Virtual Knowledge Graphs for Data Management – Challenges and Solutions

Diego Calvanese

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy

Ontopic s.r.l.

Universidade de Lisboa – Instituto Superior Técnico

24 April 2025 – Lisbon

The problem of data access in data management

In large organization data management is a complex challenge:

- Many different data sets are created independently.
- The data is heterogeneous in the way it is represented and structured.
- Data are often stored across different sources (possibly controlled by different people / organizations).

The problem of data access

However, complex data processing pipelines (e.g., for analysis, monitoring and prediction) require to **access in an integrated and uniform way** such large, richly structured, and heterogeneus data sets.

unibz

How can we address the complexity of data access?

We combine three key ideas:

1. Expose to users/applications the data in a very flexible data model, making use of terms the users are familiar with
   $\leadsto$ **Knowledge Graph** whose vocabulary is expressed in a **domain ontology / global schema**.

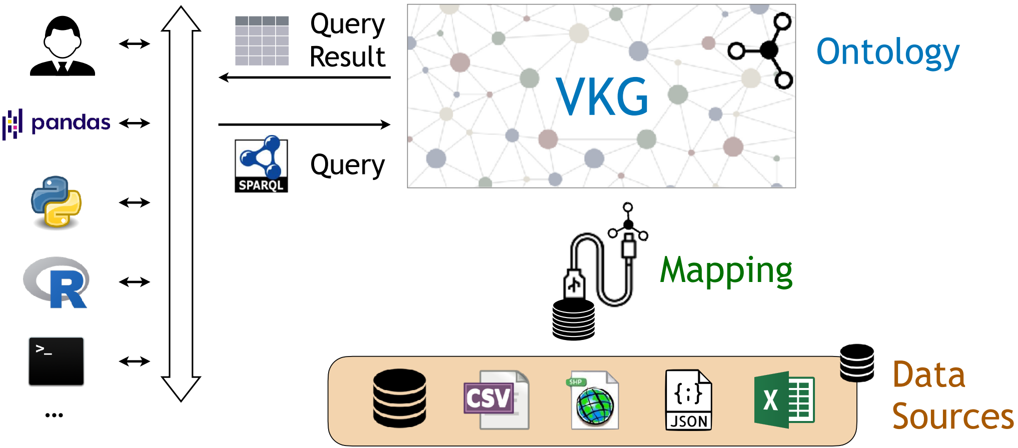
2. **Map the data sources to the global schema** in order to provide the data for the KG.

3. Exploit **virtualization**, i.e., the KG is not materialized, but kept virtual.

This gives rise to the **Virtual Knowledge Graph** (**VKG**) approach to data access,
also called **Ontology-based Data Access** (**OBDA**).
[Xiao, C., et al. 2018, IJCAI]

unibz

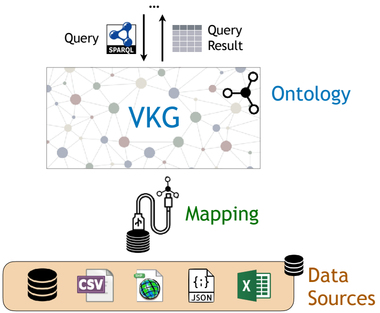
Virtual Knowledge Graph (VKG) architecture



unibz

Why an ontology?

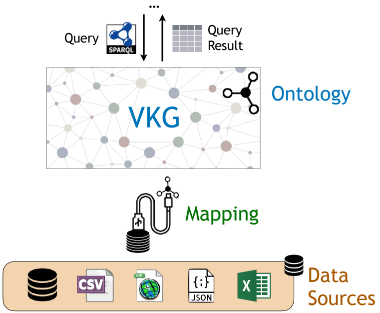
An ontology is a structured formal representation of concepts and their relationships that are relevant for the domain of interest.



- In the VKG setting, the ontology has a twofold purpose:
  - It defines a vocabulary of terms to denote classes and properties that are familiar to the user.
  - It extends the data in the sources with background knowledge about the domain of interest, and this knowledge is machine processable.
- One can make use of custom-built domain ontologies.
- In addition, one can rely on standard ontologies, which are available for many domains.

Why a Knowledge Graph for the global schema?

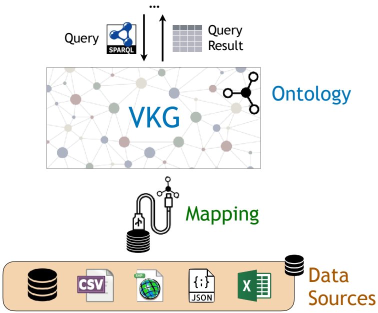
The traditional approach to data integration adopts a relational global schema.



A Knowledge Graph, instead:

- Does not require to commit early on to a specific structure.
- Can better accommodate heterogeneity.
- Can better deal with missing / incomplete information.
- Does not require complex restructuring operations to accommodate new information or new data sources.

Why mappings?

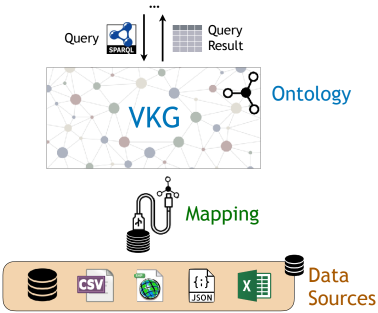


The traditional approach to data integration relies on mediators, which are specified through complex code.

Mappings, instead:

- Provide a declarative specification, and not code.
- Are easier to understand, and hence to design and to maintain.
- Support an incremental approach to integration.
- Are machine processable, hence are used in query answering and for query optimization.

Why virtualization?

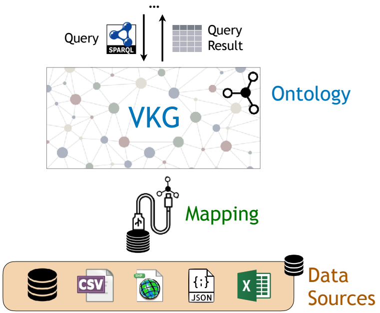


Materialized data integration relies on extract-transform-load (ETL) operations, to load data from the sources into an integrated data store / data warehouse / materialized KG.

In the virtual approach, instead:

- The data stays in the sources and is only accessed at query time.
- No need to construct a large and potentially costly materialized data store and keep it up-to-date.
- Hence the data is always fresh wrt the latest updates at the sources.
- One can rely on the existing data infrastructure and expertise.
- There is better support for an incremental approach to integration.

Incomplete information



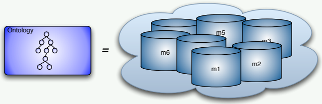
We are in a setting of **incomplete information**!!!

Incompleteness is introduced:

- by data sources, in general assumed to be incomplete;
- by domain constraints encoded in the ontology.

Plus:

Ontologies are logical theories, and hence perfectly suited to deal with incomplete information!



Minus:

Query answering amounts to **logical inference**, and hence is significantly more challenging.

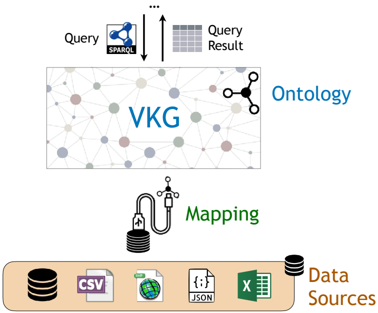
Outline

Outline

Components of the VKG framework

We consider now the main components that make up the VKG framework, and the languages used to specify them.

In defining such languages, we need to consider the **tradeoff between expressive power and efficiency**, where the key point is efficiency with respect to the data.

