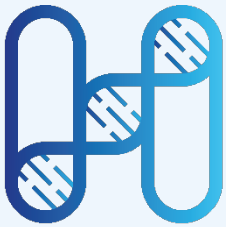

HEREDITARY

HetERogeneous sEmantic Data integratIon for the guT-bRain interplaY

Deliverable 3.1

Ontology and federated execution methods

Version 1.00, December 20, 2024

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No GA 101137074. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Funded by
the European Union

EXECUTIVE SUMMARY

The deliverable *Ontology and Federated Execution Methods* marks the achievement of **Milestone 5** in the project, focusing on two critical areas: semantic ontology modeling for neurological diseases (*Amyotrophic Lateral Sclerosis (ALS)* and *Multiple Sclerosis (MS)*) and the design of federated execution methods. It synthesizes WP2, WP3, and WP4 contributions to provide a comprehensive framework for integrating clinical and genomic data through advanced ontology design and scalable query execution frameworks.

At its core, the *Hereditary Ontology (HERO)*, central to this deliverable, ensures structured and interoperable representation of phenoclinical and genomic aspects of ALS and MS. Developed with adherence to the FAIR principles, it enables seamless integration with established standards, such as the *Simple Knowledge Organization System (SKOS)*, and lays the foundation for extensibility to other biomedical domains. HERO's modular structure, divided into clinical and genomic components, ensures clarity and scalability, while its hierarchical design captures the intricate relationships among symptoms, diagnostics, and genetic markers. Through iterative testing and real-world validation, HERO aligns closely with the domain's requirements, ensuring accurate representation and utility in the context of neurological diseases.

Hereditary federated execution environment enables robust analytics across clinical and genomic datasets. Two principal layers define the framework's architecture. The Data Federation and Virtualization Layer unifies disparate data sources into a virtualized model, ensuring efficient and scalable data integration. Secondly, the *Ontology-Based Data Access (OBDA)* Layer leverages semantic ontology-driven query rewriting and unfolding techniques to enable federated query execution. These layers work in tandem to bridge heterogeneity in data representation, fostering interoperability and enhancing federated analytic workflows.

The deliverable includes two primary use cases to validate the integrated framework. The first focuses on ALS data, demonstrating the system's ability to derive clinical insights from distributed datasets related to patient symptoms and treatment outcomes. The second explores genomics queries, underscoring the framework's capacity to handle distributed genomic data.

DOCUMENT INFORMATION

Deliverable ID	**3.1**
Deliverable Title	**Ontology and federated execution methods**
Work Package	**WP 3**
Lead Partner	**UNIPD**
Due Date	**30 December 2024**
Date of submission	**20 December 2024**
Deliverable Type	**R – Report**
Dissemination level	**PU – Public**

AUTHORS and CONTRIBUTORS

Name	Organization
Mirco Cazzaro (Author)	UNIPD
Laura Menotti (Author)	UNIPD
Todor Primov (Internal Reviewer)	ONTO
Manuel Rueda (Contributor)	CNAG
Gianmaria Silvello (Author)	UNIPD

REVISION HISTORY

Version	Date	Author	Description
0.10	2024-09-09	Gianmaria Silvello	Initial scheleton of the deliverable
0.11	2024-09-10	Laura Menotti	Draft of the Deliverable outline
0.12	2024-09-18	Laura Menotti	Add the first draft of the ontology section
0.13	2024-09-19	Mirco Cazzaro	Add the first draft of the subsection 3.1
0.14	2024-09-20	Mirco Cazzaro	Add the first draft of the subsection 3.3
0.15	2024-09-24	Mirco Cazzaro	Add the first draft of the subsection 3.4
0.16	2024-09-27	Mirco Cazzaro	First revision of subsection 3.1
0.17	2024-10-07	Mirco Cazzaro	Add the first draft of the 4.1
0.18	2024-10-09	Laura Menotti	Add the first draft of the clinical data modeling section
0.19	2024-10-22	Laura Menotti	Add the first draft of the genomic data modeling
0.20	2024-10-28	Gianmaria Silvello	Revision of the second draft
0.21	2024-11-10	Mirco Cazzaro	Revision of the genomics use case
0.22	2024-11-15	Gianmaria Silvello	Revision of the federated execution methods
0.23	2024-11-18	Gianmaria Silvello	Extended abstract and introduction
0.24	2024-11-18	Laura Menotti	Revision of the ontology section
0.25	2024-11-20	Mirco Cazzaro	New use-case about genomics data
0.26	2024-11-21	Gianmaria Silvello	Revision and added conclusions
0.26	2024-12-06	Manuel Rueda	Comments on HERO-Genomics
0.27	2024-12-09	ALL	Final revision after internal review
0.28	2024-12-18	Gianmaria Silvello	Minor checks and layout fixes
1.00	2024-12-20	Gianmaria Silvello	Finalization and printout

Contents

List of Tables

List of Figures

1 Introduction

This deliverable, **Ontology and Federated Execution Methods**, achieves **Milestone 5** of the project, focusing on the semantic ontology modeling concepts and processes for neurological diseases, specifically *Amyotrophic Lateral Sclerosis (ALS)* and *Multiple Sclerosis (MS)*, while detailing the design and initial implementation of federated execution methods. The document integrates contributions from Work Packages WP2, WP3, and WP4 to address the dual objectives of comprehensive ontology design and robust federated data query systems.

The *Hereditary Ontology (HERO)* ontology emphasizes the representation of clinical and genomic data relevant to ALS and MS. It adheres to the FAIR principles (*Findable, Accessible, Interoperable, and Reusable*), ensuring compatibility with existing standards such as the *Simple Knowledge Organization System (SKOS)* and facilitating reuse in other biomedical domains. The ontology schema is informed by domain requirement analysis, ensuring alignment with the pheno-clinical and genomic specifics of neurological disorders.

Key design choices include:

- Hierarchical structuring to capture complex relationships among clinical symptoms, diagnostic criteria, and genomic markers.
- Modular architecture, dividing the ontology into phenoclinical and genomic components for focused development and scalability.

The validation involved iterative testing to ensure the ontology's semantic consistency and alignment with domain requirements. Clinical and genomic data modeling efforts were evaluated against real-world use cases, providing an accurate representation of neurological conditions.

This deliverable's second focus is the design and implementation of a federated execution framework capable of integrating diverse data sources. This is crucial for enabling analytics across clinical and genomic distributed datasets. The architecture is built on two primary layers:

- Data Federation and Virtualization Layer: This layer consolidates heterogeneous data sources into a unified virtual representation, reducing data redundancy while ensuring scalability and efficiency.
- *Ontology-Based Data Access (OBDA)* Layer: This layer leverages OBDA systems and supports ontology-driven query rewriting, enabling semantically enriched data retrieval.

The query execution engine is designed to optimize query performance across federated datasets; the main aspects we cover in the deliverable are query rewriting based on ontological axioms to enhance semantic precision and query unfolding for mapping definitions to executable SQL commands.

The deliverable outlines two primary use cases:

- ALS Queries: Focused on clinical datasets, these queries validate the framework's ability to extract insights into patient symptoms, progression patterns, and treatment efficacy.
- Genomics Queries: Targeting genomic datasets, these queries demonstrate the framework's capability to uncover genomics aspects and to support Beacon v2 data model.

Both use cases showcase the framework's ability to handle complex queries spanning multiple distributed datasets.

This deliverable establishes a foundation for semantic ontology modeling and federated execution methods, addressing the challenges of data integration and analysis in neurological disease research. The initial architectural design paves the way for advanced data analytics capabilities, supporting the project's long-term goals of enhancing biomedical understanding of ALS and MS.

The rest of the deliverable is organized as follows: Section 2 presents the HERO ontology with a specific focus on HERO-Clinical and HERO-Genomics, covering neurological diseases and genomics, respectively. Section 3 describes the OBDA architecture supporting federated analytics in the Hereditary project and outlines initial federation execution methods. Section 4 introduces two use-cases, one presenting a federated query over ALS distributed data coming from three different medical centers and the second about genomic data distributed across three centers, each employing a different data model ranging from a relational database to Parquet-based data organization. Section 5 reports that the present deliverable verifies Milestone 5. Section 6 draws some final remarks. Finally, Section 7 lists the four annexes of the deliverable.

2 Ontology

We developed the *Hereditary Ontology (HERO)* (https://hereditary.dei.unipd.it/ontology/) by refining the *BrainTeaser Ontology (BTO)* [Faggioli et al., 2024] for modeling phenotypoc and clinical ("pheno-clinical") and genomics data related to ALS and MS. HERO is currently structured to capture both pheno-clinical and genomics aspects of these diseases.

HERO-Clinical (https://hereditary.dei.unipd.it/ontology/phenoclinical/) models the phenoclinical aspects of brain-related diseases. Currently, it focuses on ALS and MS, and in the future, it will be extended to cover also Parkinson's disease, Alzheimer's disease, and various mental disorders. This extension of the BTO incorporates the diverse and heterogeneous clinical characteristics of the medical centers involved in the Hereditary projects.

HERO-Genomics (https://hereditary.dei.unipd.it/ontology/genomics/) supports the representation of genomics data specific to ALS and MS, enabling the documentation of gene mutations linked to these diseases and the storage of gene sequencing variations, such as SNPs.

Future planned phases include further refinements to HERO-Genomics for expanded genomics data capabilities and extending HERO-Clinical to cover additional brain-related diseases and mental disorders. Additionally, HERO will be expanded to include aspects of the gut-brain interplay, focusing on developing a gut microbiome ontology, an area currently with limited foundational work or related models. This approach will ensure that HERO remains scalable, allowing it to capture complex biomedical data across various brain-related conditions and emerging research areas.

Moreover, HERO is the backbone of the Semantic Data Integration, which exploits the OBDA technology to query, aggregate, and join large heterogeneous data in a distributed manner using the SPARQL query language.

2.1 Ontology Design Principles

We have designed HERO exploiting a co-design approach, collaborating with the medical partners and domain experts to embed their knowledge and, at the same time, to validate all the design choices. To this end, we operated iteratively, producing several (intermediate) versions of the ontology and discussing them with our domain experts. In the same spirit, we plan to update the current version of HERO to reflect new requirements and possible enhancements. We anticipate that ongoing feedback from stakeholders will be pivotal in shaping the final structure of the ontology.

The iterative discussion process with the medical partners ensures that these newly defined concepts correctly describe the corresponding real-world concepts and guarantees the semantic quality of the ontology.

2.2 Validation and Testing of the Ontology

HERO has been designed to integrate ALS and MS data coming from different medical centers while avoiding sharing any raw data. HERO models both clinical and genomics data. The clinical data model builds and expands the *BrainTeaser Ontology (BTO)* [Faggioli et al., 2024], which represents retrospective clinical data for MS and ALS patients. On the other hand, the genomics data model

is based on the Beacon v2 entity *genomicVariations* from the Beacon v2.0, an *Application Program Interface (API)* specification for the federated discovery of genomic data established by the *Global Alliance for Genomics and Health (GA4GH)*. For simplicity, we divided HERO into two integrated ontologies: HERO-Clinical (Section 2.5) and HERO-Genomics (Section 2.6).

Both HERO-Genomics and HERO-Clinical have been validated with several online tools to verify their consistency and syntactical validity. The "OOPS! Ontology Pitfall Scanner" [1] [Poveda-Villalón et al., 2014] was utilized to confirm the accuracy of this ontology. Furthermore, we validated the ontology using the following tools: the SSN Validation Tool [2] [Kolozali et al., 2014] and W3C *Resource Description Framework (RDF)* Validation Service [3]. None of the validation tools reported major problems directly linked to HERO-Clinical or HERO-Genomics.

2.3 Identification of the domain requirements

The first step in validating the HERO ontology was identifying the specific requirements of the domain it seeks to model. This included gathering input from domain experts to understand the critical aspects of the knowledge space. More in detail, the medical research teams are from UNITO and UNIPD for ALS and the Hospital of Turin for MS. CNAG and CRG are the reference points for genomic data processing and analyses. In the following extension of HERO, we will interact with UCD and UNIPD to model Parkinson's clinical data and with RUMC to add the gut-brain dimension.

In this first phase, we identified the main domain requirements expressed as natural language sentences by discussing with the clinicians their data needs. The subsequent phase involved aligning the domain requirements of the different research teams by adopting uniform terminology, identifying common physical-world entities within the natural language descriptions of the domain, and identifying the relations between them. The second step involved using actual data provided by the research centers. This allowed us to determine the domain of the various classes, identify shared elements by all research centers, and prepare a draft version of HERO. Based on the clinician and medical experts' feedback, we updated the ontology, adding or removing classes when needed. Upon reaching a consensus on the domain requirements across all research teams involved in the project, we finalized the definition of the domain requirements, which is reported below.

2.4 Implementation principles

In the following, we describe how the HERO complies with the *Open Biological and Biomedical Ontology Foundry (OBO)* [4] and FAIR principles [5] [Wilkinson et al., 2016], favoring its adoption in heterogeneous scenarios.

- The ontology is *open* and publicly available. Its definition and description can be found at http://hereditary.dei.unipd.it/ontology/.

[1] https://oops.linkeddata.es/index.jsp
[2] http://iot.ee.surrey.ac.uk/SSNValidation/
[3] https://www.w3.org/RDF/Validator
[4] https://obofoundry.org/principles/fp-000-summary.html
[5] https://www.go-fair.org/fair-principles/

- The ontology schema is defined according to the OWL 1.2 *Common Format*.
- The proposed ontology relies on unique *URIs/Identifier Spaces* identified by the prefixes `https://w3id.org/hereditary/ontology/phenoclinical/schema/` and `https://w3id.org/hereditary/ontology/genomics/schema/` for HERO-Clinical and HERO-Genomics, respectively.
- A description of the *Versioning* procedure is available as part of the documentation of the HERO on the ontology web page.
- The *Scope* of the HERO is clearly defined: the ontology is meant to integrate clinical and genomics data from different medical centers.
- Following the OBO principles, we associate *Textual Definitions* to each ontology class, also to favor its re-use in other scenarios.
- Before defining a new relation, *Relations* available on the *Relations Ontology (RO)* have been considered. None of the HERO relations presents the same meaning and could have been replaced with one of the RO – nevertheless, this possibility has always been scrutinized.
- A detailed *Documentation* of the ontology is available on its web page.
- For what concerns *Documented Plurality of Users* and *Commitment To Collaboration*, these aspects are intrinsic in developing and using HERO. Indeed, HERO has been developed in the context of the Hereditary Project, which includes partners from multiple countries (from the EU and USA). The co-design approach to devise the Hereditary ontology defines its *collaborative* nature.
- The HERO identifies its *Locus of Authority* into its developers, who are indicated on the web page of the ontology, and in the authors of this paper, that comprises both medical experts in ALS and MS and computer science specialists.
- The HERO follows strict *Naming Conventions* described in Section 2.4.1.
- Finally, the Hereditary consortium is actively working on the *Maintenance* and update of HERO.

2.4.1 Design Choices

We follow some basic principles when defining classes and properties to provide consistency in HERO. These guidelines involve external referencing, annotation properties, and naming conventions.

External Referencing Reusing and referencing external classes is common practice when developing ontologies [Simperl, 2009]. Indeed, reusing entities and properties already defined in other resources enforces collaboration and data consistency. External referencing is managed with annotation properties and using the *Unique Resource Identifier (URI)* of the term in the original thesaurus. Due to its wide adoption and exhaustiveness, our primary choice as the external resource is *National Cancer Institute Thesaurus (NCIT)* [Sioutos et al., 2007], but others are also employed when no information is available in NCIT, e.g., *Systematised NOmenclature of MEDicine Clinical Terms (SNOMED-CT)* or *Anatomical Therapeutic Chemical Classification (ATC)*. The choice of NCIT as a main reference resource stems from its widespread adoption [Bairoch, 2018; Gogate et al., 2021; Sayre et al., 2020; Zhang et al., 2021], granting increased interoperability to HERO.

In HERO, external URIs are used when defining named individuals that refer to abstract concepts. On the contrary, when a new class is inserted in HERO, it is defined within the HERO namespace, and connected references are expressed using annotation properties.

Namespaces The HERO's URIs are divided into two namespaces: the schema namespace `https://w3id.org/hereditary/ontology/schema/` and the resource namespace `https://w3id.org/hereditary/ontology/resource/`. All URIs corresponding to classes, data properties, and object properties belong to the former namespace. At the same time, the latter includes all URIs referring to real-world instances of the entities described in HERO at an ontological level. Notice that, in this sense, the resource namespace is empty until the clinician starts populating it with real-world data. The only instances included in the schema namespace are the named individuals corresponding to SKOS concepts (as defined in Section 2.4.2). The choice of including these elements in the schema namespace stems from the fact that akin to relational modelling controlled dictionaries, these entities do not depend on the data underneath but can be seen as a predefined thesaurus of concepts and are a fundamental part of the reality modelled in the HERO.

Classes Definition and Annotation Properties All components of the HERO have additional information in the form of annotation properties. We defined a list of essential metadata to add when introducing a new class. Firstly, all classes must have a label denoting the name and a comment, which provides a brief explanation – together with its source (e.g., other thesauri, websites, or textbooks). The name and definition are inherited from the thesaurus if the class has an equivalent in NCIT or SNOMEDCT. In this case, the class comprises another annotation property called `rdfs:isDefinedby` expressing the *Internationalized Resource Identifier (IRI)* corresponding to the NCIT or SNOMEDCT term of reference. Most biomedical vocabularies are mapped in the *Unified Medical Language System (UMLS)* [6] with a unique identifier called *Concept Unique Identifier (CUI)* [Bodenreider, 2004]. For each class with a UMLS reference, the annotation property `dcterms:conformsTo` is instantiated with the URL of the corresponding concept. For clarity, Table 1 reports all the required annotation properties and their values for the example class `ho:Electromyography`.

