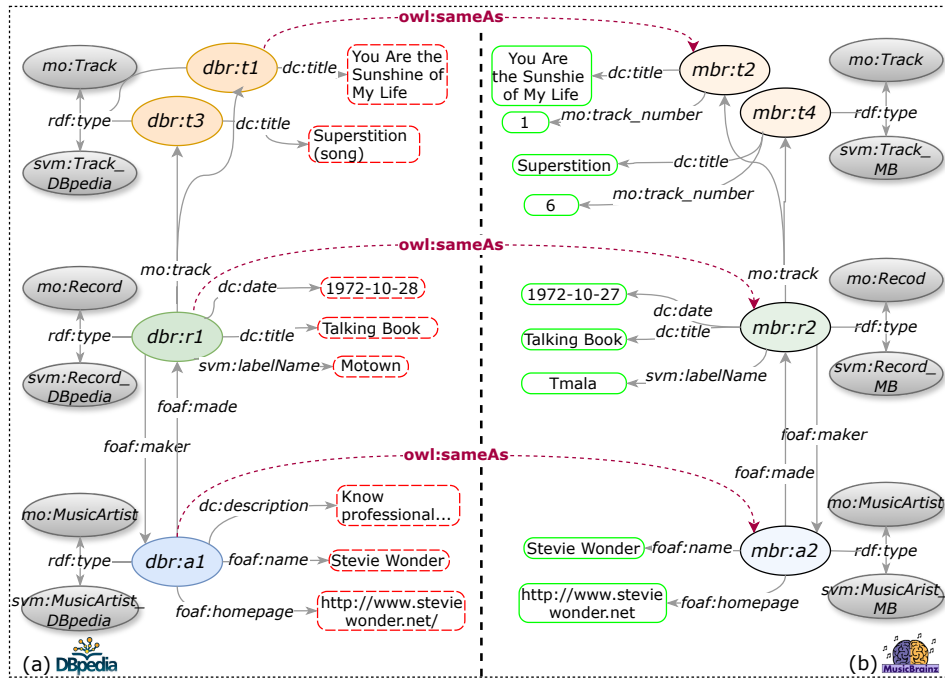


Figure 4. (a) Fragment of *ESV\_DBpedia*; (b) Fragment of *ESV\_MusicBrainz*.

- $L_{Record}$ : establish sameAs links between instances of *svm:Record\_DB* and *svm:Record\_MB*.
- $L_{Track}$ : establish sameAs links between instances of *svm:Track\_DB* and *svm:Track\_MB*.
- $L_{MusicArtist}$ : establish sameAs links between instances of *svm:MusicArtist\_DB* and *svm:MusicArtist\_MB*.

Figure 4 shows examples of links for each linkset view.

### 4.3. Unification Views of ‘*SV\_Music*’

The semantic view *SV\_Music* includes three unification views, each corresponding to a generalization class: *mo:Track*, *mo:Record*, and *mo:MusicArtist*. These unification views are virtual, and they are designed to create a perspective for resources within a generalization class that are connected by *owl:sameAs* links.

For instance, as illustrated in Figure 5(a), the resources *dbr:r1* and *mbr:r2*, which belong to the generalization class *mo:Record*, are normalized to a canonical IRI *can:r1* in the unification view of generalization class *mo:Record*. This canonical IRI aggregates all attributes, and relationships from both *dbr:r1* and *mbr:r2*.

### 4.4. Fusion Views of ‘*SV\_Music*’

The goal of a fusion view is to resolve conflicts when different sources provide divergent information for property  $p$  for the same entity in a generalization class. In the context of the generalization class *mo:Record*, two property fusion views were defined to solve conflicts for properties *svm:labelName* and *dc:date*. These conflicts were addressed using property fusion assertions  $A_1$  and  $A_2$ , defined as follows:

- $\langle A_1, mo:Record, svm:labelName, KeepSingleValueByReputation() \rangle$



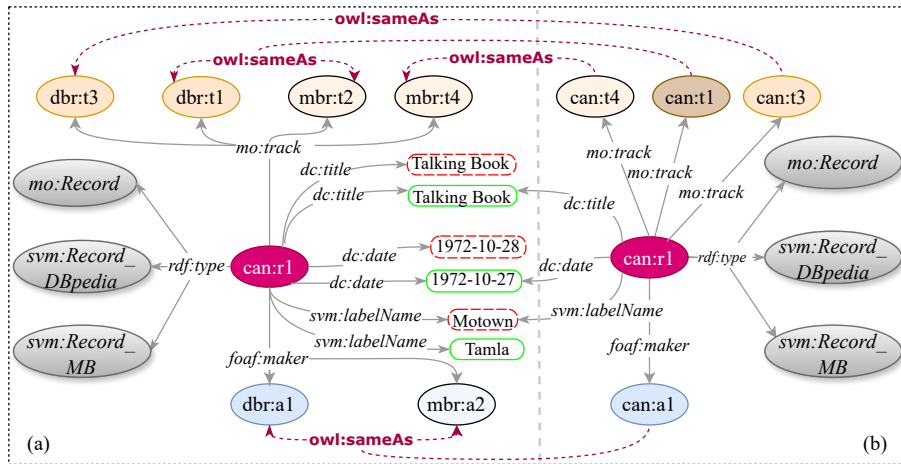


Figure 5. (a) Unification View for Canonical IRI *can:r1*. (b) Fusion View for Canonical IRI *can:r1*.

- $\langle A_2, mo:Record, dc:date, KeepSingleValueByReputation() \rangle$

Using property fusion assertions  $A_1$  and  $A_2$ , the fusion view of the resource *can:r1* produces a single triple for *svm:labelName* and *dc:date*. The conflict resolution function *KeepSingleValueByReputation()* resolves conflicts based on the reputation of the sources. Figure 5(b) shows the fusion view for the canonical IRI *can:r1*.

It is important to note that in the fusion view, the object property references the canonical IRIs. For example, in Figure 5(b), the resource *can:a1* is the canonical IRI for the resources *mbr:a2* and *dbr:a1*. Therefore, the object property *foaf:maker* of record *can:r1* references the canonical uri *can:a1*.

## 5. A Tool for Exploring Semantic View Resources Using Various Contexts

This section introduces *ContextEKG\_Explorer*, an interactive graphical tool designed to explore the semantic view of an EKG, constructed using the DDP 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. These 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.

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

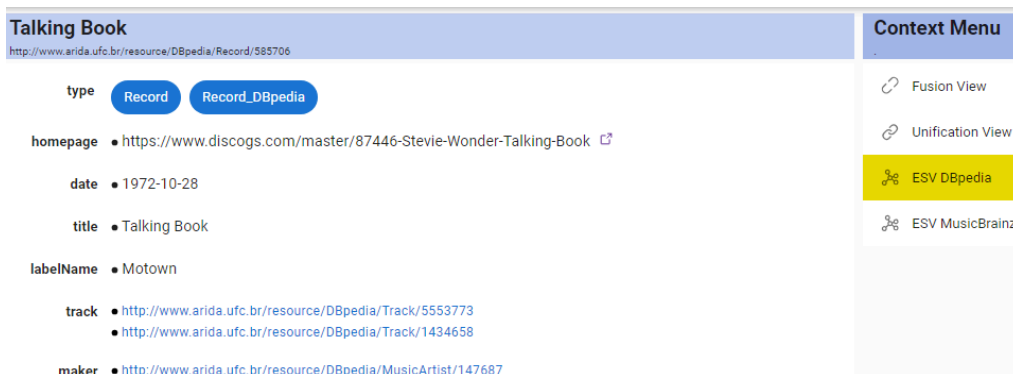


Figure 6. Resource Exploration Screen of *dbr:r1* in Exported View Context.

### 5.1. Visualization in Context of Exported Semantic Views

To visualize a resource in the Exported Semantic View (ESV) context, the user should follow these steps: first, select an ESV; next, choose a class within that ESV; and finally, select a resource from the chosen class. For example, consider the resource *dbr:r1* shown in Figure 4(a), which is an instance of the class *svm:Record\_DB* from the exported semantic view *ESV\_DBpedia*. Figure 6 illustrates the resource exploration screen, displaying information about the resource *dbr:r1*, labeled as “Talking Book”.

On the left side of the resource exploration screen in Figure 6, the current state of *dbr:r1* is displayed, with information retrieved from the data graph of the *ESV\_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 *ESV\_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 at the entity in a different way. For example, the user can switch to the context of the *ESV\_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 5.2 and 5.3.

### 5.2. 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 5(a). Figure 7 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 3.3. As shown in Figure 7, during the computation of the state of the unification view of *can:a1*,

the tool aggregates all properties from resources *dbr:r1* and *mbr:r2* and detects discrepancies between the grouped values of 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 to the context of an ESV 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 5.3.

### 5.3. 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 of the fusion view of *G*. Consider the resource *can:r1* shown in Figure 5(b). Figure 8 illustrates the resource exploration screen, displaying information for the fusion view of *can:r1*. The fusion view is virtual; therefore, the state of *can:r1* is dynamically computed as specified in Definition 3.7.

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<sup>3</sup>.