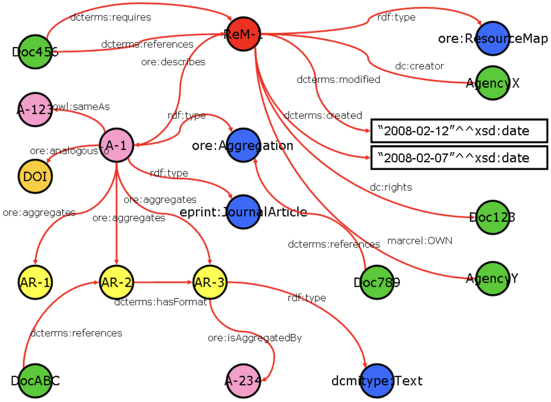


The W3C has standardized languages that are suitable for VKGs:

1. Knowledge graph: expressed in **RDF** — [W3C Rec. 2014] (v1.1)
2. Ontology $\mathcal{O}$: expressed in **OWL 2 QL** — [W3C Rec. 2012]
3. Mapping $\mathcal{M}$: expressed in **R2RML** — [W3C Rec. 2012]
4. Query: expressed in **SPARQL** — [W3C Rec. 2013] (v1.1)

RDF – Data is represented as a graph

The graph consists of a set of **subject-predicate-object triples**:



Class membership:

```
<A-1> rdf:type :Actor .
```

Object property:

```
<A-1> :playsIn <M-25> .
```

Data property:

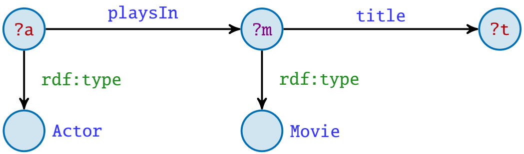
```
<M-25> :releaseDate "2008-02-12" .
```

unibz

SPARQL query language

- Is the standard query language for RDF data. [W3C Rec. 2008, 2013]
- Core query mechanism is based on **graph matching**.

```
SELECT ?a ?t
WHERE { ?a rdf:type Actor .
        ?a playsIn ?m .
        ?m rdf:type Movie .
        ?m title ?t .
      }
```

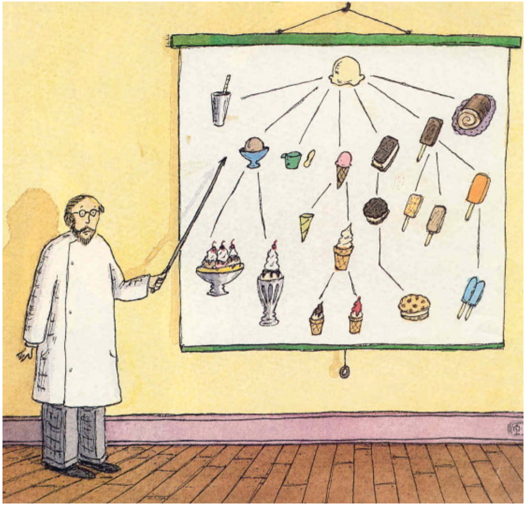


Additional language features (SPARQL 1.1):

- `UNION`: matches one of alternative graph patterns
- `OPTIONAL`: produces a match even when part of the pattern is missing
- complex `FILTER` conditions
- `GROUP BY`, to express aggregations
- `MINUS`, to remove possible solutions
- property paths (regular expressions)
- ...

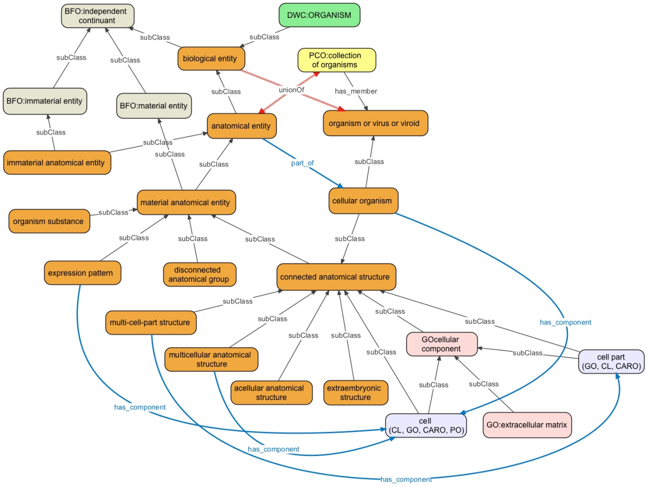
What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts in a hierarchical structure.



What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts in a hierarchical structure.
- Ontologies are often represented as graphs.



What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts in a hierarchical structure.
- Ontologies are often represented as graphs.
- However, an ontology is actually a **logical theory**, expressed in a suitable fragment of first-order logic

$\forall x.\, \mathsf{Actor}(x) \rightarrow \mathsf{Staff}(x)$
$\forall x.\, \mathsf{SeriesActor}(x) \rightarrow \mathsf{Actor}(x)$
$\forall x.\, \mathsf{MovieActor}(x) \rightarrow \mathsf{Actor}(x)$
$\forall x.\, \mathsf{SeriesActor}(x) \rightarrow \neg\mathsf{MovieActor}(x)$

$\forall x.\, \mathsf{Staff}(x) \rightarrow \exists y.\, \mathit{ssn}(x, y)$
$\forall y.\, \exists x.\, \mathit{ssn}(x, y) \rightarrow \mathtt{xsd{:}int}(y)$
$\forall x, y, y'.\, \mathit{ssn}(x, y) \land \mathit{ssn}(x, y') \rightarrow y = y'$

$\forall x.\, \exists y.\, \mathsf{actsIn}(x, y) \rightarrow \mathsf{MovieActor}(x)$
$\forall y.\, \exists x.\, \mathsf{actsIn}(x, y) \rightarrow \mathsf{Movie}(y)$
$\forall x.\, \mathsf{MovieActor}(x) \rightarrow \exists y.\, \mathsf{actsIn}(x, y)$
$\forall x.\, \mathsf{Movie}(x) \rightarrow \exists y.\, \mathsf{actsIn}(y, x)$
$\forall x, y.\, \mathsf{actsIn}(x, y) \rightarrow \mathsf{playsIn}(x, y)$
$\ldots$

What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts in a hierarchical structure.
- Ontologies are often represented as graphs.
- However, an ontology is actually a **logical theory**, expressed in a suitable fragment of first-order logic, or better, in **description logics**.

$$
\begin{aligned}
\mathsf{Actor} &\sqsubseteq \mathsf{Staff}\\
\mathsf{SeriesActor} &\sqsubseteq \mathsf{Actor}\\
\mathsf{MovieActor} &\sqsubseteq \mathsf{Actor}\\
\mathsf{SeriesActor} &\sqsubseteq \neg\mathsf{MovieActor}\\[1ex]
\mathsf{Staff} &\sqsubseteq \exists ssn\\
\exists ssn^- &\sqsubseteq \mathtt{xsd{:}int}\\
&(\textbf{funct}\ ssn)\\[1ex]
\exists\mathsf{actsIn} &\sqsubseteq \mathsf{MovieActor}\\
\exists\mathsf{actsIn}^- &\sqsubseteq \mathsf{Movie}\\
\mathsf{MovieActor} &\sqsubseteq \exists\mathsf{actsIn}\\
\mathsf{Movie} &\sqsubseteq \exists\mathsf{actsIn}^-\\
\mathsf{actsIn} &\sqsubseteq \mathsf{playsIn}\\
&\cdots
\end{aligned}
$$

unibz

The OWL 2 QL ontology language

- **OWL 2 QL** is one of the three standard sub-languages of the very expressive standard ontology language OWL 2. [W3C Rec. 2012]
- It is considered a lightweight ontology language:
  - controlled expressive power
  - efficient inference
- Optimized for accessing large amounts of data
  - Queries over the ontology can be rewritten into SQL queries over the underlying relational database (First-order rewritability).
  - Logical consistency of ontology and data can also be checked by executing SQL queries over the underlying database.

Constructs of OWL 2 QL

In an OWL 2 QL ontology, one can express knowledge about the classes and properties in the domain of interest by means of various types of assertions.

Assertion type	DL syntax	OWL syntax
Subclass assertion	$\mathsf{MovieActor} \sqsubseteq \mathsf{Actor}$	`:MovieActor` **`rdfs:subClassOf`** `:Actor .`
Class disjointness	$\mathsf{Actor} \sqsubseteq \neg\mathsf{Movie}$	`:Actor` **`owl:disjointWith`** `:Movie .`
Domain of a property	$\exists\mathsf{actsIn} \sqsubseteq \mathsf{MovieActor}$	`:actsIn` **`rdfs:domain`** `:MovieActor .`
Range of a property	$\exists\mathsf{actsIn}^- \sqsubseteq \mathsf{Movie}$	`:actsIn` **`rdfs:range`** `:Movie .`
Subproperty assertion	$\mathsf{actsIn} \sqsubseteq \mathsf{playsIn}$	`:actsIn` **`rdfs:subPropertyOf`** `:playsIn .`
Inverse properties	$\mathsf{actsIn} \equiv \mathsf{hasActor}^-$	`:actsIn` **`owl:inverseOf`** `:hasActor .`
Mandatory participation	$\mathsf{MovieActor} \sqsubseteq \exists\mathsf{actsIn}$	**`owl:someValuesFrom`** in superclass expression