[6] `https://uts.nlm.nih.gov/uts/umls/home`

Table 1: List of required annotation properties. For each class in the Brainteaser Ontology, we define `label`, `comment`, `isDefinedBy` and `conformsTo`. The table reports the values for the example class "Electromyography".

Annotation Property	Value
rdfs:label	Electromyography
rdfs:comment	An assessment of skeletal muscle function and nerve control, obtained by recording and studying the intrinsic electrical properties of the muscles. [Definition Source: NCI]
rdfs:isDefinedBy	http://purl.bioontology.org/ontology/SNOMEDCT/42803009
dcterms:conformsTo	https://uts.nlm.nih.gov/uts/umls/concept/C0013839

Table 2: Example of required information for the `bto:hasDisease` object property. For each object property in HERO, we define a label, a comment describing it, the domain, and the range.

Property	Value
rdfs:label	hasDisease
rdfs:comment	It defines the relationship between a person and a disease he or she suffers from.
Domain	`Person`
Range	"`Disease, Disorder or Finding`"
rdfs:isDefinedBy	http://purl.obolibrary.org/obo/RO_0016002

Naming Conventions All components must have a label and a comment. We use explanatory labels for the object and data properties where the property range and domain are included. In this case, the comment explains the relationship between the two classes. Table 2 reports an example of all required information for object properties. Concerning data properties, the label usually includes the name of the domain class so that its meaning is intuitive. A comment with the attribute description and, when available, the definition source are also included. Table 3 reports an example of the required information for data properties, using the property `bto:deathCause` as an example. All the HERO components can comprise the `note` annotation property for additional remarks or business logic rules.

Alignment to the OMOP Common Data Model The OMOP *Common Data Model (CDM)* is a community effort to standardize the structure of observational data to produce reliable evidence and ease data sharing. [7] The idea behind this data model is to transform data contained in local databases into a common representation, allowing collaborative research and large-scale analytics. HERO follows the standardized clinical data model provided by OMOP CDM to allow interoperability with data modeled following the OMOP CDM. Table 4 provides a list of OMOP components that are modeled in HERO.

[7] https://www.ohdsi.org/data-standardization/

Table 3: Example of required information for the `bto:deathCause` data property. For each data property in HERO, we define the domain, label, and comment on the domain, and range.

Property	Value
rdfs:label	deathCause
rdfs: comment	The circumstance or condition that results in the death of a living being.
Domain	`Patient`
Range	`rdf:PlainLiteral`
dcterms:conformsTo	`https://uts.nlm.nih.gov/uts/umls/concept/C0007465`
rdfs:isDefinedBy	`http://purl.obolibrary.org/obo/NCIT_C81239`

Table 4: HERO classes following the OMOP Common Data Model. For each OMOP element, we report the corresponding HERO component and its type, e.g., class, data property, or object property.

OMOP	HERO Component	Type
Person	Patient	Class
Visit_occurrence	Protocol Event	Class
Death	deathDate; deathCause	Data properties
Condition_occurrence	Condition	Class
Drug_exposure	Therapeutic Treatment	Class
Procedure_occurrence	Intervention or Procedure	Class
Specimen	Biosample	Class

2.4.2 Usage of the Simple Knowledge Organization System (SKOS)

HERO employs external resources to model the diseases affecting a patient, anatomical sites of traumas, and pharmacological substances. Often we are interested in the abstract concept behind the medical term. When an ontology imports external resources, a modeling pattern is *Classism* [Allemang et al., 2020]. Classism is a design pattern where an external hierarchy is modeled as a hierarchy of ontological classes. In this way, data is stored by instantiating multiple named individuals – all with different URIs – for each class, one for each piece of information of interest. In HERO classism is avoided for two important advantages: *i)* it dramatically reduces the number of required URIs, by not defining multiple named individuals; *ii)* it reduces the complexity of the queries. For instance, if we employ classism to model the anatomical location of patients' traumas, the query that returns patients who suffered from a head trauma needs to match three triples: one for patients suffering a trauma, one for traumas located in an anatomical location, and one for keeping only anatomical locations of type "Head". On the other hand, if we avoid classism by defining a unique concept for each anatomical location as a named individual, the above query needs to match only two triples: one for patients suffering a trauma and all traumas located in the head (modeled as the same named individual for all head traumas). Therefore, in the HERO, classification schemes that refer to abstract concepts already defined in other semantic resources are modeled using the

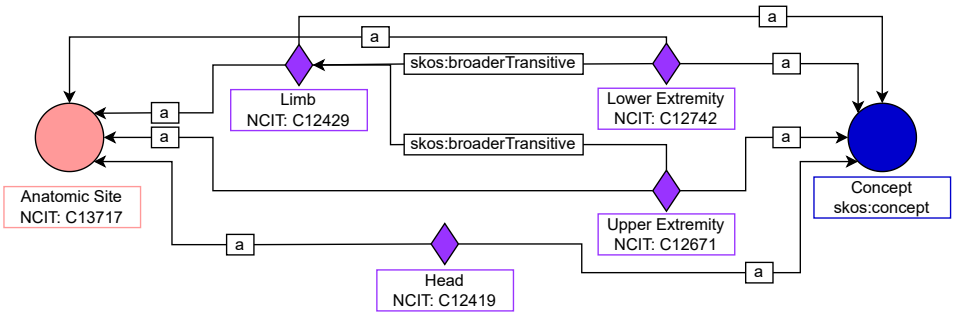

Figure 1: An example of the SKOS data model. Each medical term is modeled as a named individual, and the hierarchical scheme is modeled using SKOS.

SKOS data model. [8] In particular, concepts of hierarchical schemes are modeled as named individuals of type `skos:Concept`, and the relationships among concepts are represented by the object property `skos:broaderTransitive`. Such property is transitive and asserts that one concept is broader in meaning, i.e., more general than another. Differently from the `rdfs:subClassOf` property, `skos:broaderTransitive` links two named individuals rather than two classes. For each class modeled using the SKOS data model, we report the corresponding *Concept Schema* by instantiating an individual belonging to class `skos:ConceptScheme`.

Figure 1 illustrates this design schema considering the class `bto:AnatomicSite` as an example. As reported, each body region is modelled as a named individual of type `skos:Concept` and `bto:AnatomicSite`, and the terms' hierarchical structure is asserted using the object property `skos:broaderTransitive`. For instance, given that `limb` is a more general concept than `upper extremity`, the individual representing the abstract concept `upper extremity` is connected by the property mentioned above to the one for `limb`. As a result, the SKOS data model allows for storing the location information without instantiating one individual for each patient but by simply referring to the individual already instantiated as a concept. Note that this approach prevents us from describing the peculiarities of the specific entity. However, such a design principle is employed on components that do not have this requirement, i.e. for each class referring to a set of abstract terms without any associated data or object property. Table 5 reports all classes modelled using the SKOS standard and the corresponding semantic resource of reference. NCIT has been employed as the main reference thesaurus whenever it contained the required concepts. We resorted to other well-known resources otherwise. Within the HERO namespace, new concepts are defined only if they refer to terms specific to the domain of interest and the corresponding concept is not available in the considered resources.

To provide a practical example, assume the clinician needs to model the fact that a patient suffered from head trauma. We do not need to refer to the head of the specific patient; thus, we do not require a URI. Still, we only need to associate the individual referring to the specific patient's `trauma` with a generic individual representing the entity `head`. Note that we instantiate the specific patient's trauma and assign a URI to it since we are interested in storing specific information

[8] https://www.w3.org/TR/2009/REC-skos-reference-20090818/

Table 5: List of classes in HERO-Clinical modeled using the SKOS data model. For each class, we specify its name and the reference semantic resource we use to define the concepts.

Class	Reference Source
Kinship Type	National Cancer Institute Thesaurus (NCIT)
Occupation	Occupations pillar of the ESCO Classification (ESCO)
Group (social concept)	SNOMED Clinical Terms (SNOMED-CT)
Gene	National Cancer Institute Thesaurus (NCIT)
Disease, Disorder or Findings	National Cancer Institute Thesaurus (NCIT)
Therapeutic Procedure Type	National Cancer Institute Thesaurus (NCIT)
Surgical Procedure Type	SNOMED Clinical Terms (SNOMED-CT)
Anatomic Site	National Cancer Institute Thesaurus (NCIT)
Pharmacologic Substance	Anatomical Therapeutic Chemical (ATC) Classification
Adverse Drug Reaction	Ontology of Adverse Events (OAE)

related to each trauma. Indeed, the patient's trauma has some attributes (such as a date) and might have happened in other places besides the head. Therefore, for all patients affected by head trauma, we create an URI for the specific patient's trauma, and we link it with the object property `bto:anatomicalLocation` the URI of the generic concept `head`. The same applies to all head traumas.

2.5 Clinical Data Modeling

The Clinical Data Modeling of HERO considers the clinical course and the anamnestic history of patients affected by ALS and MS by exploiting an event-based approach. With "event," we refer to anything that can happen to the patient during their clinical history. For example, at a certain point, the patient will experience an onset: we consider the onset an event, assign it additional information (e.g., the date and onset region), and link it to the patient. Subsequent diagnoses, visits, treatments, and so on will also be considered events. Therefore, they will be characterized with additional information and linked to the patient. This method provides a unified model instead of using different resources for each disease. It enhances ontology re-use as it is easier to extend HERO to represent other events or other diseases, not needed, or even unknown, at the time of the definition of the ontology.

2.5.1 Definition of the domain requirements

HERO's design centers on patients and events that can occur during each patient's clinical history. The patient's clinical history consists of several events, e.g., occurred traumas, pregnancies, surgical procedures, or treatments. Patient's clinical course differs among those affected by MS and ALS; however, the event-based approach exploited in HERO enables the joint model of the two diseases. Patients' data requirements are identical for MS and ALS. Therefore, part of HERO is designed to model static variables, e.g., date of birth, biological gender, occupation, and clinical family history.

Several works demonstrate the presence of genetic risk factors for both diseases [Renton et al., 2014; Waubant et al., 2019]. Hence, modeling patients' genomes can enhance the understanding of risk factors for MS and ALS. In addition, pollutant exposure levels, smoking habits, or physical activity can influence the development or progress of both diseases [Olsson et al., 2017; Westeneng et al., 2021].

In the remainder of this section, we provide an overview of the domain requirements, which revolves around clinical data collection for ALS and MS. A practitioner interested in more specific biochemical details, such as the etiology of the diseases or biological pathways, can extend BTO, either using a biologic-oriented ontology or with their classes.

Multiple Sclerosis MS is an autoimmune disorder mainly affecting young adults characterized by the destruction of myelin in the *Central Nervous System (CNS)*. Pathologic findings include multiple sharply demarcated areas of demyelination throughout the white matter of the CNS. In terms of clinical manifestations, visual loss, paresthesias, spasticity, loss of sensation, and bladder dysfunction are recurring symptoms [Adams et al., 1997; Schaeffer et al., 2015]. The MS typical pattern consists of recurring attacks, known as *relapses*, followed by partial recovery. However, acute and chronic progressive forms also occur. More than 2.5 million people live with MS worldwide [Houtchens and Khoury, 2013].

MS diagnosis is made through a combination of clinical history, neurological examination, and *Magnetic Resonance Imagings (MRIs)* [Gelfand, 2014]. In particular, the clinical history of patients affected by MS comprises:

- *Cerebrospinal Fluid (CSF)* analysis [Stangel et al., 2013];
- The recording of *Evoked Potentialss (EPs)*;
- Clinical Evaluation (e.g., weight and *Body Mass Index (BMI)* assessments);
- *Expanded Disability Status Scale (EDSS)* score [Kurtzke, 1983];
- Hematology Tests;
- Cognitive Assessment, e.g., *Symbol Digit Modalities Test (SDMT)* [Nocentini et al., 2006], *Delis-Kaplan Executive Function System (D-KEFS)* [Delis et al., 2001].

In addition, MS can manifest itself in different phases, each involving different courses of treatment:

- Clinically Isolated Syndrome (CIS);
- Radiologically Isolated Syndrome (RIS);
- Primary Progressive MS (PP);
- Secondary Progressive MS (SP);
- Relapsing-Remitting MS (RR).

MS is often characterized by a cyclic progression, with periods of worsening of the disease, called relapses and improvements. Therefore, it is of the utmost importance to record symptoms and body areas (sites) involved during relapses. MS relapses are also linked to pregnancies, with a decreased risk of relapses in correspondence with pregnancies, making them an additional important piece of information to be recorded. MS progression is recorded using the EDSS score, which clinicians usually assess during visits.

Amyotrophic Lateral Sclerosis ALS is a heterogeneous neurodegenerative disease associated with motor dysfunction, such as muscle weakness or dysphagia, and cognitive and behavioural changes [Hardiman et al., 2017]. ALS affects upper and lower motor neurons in the brain stem and spinal cord [Adams et al., 1997; Peters and Brown, 2015]. The disease onset usually occurs after age fifty and becomes fatal within three to six years. Clinical manifestations include, among others, progressive weakness, atrophy, hyperreflexia, and the eventual paralysis of respiratory functions. Pathologic features include the replacement of motor neurons with fibrous astrocytes and the atrophy of anterior spinal nerve roots as well as corticospinal [Adams et al., 1997; Peters and Brown, 2015]. Global estimates indicate that the incidence of ALS ranges between 4.1 and 8.4 per 100,000 persons [Longinetti and Fang, 2019].

The clinical history of patients affected by ALS comprises:

- Anatomical region of the onset (e.g., bulbar or spinal);
- Presence of behavioral or cognitive impairments;
- Pulmonary function tests (e.g., Relative *Forced Vital Capacity (FVC)* measures);
- Lower vs upper motor neuron predominant phenotype;
- *Revised Amyotrophic Lateral Sclerosis Functional Rating Scale (ALSFRS-R)* rating scale [Cedarbaum et al., 1999];
- *Milano-Torino functional staging system (MiToS)* [Chiò et al., 2015];
- *King's clinical staging method (KINGS)* [Roche et al., 2012].
- Genetic Testing;
- Electromyography.

ALS is characterized by fast progression requiring several medical interventions, with a positive impact on the quality of life of the patients and prolonging survival, such as the *Non-Invasive mechanical Ventilation (NIV)* and *Percutaneous Endoscopic Gastrostomy (PEG)*. Being able to predict when a patient will need one of such interventions would allow for preventing medical complications. Thus, we record the occurrence of such events within HERO, to aid the development of predictive models to foresee when the patient will need specific medical interventions.

2.5.2 Ontology Schema

The Clinical Data Modeling is based on the *BrainTeaser Ontology (BTO)* [Faggioli et al., 2024]. We followed the same event-based structure, refining the modeling for the Hereditary needs, which are wider than the ones initially thought for the BTO. We added several properties and classes to include helpful information in our use case. In particular, we added a class to represent the affiliated Socio-Health Local Authority for each patient, one for risk factors, and some properties for MS relapses. The most extensive refinements and extensions are in the Diagnostic Procedure area, where we included several new tests for patients with ALS and MS. Annex 1 reports the schema of the Clinical Data Modeling. The clinical course of ALS and MS patients is organized into eight semantic areas, each denoting a set of ontological classes: “Patient" with “Environmental Data", “Event", “Activity", “Contingencies", and “Intervention or Procedures", divided into “Surgical Procedures", “Diagnostic Procedures", and “Therapeutic Procedures". The complete documentation of HERO-Clinical, including technical details, is available at: `https://hereditary.dei.unipd.it/ontology/phenoclinical/`.

Diagnostic Procedure Diagnostic procedures are exams performed by medical personnel to assess each patient’s diagnosis or disease progression. Thus, diagnostic procedures differ based on MS and ALS patients. Figure 2 reports the diagnostic procedure semantic area modeled in HERO. We highlighted the novel components added to HERO with different colors compared to BTO. We added some properties to classes “CSF Analysis" and “Clinical Evaluation" to include walking status assessments and additional CSF method information. Concerning ALS patients, we added six new classes, two representing neurological exam and blood gas analysis, while the others represent the assessment of muscle response to nerve stimulation. On the other hand, MS novel exams include cognitive assessment tests and three questionnaires, two of which are self-reported by patients. The “Cognitive Assessment" class includes exams to assess lesions distribution, e.g., “*Corpus Callosum Topography Score (CLTR)*" and “*Lesion Topography Score (LTS)*", and tests for cognitive functions, e.g., “*Symbol Digit Modalities Test (SDMT)*" [Nocentini et al., 2006] and “*Paced Auditory Serial Addition Test (PASAT)*" [Ozakbas et al., 2016]. The questionnaires include the “*Multiple Sclerosis Neuropsychological Questionnaire (MSNQ)*" [Benedict et al., 2004], “Fatigue Severity Score" (self-reported), and “*Beck Depression Inventory-II (BDI-II)*" [Beck et al., 1996] (self-reported).