## 6. Conclusions

This paper introduced an innovative Data Design Pattern specifically developed for constructing and managing the semantic view of an EKG. The proposed architecture organized data and metadata into three hierarchical levels of views, providing a robust framework for efficient data management and utilization of the semantic view within EKG systems.

<sup>3</sup><https://graphdb.ontotext.com/documentation/10.6/>

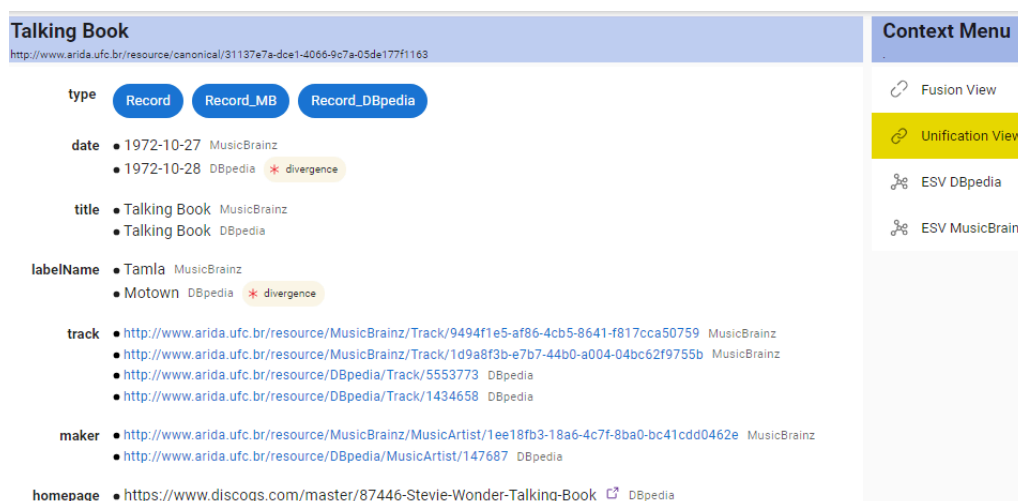


Figure 7. Resource exploration screen in the Unification View context.

**Figure 8. Resource exploration screen in Fusion View context.**

Additionally, the paper presented an interactive graphical interface designed to support the exploration of resources within the semantic view in various contexts. This interface allows users to effectively track the data lineage of particular resources through their transformational flows. This capability proved invaluable for understanding the origins, transformations, and integrations of data within the knowledge graph, offering deeper insights into data management and utilization. By providing a clear view of data lineage, the interface enhances users' ability to manage and utilize semantic data effectively within the EKG framework.

As future work, we will explore leveraging the EKG's semantic views for constructing specialized data mashups to be used in analytical applications. This involves using the semantic view as a roadmap for navigating and manipulating the complex landscape of enterprise data.

## 7. Acknowledgments

We would like to thank the *Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (Funcap) - Programa de Bolsas de Formação Acadêmica: Mestrado e Doutorado - Brasil* for financial support. Grant #BMD-0008-00739.01.10/24.

## References

- Avila, C. and Vidal, V. (2023). Lirb: Um navegador leve baseado em texto para knowledge graphs rdf. In *Anais Estendidos do XXXVIII Simpósio Brasileiro de Bancos de Dados*, pages 102–107, Porto Alegre, RS, Brasil. SBC.
- 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 Simpósio Brasileiro de Bancos de Dados*, pages 343–354. SBC.
- Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48(1-4):2.
- Galkin, M., Auer, S., and Scerri, S. (2016). Enterprise knowledge graphs: a backbone of linked enterprise data. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 497–502. IEEE.
- 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. SCITEPRESS.
- 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.
- Haase, P., Herzig, D. M., Kozlov, A., Nikolov, A., and Trame, J. (2019). metaphactory: A platform for knowledge graph management. *Semantic Web*, 10(6):1109–1125.
- Ruan, T., Xue, L., Wang, H., Hu, F., Zhao, L., and Ding, J. (2016). Building and exploring an enterprise knowledge graph for investment analysis. In *The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II 15*, pages 418–436. Springer.
- 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.
- Song, D., Schilder, F., Hertz, S., Saltini, G., Smiley, C., Nivarthi, P., Hazai, O., Landau, D., Zaharkin, M., Zielund, T., et al. (2017). Building and querying an enterprise knowledge graph. *IEEE Transactions on Services Computing*, 12(3):356–369.
- Vidal, V. M., Casanova, M. A., Arruda, N., Roberval, M., Leme, L. P., Lopes, G. R., and Renso, C. (2015). Specification and incremental maintenance of linked data mashup views. In *International Conference on Advanced Information Systems Engineering*, pages 214–229. Springer.