Syntax and semantics of OWL 2 QL KBs

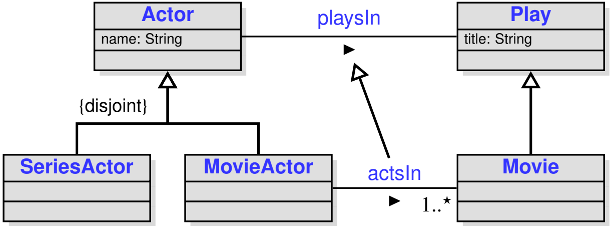
Axiom type	OWL Syntax	DL Syntax	Semantics
Membership (class)	<a> **rdf:type** <C>	$C(a)$	$a \in C^{\mathcal{I}}$
Membership (data property)	<a> :A <l>	$A(a, \ell)$	$(a, \ell) \in A^{\mathcal{I}}$
Membership (object property)	<a1> :P <a2>	$P(a_1, a_2)$	$(a_1, a_2) \in P^{\mathcal{I}}$
Subclass assertion	C1 **rdfs:subClassOf** C2	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
Class disjointness	C1 **owl:disjointWith** C2	$C_1 \sqsubseteq \neg C_2$	$C_1^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} - C_2^{\mathcal{I}}$
Domain of a property	P **rdfs:domain** C1	$\exists P \sqsubseteq C_1$	$\{x \mid \exists y.(x, y) \in P^{\mathcal{I}}\} \subseteq C_1^{\mathcal{I}}$
Range of a property	P **rdfs:range** C2	$\exists P^- \sqsubseteq C_2$	$\{y \mid \exists x.(x, y) \in P^{\mathcal{I}}\} \subseteq C_2^{\mathcal{I}}$
Mandatory participation	using **owl:someValuesFrom**	$C \sqsubseteq \exists R$	$C^{\mathcal{I}} \subseteq \exists R^{\mathcal{I}}$
Subproperty assertion	P1 **rdfs:subPropertyOf** R2	$P_1 \sqsubseteq R_2$	$P_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
Property disjointness	P1 **owl:propertyDisjointWith** P2	$P_1 \sqsubseteq \neg P_2$	$P_1^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) - P_2^{\mathcal{I}}$
Inverse property	P2 **owl:inverseOf** P1	$P_1 \equiv P_2^-$	$P_1^{\mathcal{I}} = \{(y, x) \mid (x, y) \in P_2^{\mathcal{I}}\}$

- We have used R to denote either an object property P or the inverse P^- of an object property.
- We have listed the axioms involving object properties, but OWL 2 QL allows for analogous axioms involving data properties.

Representing OWL 2 QL ontologies as UML class diagrams/ER schemas

There is a close correspondence between OWL 2 QL and conceptual modeling formalisms, such as UML class diagrams and ER schemas [Berardi, C. & De Giacomo 2005; Bergamaschi & Sartori 1992; Borgida 1995; C., Lenzerini & Nardi 1999; Lenzerini & Nobili 1990; Queralt et al. 2012].

`:MovieActor rdfs:subClassOf :Actor`	subclass
`:SeriesActor owl:disjointWith :MovieActor`	disjointness
`:actsIn rdfs:domain :MovieActor`	domain
`:actsIn rdfs:range :Movie`	range
`:actsIn rdfs:subPropertyOf :playsIn`	sub-association
`... owl:someValuesFrom ...`	mandatory participation



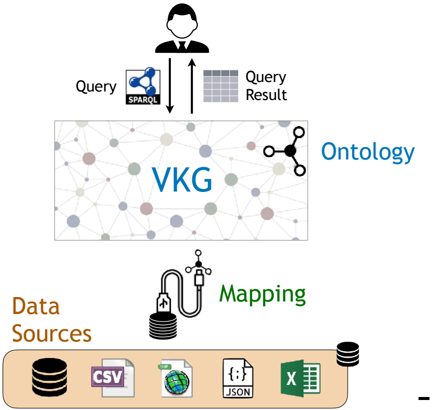
In fact, to visualize an OWL 2 QL ontology, we can use standard UML class diagrams.

Use of mappings

In the VKG framework, the mapping encodes how the data in the sources should be used to create the Virtual Knowledge Graph, which is formulated in the vocabulary of the ontology.

VKG defined from the mapping and the data.

- Queries are answered with respect to the ontology and the data of the VKG.
- The data of the VKG is not materialized (it is virtual!).
- Instead, the information in the ontology and the mapping is used to translate queries over the ontology into queries formulated over the sources.



Note: The graph is **always up to date** wrt the data sources.

Mapping language

The mapping consists of a set of assertions of the form

$$Q_{sql}(\vec{x}) \rightsquigarrow \textbf{iri}(\vec{x})\ \texttt{rdf:type}\ C$$
$$Q_{sql}(\vec{x}) \rightsquigarrow \textbf{iri}_1(\vec{x})\ prop\ \textbf{iri}_2(\vec{x})$$

- $Q_{sql}(\vec{x})$ is the source query expressed in SQL,
- The right hand side is the target, consisting of a triple pattern involving an ontology class C or a (data or object) property $prop$, and making use of the answer variables $\vec{x}$ of the SQL query.

Intuition behind the mapping

The answers returned by the SQL query in the left-hand side are used to create the objects (and values) that populate the class / property in the right-hand side.

Note: The mapping contains **iri-templates** of the form **iri**$(\vec{x})$, which are used to transform values retrieved from the database into objects of the VKG (thus solving the so-called impedance mismatch).

Mapping language – Example

Ontology $\mathcal{O}$:

```
:actsIn rdfs:domain :MovieActor .
:actsIn rdfs:range :Movie .
:Movie rdfs:subClassOf :Play .
:title rdfs:domain :Play .
:title rdfs:range xsd:string .
...
```

Mapping $\mathcal{M}$:

m_1:
```
SELECT mcode, mtitle FROM MOVIE
WHERE type = "m"
  ⇝ :m-{mcode} rdf:type :Movie .
    :m-{mcode} :title {mtitle} .
```

m_2:
```
SELECT M.mcode, A.acode FROM MOVIE M, ACTOR A
WHERE M.mcode = A.pcode AND M.type = "m"
  ⇝ :a-{acode} :actsIn :m-{mcode} .
```

Database $\mathcal{D}$:

MOVIE				
mcode	*mtitle*	*myear*	*type*	···
511	The Matrix	1999	m	···
824	Altered Carbon	2018	s	···
227	Blade Runner	1982	m	···

ACTOR			
pcode	*acode*	*aname*	···
511	43	K. Reeves	···
511	57	C.A. Moss	···
227	61	H. Ford	···

The mapping $\mathcal{M}$ applied to database $\mathcal{D}$ generates the (virtual) knowledge graph $\mathcal{V} = \mathcal{M}(\mathcal{D})$:

```
:m-511 rdf:type :Movie .          :m-511 :title "The Matrix" .
:m-227 rdf:type :Movie .          :m-227 :title "Blade Runner" .
:a-43 :actsIn :m-511 .     :a-57 :actsIn :m-511 .     :a-61 :actsIn :m-227 .
```

Query answering in VKGs

In VKGs, we want to answer queries formulated over the ontology, by using the data provided by the data sources through the mapping.

- The ontology contains **domain knowledge** that can be used to enrich answers.

  Example: Suppose that our data contains a-43 among the MovieActors, and that the ontology states that each MovieActor is an Actor.
  If we ask for all Actors, we should return also a-43, considering both the data and the knowledge in the ontology.

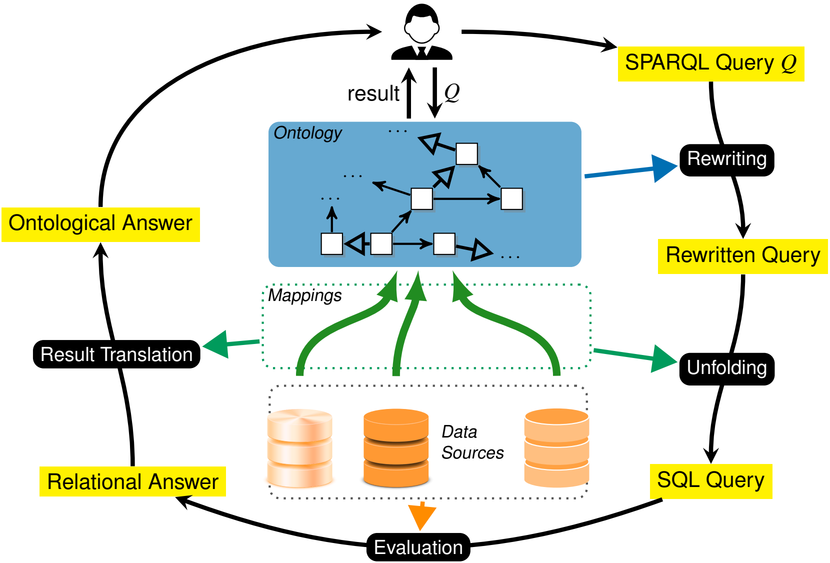
- The **mapping** encodes the information of how to translate a query over the ontology into a query over the **database**.