2.6 Genomic Data Modeling

The Genomic Data Modeling of HERO represents genomics data for ALS and MS patients. HERO-Clinical allows to store whether a patient’s genome presents specific genomic variation linked to ALS or MS. However, HERO-Clinical does not provide a detailed representation for genetic testing nor stores gene sequencing variations, e.g., *Single Nucleotide Polymorphisms (SNPs)*. We identified two initiatives by GA4GH towards genomic data modeling: Phenopacket [Ladewig et al., 2023] and Beacon [Rambla et al., 2022; Rueda et al., 2022].

Phenopacket [Ladewig et al., 2023] defines a machine-readable phenotypic description of a patient or a sample in the context of rare, common, complex disease or cancer. Phenopacket is a GA4GH standard for linking phenotypic descriptions, genetic information, diagnoses, and treatments

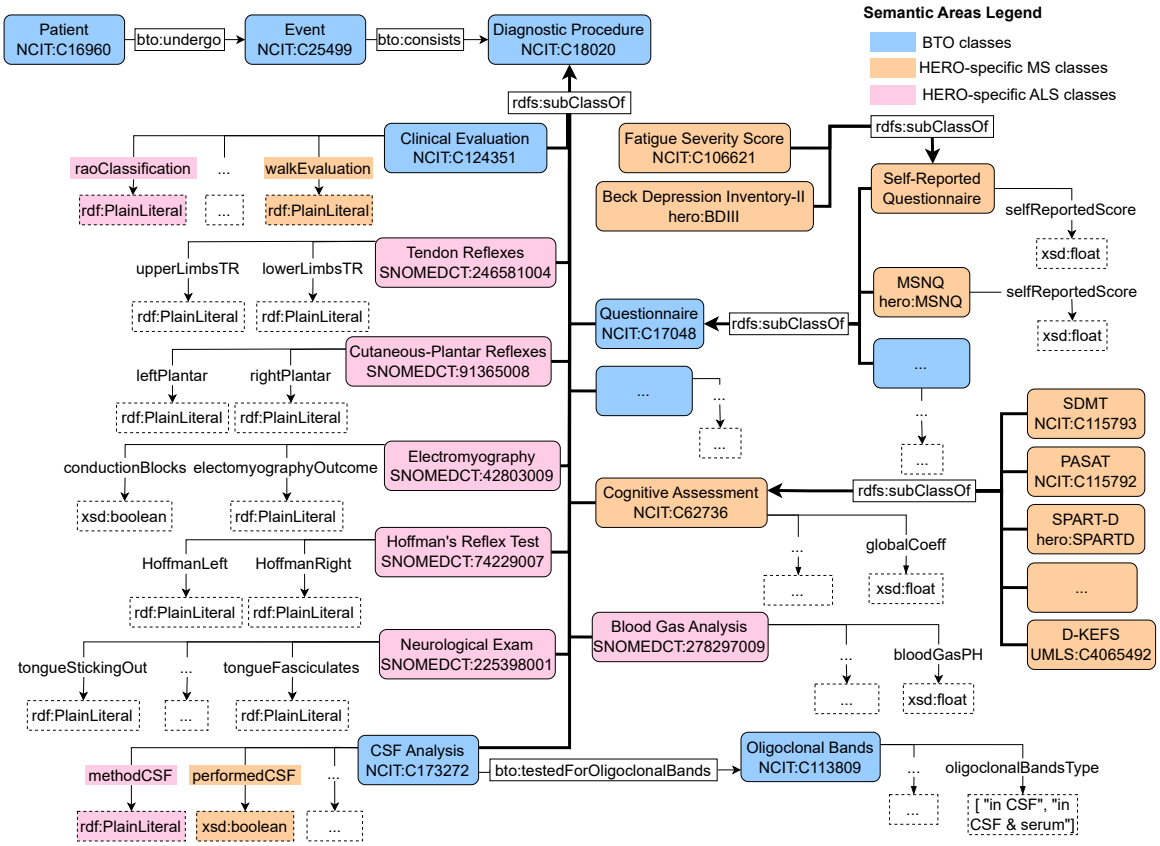

Figure 2: The Diagnostic Procedure area in HERO-Clinical. The components highlighted in orange and pink are novel diagnostic procedures or properties for MS and ALS patients, respectively, included in HERO.

to each patient. The Phenopacket schema is a data model represented using protocol buffers, a language-neutral, platform-neutral mechanism for serializing structured data. The data schema is described in `.proto` files and can be serialized with a hierarchical structure, e.g., JSON or XML. The high-level component of interest for our use-case is the "Genomic Interpretation", where *Variant Call Format (VCF)* files are represented using the class "VCF Record". Phenopacket stores both phenoclinical and genomic data, however, we do not employ it. Indeed, HERO is aligned with OMOP-CDM and provides a more complete representation of longitudinal data.

Beacon v2 [Rambla et al., 2022; Rueda et al., 2022] is an API specification for the federated discovery of both genomic and phenotypic data. The specification consists of two components: the framework and the models. The former defines the format for requests and responses, while the latter defines the response structure for biological data. The component of interest for our use case is the *genomicVariations* entity from the Beacon v2.0 data models, which enables the representation of genetic variation in an individual.

Although Genomic Data Modeling adheres to the Beacon v2.0 data model, many of its classes, originally derived from Phenopacket v2, can also be directly translated into the Phenopacket data model. [9]

2.6.1 Definition of the domain requirements

VCF is a text format widely used for storing genetic variations, such as *Single Nucleotide Variants (SNVs)*, insertions, deletions, and structural variants, together with rich annotations [Danecek et al., 2011]. The format was developed for the 1000 Genome Project [10] and has been adopted in several other projects, e.g., dbSNP. [11] Each VCF file comprises a header and a body. The head provides metadata describing the file's body and keywords that optionally describe the fields used in the body. The body has a tab-separated format with eight mandatory columns and unlimited optional columns that may record additional information about the sample. When optional columns are used, the first of these columns describes the format of the additional columns. The mandatory columns are:

- CHROM, the name of the sequence, typically a chromosome, on which the variation is being called;
- POS, the one-based position of the variation;
- ID, the identifier of the variation, e.g., dbSNP rs identifier;
- REF, the reference base;
- ALT, the alternative allele;
- QUAL, the quality score associated with the inference of the given allele;
- FILTER, the flag indicating which filters have failed or PASS if all filters were successfully passed;
- INFO, the description of the variation. This column is highly variable and its content can vary across VCFs;
- FORMAT, an optional list of fields for describing the samples;
- SAMPLEs, optional values describing the samples.

In the context of the Hereditary project, the genetic information is collected in VCF files comprising SNPs. A SNP is a one-letter place where an individual's genome varies concerning another sequence, usually called "Reference Genome". SNPs are SNVs present in a sufficiently large fraction, e.g., at least 1%, of a specific population [Brookes, 1999].

HERO will serve as a unified means to answer multiple genomic information queries. Hence, HERO is first tested to answer common genomic queries, such as:

[9] https://docs.genomebeacons.org/formats-standards/#phenopackets
[10] https://www.internationalgenome.org/
[11] https://www.ncbi.nlm.nih.gov/snp/

- Sequence Queries for a specified sequence at a given genomic position.
- Range Queries for matching variants at least overlapping with a specified region.
- GeneId Queries for returning variants affecting a gene's coding region.
- Bracket Queries for matching variants falling in a start and end range.
- Genomic Allele Queries for matching variants with the specified allele.
- Amino Acid Change Queries for matching variants with the specified amino acid change.

2.6.2 Ontology Schema

Annex 2 reports the complete schema of the Genomic Data Modeling. The Genomic Data Modeling in HERO mirrors the Beacon Data Model, employing the *genomicVariations* entity as the core class of the ontology. HERO-Genomics allows the storage of genetic variations, e.g., SNVs, insertions, deletions, alongside with their annotations, modeled with the "Case Level Variant" class. HERO-Genomics provides a direct mapping from VCFs files, or Beacon, to RDF format, thanks to classes "VCF Record", "VCF File", and "Variant Quality". HERO-Clinical and HERO-Genomics are connected thanks to the shared classes "Patient" and "Disease, Disorder or Finding". The complete documentation of HERO-Genomics, including technical details, is available at: `https://hereditary.dei.unipd.it/ontology/genomics/`.

Genomic Variations Figure 3 reports the schema of genomic variations and VCF files. HERO-Genomics records systemic variations, e.g., copy number change, molecular variations, and legacy variations such as SNPs.

Systematic Variations comprise *Copy Number Variation (CNV)* and genotypes. CNVs represent specific DNA segments that appear in a variable number of copies among individuals. In HERO-Genomics, CNVs are represented with two classes, namely `Copy Number Change` and `Copy Number Count`. The former describes the change in the copy number of a sequence in a genome. This class records the location of the copy number change with the object property `location` that links the class to the `Location` class. One can also specify the type of variation copy change exploiting a taxonomy modeled using SKOS concepts, where values are from the *Experimental Factor Ontology (EFO)* [Malone et al., 2010], to be specific children of "`Copy Number Assessment`". [12] On the other hand, `Copy Number Count` represents the integer number of copies of the DNA sequence in a genome. This class stores location information with the object property `location` and the integer number of copies of the sequence with data property `variationCopies`. Systematic Variations also comprises the class `Genotype`, which stores the number of molecular variations with data property `molecularVariationCount` and the molecular variations as instances of class `Genotype Member`.

Molecular Variations are variations on a contiguous molecule and can be classified into Haplotype and Allele. The former is a set of non-overlapping Allele members that co-occur on the same

[12] `http://www.ebi.ac.uk/efo/EFO_0030063`

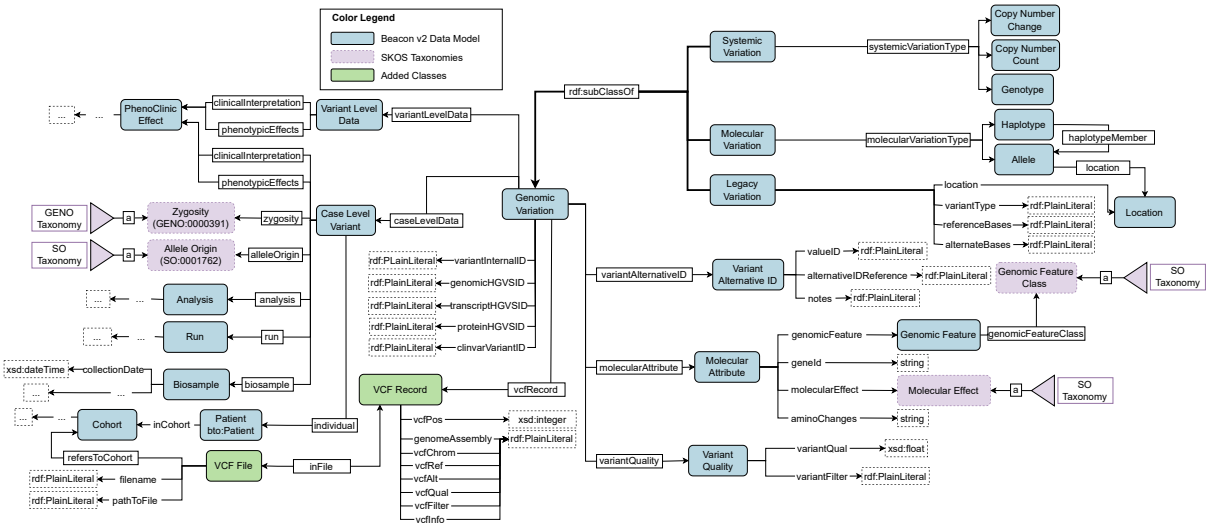

Figure 3: Genomic Variation Modeling in HERO-Genomics. HERO-Genomics records systemic variations, e.g., copy number change, molecular variations, and legacy variations such as SNPs. For each variation, one can store the corresponding VCF row and store additional annotations with classes `Molecular Attribute`, `Case Level Variant`, and `Variant Level Data`. Classes in blue represent components from the Beacon Data Model, those in green were added to satisfy Hereditary's data requirements, and those in purple are classes that represent a SKOS taxonomy.

molecule. Thus, each haplotype is linked to its allele members with object property `haplotypeMember` ranging to class `Allele`. The latter is a variant of the sequence of nucleotides at a particular location. Thus, class `Allele` has object property `location` to store the location of the allele and object property `sequenceState` to record the expression of the sequence state, represented by class `Sequence Expression`. Sequence expressions can be `Literal Sequence Expression`, i.e., an explicit sequence, `Derived Sequence Expression`, i.e., a sequence that is derived from a sequence location, `Repeated Sequence Expression`, or `Composed Sequence Expression`.

Class `Legacy Variation` represents any other genomic variation. For instance, a SNP where base "A" is mutated to "T" can be represented as a Legacy Variation, by storing its location with object property `location`, the variant type "SNP" with data property `variantType`, data property `referenceBase` with value "A" and `alternateBase` with value "T".

Variant Call Format For each variation, one can store the VCF row with class `VCF Record` and link it to the corresponding file with class `VCF File`. Annotations such as molecular effects and amino acid changes are represented by the class `Molecular Attribute`. Class `Case Level Data` stores information about phenotypic effects, clinical interpretations, and the observed zygosity, a SKOS taxonomy from the Gene Ontology [Ashburner et al., 2000; Consortium et al., 2023]. [13] In addition, the class links the genomic variation to the corresponding biosample (`Biosample`), run (`Run`), analysis (`Analysis`), and patient (`Patient` from HERO-Clinical). The location of the variation

[13] https://geneontology.org/

can stored with the class `Location`, which provides a representation both for sequence location, to specify the sequence interval and reference genome, and chromosome location, to specify the cytoband interval of the chromosome where the variation occurs.

3 Federated Execution Methods

This section presents the federated analytics infrastructure of the Hereditary project. The rest of the section is organized as follows:

- Subsection 3.1 provides a theoretical perspective of the structure and the components of an OBDA-based Federated Analytics infrastructure;
- Subsection 3.2 discusses and compares current technological solutions, evaluating those implementing the OBDA paradigm and Data Virtualization systems that best align with our Federated Analytics requirements;
- Subsection 3.3 details the Federated Analytics system architecture implementation.

3.1 Federated Analytics Infrastructure

Federated analytics of clinical and genomic data requires a robust, multi-faceted architecture. Combining low-level technologies for data connectivity and collection from various sources with high-level concepts is essential. This integration creates a unified vocabulary that effectively describes clinical and genomic data and their interrelationships. The result is a comprehensive system that can handle the complexity of federated analytics in these critical fields.

This conceptualization can be formalized into three distinct paradigms:

- **Data Federation** involves those technologies that organize data from multiple and autonomous data sources. To achieve this, Data Federation systems connect to different *Database Management Systems (DBMSs)*, File Systems, and APIs, collecting data into a central node at query time. If necessary, interventions on data such as data anonymization, data obfuscation, and other privacy-preserving techniques may be performed at this stage.
- **Data Virtualization** refers to combining data in different formats and representing different syntaxes from various sources, harmonizing and recombining them under a virtual schema, usually relational. This process does not enrich data by adding semantic layers or additional information. Instead, it allows grouping chunks of data under a common schema.
- **Data Integration** combines data from different sources and provides a unified view to the user. The design of data integration systems involves addressing theoretical challenges such as modeling the application, query processing, handling inconsistent data sources, and reasoning on queries.

Data Integration has evolved from its initial concept, which lacked a semantic focus, to incorporate advancements in semantic research. Semantic Data Integration is an approach that provides users with a unified view of data and establishes relationships between individual data points. This method adds a semantic layer to data, enhancing its interconnectedness and meaning. These advancements have subsequently resulted in the development of tools and mapping languages designed explicitly for semantic data integration, laying the groundwork for a paradigm known as

OBDA [Calvanese et al., 2007]. The core idea behind the OBDA paradigm involves establishing a global schema, represented by an ontology, which acts as a blueprint for how data from diverse sources is viewed and accessed. This ontology defines not just the entities and their attributes but also the relationships and constraints between these entities. This integration between the ontology and the underlying data model is then achieved by defining mappings that establish semantic relationships among data entities.

The integration process in the OBDA approach is complex, as it aims to replicate the functionalities of graph (RDF-based) databases. Specifically, it allows for querying and reasoning on data by leveraging the open-world assumption, enabled through the Web infrastructure and RDF's triple-based structure. In this approach, the ontology serves as a model or global schema, defining entities and relationships across various data sources without centralizing them as triples in a single storage. As a result, direct RDF-based querying and reasoning capabilities cannot be applied straightforwardly to these dispersed data sources. To overcome this issue, a query over the ontology must go through diverse preprocessing steps before execution. Figure 4 illustrates by means of a block diagram the query flow before being submitted and executed on the underlying data sources. The OBDA approach begins with an ontology that describes the domain of interest and a virtual global relational schema mapped to multiple heterogeneous data sources. This approach relies on mappings between the ontology and the global schema. When a query is made, typically in SPARQL, it is first rewritten to include implicit results, which may involve inferred triples. Next, the process of query unfolding converts this rewritten query into corresponding *Structured Query Language (SQL)* operations, transforming the entire data flow graph into a single SQL statement. Mappings are applied to translate SPARQL subqueries into SQL subqueries, which, when combined, can be executed over the global virtual schema.