A VKG query answering engine has to take into account all these types of information.

Query answering by query rewriting

unibz

Query answering by query rewriting



Outline

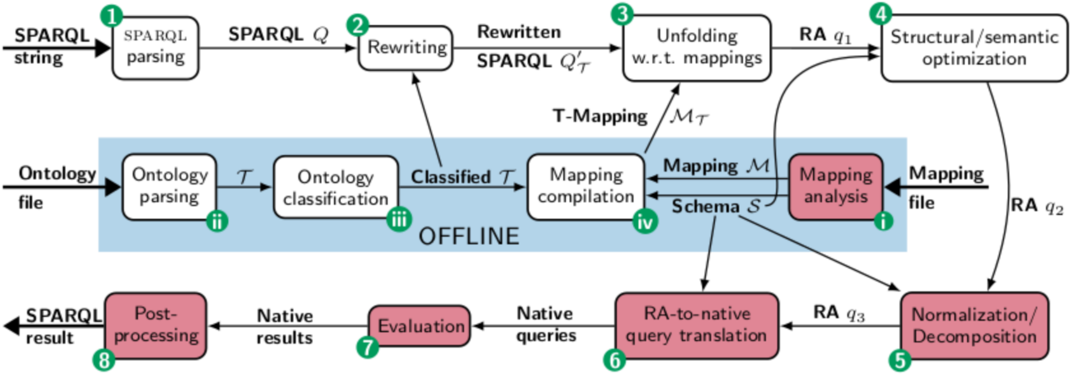
unibz

The *Ontop* system [C., Cogrel, et al. 2017, Semantic Web J.], [Xiao, Lanti, et al. 2020, ISWC]

https://ontop-vkg.org/

- State-of-the-art VKG system.
- Addresses the key challenges in query answering of scalability and performance.
- Compliant with all relevant Semantic Web standards:
  RDF, RDFS, OWL 2 QL, R2RML, SPARQL, and GeoSPARQL.
- Supports all major relational DBMSs:
  Oracle, DB2, MS SQL Server, Postgres, MySQL, Teiid, Dremio, Denodo, etc.
- Open-source and released under Apache 2 license.

Query answering in *Ontop*



unibz

The *Ontopic* spinoff of unibz

ONTOPIC

https://ontopic.ai/

Funded in April 2019 as the first spin-off of the Free University of Bozen-Bolzano.

- Ontopic Studio
  - Ensures scalability, reliability, and cost-efficiency at design and runtime of VKG solutions.
  - Strong focus on usability
- Ontopic Server
  - VKG Server functionalities
  - Deployment of SPARQL endpoints
  - SQL connector: Deployment of JDBC functionality over VKGs
- Technical services
  - Technical support for Ontop and Ontopic tools
  - Customized developments

unibz

Outline

unibz

Applications of the VKG approach

Adopted in many academic and industrial use cases from different application areas.
See also [Xiao, Ding, et al. 2019, Data Intelligence].

- Industry 4.0
  - Ability to deal with data coming from different vendors, or with historical heterogeneous data.

  Examples: Equinor, Siemens, Bosch

- Analytical processing / Business Intelligence
  - Combine internal data, manual processes (e.g., Excel), and external data.
  - Data privacy issues / GDPR: we need to avoid data copies

  Examples: Toscana Open Research, a large European university, a large TLC company

- Geospatial data
  - GeoSPARQL over PostGIS

  Examples: LinkedGeoData.org, South Tyrolean Open Data Hub

unibz

Who provides the ontology?

- Designing an ontology is not an easy task.
- In many domains (e.g., the biomedical one) ontologies are developed independently by trained experts and are already available to be re-used.
- Having "standardized ontologies" enables interoperability across different data sources.
- However, ontology design is a well investigated task, and methodologies and supporting tools are readily available. See, e.g.,
  - the series of *Workshops on Ontology Design Patterns* [http://ontologydesignpatterns.org/];
  - the OntoClean methodology for ontology analysis based on formal, domain-independent properties of classes [Guarino & Welty 2009].

unibz

Who provides the mappings?

VKG mappings:

- Map complex queries to complex queries – cf. GLAV relational mappings [Lenzerini 2002].
- Overcome the abstraction mismatch between relational data and target ontology.
- Are inherently more sophisticated than mappings for schema matching [Rahm & Bernstein 2001] and ontology matching [Euzenat & Shvaiko 2007].

As a consequence:

- Management of VKG mappings is an essentially manual effort that is **labor-intensive** and **error-prone**.
- Requires highly-skilled professionals [Spanos, Stavrou & Mitrou 2012].
- Writing mappings is challenging in terms of semantics, correctness, and performance.

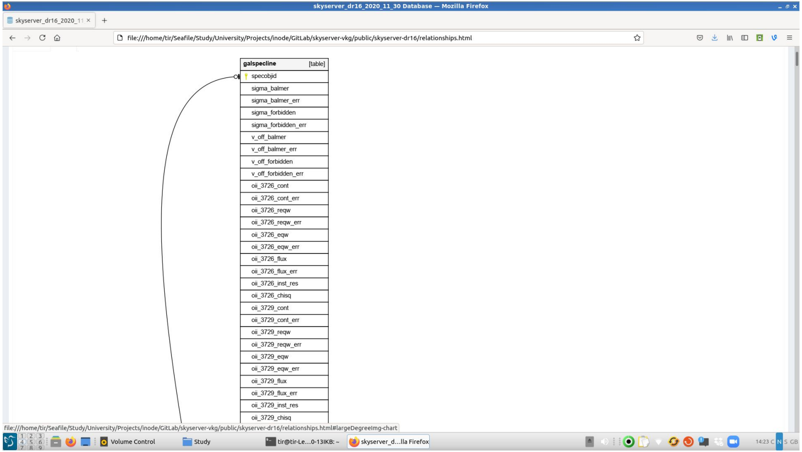
Designing and managing mappings is the most critical bottleneck
for the adoption of the VKG approach.

unibz

Who provides the mapping?

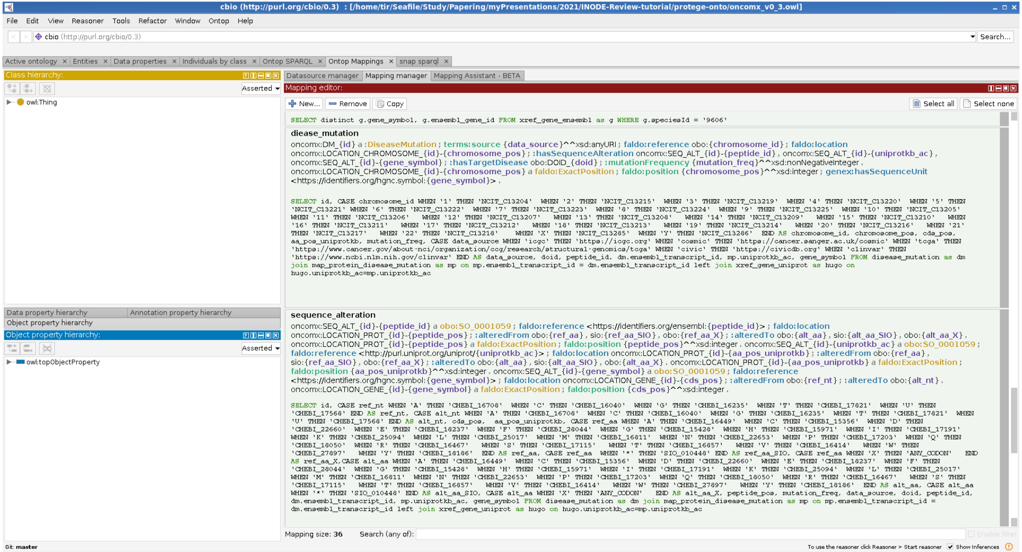
Writing mappings manually is a **time-consuming** and **error-prone** task.

Who provides the mapping?



unibz

Who provides the mapping?



Mapping patterns

In relational database design, **well-established conceptual modeling principles** and **methodologies** are usually employed.

- The resulting schema should suitably reflects the application domain at hand.
- This design phase relies on semantically-rich representations such as ER diagrams.
- However, these representations, typically:
  - get lost during deployment, since they are not conveyed together with the database itself, or
  - quickly get outdated due to continuous adjustments triggered by changing requirements.

Key Observation

While the relational model may be semantically-poor with respect to ontological models, the original semantically-rich design of the application domain **leaves recognizable footprints** that can be converted into ontological mapping patterns.

unibz

VKG mapping patterns

Several approaches and tools that deal with the problem of extracting a KG from a relational data source have been proposed, several of them based on mapping patterns.

However, to the best of our knowledge:

- There is no comprehensive approach for KG mapping patterns exploiting all of:
  - the relational schema with its constraints
  - extensional data stored in the DB
  - the domain knowledge that is encoded in ontology axioms
  - the conceptual schema at the basis of the relational schema
- Only a few come with a systematic categorization of the mappings that they produce.
- None of them have drawn an explicit and precise connection between their outputs and conceptual modeling practices found in DB design.
- None of them attempts an analysis over real-world scenarios

Catalog of mapping patterns

We build on well-established methodologies and patterns studied in:

- data management – e.g., W3C Direct Mapping Specification [Arenas et al. 2012] and extensions
- data analysis – e.g., algorithms for discovering dependencies, and
- conceptual modeling

In specifying each pattern, we consider:

- the three components of a VKG specification: DB schema, ontology, mapping between the two;
- the conceptual schema of the domain of interest;
- underlying data, when available.

For the moment, we do not fix what is given as input and what is produced as output, but we simply describe how the elements relate to each other, on a per-pattern basis.

unibz

Two major groups of mapping patterns [C., Gal, et al. 2023, DKE]

Schema-driven patterns

Are shaped by the structure of the DB schema and its explicit constraints.

Data-driven patterns

- Consider also constraints emerging from specific configurations of the data in the DB.
- For each schema-driven pattern, we identify a data-driven version:
  The constraints over the schema are not explicitly specified, but hold in the data.
- We provide also data-driven patterns that do not have a schema-driven counterpart.

- We use also additional semantic information from the ontology $\rightsquigarrow$ **Pattern modifiers**
- Some patterns come with **views over the DB-schema**:
  - Views reveal structures over the DB-schema, when the pattern is applied.
  - Views can be used to identify the applicability of further patterns.

Constraints on the data

When defining the mapping patterns, we consider the traditional types of DB constraints:

- **Primary key constraint**: $T(\underline{\mathbf{K}}, \mathbf{A})$
- **Key constraint**: $\text{key}_T(\mathbf{K})$
- **Foreign key constraint**: $T_1[\mathbf{A}] \subseteq T_2[\mathbf{K}]$, where **K** is a (typically primary) key of relation T_2.
  We use the notation:

$$T_1\,(\mathbf{A}, \mathbf{B}) \longrightarrow T_2\,(\underline{\mathbf{K}}, \mathbf{A}')$$

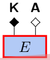
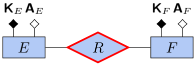
Note: We use normal font (e.g., A) for single attributes, and boldface for sets of attributes (e.g., **A**).

unibz

Fragment of schema-driven patterns

Primary Key

Source part of the mapping assertion

E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY
Schema Entity (SE)			
	$T_E(\underline{\mathbf{K}}, \mathbf{A})$	s: T_E t: $C_E(\mathbf{iri}_E(\mathbf{K}))$, $\{d_A(\mathbf{iri}_E(\mathbf{K}), A)\}_{A \in \mathbf{K} \cup \mathbf{A}}$	$\{\exists d_A \sqsubseteq C_E\}_{A \in \mathbf{K} \cup \mathbf{A}}$
Schema Relationship (SR)			
	$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $T_F(\underline{\mathbf{K}_F}, \mathbf{A}_F)$ $T_R(\underline{\mathbf{K}_{RE}, \mathbf{K}_{RF}})$	s: T_R t: $p_R(\mathbf{iri}_E(\mathbf{K}_{RE}), \mathbf{iri}_F(\mathbf{K}_{RF}))$	$\exists p_R \sqsubseteq C_E$ $\exists p_R^- \sqsubseteq C_F$

Target part of the mapping assertion

Domain Axiom

In case of $(\_, 1)$ cardinality on role R_E (resp., R_F), the primary key for T_R is restricted to the attributes $\mathbf{K}_{RE}$ (resp., $\mathbf{K}_{RF}$).

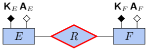
Foreign Key

iri Template Function

Range Axiom

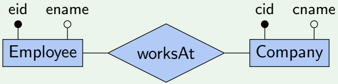
unibz

Example: Schema Relationship Pattern

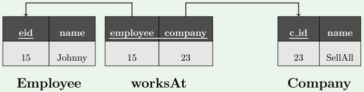
E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY
Schema Entity (SE)			
	$T_E(\underline{\mathbf{K}}, \mathbf{A})$	s: T_E t: $C_E(\mathbf{iri}_E(\mathbf{K}))$, $\{d_A(\mathbf{iri}_E(\mathbf{K}), A)\}_{A \in \mathbf{K} \cup \mathbf{A}}$	$\{\exists d_A \sqsubseteq C_E\}_{A \in \mathbf{K} \cup \mathbf{A}}$
Schema Relationship (SR)			
	$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $T_F(\underline{\mathbf{K}_F}, \mathbf{A}_F)$ $T_R(\underline{\mathbf{K}_{RE}, \mathbf{K}_{RF}})$	s: T_R t: $p_R(\mathbf{iri}_E(\mathbf{K}_{RE}), \mathbf{iri}_F(\mathbf{K}_{RF}))$	$\exists p_R \sqsubseteq C_E$ $\exists p_R^- \sqsubseteq C_F$

In case of $(\_, 1)$ cardinality on role R_E (resp., R_F), the primary key for T_R is restricted to the attributes $\mathbf{K}_{RE}$ (resp., $\mathbf{K}_{RF}$).

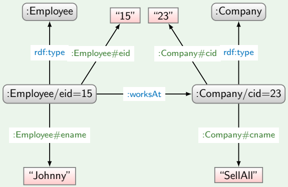
Conceptual



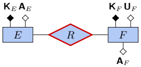
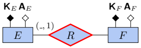
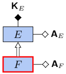
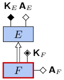
DB Schema



RDF (data only)

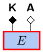


The other schema-driven patterns

E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY
Schema Relationship with Identifier Alignment (SRa)			
	$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $T_F(\underline{\mathbf{K}_F}, \mathbf{U}_F, \mathbf{A}_F)$ $T_R(\underline{\mathbf{K}_{RE}, \mathbf{U}_{RF}})$ $\mathsf{key}_{R_F}(\mathbf{U}_F)$	s: $T_R \bowtie_{\mathbf{U}_{RF}=\mathbf{U}_F} T_F$ t: $p_R(\mathsf{iri}_E(\mathbf{K}_{RE}), \mathsf{iri}_F(\mathbf{K}_F))$	$\exists p_R \sqsubseteq C_E$ $\exists p_R^- \sqsubseteq C_F$
In case of $(\_, 1)$ cardinality on role R_E (resp., R_F), the primary key for T_R is restricted to the attributes $\mathbf{K}_{RE}$ (resp., $\mathbf{U}_{RF}$).			
Schema Relationship with Merging (SRm)			
	$T_F(\underline{\mathbf{K}_F}, \mathbf{A}_F)$ $T_E(\underline{\mathbf{K}_E}, \mathbf{K}_{EF}, \mathbf{A}_E)$	s: T_E t: $p_{EF}(\mathsf{iri}_E(\mathbf{K}_E), \mathsf{iri}_F(\mathbf{K}_{EF}))$	$\exists p_{EF} \sqsubseteq C_E$ $\exists p_{EF}^- \sqsubseteq C_F$
Schema Hierarchy (SH)			
	$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $T_F(\underline{\mathbf{K}_{FE}}, \mathbf{A}_F)$	s: T_F t: $C_F(\mathsf{iri}_E(\mathbf{K}_{FE}))$, $\{d_A(\mathsf{iri}_E(\mathbf{K}_{FE}), A)\}_{A \in \mathbf{A}_F}$	$C_F \sqsubseteq C_E$ $\{\exists d_A \sqsubseteq C_F\}_{A \in \mathbf{A}_F}$
Schema Hierarchy with Identifier Alignment (SHa)			
	$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $\mathsf{key}_{T_F}(\mathbf{U}_F)$ $T_F(\underline{\mathbf{K}_F}, \mathbf{U}_F, \mathbf{A}_F)$ — $T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $\mathsf{key}_{V_F}(\mathbf{K}_F)$ $V_F(\mathbf{K}_F, \underline{\mathbf{U}_F}, \mathbf{A}_F) = T_F$	s: T_F t: $C_F(\mathsf{iri}_E(\mathbf{U}_F))$, $\{d_A(\mathsf{iri}_E(\mathbf{U}_F), A)\}_{A \in \mathbf{K}_F \cup \mathbf{A}_F}$	$C_F \sqsubseteq C_E$ $\{\exists d_A \sqsubseteq C_F\}_{A \in \mathbf{K}_F \cup \mathbf{A}_F}$
In this pattern, the "alignment" is meant to align the primary identifier used in the child entity to the primary identifier used in the parent entity. ...			