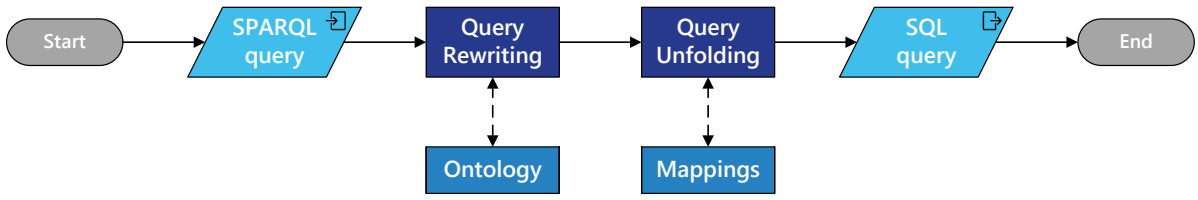

Figure 4: The flow of a query through OBDA

More formally, an OBDA implementation must realize this chain of procedures:

1. the input query q is rewritten into q' over the virtual ABox (namely, a virtual ABox $\mathcal{V}$ is a tuple $\langle \mathcal{D}, \mathcal{M} \rangle$, where $\mathcal{D}$ is a DB and $\mathcal{M}$ a set of mappings [Rodriguez and Calvanese, 2011]) A' [Makris et al., 2010];

2. the rewritten query q' is unfolded over the previously defined mapping definitions into the SQL query q^*;

3. the SQL query q^* is evaluated over the relational data source (i.e., the Virtual Schema);

It is important to note that the OBDA paradigm is traditionally designed to operate over a single relational database layer. When this approach is extended to a data federation context – where multiple data sources with diverse data models (e.g., relational, CSV, document-based) are involved – it falls under the recently defined *Ontology-Based Data Federation (OBDF)* paradigm [Gu et al., 2024].

The OBDA paradigm is valuable for adopting a broader vocabulary to represent clinical and genomics data effectively. However, it limits federated analytics tasks, especially within the Hereditary project, where data is distributed across disparate sources and cannot be centralized or duplicated due to strict regulatory constraints. To address this, we adopt the OBDF paradigm, which combines the OBDA approach with Data Federation. This approach connects various data sources using a Data Virtualization system, allowing heterogeneous data to be harmonized under a virtual unified relational schema. This schema is then mapped to a common ontology, enabling reasoning and executing global, graph-based queries.

Figure 5 illustrates the structure of the OBDF framework within the Hereditary Federated Analytics architecture. This framework emphasizes the connection between the SPARQL endpoint and HERO, the core model supporting all query operations. HERO provides a comprehensive view of all project data, allowing users to explore interconnected data paths that would otherwise remain isolated – such as linking an ALS patient's clinical data with common genomic variants associated with ALS. As shown, an OBDA component functions as an intermediary, bridging the SPARQL endpoint over HERO with the underlying virtualization layer and the global relational schema, which unifies the heterogeneous local data sources.

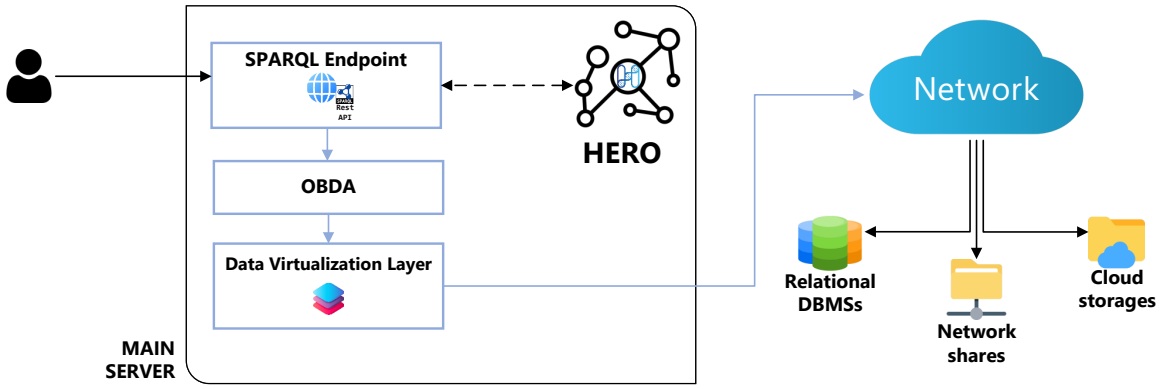

Figure 5: A conceptual representation of the OBDF framework applied to the Hereditary Federated Analytics context.

The Hereditary data federation architecture offers several advantages. It enables the use of data virtualization and federation technologies, making the architecture scalable in terms of connectable sources and the variety of supported data types, as users can set up and integrate custom drivers. Additionally, the architecture maintains live connections to the underlying data sources, ensuring that information remains current, provided that no caching mechanisms are employed between the sources and the architecture. Furthermore, implementing privacy-preserving methods and com-

ponents at the source level enhances customization options and strengthens the autonomy of the central components within the federated architecture.

3.2 Technological choices

This section investigates comparative amongst existing OBDA systems and Data Virtualization and Federation systems. This analysis aims to understand the strengths and weaknesses of existing solutions from the academic community and the industry to understand which of them better fits the Federated Analytics framework requirements of Hereditary.

3.2.1 OBDA Systems

A first analysis in compares different OBDA solutions and Table 6 summarizes its content [Kharlamov et al., 2014].

SYSTEM	REASONING CAPABILITIES
Optique (Siemens)	YES
Ontop	YES
Mastro	YES
morph-RDB	YES
D2RQ	NO
OntoQF	NO
Virtuoso	NO
Spyder	NO
Ultrawrap	NO

Table 6: Comparative among OBDA Systems. The comparison highlights the reasoning capabilities of different systems.

Given that ontology reasoning via query rewriting is one of the most important advantages in using the OBDA approach, and considering that the Optique system is an enterprise, on-premises and closed source solution, more in-depth comparisons analyzed in details Ontop and Mastro [Namici, 2018; Namici and De Giacomo, 2018].

To the best of our knowledge, Mastro was the first OBDA tool, developed at La Sapienza University of Rome, Italy. It supports a subset of SPARQL queries and integrates a custom XML-based mapping syntax [Calvanese et al., 2011]. Mastro initially was not supporting the *RDB to RDF Mapping Language (R2RML)*, which was a strong limitation in standard compliance. The tool used a complex system of view predicate mappings, which may affect its adaptability to standardized environments. Although recent updates have integrated R2RML support, the system's architecture still lacks certain optimizations due to its design constraints. For instance, Mastro cannot perform advanced semantic query optimizations because it does not support detailed data constraints in its mappings. This limitation could lead to less efficient query processing and increased execution times, especially with complex queries involving multiple joins or extensive data operations.

Moreover, Mastro is less efficient in handling datatype operations and IRI constructions within SQL, leading to potential slowdowns in query processing.

Ontop [Calvanese et al., 2015] is an open-source and well-maintained OBDA framework developed at the Free University of Bolzano, Italy. It is compliant with *World Wide Web Consortium (W3C)* standards, including R2RML for ontology mapping, *Web Ontology Language (OWL)* for ontology representation, and SPARQL for querying. Ontop is designed for high-performance query answering over virtualized RDF graphs. Above this, it provides an efficient query-answering system that leverages R2RML mappings and supports comprehensive optimizations. The system uses advanced query rewriting techniques based on a tree witness algorithm, effectively reducing query complexity and execution time. The strengths of Ontop are particularly evident in its handling of complex SPARQL queries, where it efficiently translates these into SQL, utilizing $\mathcal{T}$-mappings and database integrity constraints for optimization. Regardless of its adherence to standards, Ontop also comprises a custom mapping language (the Ontop Native mapping language). The comparison between the two systems has been performed in a W3C-compliant setting in two different benchmarking scenarios: the *Norwegian Petroleum Directorate (NPD)* benchmark and the *Automobile Club d'Italia (ACI)* [14] application developed at the Sapienza University of Rome [Lanti et al., 2015; Namici and De Giacomo, 2018]. The observed benchmarking results are often conflicting. On average, Ontop shows faster response time, although there are two main cases where it performs way worse: when the original query comes with the DISTINCT clause (no matter the size of the relational dataset), and when any aggregation function is used alongside the GROUP BY clause (only if the dataset size is very large, otherwise it performs better than Mastro). In addition, Ontop performs better in scenarios where many mappings are involved for unfolding a certain query, thanks to its capability of compiling the mappings definition and ontology to form the so-called $\mathcal{T}$-mappings. This means that a choice of which system fits better depends both on how constraining the timings in the query processing phase are and on how many mappings have to be unfolded on average; that strictly depends on the heterogeneity of the underlying relational source. We chose Ontop for the Hereditary Federated Analytics architecture due to its native support for two high-level mapping languages, which provide greater flexibility for optimization without requiring low-level *Description Logic (DL)* languages. Additionally, Ontop is open-source and well-maintained by the community. It is integrated into GraphDB, which offers robust APIs and includes a wide array of APIs that facilitate its adoption within a broader framework.

3.2.2 Federation and Virtualization Systems

With respect to OBDA solutions, Data Federation and Virtualization Systems offer various functionalities depending on their implementation, making them difficult to compare and evaluate globally. Therefore, this analysis will focus on four main aspects, considering their crucial role in the broader Federated Analytics framework: the software support and documentation, being a *Free And Open Source Software (FOSS)*, scalability and logging capabilities.

Our comparison has been performed on three Federation and Virtualization systems natively supported by Ontop: Denodo, Teiid, and Dremio. Table 7 summarizes the comparison analysis.

[14] https://www.informatica.aci.it/

	SUPPORT & DOCUMENTATION	FREE AND OPEN SOURCE	SCALABILITY	ROBUST LOGGING CAPABILITIES
Denodo	X		X	X
Teiid		X		
Dremio	X	X	X	X

Table 7: Comparative of Data Federation Systems . The comparison focuses on four aspects crucial for our Federated Analytics framework. These aspects were evaluated by manual inspection of the software and its documentation when available.

Denodo is an enterprise virtualization platform that also serves as a data federation system, integrating data from diverse sources to provide a unified view without requiring physical data deduplication. It allows for real-time access to structured and unstructured data from various sources, including relational, column, and No-SQL databases, web API, and CSV files. Denodo provides robust security features, including hashing, encrypting functions, and user privileges, ensuring that sensitive data is protected according to compliance standards. Moreover, it utilizes advanced query optimization techniques, such as caching and query rewriting, to improve its performance. These optimizations ensure efficient data retrieval, reducing latency and improving the overall speed of data access across the federated sources. Although being highly scalable and capable of accommodating new data sources and coming with a custom source wrapper template for unsupported data sources, its enterprise closed-source nature, which allows for a maximum of five connections unless a premium plan is signed, hampers its adoption in a broader context where modifications to its native implementation might be needed.

The University of Padua is the first institution to sign the Denodo academic program, which enables the use of the Denodo premium version for teaching purposes at no cost. This agreement also permits using the Denodo premium version within the Hereditary project for its entire duration. This presents a significant advantage for Hereditary, as it allows us to implement the federated architecture in a plug-and-play manner without being limited to a single virtualization system. Furthermore, it enables us to conduct comparative experiments across various virtualization systems.

Teiid is an open-source Java component developed by Red Hat that provides integrated access to multiple data sources through a single uniform API. Rather than a DBMS, Teiid is more like a query engine for integrating data from multiple sources, accessible both through API and *Java Database Connectvity (JDBC)* or *Open Database Connectivity (ODBC)* interfaces. A Teiid implementation exists as an Eclipse plugin and a deployable package on web containers such as Wildfly and OpenShift. One of the main issues with Teiid is its poor documentation when it is not deployed alongside enterprise solutions (like OpenShift). Moreover, no major releases have been published for four years, and many open issues on the Teiid official GitHub repository [15] are no longer being addressed.

Dremio is a virtualization platform that also serves as a data federation system. Although it is developed within an enterprise context, its standard version, comprehensive of most of the Dremio

[15] https://github.com/teiid/teiid/issues

features, is FOSS under the Apache 2.0 license, combining typical enterprise software robustness with a strong community active on continuous maintenance. Even if it is possible to use Dremio as a standalone instance (e.g., on a server), it is by design a distributed system and thus it can run on clusters. In the case of standalone configurations, it is possible to install it using packet managers in Linux server environments, but the easiest way is to use the official Docker image. Dremio makes use of Apache Arrow to enhance processing speeds and reduce latency. Moreover, it optimizes query performance through its advanced query planner and execution engine, which can push down operations to the data source level, minimizing data movement and speeding up response times. It provides comprehensive security features that include encryption, access controls, and data masking to ensure privacy. It also maintains detailed logs of all queries, which is crucial for compliance purposes in many fields, such as the clinical domain, where patient medical data is managed alongside their personal information. As in Denodo, but under an open-source perspective, it is possible to build custom connectors for unsupported data sources and Dremio. This is doable through *Advanced Relational Pushdown (ARP)* connectors: a public repository provides a customizable Maven template for each data source for which a JDBC driver is available. In conclusion, Dremio offers an open-source virtualization system typically robust in enterprise software. It is highly scalable in terms of computation distribution over clusters and the types of supported sources, and it has a comprehensive logging system that is well-suited for tracking data flow.

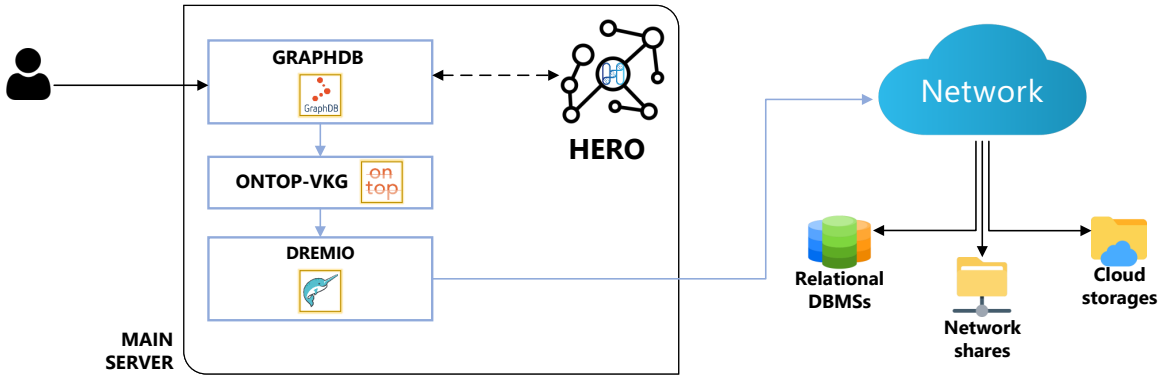

Figure 6: Federated Analytics Architecture design draft considering the technological background analysis.

Considering the technological choices from this background analysis, the architecture design draft results are as represented in Figure 6: it adopts Ontop as its semantic data integration layer and Dremio as its data virtualization and federation layer. Dremio has been employed in our framework as it has been evaluated positively under all four main aspects: it is a free and open-source and scalable virtualization platform, well documented and supported by the company and an active community. It also has strong logging capabilities, crucial in our use case scenarios.

The SPARQL endpoint role is played by GraphDB, which offers a more comprehensive and user-friendly platform with respect to the Ontop web endpoint. The Hereditary federated architecture utilizes the professional edition of GraphDB, which enables parallel optimizations and concurrent queries over the ontology. This capability is made possible through Ontotext's participation in the

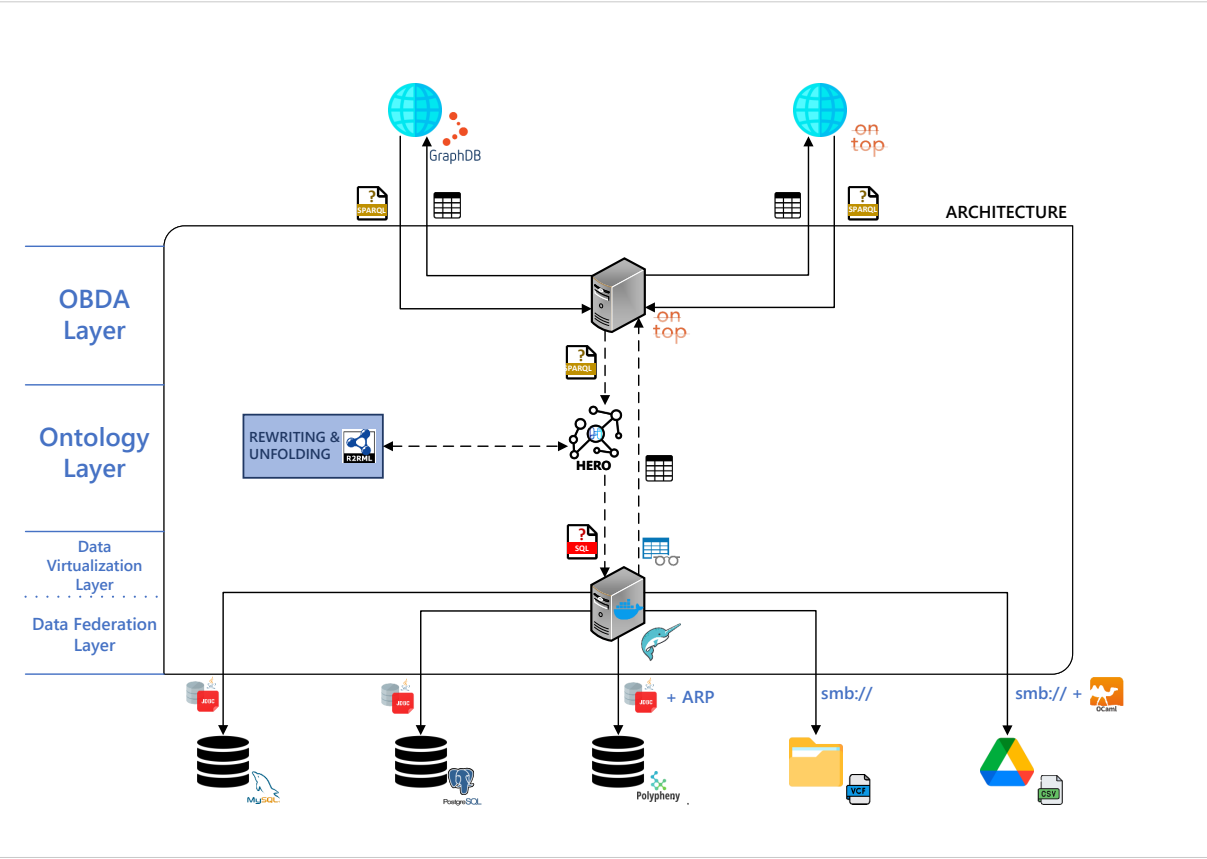

Figure 7: The Federated Analytics architecture

project consortium, which provides several professional licenses to the consortium members.