unibz

A “data”-driven pattern

E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY
Clustering Entity to Class (CE2C)			
 $\mathbf{B} \subseteq \mathbf{K} \cup \mathbf{A}$, *partition*$_{\mathcal{D}}(\mathbf{B}, E)$	$T_E(\mathbf{K}, \mathbf{A})$ unique$_{T_E}(\mathbf{K})$ $\mathbf{B} \subseteq \mathbf{K} \cup \mathbf{A}$ *partition*$_{\mathcal{D}}(\mathbf{B}, E)$ $\{V_{E_\mathbf{v}}(\mathbf{K}, \mathbf{A}) = \sigma_{\mathbf{B}=\mathbf{v}}(T_E)\}_{\mathbf{v} \in \pi_\mathbf{B}(T_E)}$	$\{s : \sigma_{\mathbf{B}=\mathbf{v}}(T_E)$ $t : C_E^\mathbf{v}(\mathbf{iri}_E(\mathbf{K}))\}_{\mathbf{v} \in \pi_\mathbf{B}(T_E)}$	$\{C_E^\mathbf{v} \sqsubseteq C_E\}_{\mathbf{v} \in \pi_\mathbf{B}(T_E)}$

eid	name	gender	
1	Johnny	M	Male
2	Elena	F	Female
3	Ann	F	Female
4	Paul	M	Male

Employee

unibz

Design scenarios for VKG mapping patterns

Depending on what information is available, we can consider different design scenarios where the patterns can be applied:

1. **Debugging of a VKG specification** that is already in place.
2. **Conceptual schema reverse engineering** for a DB that represents the domain of interest by using a given full VKG specification.
3. **Mapping bootstrapping** for a given DB and ontology that miss the mappings relating them.
4. **Ontology + mapping bootstrapping** from a given DB with constraints, and possibly a conceptual schema.
5. **VKG bootstrapping**, where the goal is to set up a full VKG specification from a conceptual schema of the domain.

unibz

MPBoot mapping bootstrapper

We are currently developing the **MPBoot mapping bootstrapper**, that relies on mapping patterns:

- Current version supports the **Direct Mapping W3C Specification**
- Enriched with various configuration options:
  - selection of elements (tables, attributes) to actually map
  - renaming of elements
  - treatment of null values in tables
  - treatment of tables without primary keys
- Partial support for schema-driven mapping patterns:
  - generation of domain and range assertions for properties (*Schema Relationship Pattern*)
  - generation of class and property hierarchies (*Schema Hierarchy Pattern*)
- Extension to fully support schema driven patterns is ongoing.
- Extension to consider also data driven patterns is starting now [PhD of Marco Di Panfilo].

Outline

unibz

Further research directions

Extensions of the VKG framework:

- Ontology-based update [Wandji & C. 2024, RuleML+RR] $\rightsquigarrow$ Ontology-based Data Management
- Privacy issues [Baura, C. & Marconi 2024, JOWO] – Based on Controlled Query Evaluation
- Support for additional types of data:
  - geospatial data: native support in *Ontop*, but we are extending it
  - temporal data
  - graph structured data and graph databases – OnTeGra project with TU Vienna and Virtual Vehicles
  - raster data: *OntoRaster* framework [Ghosh et al. 2024, RuleML+RR] and CRiMA project
- Support for multiple, heterogeneous data sources – Ontology-based Data Federation [Gu, C., et al. 2023, SEBD], [Gu, Corcoglioniti, et al. 2024, SWJ]

Supporting technologies are needed to ease the adoption of VKGs:

- Techniques for (semi-)automatic extraction/learning of ontology axioms and mapping assertions [C., Gal, et al. 2023, DKE].
- Techniques and tools for efficient management of mappings and ontology axioms, to support design, maintenance, and evolution $\rightsquigarrow$ Ontopic Studio
- User-friendly ontology querying modalities (graphical languages, natural language queries).

unibz

Conclusions

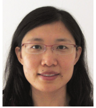
- VKGs are by now a mature technology to address the challenges in data access and integration.
- The technology is general purpose, and it can be tailored towards specific domains, relying also on standard ontologies.
- Several foundational and practical challenges still remain towards a wider adoption of the VKG technology.

Thank you!

A great thank you to all my collaborators

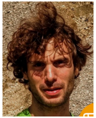
Elena Botoeva

Julien Corman



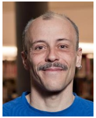
Linfang Ding

Elem Güzel



Davide Lanti

Marco Montali



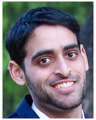
Alessandro Mosca

Mariano Rodriguez Muro

Guohui Xiao

Technion Haifa

Avigdor Gal



Roee Shraga

Birkbeck College London

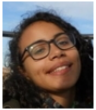
Roman Kontchakov

Vladislav Ryzhikov

Michael Zakharyaschev

Ontopic s.r.l.

Benjamin Cogrel



Sarah Komla Ebri

U. Roma “La Sapienza”

Giuseppe De Giacomo

Domenico Lembo

Maurizio Lenzerini

Antonella Poggi

Riccardo Rosati

unibz

References I

[1] Marcelo Arenas, Alexandre Bertails, Eric Prud'hommeaux & Juan Sequeda. *A Direct Mapping of Relational Data to RDF*. W3C Recommendation. Available at `http://www.w3.org/TR/rdb-direct-mapping/`. World Wide Web Consortium, Sept. 2012.

[2] Divya Baura, Diego C. & Lorenzo Marconi. "Implementing Controlled Query Evaluation in OBDA". In: *Proc. of the Joint Ontology Workshops Episode 10: The Tukker Zomer of Ontology (JOWO 2024)*. Vol. 3882. CEUR Workshop Proceedings, `http://ceur-ws.org/`. CEUR-WS.org, 2024.

[3] Daniela Berardi, Diego C. & Giuseppe De Giacomo. "Reasoning on UML Class Diagrams". In: *Artificial Intelligence* 168.1–2 (2005), pp. 70–118.

[4] Konstantina Bereta, Guohui Xiao & Manolis Koubarakis. "Ontop-spatial: Ontop of Geospatial Databases". In: *J. of Web Semantics* 58 (2019). DOI: `10.1016/j.websem.2019.100514`.

[5] Sonia Bergamaschi & Claudio Sartori. "On Taxonomic Reasoning in Conceptual Design". In: *ACM Trans. on Database Systems* 17.3 (1992), pp. 385–422.

[6] Alexander Borgida. "Description Logics in Data Management". In: *IEEE Trans. on Knowledge and Data Engineering* 7.5 (1995), pp. 671–682.

unibz

References II

[7] Sebastian Brandt, Diego C., Elem Güzel Kalayci, Roman Kontchakov, Benjamin Mörzinger, Vladislav Ryzhikov, Guohui Xiao & Michael Zakharyaschev. "Two-Dimensional Rule Language for Querying Sensor Log Data: A Framework and Use Cases". In: *Proc. of the 26th Int. Symp. on Temporal Representation and Reasoning (TIME)*. Vol. 147. Leibniz Int. Proc. in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019, 7:1–7:15. DOI: 10.4230/LIPIcs.TIME.2019.7.

[8] Sebastian Brandt, Elem Güzel Kalayci, Vladislav Ryzhikov, Guohui Xiao & Michael Zakharyaschev. "Querying Log Data with Metric Temporal Logic". In: *J. of Artificial Intelligence Research* 62 (2018), pp. 829–877.

[9] Diego C., Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro & Guohui Xiao. "Ontop: Answering SPARQL Queries over Relational Databases". In: *Semantic Web J.* 8.3 (2017), pp. 471–487. DOI: 10.3233/SW-160217.

[10] Diego C., Avigdor Gal, Davide Lanti, Marco Montali, Alessandro Mosca & Roee Shraga. "Conceptually-grounded Mapping Patterns for Virtual Knowledge Graphs". In: *Data and Knowledge Engineering* 145 (2023), p. 102157. DOI: 10.1016/j.datak.2023.102157.

References III

[11] Diego C., Davide Lanti, Ana Ozaki, Rafael Peñaloza & Guohui Xiao. "Enriching Ontology-based Data Access with Provenance". In: *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. IJCAI Org., 2019, pp. 1616–1623. DOI: 10.24963/ijcai.2019/224.

[12] Diego C., Maurizio Lenzerini & Daniele Nardi. "Unifying Class-Based Representation Formalisms". In: *J. of Artificial Intelligence Research* 11 (1999), pp. 199–240.

[13] Jérôme Euzenat & Pavel Shvaiko. *Ontology Matching*. Springer, 2007.

[14] Arka Ghosh, Albulen Pano, Guohui Xiao & Diego Calvanese. "OntoRaster: Extending VKGs with Raster Data". In: *Proc. of the 8th Int. Joint Conf. on Rules and Reasoning (RuleML+RR 2024)*. Vol. 15183. Lecture Notes in Computer Science. Springer, 2024, pp. 108–123. DOI: 10.1007/978-3-031-72407-7_9.

[15] Zhenzhen Gu, Diego C., Marco Di Panfilo, Davide Lanti, Alessandro Mosca & Guohui Xiao. "Ontology-Based Data Federation – A Framework Proposal". In: *Proc. of the 31st Italian Symp. on Advanced Database Systems (SEBD 2023)*. Vol. 3478. CEUR Workshop Proceedings, http://ceur-ws.org/. CEUR-WS.org, 2023, pp. 210–219.

References IV

[16] Zhenzhen Gu, Francesco Corcoglioniti, Davide Lanti, Alessandro Mosca, Guohui Xiao, Jing Xiong & Diego C. "A Systematic Overview of Data Federation Systems". In: *Semantic Web J.* 15.1 (2024), pp. 107–165. DOI: 10.3233/SW-223201.

[17] Nicola Guarino & Christopher A. Welty. "An Overview of OntoClean". In: *Handbook on Ontologies*. Ed. by Steffen Staab & Rudi Studer. International Handbooks on Information Systems. Springer, 2009, pp. 201–220. DOI: 10.1007/978-3-540-92673-3_9.

[18] Elem Güzel Kalayci, Sebastian Brandt, Diego C., Vladislav Ryzhikov, Guohui Xiao & Michael Zakharyaschev. "Ontology-based Access to Temporal Data with Ontop: A Framework Proposal". In: *Applied Mathematics and Computer Science* 29.1 (2019), pp. 17–30. DOI: 10.2478/amcs-2019-0002.

[19] Maurizio Lenzerini. "Data Integration: A Theoretical Perspective.". In: *Proc. of the 21st ACM Symp. on Principles of Database Systems (PODS)*. 2002, pp. 233–246. DOI: 10.1145/543613.543644.

[20] Maurizio Lenzerini & Paolo Nobili. "On the Satisfiability of Dependency Constraints in Entity-Relationship Schemata". In: *Information Systems* 15.4 (1990), pp. 453–461.

References V

[21] Anna Queralt, Alessandro Artale, Diego C. & Ernest Teniente. "OCL-Lite: Finite Reasoning on UML/OCL Conceptual Schemas". In: *Data and Knowledge Engineering* 73 (2012), pp. 1–22. DOI: 10.1016/j.datak.2011.09.004.

[22] Erhard Rahm & Philip A. Bernstein. "A Survey of Approaches to Automatic Schema Matching". In: *Very Large Database J.* 10.4 (2001), pp. 334–350.

[23] Dimitrios-Emmanuel Spanos, Periklis Stavrou & Nikolas Mitrou. "Bringing Relational Databases into the Semantic Web: A Survey". In: *Semantic Web J.* 3.2 (2012), pp. 169–209.

[24] Romuald Esdras Wandji & Diego C. "Ontology-Based Update in Virtual Knowledge Graphs via Schema Mapping Recovery". In: *Proc. of the 8th Int. Joint Conf. on Rules and Reasoning (RuleML+RR 2024)*. Vol. 15183. Lecture Notes in Computer Science. Springer, 2024, pp. 59–74. DOI: 10.1007/978-3-031-72407-7_6.

[25] Guohui Xiao, Diego C., Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati & Michael Zakharyaschev. "Ontology-Based Data Access: A Survey". In: *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. IJCAI Org., 2018, pp. 5511–5519. DOI: 10.24963/ijcai.2018/777.

unibz

References VI

[26] Guohui Xiao, Linfang Ding, Benjamin Cogrel & Diego C. "Virtual Knowledge Graphs: An Overview of Systems and Use Cases". In: *Data Intelligence* 1.3 (2019), pp. 201–223. DOI: 10.1162/dint_a_00011.

[27] Guohui Xiao, Davide Lanti, Roman Kontchakov, Sarah Komla-Ebri, Elem Güzel-Kalayci, Linfang Ding, Julien Corman, Benjamin Cogrel, Diego C. & Elena Botoeva. "The Virtual Knowledge Graph System Ontop". In: *Proc. of the 19th Int. Semantic Web Conf. (ISWC)*. Vol. 12507. Lecture Notes in Computer Science. Springer, 2020, pp. 259–277. DOI: 10.1007/978-3-030-62466-8_17.

Outline

unibz

Outline

Data federation

- We have multiple, heterogeneous data sources (RDB, NoSQL DB, CSV sources ...) to be federated.
- Only requirement: each data source supports a query language.
- Each source S_i comes with a function transforming its (possibly, non-relational) schema into a relational schema Σ_i.
- We call $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$ the **VDB schema** of the federation.
- Operations in a data federation system (joins and unions):
  - **Local operations** are performed within a source.
  - **Federated operations** are performed at the level of the federation system.

unibz

Ontology-based data federation (OBDF)

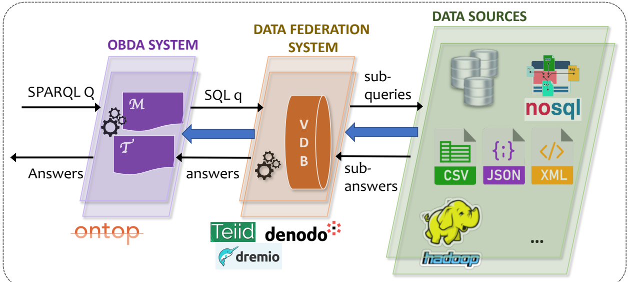
Formalization

We are given:

- a set $\mathbb{S}$ of data sources with VDB schema Σ,
- an ontology $\mathcal{T}$,
- a set $\mathcal{M}$ of mappings from Σ to $\mathcal{T}$.

An **OBDF specification** is a triple $\mathcal{F} = \langle \mathcal{T}, \mathcal{M}, \Sigma \rangle$.
An **OBDF instance** is a pair $\langle \mathcal{F}, \mathbb{D} \rangle$, where $\mathbb{D}$ is a set of database instances compliant with Σ.

Query answering in OBDF



Source hints

Given an instance $\mathbb{D}$ of Σ, by analyzing source mappings (specifically, IRI-templates), we can precompute **source hints**, i.e., meta-information to be exploited for query optimization in OBDF:

Hints of type 1: Empty federated join $(FJ =_{\mathbb{D}} \emptyset)$

A federated join expression FJ is an **empty federated join w.r.t.** $\mathbb{D}$, if $ans(FJ, \mathbb{D}) = \emptyset$.

Hints of type 2: Containment Redundancy $(A \subseteq_{\mathbb{D}} B)$

An algebra expressions A is **data-contained** in an algebra expression B, if $ans(A, \mathbb{D}) \subseteq ans(B, \mathbb{D})$.
We use $A \equiv_{\mathbb{D}} B$ to indicate that $A \subseteq_{\mathbb{D}} B$ and $B \subseteq_{\mathbb{D}} A$.

Materialized views can improve query answering performance, and can be specified in some data federation systems (but usually not directly in the sources).

Hints of type 3: VDB Schema with Views (Σ^M)

Let M be a set of view definitions, and Σ_M the relational schema of a special data source S^M materializing the views defined in M. Then we denote by Σ^M the VDB schema $\Sigma \cup \Sigma_M$.

Hint-based query optimization

Novel hint-based query optimization algorithm:

1. **Precomputation of hints**: by analyzing ontology and mappings, we precompute in advance all possible joins and unions between pairs of relations.
2. We adopt **classic optimization rules for OBDA** (self-join elimination, left distribution, ...) and novel **hint-based optimizations** (empty join elimination, ...).
3. We adopt a **cost model** based on heuristic arguments (federated joins are costly, ...).
4. We have developed a novel **query unfolding algorithm** that applies the optimization rules (considering the precomputed hints), guided by the cost model.

unibz

Outline

Geospatial extension [Bereta, Xiao & Koubarakis 2019, J. Web Sem.], [Xiao, Lanti, et al. 2020, ISWC]

Spatial data play an important role in many scenarios.