This selection of tools ensures a robust and scalable architecture that meets the diverse needs of data consumers across various applications. The next step in our research involves an empirical evaluation of the system's performance under different load conditions and query complexities.

3.3 Implementation

The federated analytics system architecture, represented in Figure 7, implements the OBDF framework. In this section we will analyze implementation details layer by layer and using a bottom-up approach, provide an overview of how different data sources can be plugged into the system, and discuss the implementation details of the Hereditary architecture.

3.3.1 Data Federation and Virtualization Layer

At the lowest level of the architecture, we have a Dremio. This virtual data lakehouse system encompasses both the federation and virtualization functionalities, as it can connect to multiple diverse

data sources seamlessly and allows the definition of non-materialized virtual views of the federated data. Within Dremio, it is possible to create and manage virtual views. For example, in Figure 8, we have a view called "virtual_view," stored in the data lakehouse, combining information from RS1 and RS2 in a broader schema. The execution of the view triggers individual and independent queries on the two sources, and the virtual query engine combines result sets on the fly, joining the sets based on the specified condition. For instance, a user might combine genomics data in CSV files stored on a *Network Attached Storage (NAS)* folder with patient data from a MySQL database to perform concurrent analytics by joining relations. Dremio allows you to materialize the view to avoid computing costs. Materialized result sets can be cached, so there is a tradeoff between computing costs and streaming capabilities. For example, in Listing 1, we have a view definition in the data virtualization layer. In this case, the schemata 'LISBON', 'TURIN', and 'MADRID' abstracts come respectively from a PostgreSQL instance, a MySQL instance, and a Polypheny instance. Every time that this view is used, three different queries are submitted, one each on the three different sources, and after retrieving the results, the Dremio query engine builds on the fly the result set, and then elaborates the response based on the incoming query specifications.

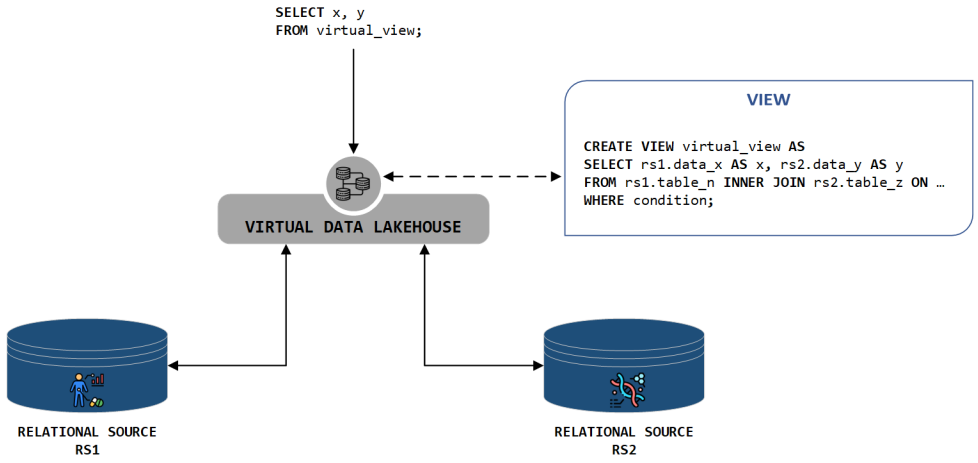

Figure 8: A logical representation of how virtual views abstracts underlying data sources.

```
# Patient Static Data
CREATE VIEW clinical_data.STATIC_VARS AS
SELECT * FROM LISBON.public.lisbon_static_vars UNION SELECT * FROM
MADRID.madrid_alsfrs.Madrid_static_vars UNION SELECT * FROM TURIN.
alsfrs_turin.Turin_static_vars;
```

Listing 1: Virtual views over clinical data

Dremio offers many APIs [16] that allow developers to interact with all the system's functionalities. It also embeds a web *Graphical User Interface (GUI)* that simplifies the usage process, especially when setting up experimental environments. Dremio also provides a JDBC driver to allow incoming

[16] https://docs.dremio.com/24.3.x/reference/api/

connections, thus making it possible to query defined virtual views from other applications. This is crucial for our data pipeline within our federated analytics architecture, as this is the mechanism that Ontop uses to query and retrieve data from Dremio. Relations from different sources are abstracted to hide SQL dialect details and harmonize them: this is made possible by ARP connectors, namely connection wrappers that implement interfaces, to provide a common entry point for each source. Dremio comes with some predefined ARP connectors, wrapping some of the most popular DBMSs. Still, developers can create and inject custom connectors into a Dremio instance, to support new sources. A template provided on GitHub [17] can be cloned, customized and packed in a *Java Archive (JAR)* file.

As mentioned before, in our experimental setup, we distributed portions of a dataset about clinical and genomics data to various data sources. The decision on which sources had to be federated in our experiment was driven both by real-world scenarios, where these systems are often of interest, and also in a way that led us to experiment with nontrivial use cases. In particular, the data sources involved in our experiment were:

- MySQL;
- PostgreSQL;
- Polypheny;
- a NAS folder;
- Google Drive.

MySQL and PostgreSQL are two of the most popular relational FOSS DBMSs. They are both extensively employed across various applications, from small projects to critical enterprise environments. Thus, we chose to federate these two DBMSs considering they can potentially be adopted in application software in clinical environments and in advanced biomedical research contexts. The interfacing between these DBMSs and Dremio is performed through their JDBC driver and the ARP connectors that are provided natively with Dremio.

Polypheny [18] is a FOSS polystore system designed to support diverse data models, including relational, document, and graph data [Vogt et al., 2018]. Developed at first as only a research project from the University of Basel, it is now available as an enterprise solution. It has been engineered to handle mixed workloads and various query languages, making it a versatile platform suitable for dynamic data environments. In the context of the Hereditary project, Polypheny can serve as a low-level federation within local environments, such as hospitals or research centers. This makes it particularly crucial to be able to federate it under Dremio. Unlike the straightforward integrations with MySQL and PostgreSQL, federating Polypheny required a more sophisticated approach; in fact, no native support through ARP is provided for Polypheny, and thus a custom ARP connector needed to be developed. Dremio's ARP framework offers a powerful mechanism to extend Dremio's capability to interact with various data sources by interfacing Dremio's internal query representations

[17] https://github.com/dremio-hub/dremio-sqllite-connector
[18] https://polypheny.com/

into the native query language of the target data source and translating the SQL dialects. The ARP connector template consists of a Java Maven project that must be compiled, packed within a JAR file, and injected, together with the target source JDBC driver, into the running Dremio instance. Two files need to be customized: the storage plugin configuration Java class, where connection parameters to the target source need to be specified by adding *User Interface (UI)* form input fields, and the ARP YAML configuration file, where translations between SQL keywords, functions and data types need to be defined, as well as other miscellaneous information like the plugin name, shown in the Dremio UI. We exploited this framework, and we developed a custom ARP connector specifically for Polypheny. The connector source code is publicly available in a GitHub repository [19], and all the implementation details are available in Annex 3.

Dremio also offers native support to a variety of relational file types. These files can be ingested in the Dremio instance itself, but, in this case, data is duplicated from the original files, and it is impossible to exploit any streaming capability. An alternative is to attach and browse NAS sources. Dremio integrates an ARP connector to local folders, reading supported file formats. More details about the connection procedures are described in Annex 4.

Google Drive is a widely used cloud file storage service. With its office suite, users can create, manage and store documents, spreadsheets and presentations and collaborate with others. These features make this system suitable for research environments where collaboration on shared spreadsheets is frequent. Thus, accessing and federate shared spreadsheets from Google Drive is crucial. To accomplish this task, we mapped Google Drive folders to a local host folder to reuse all the concepts valid for NAS folders. All the technicalities are reported in Annex 4.

3.3.2 OBDA Layer

The HERO, in the Federated Analytics context, acts as a vocabulary for designing semantic queries over data that comes in a relational fashion. This translates into the need to encapsulate an OBDA system within the Federated Analytics architecture. As outlined in previous sections, the leading FOSS software component accomplishing to this task is Ontop. Ontop is composed of a suite of tools:

- Ontop Mapping (plugin for Protégé [20]);
- Ontop SPARQL Endpoint (plugin for Protégé);
- Ontop SPARQL Endpoint (standalone);
- GraphDB Ontop Endpoint.

To define mappings between the Ontology and the Virtual Views exposed by Dremio, the Ontop Mapping plugin for Protégé has been used. SPARQL queries have been initially tested through the SPARQL Endpoint plugin for Protégé and then executed without limiting the fetching process

[19] https://github.com/mircocazzaro/dremio-polypheny-arp

[20] Protégé is a free, open-source platform with a suite of tools to design ontologies. https://protege.stanford.edu/

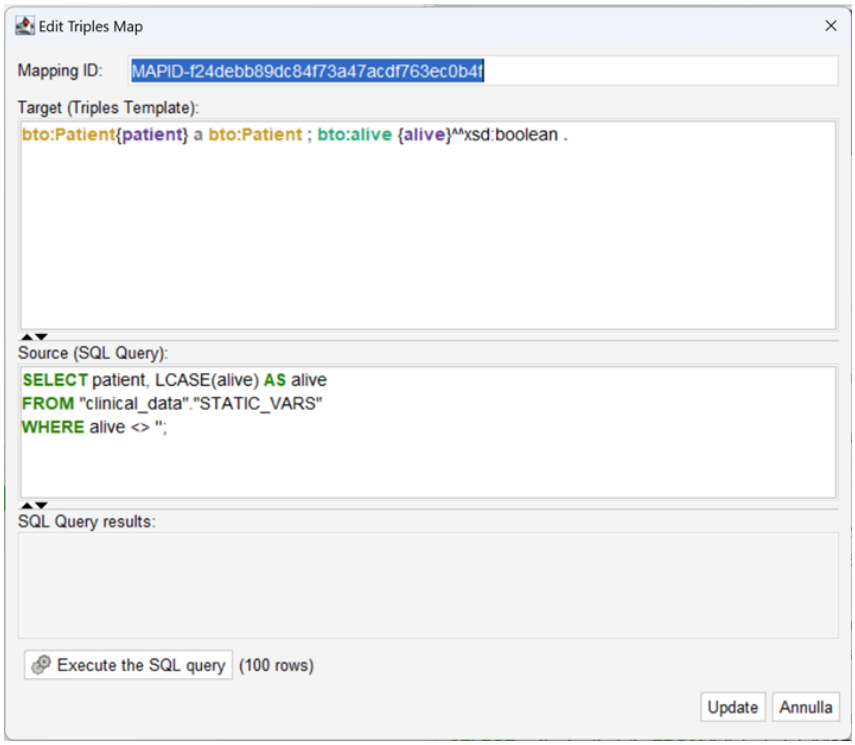

Figure 9: Mapping form editor of the Ontop Mapping plugin for Protégé.

using the Endpoint in GraphDB. For the mapping definition process, we adopted the Ontop Native mapping language. This is because the Ontop Mapping plugin for Protégé provides an intuitive visual mapping definition tool that automatically translates them. As shown in Figure 9, mappings are defined by matching a portion of the graph (using Turtle notation) to an SQL query over, in this case, the virtual schema exposed by Dremio; fields selected in the query are matched one by one with labels specified in curly brackets in the triples previously defined. In particular, in this example, the SQL query retrieves all patient identifiers whose "alive" field is not equal to the empty string (so as not to map uncertain information). The mapping of the patient identifier is performed on an individual named "Patient" concatenated with its ID, and it is stated to be part of the Patient class. The "alive" attribute is mapped to the corresponding boolean data property.

The resulting mapping definition of the prompt shown in Figure 9 is represented in Listing 2. For each mapping entry, a default mapping ID is assigned with a unique key; the target fields contain the mapped triples, while the source field contains the SQL query.

ID	PATIENT	ALIVE
1	0x012	TRUE
2	0xABC	FALSE
3	0xDEF	TRUE

Table 8: Simplified representation of three entries corresponding to three patients coming from the virtual relation "clinical_data"."STATIC_VARS".

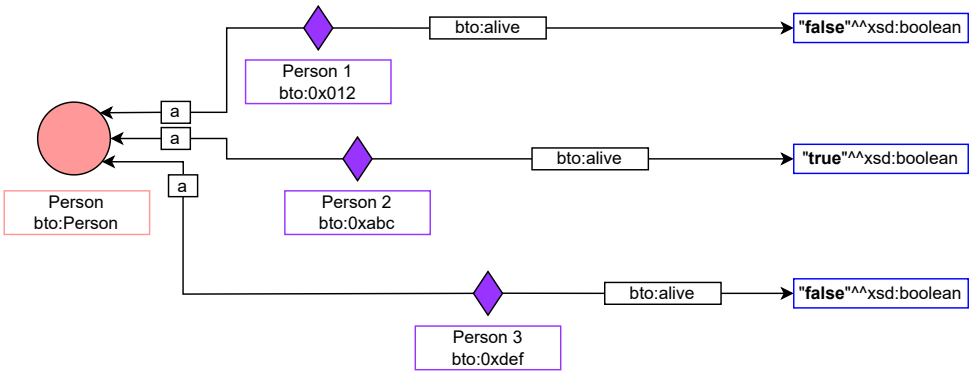

Figure 10: Triples returned from Ontop after mapping in Code 2 is interpreted.

```
mappingId MAPID-f24debb89dc84f73a47acdf763ec0b4f
target    bto:Patient{patient} bto:alive {alive}^^xsd:boolean .
source    SELECT patient, LCASE(alive) AS alive
      FROM "clinical_data"."STATIC_VARS"
      WHERE alive <> '';
```

Listing 2: Mappings definition between the virtual relational schema exposed by Dremio and the BRAINTEASER Ontology

Supposing that the virtual relation “clinical_data”.”STATIC_VARS” is composed of the entries represented in Table 8, where we have three patient, each of which with its own alive status, then the translation in triples occurs as represented in Figure 10.

The Ontop Mapping plugin for Protégé can connect with the underlying relational source (i.e., Dremio), to test the SQL queries and assess the validity of the defined mappings. It also prevents mismatches with the ontology and typos within the Turtle syntax at definition time. A dedicated tab that allows to specify Ontop connection properties is available. In our experiments, no specific option needs to be set up. SPARQL queries over the ontology have been tested at first with the Ontop SPARQL plugin, and subsequently executed on an Ontop endpoint (on a GraphDB instance), fed with HERO and the exported mappings.

4 Use Cases

4.1 Federated analytics: ALS Example Queries

The use case selected for this analysis involves a SPARQL query, reported in Listing 3, to access and analyze the *Amyotrophic Lateral Sclerosis Functional Rating Scale (ALSFRS)* data, which is crucial for assessing the progression of ALS symptoms over time. This scenario is meticulously chosen to demonstrate the robust capability of the proposed architecture to manage and execute queries that necessitate access to multiple physical locations.

```
PREFIX bto: <https://w3id.org/brainteaser/ontology/schema/>

SELECT ?patient ?date ?tot ?bulbar ?motor ?respiratory ?q1 ?q2 ?q3 ?q4 ?q5 ?q6 ?q7 ?q8 ?q9 ?q10 ?q11 ?q12 WHERE {
    ?p bto:undergo ?e.
    ?e bto:consists ?alsfrs.
    ?alsfrs a bto:ALSFRS;
           bto:procedureStart ?date;
           bto:revisedALSFRS ?tot;
           bto:bulbarSubscore ?bulbar;
           bto:motorSubscore ?motor;
           bto:respiratorySubscore ?respiratory;
           bto:alsfrs1 ?q1;
           bto:alsfrs2 ?q2;
           bto:alsfrs3 ?q3;
           bto:alsfrs4 ?q4;
           bto:alsfrs5 ?q5;
           bto:alsfrs6 ?q6;
           bto:alsfrs7 ?q7;
           bto:alsfrs8 ?q8;
           bto:alsfrs9 ?q9;
           bto:alsfrs10 ?q10;
           bto:alsfrs11 ?q11;
           bto:alsfrs12 ?q12.
    BIND(SUBSTR( (STR(?p)), 48) AS ?patient)
}
ORDER BY ?patient ?date
```

Listing 3: SPARQL query on ALSFRS visits performed by patients over the *BRinging Artificial INTelligencE home for a better cAre of amyotrophic lateral sclerosis and multiple SclERosis (BRAINTEASER)* ontology vocabulary. The query retrieves patients' exam scores and their answers to twelve different questions.

The query is based on HERO by focusing on its BRAINTEASER ontology component, indicated by the prefix bto, to query ALSFRS data. It is designed to retrieve a comprehensive set of variables related to the ALSFRS assessments, including total scores and subscores for bulbar, motor, and respiratory functions and individual question scores from the ALSFRS-R. This query fetches detailed patient assessment data and orders it chronologically for each patient. OPTIONAL clause is not employed considering that for this use case we fed the architecture with complete relations.

4.1.1 Experimental Setup

Figure 11 shows the main query processing steps within the Federated Analytics architecture, outlining a flow from the initial query submission to the system to the physical data sources: the pairs "block-arrow" in the figure corresponds to the pairs "preprocessing algorithm-output subquery". The

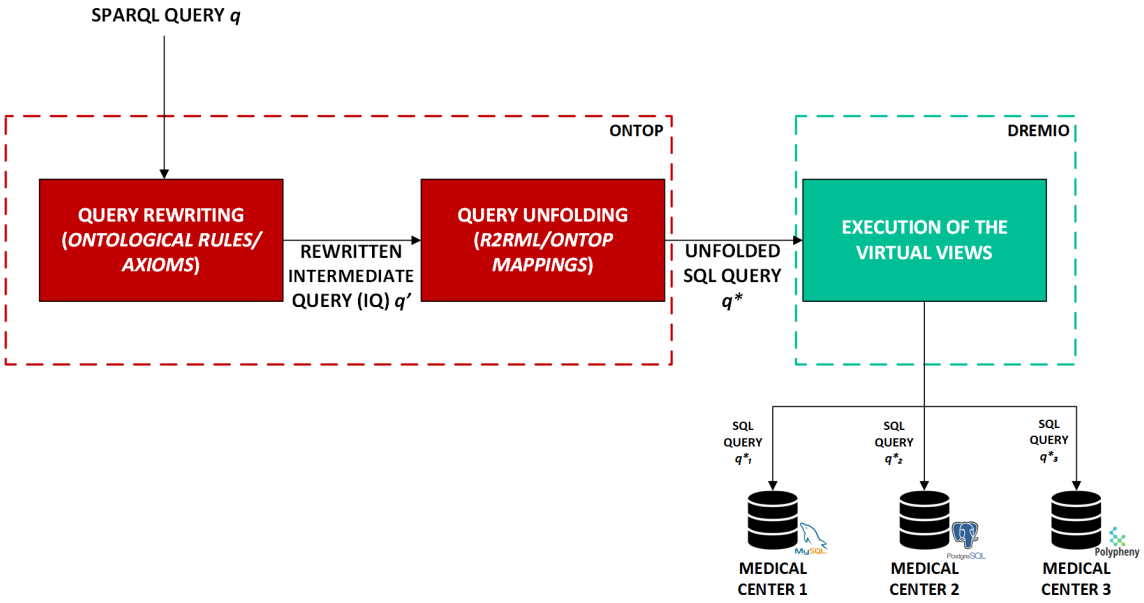

Figure 11: Data Flow Block Diagram of a Query Through the Federated Analytics architecture

three main steps are the following:

1. Query Rewriting: input SPARQL query q is rewritten over ontological axioms to include non-explicit results explicitly; during this phase, input query q is transformed into the SPARQL query q';

2. Query Unfolding: rewritten SPARQL query q' is unfolded to SQL using R2RML or Ontop Native mappings, to query the underlying SQL source; during this phase, rewritten query q' is transformed into the SQL query $q*$;

3. Federated Query Execution: unfolded SQL query is split in $q_1^*, \ldots, q_n^*$ and submitted to the physical underlying data sources, each written in the corresponding SQL source dialect.

We use the Ontop framework in its standalone version in this setup, setting the root level to DEBUG in the `log/logback.xml` configuration file. This allows for a granular inspection of how queries are translated from SPARQL to SQL, focusing on the use of ontological rules and mappings and allowing the inspection of both q' and $q*$. We also configured the underlying DBMSs to log the incoming queries to inspect how the federation system splits and dispatches $q*$ chunks.

4.1.2 Query Rewriting over Ontological Axioms

The first transformation step involves expanding the SPARQL query according to the ontological rules. This processing is performed to include implicit results not explicitly matched in the input SPARQL query q. In other words, this process mimics the reasoning phase performed by standard graph databases. Ontop implemented different query rewriting strategies within time. Still, in the current state of the art, it employs the tree-witness query rewriting algorithm, which consists of identifying and utilizing tree witnesses, namely specific substructures within the input query that correspond to existential variables introduced by the ontology's existential rules [Calvanese et al., 2017; Kikot et al., 2012]. Once tree witnesses are identified, the algorithm rewrites the original query into a union of conjunctive queries that encapsulates all the necessary ontological inferences. Furthermore, by inspecting the Ontop debug logs, we noticed how the query rewriting process is not performed in a single shot. Still, instead, it is the result of a chain of sub-procedures aiming at optimizing the query structure, each of which produces in output a sort of semi-finished query, called *Intermediate Query (IQ)* [21]. Listing 4 showcases the rewritten IQ before undergoing the unfolding process. We can see that the query begins with a CONSTRUCT clause, which indicates the creation of a subgraph comprising the specified variables. The subsequent JOIN operations combine multiple INTENSIONAL triple patterns, as shown in lines 3 to 19. These triples correspond directly to the ontological rules without any inferred results, demonstrating that the explicit result set is formed in this specific case by joining intensional virtual triples over the ontological mappings alone.

```
1  CONSTRUCT [patient, date, tot, bulbar, motor, respiratory, q1, q2, q3, q4, q5,
       q6, q7, q8, q9, q10, q11, q12] [patient/SP_SUBSTR2(SP_STR(p),"48"^^xsd:
       integer)]
2   ORDER BY [ASC(SP_SUBSTR2(SP_STR(p),"48"^^xsd:integer)), ASC(date)]
3    JOIN
4     INTENSIONAL triple(p, <https://w3id.org/brainteaser/ontology/schema/undergo>,
        e)
5     INTENSIONAL triple(e, <https://w3id.org/brainteaser/ontology/schema/consists
       >, alsfrs)
6     INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
       procedureStart>, date)
7     INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
       revisedALSFRS>, tot)
8     INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
       bulbarSubscore>, bulbar)
9     INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
       motorSubscore>, motor)
10    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
       respiratorySubscore>, respiratory)
11    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
       alsfrs1>, q1)
12    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
       alsfrs2>, q2)
13    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
       alsfrs3>, q3)
```

[21] https://ontop-vkg.org/dev/internals/iq.html#iq

```
14    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
      alsfrs4>, q4)
15    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
      alsfrs5>, q5)
16    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
      alsfrs6>, q6)
17    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
      alsfrs7>, q7)
18    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
      alsfrs8>, q8)
19    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
      alsfrs9>, q9)
20    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
      alsfrs10>, q10)
21    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
      alsfrs11>, q11)
22    INTENSIONAL triple(alsfrs, <https://w3id.org/brainteaser/ontology/schema/
      alsfrs12>, q12)
```

Listing 4: Rewritten query in a DBMS-independent query representation format.

4.1.3 Query Unfolding over Mappings Definitions

Once the query rewriting process expands the original SPARQL query, Ontop performs the unfolding process. In this phase, the expanded SPARQL query is translated into SQL based on the mapping definitions that link the ontology to the virtual database schema. Here, multiple query optimizations occur, such as query lifting, fixed point join optimization, aggregation node simplification, and distinct clause prevention. As shown in Listing 5, the resulting query is expressed in SQL, composed of an intricate pattern. This highlights how a less complex data model (i.e., relational) reflects a poorer schema expressivity. For simplicity, the query is reported without redundant patterns that were not explicative.

```
1  CONSTRUCT [patient, date, tot, bulbar, motor, respiratory, q1, q2, ..., q12]
2  [
3      date/RDF(CHARACTER VARYINGToVARCHAR(date2m51), xsd:datetime),
4      q1/RDF(INTEGERToVARCHAR(q11m25), xsd:integer),
5      ...
6      q12/RDF(INTEGERToVARCHAR(q121m41), xsd:integer),
7      patient/RDF(SUBSTRING2("https://w3id.org/brainteaser/ontology/schema/Patient
       " || (CHARACTER VARYINGToVARCHAR(patient26m9)), "48"^^BIGINT), xsd:string),
8      tot/RDF(CHARACTER VARYINGToVARCHAR(tot1m45), xsd:integer),
9      respiratory/RDF(INTEGERToVARCHAR(respiratory1m35), xsd:integer)
10 ]
11 DISTINCT CONSTRUCT
12 [patient26m9, v49m9, date2m51, tot1m45, bulbar1m17, motor1m55, respiratory1m35,
       ..., q121m41]
13 [
14     v14m10/https://w3id.org/brainteaser/ontology/schema/alsfrs(CHARACTER
       VARYINGToVARCHAR(patient1m45), CHARACTER VARYINGToVARCHAR(date1m45))
15 ]
```

```
ORDER BY
[
    ASC(patient), ASC(date)
]
JOIN
    STRICT_EQ on alsfrs(patient, date) columns
    ...
    IS_NOT_NULL on all necessary columns
DISTINCT CONSTRUCT
[patient26m9, v49m9]
[v49m9/https://w3id.org/brainteaser/ontology/schema/e_spiro(patient26m9, date1m9
    )]
FILTER IS_NOT_NULL(patient26m9, date1m9)
EXTENSIONAL "clinical_data"."SPIRO"(patient, date)
```

Listing 5: Unfolded SQL Translation of the rewritten SPARQL Query after optimizations such as query lifting, fixed point join optimization, aggregation node simplification, and distinct clause prevention.

In the unfolded SQL query, the CONSTRUCT clause is crucial in shaping the structure of the resultant RDF data, where each field is transformed to match a specific data type and linked to its RDF representation. This transformation ensures that the output aligns with the expected semantic standards, facilitating subsequent data integration and analysis. The NATIVE clause instead specifies the direct SQL translation components, signifying how the translated query interfaces with the underlying SQL database. It highlights the direct mapping of SPARQL to SQL, ensuring that the original semantic query's intent is preserved while being adapted for execution over relational data structures. The SELECT clause in SQL is constructed to directly map each variable specified in the SPARQL SELECT query. Each variable, such as ?patient, ?date, ?tot, etc., corresponds to specific fields in the underlying database tables. For example, bto:procedureStart translates to selecting the date column in SQL. The SPARQL BIND function used to extract the patient ID from a URL or string is represented in SQL with a SUBSTRING function, allowing the extracted patient ID to be represented correctly in the result set. The FROM clause in SQL involves the specification of tables and joins corresponding to the triples patterns defined in the SPARQL query. The SPARQL predicate bto:undergo and bto:consists suggest relational joins between patient data and ALSFRS assessment data. This relationship is mapped to SQL joins or subqueries that fetch data from multiple related tables, capturing the relational nature of the data as specified by the ontology. The WHERE clause in SQL is essential for filtering data based on conditions expressed in SPARQL. Conditions such as ensuring data completeness or filtering based on specific criteria (like date ranges or specific patient characteristics) are directly translated from SPARQL conditions. The SQL version ensures that only relevant, complete records are processed. The ORDER BY clause in SQL mirrors the SPARQL order condition, ensuring that the results are returned in a specific order, which in this case is based on patient ID and date.

4.1.4 SQL queries on Data Sources

After the SQL query is delivered to Dremio, its virtualization engine evaluates the unfolded SQL query against the virtual view, to start the data retrieval process from the data sources. This is

patient	date	tot	bul	mo	res	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12
0x2913c1e	2020-11-21	27	6	12	9	0	3	3	4	4	2	2	0	0	3	3	3
0xd5043ae	2020-11-6	41	5	24	12	1	2	2	4	4	4	4	4	4	4	4	4
0xd5043a	2020-9-27	44	8	24	12	3	3	2	4	4	4	4	4	4	4	4	4
0x30779b	2021-1-12	40	12	16	12	4	4	4	4	4	3	3	2	0	4	4	4
0xa990c9	2021-10-8	45	9	24	12	3	3	3	4	4	4	4	4	4	4	4	4
0xe97dee5	2021-11-13	27	10	5	12	4	3	3	2	1	1	1	0	0	4	4	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 9: Initial chunk of the MySQL result set of query 6.

patient	date	tot	bul	mo	res	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12
0xe3e72f	2020-10-27	46	12	22	12	4	4	4	4	3	4	4	4	3	4	4	4
0xe3e72f	2021-03-28	44	12	20	12	4	4	4	3	3	4	4	3	3	4	4	4
0xe3e72f	2022-02-13	39	12	15	12	4	4	4	3	3	3	3	2	1	4	4	4
0x1e56c	2022-06-28	43	12	19	12	4	4	4	2	2	3	4	4	4	4	4	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 10: Initial chunk of the PostgreSQL result set of query 7.

performed by querying the data from the sources by executing the views and treating results as if they were real materialized tables. As shown in Figure 11, in this use case experiment, we have three different data sources: MySQL, PostgreSQL, and Polypheny. Considering that Polypheny does not have strong logging capabilities, we inspected both MySQL and PostgreSQL query logs to see how data is retrieved from sources. By inspecting both the DBMS log files, what immediately emerges is how the queries defined in the views get executed many times: this is explainable as we deactivated all Dremio caching mechanisms, and thus the data source is queried each time in the original query is recalled the virtual view, multiplied for the source invocations in the virtual view itself.

Listing 6 and 7 contains one of the repeated queries executed in MySQL and PostgreSQL, respectively.

```
1 SELECT `Turin_alsfrs`.`patient`, `Turin_alsfrs`.`date`, CAST(`Turin_alsfrs
    `.`tot` AS CHAR(65536)) AS `EXPR$2`, `Turin_alsfrs`.`bulbar`, `
    Turin_alsfrs`.`motor`, `Turin_alsfrs`.`respiratory`, `Turin_alsfrs`.`q1
    `, `Turin_alsfrs`.`q2`, `Turin_alsfrs`.`q3`, `Turin_alsfrs`.`q4`, `
    Turin_alsfrs`.`q5`, `Turin_alsfrs`.`q6`, `Turin_alsfrs`.`q7`, `
    Turin_alsfrs`.`q8`, `Turin_alsfrs`.`q9`, `Turin_alsfrs`.`q10`, `
    Turin_alsfrs`.`q11`, `Turin_alsfrs`.`q12`
2 FROM `alsfrs_turin`.`Turin_alsfrs`
3 WHERE `Turin_alsfrs`.`patient` IS NOT NULL AND `Turin_alsfrs`.`date` IS NOT
    NULL
```

Listing 6: SQL query submitted and executed on the MySQL instance.

patient	date	tot	bul	mo	res	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12
0xe3e72f	2020-10-27	46	12	22	12	4	4	4	4	3	4	4	4	3	4	4	4
0x2913c1e	2020-11-21	27	6	12	9	0	3	3	4	4	2	2	0	0	3	3	3
0xd5043ae	2020-11-6	41	5	24	12	1	2	2	4	4	4	4	4	4	4	4	4
0xd5043a	2020-9-27	44	8	24	12	3	3	2	4	4	4	4	4	4	4	4	4
0x30779b	2021-1-12	40	12	16	12	4	4	4	4	4	3	3	2	0	4	4	4
0xe3e72f	2021-03-28	44	12	20	12	4	4	4	3	3	4	4	3	3	4	4	4
0xa990c9	2021-10-8	45	9	24	12	3	3	3	4	4	4	4	4	4	4	4	4
0xe97dee5	2021-11-13	27	10	5	12	4	3	3	2	1	1	1	0	0	4	4	4
0xe3e72f	2022-02-13	39	12	15	12	4	4	4	3	3	3	3	2	1	4	4	4
0x1e56c	2022-06-28	43	12	19	12	4	4	4	2	2	3	4	4	4	4	4	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 11: Initial chunk of the Ontop result set of query 3.

```
SELECT "lisbon_alsfrs"."patient" COLLATE "C", CAST("lisbon_alsfrs"."date" AS
     VARCHAR(65536)) AS "EXPR$1", CAST("lisbon_alsfrs"."tot" AS VARCHAR
    (65536)) AS "EXPR$2", "lisbon_alsfrs"."bulbar", "lisbon_alsfrs"."motor",
     "lisbon_alsfrs"."respiratory", "lisbon_alsfrs"."q1", "lisbon_alsfrs"."
    q2", "lisbon_alsfrs"."q3", "lisbon_alsfrs"."q4", "lisbon_alsfrs"."q5", "
    lisbon_alsfrs"."q6", "lisbon_alsfrs"."q7", "lisbon_alsfrs"."q8", "
    lisbon_alsfrs"."q9", "lisbon_alsfrs"."q10", "lisbon_alsfrs"."q11", "
    lisbon_alsfrs"."q12"
    FROM "public"."lisbon_alsfrs"
    WHERE "lisbon_alsfrs"."patient" COLLATE "C" IS NOT NULL AND CAST("
    lisbon_alsfrs"."date" AS VARCHAR(65536)) IS NOT NULL AND "lisbon_alsfrs"
    ."q5" IS NOT NULL
```

Listing 7: SQL query submitted and executed on the PostgreSQL istance.

4.1.5 Result Set

To validate the integration capabilities in our architecture's federated environment, we executed the queries Dremio submitted to MySQL and PostgreSQL. Subsequently, we executed the SPARQL query presented in Listing 3, ordering the result set only by date to shuffle rows among different data sources. We finally inspected the result set returned by Ontop, and we matched the returned rows with those returned by the two sources. For this experiment, we turned off the Polypheny instance. Tables 9 and 10 showcases result sets coming from the execution of logged queries presented in Listing 6 and 7, belonging to MySQL and PostgreSQL respectively.

For privacy reasons, we obfuscated original data by hashing patient references and adding a random amount of time to each visit date. We then executed the SPARQL query in Ontop: Table 11 reports its result set. In yellow (●), we highlighted the rows coming from MySQL, while in green (●), those coming from PostgreSQL.

This example shows how the OBDF architecture can fetch data from different data sources and

plug them together, effectively mimicking the behavior of a graph-based DBMS.

4.2 Federated Analytics in Genomics: Challenges and Ontology Deployment

The conceptualization of use cases involving genomics variant data poses significant challenges due to such data's high complexity and volume. Genomic data often needs to be represented in compact formats, such as VCF, with dataset sizes reaching tens of gigabytes, even in their compressed form. This section discusses the challenges in handling genomics data, identifying suitable data formats for efficient querying, and deploying the HERO-Genomics ontology using the OBDA paradigm to enable federated analytics across diverse genomics datasets.

4.2.1 Challenges in Genomics Data Integration

For our use case, we needed to investigate genomics datasets to identify possible relations among rows of unstructured data. Subsequently, we aimed to find a suitable structured data format accepted by Dremio, ensuring that queries could be executed reasonably. Our final goal was to have the query represented in Listing 8 executed adequately through the federated OBDA architecture.

```
PREFIX : <https://w3id.org/hereditary/ontology/schema/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xml: <http://www.w3.org/XML/1998/namespace>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX obda: <https://w3id.org/obda/vocabulary#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?ref (COUNT(?ref) AS ?count)
WHERE {
 ?variation a :GenomicVariation .
 ?variation :referenceBases ?ref .
}
GROUP BY ?ref
ORDER BY DESC(?count)
```

Listing 8: SPARQL query on counting the total number of variations grouped by their reference base.

The query is based on HERO by focusing on its genomic specifications compliant with the Beacon v2 Data Model [Rambla et al., 2022; Rueda et al., 2022]. The query has a simple triple pattern, as we decided for this use case to focus on proving that crucial components of the SPARQL query language were working correctly (i.e., aggregation functions, grouping, and ordering).

4.2.2 Available Datasets

For this use case, we employed publicly available genomic datasets containing variant information and zygosity sampling, as these are the genomics information that HERO provides as a vocabulary. The *Common Infrastructure for National Cohorts in Europe, Canada, and Africa (CINECA)* project

website [22] offers synthetic datasets comprising both genotypical and phenotypical data grouped in cohorts covering different areas of the world. Variant data comes in VCF format, and each dataset is about 10 gigabytes, with some outlier exceptions.

To exploit federation capabilities, we selected three different datasets:

- CINECA synthetic cohort NA Canada CHILD v1 [23];
- CINECA synthetic cohort Europe CH SIB [24];
- CINECA synthetic cohort Africa H3ABioNet v1 [25].

4.2.3 Identification of a Suitable Structured Data Format

VCF data cannot be directly ingested and queried within Dremio due to its highly compressed nature that worsens query performances despite being well-suited for packing and sharing variants information. Therefore, developing a custom ARP connector for supporting VCF in Dremio would be impractical. This necessitated identifying a structure within raw variant data and finding a suitable data format to host such a structure. We manually investigated the VCF raw schema and observed the following characteristics:

1. Each row corresponds to a genomic variation; several initial fields hold information about the variation, such as the reference and alternate base;
2. A column named "INFO" contains comma-separated pairs of keys and values;
3. Each sample represents a column. Zygosities for each variant-sample pair are reported, composing a matrix.

This inspection allowed us to pinpoint entities and to define the relational schema shown in Figure 12.

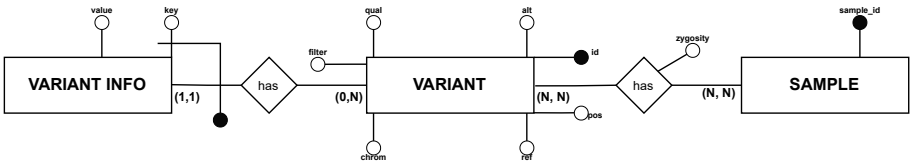

Figure 12: Entity-Relationship diagram extracted from the raw schema of a VCF file format. The three entities identified within the raw VCF data format are "VARIANT", "VARIANT INFO" and "SAMPLE".

The three entities represent the whole content of a VCF file row in a structured fashion. The entity "VARIANT" carries general information about the variation, such as reference and alternate bases,

[22] https://www.cineca-project.eu/
[23] https://zenodo.org/records/5122832
[24] https://zenodo.org/records/4955933
[25] https://zenodo.org/records/5082689

chromosome, position, and quality. The entity "VARIANT INFO" contains the comma-separated key-value pairs from the "INFO" field; the relation between the "VARIANT" entity and the "VARIANT INFO" entity is one-to-many, as each variant can have many linked information pairs. The entity "SAMPLE" contains the IDs of all the samples (i.e., the columns of the VCF file); the relation between "SAMPLE" and "VARIANT" is many-to-many, as each variant is related to each sample, carrying the zygosity information.

4.2.4 Experimental Setup

The conceptual schema reported in Figure 12 can be instantiated into different structured data formats. For our experiment, we employed three different storage paradigms (i.e., a relational DBMS, a text-based hierarchical structure, and columnar-oriented storage) to verify the impact of different data models on the size of the dataset. In particular, we converted the VCF files into SQL and ingested them into a PostgreSQL instance, into JSON files, and Apache Parquet files. Given the large original datasets, we took samples of different sizes for each of the three VCF files; we also considered a fourth small sample provided by CINECA from the Europe CH SIB cohort. We converted all four datasets into the three formats. Figure 13 illustrates conversion results regarding output dataset sizes. The Parquet format outperforms relational DBMS and JSON solutions, where output datasets

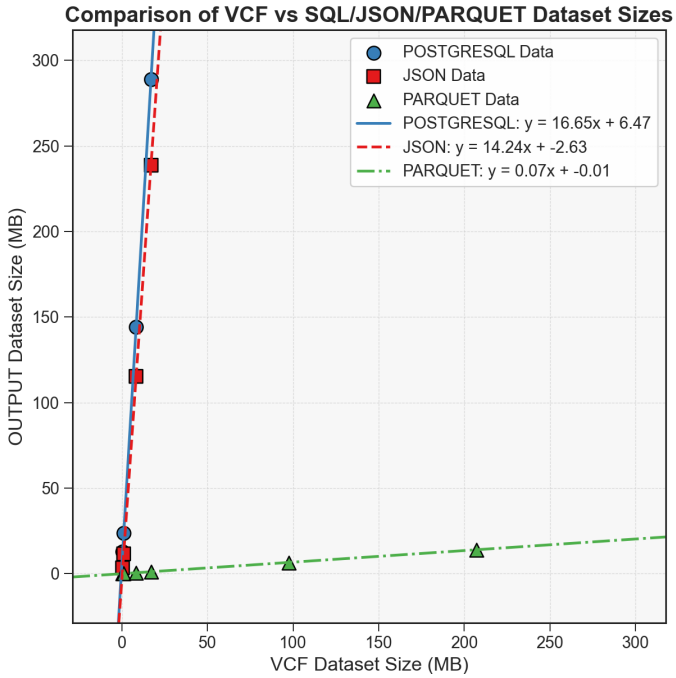

Figure 13: Comparison between the three converted datasets (PostgreSQL, JSON and Parquet) with respect to a VCF file.

are about 15 times larger than the originals. Parquet files, in contrast, are significantly smaller than the corresponding VCF files. Currently, the "VARIANT INFO" entity is excluded from Parquet files,

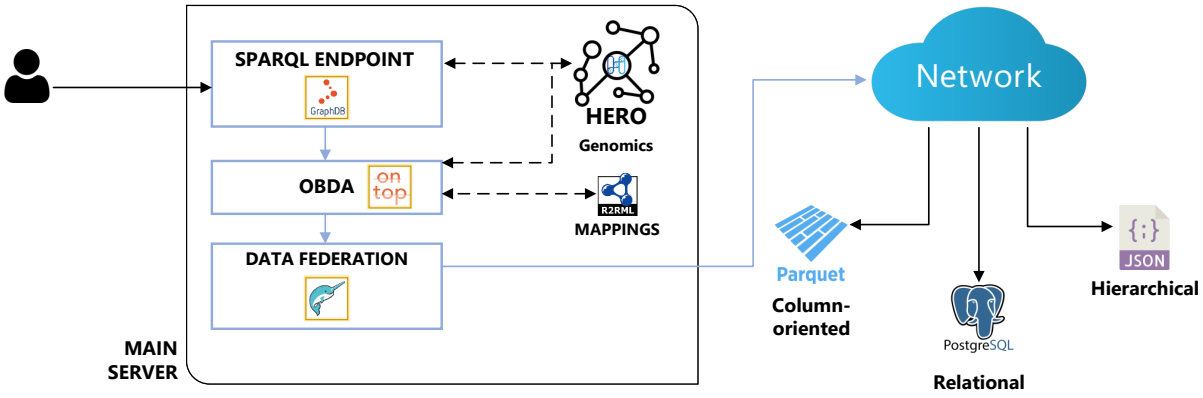

Figure 14: Federated software architecture implementing the OBDA paradigm. The framework comprises a SPARQL endpoint, an OBDA component exploiting the ontology and mapping definitions, and a Data Federation system that collects and virtualizes data from three diverse data sources: a column-oriented file, a hierarchical DB and a relational DB.

which is acceptable since it is not represented in HERO. Removing this entity in PostgreSQL and JSON storage showed minimal impact on dataset size. The "VARIANT" and "SAMPLE" relationship is flattened into a single table in Parquet.

4.2.5 Query Results

We organized the sampled datasets in the three data sources to prove the feasibility of all three structured sources. Figure 14 overviews how different software components are integrated into our use case scenario. Dremio and GraphDB were running in a local environment, while the PostgreSQL data source was remotely instantiated. Query 8 returned 50 rows in about 1 second. Table 12 shows a portion of the result set.

4.2.6 Ontology Deployment

This section presents how the HERO-Genomics ontology can be employed to define SPARQL queries expressing Beacon v2 genomic queries. The Beacon v2 specification consists of the framework and the models. The framework defines request and response formats, while the models specify the structure of biological data responses. Requests structure follows the Beacon v2 API [26]. Relevant Beacon queries comprise sequence queries for the existence of a specified sequence at a given genomic position, range queries for matching variants at least overlapping with a specified region, GeneId queries for returning variants affecting a gene's coding region, bracket queries for matching variants falling in a start range and end range, genomic allele queries for matching variants with the specified allele and amino acid change queries for matching variants with the specified amino acid change.

[26] https://docs.genomebeacons.org/variant-queries/#beacon-v2

REF	COUNT
G	789
C	730
T	549
A	512
CA	8
AT	7
GA	7
TA	7
CT	7
AC	6
AATG	6
...	...

Table 12: Partition of the retrieved result set from Ontop after execution of query represented in Listing 8.

In this use-case, we employ the query in Listing 9, formulated on the HERO-Genomics data model, as it represents a valid example of a *Beacon Sequence Query*, given it verifies the existence of specified sequences at given genomic positions. The correct execution of this query should return a tuple for each position if the specified sequence exists.

```
PREFIX : <https://w3id.org/hereditary/ontology/schema/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?start ?ref ?alt
WHERE { ?variation a :GenomicVariation ;  :referenceBases ?ref ; :alternateBases ?alt ; :location ?seqLoc .
    ?seqLoc :sequenceInterval ?seqInt .
    ?seqInt :sequenceIntervalStart ?start .
    OPTIONAL { ?seqLoc :referenceSequence ?refName .
        FILTER (?refName = "GRCh38"^^rdf:PlainLiteral)}
    FILTER ( (?start = "9419057"^^rdf:PlainLiteral && ?ref = "C"^^rdf:PlainLiteral && ?alt = "T"^^rdf:PlainLiteral) ||
        (?start = "719853"^^rdf:PlainLiteral && ?ref = "CAG"^^rdf:PlainLiteral && ?alt = "C"^^rdf:PlainLiteral) ||
        (?start = "16115625"^^rdf:PlainLiteral && ?ref = "AT"^^rdf:PlainLiteral && ?alt = "A"^^rdf:PlainLiteral) )}
```

Listing 9: SPARQL query verifying the existence of a specified sequence at a given genomic position.

HERO-Genomics offers the same expressive capabilities as Beacon, combined with the flexibility of SPARQL queries. A key advantage is the ability to query multiple genomics datasets distributed across different centers. The synthetic datasets we use here employ various data models and compression techniques. We illustrate how the OBDA-based federated architecture enables mapping the ontology to these diverse dataset sources. We employed the same setup configuration and sample datasets presented in the previous use case example described in Figure 11. We mapped this schema with the ontology vocabulary using the Ontop native mapping language. An example of a mapping definition is presented in Listing 10

```
mappingId MAPID-868c2fe1dffc4fa2b01888540fc4639b
target  :chr{chrom}{pos}{ref}{alt} :variantAlternativeID :chr{chrom}{pos}{ref}{
    alt}AlternativeID ; :referenceBases {ref}^^rdf:PlainLiteral ; :
    alternateBases {alt}^^rdf:PlainLiteral . :chr{chrom}{pos}{ref}{alt}
    AlternativeID :valueID {id}^^rdf:PlainLiteral .
source    SELECT chrom, pos, "ref", alt, id FROM public_json_parquet_psql.
    variants;
```

Listing 10: A mapping definition in Ontop native mapping language in this case mapping a SPARQL query to a SQL query. The mapping definition comprises three fields: the mapping id, the target triples in the ontology that must be mapped, and the source SQL query that retrieves the data from a relational database.

To implement the principles of the OBDA paradigm, target triples in HERO-Genomics are parameterized with placeholders enclosed in curly brackets. These placeholders correspond to fields in the virtual genomic relational schema referenced by the associated SQL source query. The combination of this mapping layer alongside a virtualization layer bridges HERO-Genomics and the local genomic data sources. We executed the query in Listing 9 to demonstrate the querying process from HERO-Genomics to the local data sources and back, exploiting the defined OBDA architecture. The results are shown in Table 13.

start	ref	alt
16115625	AT	A
9419057	C	T
719853	CAG	C

Table 13: Retrieved result set from Ontop after executing the query in Listing 8. Rows are color-coded based on the originating data source: PostgreSQL (●), Apache Parquet (●) and JSON (●).

In blue (●) we highlighted the row coming from PostgreSQL; in yellow (●) the row coming from the Apache Parquet dataset; in green (●) the row coming from the JSON dataset. This demonstrates how the data federation system enables querying of three heterogeneous and remote data sources. We can interpret relationships between the data and construct meaningful queries by employing HERO-Genomics as a unified semantic layer. The resulting dataset is a unified collection of triples, seamlessly integrated despite originating from sources that could not otherwise be queried together. Integrating the HERO-Genomics ontology with a federated OBDA architecture addresses the challenges posed by complex and heterogeneous genomics data. We enable efficient and unified querying across multiple datasets by structuring the raw variant data into suitable formats and employing a data federation system. The OBDA paradigm, supported by tools like Ontop, allows for the seamless mapping of ontology terms to diverse data sources, facilitating advanced queries that leverage the expressive power of SPARQL. This approach demonstrates the potential for federated analytics in genomics, providing a scalable solution for integrating and analyzing large-scale genomics data across different platforms and formats.

5 Milestone verification

This deliverable verifies Milestone 5: "Semantic ontology modeling concepts and processes for ALS and MS; design of the federated execution methods."

Means of verification: Online preliminary ontology version focusing on neurological diseases, specifically ALS and MS. The HERO ontology is openly available at the URL https://hereditary.dei.unipd.it/ontology/; it can be downloaded as an OWL file, and the documentation is publicly available. At the moment, the HERO ontology is divided into HERO-Clinical and HERO-Genomics. These two connected ontologies constitute the basis for further developments within the project that will include further brain-related diseases like Parkinson's disease and the modeling of the gut-brain interplay that is undergoing at the moment.

Moreover, this deliverable presents the initial architecture for supporting federated analytics over distributed independent heterogeneous datasets employing different data models.

6 Final remarks

This deliverable achieves Milestone 5, laying the groundwork for semantic ontology modeling and federated execution methods tailored to ALS and MS research. The HERO ontology, guided by FAIR principles, provides a robust framework for representing clinical and genomic data while ensuring interoperability and reusability. Its modular and hierarchical design facilitates focused development and scalability, addressing the complex requirements of neurological disease modeling.

The federated execution framework complements this effort by enabling seamless integration and analysis of distributed datasets. The architecture ensures efficient, semantically enriched data querying across diverse sources through the Data Federation and Virtualization Layer and the Ontology-Based Data Access (OBDA) Layer. The proposed use cases validate the framework's capability to handle real-world queries in both clinical and genomic contexts.

This deliverable addresses the immediate project objectives and establishes a foundation for advancing data integration and analytics in the following steps of the HEREDITARY project.

7 Annexes

Number	Name
1	Clinical Data Modeling of the HERO ontology organized into eight semantic areas.
2	Genomic Data Modeling of the HERO ontology.
3	Custom ARP Connector Implementation Details.
4	NAS and Google Drive connection to Dremio.

References

Adams, R. D., Victor, M., and Ropper, A. H. (1997). *Principles of Neurology. 6th Edition*, volume 24. McGraw Hill.

Allemang, D., Hendler, J., and Gandon, F. (2020). Good and bad modeling practices. In *Semantic Web for the Working Ontologist: Effective Modeling for Linked Data, RDFS, and OWL*, pages 436–440. Association for Computing Machinery, New York, NY, USA.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., and Sherlock, G. (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29.

Bairoch, A. (2018). The Cellosaurus, a Cell-Line Knowledge Resource. *Journal of biomolecular techniques : JBT*, 29:25–38.

Beck, A. T., Steer, R. A., and Brown, G. (1996). Beck depression inventory–ii. *Psychological assessment*.

Benedict, R. H., Cox, D., Thompson, L. L., Foley, F., Weinstock-Guttman, B., and Munschauer, F. (2004). Reliable screening for neuropsychological impairment in multiple sclerosis. *Multiple Sclerosis Journal*, 10(6):675–678. PMID: 15584493.

Bodenreider, O. (2004). The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32(Database-Issue):267–270.

Brookes, A. J. (1999). The essence of snps. *Gene*, 234(2):177–186.

Calvanese, D., Cogrel, B., Kalayci, E. G., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., and Xiao, G. (2015). OBDA with the ontop framework. In Lembo, D., Torlone, R., and Marrella, A., editors, *23rd Italian Symposium on Advanced Database Systems, SEBD 2015, Gaeta, Italy, June 14-17, 2015*, pages 296–303. Curran Associates, Inc.

Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., and Xiao, G. (2017). Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8(3):471–487.

Calvanese, D., Giacomo, G. D., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., and Savo, D. F. (2011). The MASTRO system for ontology-based data access. *Semantic Web*, 2(1):43–53.

Calvanese, D., Giacomo, G. D., Lembo, D., Lenzerini, M., Poggi, A., and Rosati, R. (2007). Ontology-based database access. In Ceci, M., Malerba, D., and Tanca, L., editors, *Proceedings of the Fifteenth Italian Symposium on Advanced Database Systems, SEBD 2007, 17-20 June 2007, Torre Canne, Fasano, BR, Italy*, pages 324–331.

Cedarbaum, J. M., Stambler, N., Malta, E., Fuller, C., Hilt, D., Thurmond, B., and Nakanishi, A. (1999). The alsfrs-r: a revised als functional rating scale that incorporates assessments of respiratory function. *Journal of the Neurological Sciences*, 169(1):13–21.

Chiò, A., Hammond, E. R., Mora, G., Bonito, V., and Filippini, G. (2015). Development and evaluation of a clinical staging system for amyotrophic lateral sclerosis. *Journal of Neurology, Neurosurgery & Psychiatry*, 86(1):38–44.

Consortium, T. G. O., Aleksander, S. A., Balhoff, J., Carbon, S., Cherry, J. M., Drabkin, H. J., Ebert, D., Feuermann, M., Gaudet, P., Harris, N. L., Hill, D. P., Lee, R., Mi, H., Moxon, S., Mungall, C. J., Muruganugan, A., Mushayahama, T., Sternberg, P. W., Thomas, P. D., Van Auken, K., Ramsey, J., Siegele, D. A., Chisholm, R. L., Fey, P., Aspromonte, M. C., Nugnes, M. V., Quaglia, F., Tosatto, S., Giglio, M., Nadendla, S., Antonazzo, G., Attrill, H., dos Santos, G., Marygold, S., Strelets, V., Tabone, C. J., Thurmond, J., Zhou, P., Ahmed, S. H., Asanitthong, P., Luna Buitrago, D., Erdol, M. N., Gage, M. C., Ali Kadhum, M., Li, K. Y. C., Long, M., Michalak, A., Pesala, A., Pritazahra, A., Saverimuttu, S. C. C., Su, R., Thurlow, K. E., Lovering, R. C., Logie, C., Oliferenko, S., Blake, J., Christie, K., Corbani, L., Dolan, M. E., Drabkin, H. J., Hill, D. P., Ni, L., Sitnikov, D., Smith, C., Cuzick, A., Seager, J., Cooper, L., Elser, J., Jaiswal, P., Gupta, P., Jaiswal, P., Naithani, S., Lera-Ramirez, M., Rutherford, K., Wood, V., De Pons, J. L., Dwinell, M. R., Hayman, G. T., Kaldunski, M. L., Kwitek, A. E., Laulederkind, S. J. F., Tutaj, M. A., Vedi, M., Wang, S.-J., D'Eustachio, P., Aimo, L., Axelsen, K., Bridge, A., Hyka-Nouspikel, N., Morgat, A., Aleksander, S. A., Cherry, J. M., Engel, S. R., Karra, K., Miyasato, S. R., Nash, R. S., Skrzypek, M. S., Weng, S., Wong, E. D., Bakker, E., Berardini, T. Z., Reiser, L., Auchincloss, A., Axelsen, K., Argoud-Puy, G., Blatter, M.-C., Boutet, E., Breuza, L., Bridge, A., Casals-Casas, C., Coudert, E., Estreicher, A., Livia Famiglietti, M., Feuermann, M., Gos, A., Gruaz-Gumowski, N., Hulo, C., Hyka-Nouspikel, N., Jungo, F., Le Mercier, P., Lieberherr, D., Masson, P., Morgat, A., Pedruzzi, I., Pourcel, L., Poux, S., Rivoire, C., Sundaram, S., Bateman, A., Bowler-Barnett, E., Bye-A-Jee, H., Denny, P., Ignatchenko, A., Ishtiaq, R., Lock, A., Lussi, Y., Magrane, M., Martin, M. J., Orchard, S., Raposo, P., Speretta, E., Tyagi, N., Warner, K., Zaru, R., Diehl, A. D., Lee, R., Chan, J., Diamantakis, S., Raciti, D., Zarowiecki, M., Fisher, M., James-Zorn, C., Ponferrada, V., Zorn, A., Ramachandran, S., Ruzicka, L., and Westerfield, M. (2023). The Gene Ontology knowledgebase in 2023. *Genetics*, 224(1):iyad031.

Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., Handsaker, R. E., Lunter, G., Marth, G. T., Sherry, S. T., McVean, G., Durbin, R., and Group, . G. P. A. (2011). The variant call format and VCFtools. *Bioinformatics*, 27(15):2156–2158.

Delis, D., Kaplan, E., and Kramer, J. (2001). Delis-Kaplan Executive Function System:Technical Manual. *San Antonio, TX: Harcourt Assessment Company*.

Faggioli, G., Menotti, L., Marchesin, S., Chió, A., Dagliati, A., de Carvalho, M., Gromicho, M., Manera, U., Tavazzi, E., Di Nunzio, G. M., Silvello, G., and Ferro, N. (2024). An extensible and unifying approach to retrospective clinical data modeling: the brainteaser ontology. *Journal of Biomedical Semantics*, 15(1):16.

Gelfand, J. M. (2014). Chapter 12 - multiple sclerosis: diagnosis, differential diagnosis, and clinical presentation. In Goodin, D. S., editor, *Multiple Sclerosis and Related Disorders*, volume 122 of *Handbook of Clinical Neurology*, pages 269–290. Elsevier.

Gogate, N., Lyman, D., Bell, A., Cauley, E., Crandall, K. A., Joseph, A., Kahsay, R., Natale, D. A., Schriml, L. M., Sen, S., and Mazumder, R. (2021). COVID-19 biomarkers and their overlap with comorbidities in a disease biomarker data model. *Briefings in Bioinformatics*, 22(6). bbab191.

Gu, Z., Calvanese, D., Panfilo, M. D., Lanti, D., Mosca, A., and Xiao, G. (2024). OBDF: OBDA + data federation - extended abstract. In *40th International Conference on Data Engineering, ICDE 2024 - Workshops, Utrecht, Netherlands, May 13-16, 2024*, pages 381–383. IEEE.

Hardiman, O., Al-Chalabi, A., Chio, A., Corr, E. M., Logroscino, G., Robberecht, W., Shaw, P. J., Simmons, Z., and van den Berg, L. H. (2017). Amyotrophic lateral sclerosis. *Nature Reviews Disease Primers*, 3(1):17071.

Houtchens, M. K. and Khoury, S. J. (2013). Chapter 52 - multiple sclerosis. In Goldman, M. B., Troisi, R., and Rexrode, K. M., editors, *Women and Health (Second Edition)*, pages 785–801. Academic Press, second edition edition.

Kharlamov, E., Solomakhina, N., Özçep, Ö. L., Zheleznyakov, D., Hubauer, T., Lamparter, S., Roshchin, M., Soylu, A., and Watson, S. (2014). How semantic technologies can enhance data access at siemens energy. In Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C. A., Vrandecic, D., Groth, P., Noy, N. F., Janowicz, K., and Goble, C. A., editors, *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, volume 8796 of *Lecture Notes in Computer Science*, pages 601–619. Springer.

Kikot, S., Kontchakov, R., and Zakharyaschev, M. (2012). Conjunctive query answering with OWL 2 QL. In Brewka, G., Eiter, T., and McIlraith, S. A., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press.

Kolozali, S., Elsaleh, T., and Barnaghi, P. M. (2014). A validation tool for the W3C SSN ontology based sensory semantic knowledge. In *Joint Proceedings of the 6th International Workshop on the Foundations, Technologies and Applications of the Geospatial Web, TC 2014, and 7th International Workshop on Semantic Sensor Networks, SSN 2014, co-located with 13th International Semantic Web Conference (ISWC 2014)*, volume 1401 of *CEUR Workshop Proceedings*, pages 83–88.

Kurtzke, J. F. (1983). Rating neurologic impairment in multiple sclerosis. *Neurology*, 33(11):1444–1444.

Ladewig, M. S., Jacobsen, J. O. B., Wagner, A. H., Danis, D., El Kassaby, B., Gargano, M., Groza, T., Baudis, M., Steinhaus, R., Seelow, D., Bechrakis, N. E., Mungall, C. J., Schofield, P. N., Elemento,

O., Smith, L., McMurry, J. A., Munoz-Torres, M., Haendel, M. A., and Robinson, P. N. (2023). GA4GH Phenopackets: A Practical Introduction. *Advanced Genetics*, 4(1):2200016.

Lanti, D., Rezk, M., Xiao, G., and Calvanese, D. (2015). The NPD benchmark: Reality check for OBDA systems. In Alonso, G., Geerts, F., Popa, L., Barceló, P., Teubner, J., Ugarte, M., den Bussche, J. V., and Paredaens, J., editors, *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015*, pages 617–628. OpenProceedings.org.

Longinetti, E. and Fang, F. (2019). Epidemiology of amyotrophic lateral sclerosis: an update of recent literature. *Current opinion in neurology*, 32(5):771.

Makris, K., Gioldasis, N., Bikakis, N., and Christodoulakis, S. (2010). Ontology mapping and SPARQL rewriting for querying federated RDF data sources - (short paper). In Meersman, R., Dillon, T. S., and Herrero, P., editors, *On the Move to Meaningful Internet Systems, OTM 2010 - Confederated International Conferences: CoopIS, IS, DOA and ODBASE, Hersonissos, Crete, Greece, October 25-29, 2010, Proceedings, Part II*, volume 6427 of *Lecture Notes in Computer Science*, pages 1108–1117. Springer.

Malone, J., Holloway, E., Adamusiak, T., Kapushesky, M., Zheng, J., Kolesnikov, N., Zhukova, A., Brazma, A., and Parkinson, H. (2010). Modeling sample variables with an Experimental Factor Ontology. *Bioinformatics*, 26(8):1112–1118.

Namici, M. (2018). R2RML mappings in OBDA systems: Enabling comparison among OBDA tools. *CoRR*, abs/1804.01405.

Namici, M. and De Giacomo, G. (2018). Comparing query answering in OBDA tools over w3c-compliant specifications. In Ortiz, M. and Schneider, T., editors, *Proceedings of the 31st International Workshop on Description Logics co-located with 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018), Tempe, Arizona, US, October 27th - to - 29th, 2018*, volume 2211 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Nocentini, U., Giordano, A., Vincenzo, S., Panella, M., and Pasqualetti, P. (2006). The symbol digit modalities test - oral version: Italian normative data. *Functional neurology*, 21:93–6.

Olsson, T., Barcellos, L. F., and Alfredsson, L. (2017). Interactions between genetic, lifestyle and environmental risk factors for multiple sclerosis. *Nature Reviews Neurology*, 13(1):25–36.

Ozakbas, S., Cinar, B. P., Gurkan, M. A., Ozturk, O., Oz, D., and Kursun, B. B. (2016). Paced auditory serial addition test: National normative data. *Clinical Neurology and Neurosurgery*, 140:97–99.

Peters, O. and Brown, R. (2015). Chapter 18 - amyotrophic lateral sclerosis. In Zigmond, M. J., Rowland, L. P., and Coyle, J. T., editors, *Neurobiology of Brain Disorders*, pages 262–280. Academic Press, San Diego.

Poveda-Villalón, M., Gómez-Pérez, A., and Suárez-Figueroa, M. C. (2014). Oops! (ontology pitfall scanner!): An on-line tool for ontology evaluation. *Int. J. Semantic Web Inf. Syst.*, 10(2):7–34.

Rambla, J., Baudis, M., Ariosa, R., Beck, T., Fromont, L. A., Navarro, A., Paloots, R., Rueda, M., Saunders, G., Singh, B., Spalding, J. D., Törnroos, J., Vasallo, C., Veal, C. D., and Brookes, A. J. (2022). Beacon v2 and beacon networks: A "lingua franca" for federated data discovery in biomedical genomics, and beyond. *Human Mutation*, 43(6):791–799.

Renton, A. E., Chiò, A., and Traynor, B. J. (2014). State of play in amyotrophic lateral sclerosis genetics. *Nature Neuroscience*, 17(1):17–23.

Roche, J. C., Rojas-Garcia, R., Scott, K. M., Scotton, W., Ellis, C. E., Burman, R., Wijesekera, L., Turner, M. R., Leigh, P. N., Shaw, C. E., and Al-Chalabi, A. (2012). A proposed staging system for amyotrophic lateral sclerosis. *Brain*, 135(3):847–852.

Rodriguez, M. and Calvanese, D. (2011). Dependencies: Making ontology based data access work. In Barceló, P. and Tannen, V., editors, *Proceedings of the 5th Alberto Mendelzon International Workshop on Foundations of Data Management, Santiago, Chile, May 9-12, 2011*, volume 749 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Rueda, M., Ariosa, R., Moldes, M., and Rambla, J. (2022). Beacon v2 Reference Implementation: a toolkit to enable federated sharing of genomic and phenotypic data. *Bioinformatics*, 38(19):4656–4657.

Sayre, R. R., Wambaugh, J. F., and Grulke, C. M. (2020). Database of pharmacokinetic time-series data and parameters for 144 environmental chemicals. *Scientific Data*, 7.

Schaeffer, J., Cossetti, C., Mallucci, G., and Pluchino, S. (2015). Chapter 30 - multiple sclerosis. In Zigmond, M. J., Rowland, L. P., and Coyle, J. T., editors, *Neurobiology of Brain Disorders*, pages 497–520. Academic Press, San Diego.

Simperl, E. (2009). Reusing ontologies on the semantic web: A feasibility study. *Data & Knowledge Engineering*, 68(10):905–925.

Sioutos, N., de Coronado, S., Haber, M. W., Hartel, F. W., Shaiu, W.-L., and Wright, L. W. (2007). Nci thesaurus: a semantic model integrating cancer-related clinical and molecular information. *Journal of Biomedical Informatics*, 40(1):30–43.

Stangel, M., Fredrikson, S., Meinl, E., Petzold, A., Stüve, O., and Tumani, H. (2013). The utility of cerebrospinal fluid analysis in patients with multiple sclerosis. *Nature Reviews Neurology*, 9(5):267–276.

Vogt, M., Stiemer, A., and Schuldt, H. (2018). Polypheny-db: Towards a distributed and self-adaptive polystore. In Abe, N., Liu, H., Pu, C., Hu, X., Ahmed, N. K., Qiao, M., Song, Y., Kossmann, D., Liu, B., Lee, K., Tang, J., He, J., and Saltz, J. S., editors, *IEEE International Conference on Big Data (IEEE BigData 2018), Seattle, WA, USA, December 10-13, 2018*, pages 3364–3373. IEEE.

Waubant, E., Lucas, R., Mowry, E., Graves, J., Olsson, T., Alfredsson, L., and Langer-Gould, A. (2019). Environmental and genetic risk factors for ms: an integrated review. *Annals of Clinical and Translational Neurology*, 6(9):1905–1922.

Westeneng, H.-J., van Veenhuijzen, K., van der Spek, R. A., Peters, S., Visser, A. E., van Rheenen, W., Veldink, J. H., and van den Berg, L. H. (2021). Associations between lifestyle and amyotrophic lateral sclerosis stratified by c9orf72 genotype: a longitudinal, population-based, case-control study. *The Lancet Neurology*, 20(5):373–384.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G., Groth, P., Goble, C., Grethe, J. S., Heringa, J., 't Hoen, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., and Mons, B. (2016). The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3(1):160018.

Zhang, W., Zeng, B., Yang, M., Yang, H., Wang, J., Deng, Y., Zhang, H., Yao, G., Wu, S., and Li, W. (2021). ncrnavar: A manually curated database for identification of noncoding rna variants associated with human diseases. *Journal of Molecular Biology*, 433(11):166727. Computation Resources for Molecular Biology.