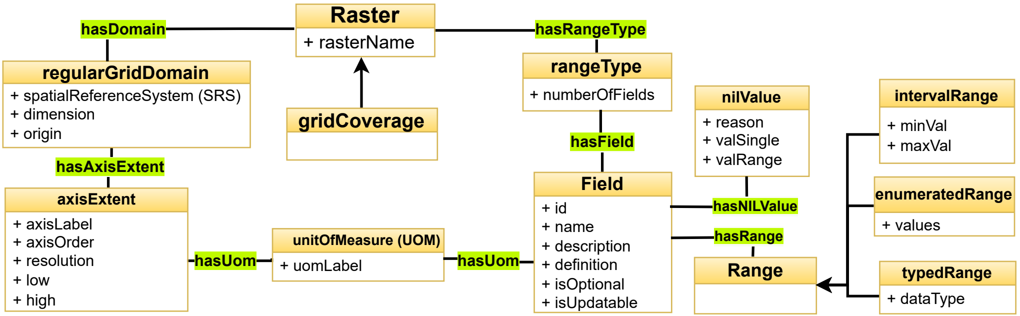
Geo-spatial extension on *Ontop*

- *Ontop* (since v4) provides full support for accessing geospatial data.
- Supports GeoSPARQL query language standardized by Open Geospatial Consortium (OGC).
- Translates GeoSPARQL functions into functions supported by PostGIS.
- Use cases: urban development, land management, disaster management.

Raster data

Raster data are data organizes as **multidimensional arrays** of values:

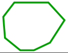
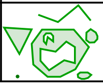
- The dimensions of an array are determined by its axes, each with an extent.
- The values in each cell of the array are determined by fields.



Raster data are efficiently managed by **dedicated database systems**, e.g., Rasdaman.

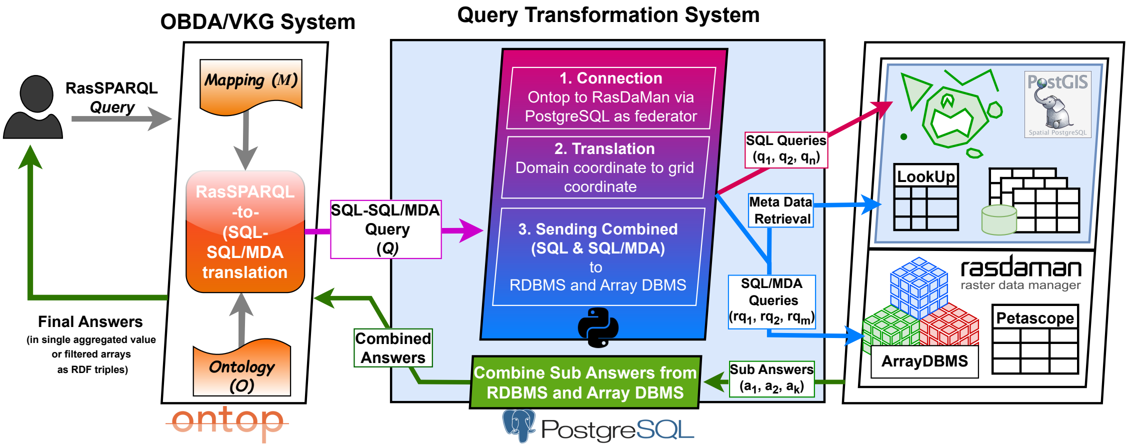
Geometry vector data

- Geographical raster data needs to be combined with geometry data.
- We consider geometries conforming to the OGC ISO 19125 OpenGIS Standard (https://www.ogc.org/standard/sfa/).
- Geometries are represented as Well-Known-Text (WKT) literals.

OGC Geometry Type	Region Geometry Illustration	Exterior & interior boundaries	OGC WKT Literal Representation (i.e. regionWkt)	Description
Point		**NA**	**POINT** (x, y)	A WKT point (e.g. pin location)
LineString		**NA**	**LINESTRING**(POINT$_{01}$,POINT$_{02}$,..... ..,POINT$_{m}$ **)**	A LineString with at least two or more points (e.g., road network)
LinearRing		**exterior = 1 interior = 0**	**LINEARRING (** POINT$_{01}$,POINT$_{02}$,.... .....,POINT$_{n}$, POINT$_{01}$**)**	An enclosed LineString where start point = end point with zero measurable area
Polygon		**exterior = 1 interior = 2**	**POLYGON** ((LINEARRING$_{01}$), [(LINEARRING)]***)**	A LinearRing with a valid measurable area with zero or more holes or interiors (e.g., countries, lake within forest)
Multi-Polygon		**exterior = 3 interior = 3**	**MULTIPOLYGON (((**POLYGON$_{01}$), [(POLYGON)]***))**	Collection of two or more Polygons with zero or more interiors or holes (e.g., scattered islands, enclaves)
Geometry-Collection		**exterior = 4 interior = 2**	**GEOMETRYCOLLECTION**([POINT*], [LINEARSTRING*], [POLYGON*], [MULTIPOLYGON*])	Heterogeneous collection of every OGC standard geometries

unibz

The *OntoRaster* framework [Ghosh et al. 2024, RuleML+RR]



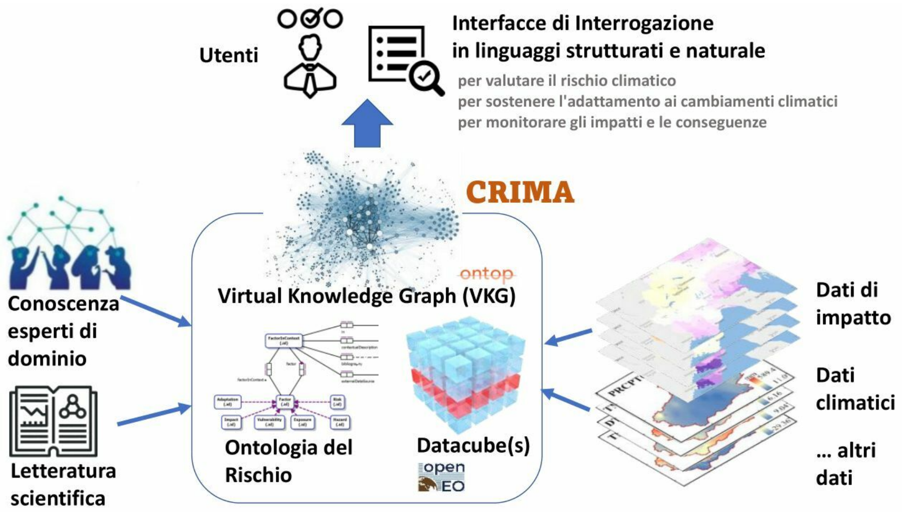
OntoRaster accepts queries expressed in *RasSPARQL*.

RasSPARQL raster functions

RasSPARQL raster functions are translated into specific functions supported by Rasdaman.

RasSPARQL function	Input arguments	Output type	PL/Python stored proc.
`rasDimension()`	`rasterName`	`xsd:string`	`query2string()`
`rasCellOp()`	`timeStamp, operator, operand, rasterName`	`xsd:string`	`query2array()`
`rasSpatialAverage()`	`timeStamp, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasSpatialMinimum()`	`timeStamp, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasSpatialMaximum()`	`timeStamp, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasTemporalAverage()`	`startTime, endTime, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasTemporalMinimum()`	`startTime, endTime, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasTemporalMaximum()`	`startTime, endTime, regionGeometry, rasterName`	`xsd:double`	`query2numeric()`
`rasClipRaster()`	`timeStamp, regionGeometry, rasterName`	`xsd:string`	`query2array()`
`rasClipRasterAnyGeom()`	`timeStamp, regionGeometry, rasterName`	`xsd:string`	`query2array()`

Raster data combined with Web Services



unibz

Outline

Provenance and explanation [C., Lanti, et al. 2019, IJCAI]

- The base version of *Ontop*, does not provide any information about how query answers are constructed.
- In many cases, we are interested in:
  - which data from which relation/source has been used to obtain an answer
  - which mappings have been activated
  - which ontology axioms have contributed to the answer
- We have developed a framework for provenance/explanation in VKGs, building on provenance semi-rings in relational databases.
- We have a prototype extension of *Ontop* that supports this framework.
- We are considering incorporation of the framework in the *Ontop* main branch requires effort.

Temporal extension [Brandt, C., et al. 2019; Brandt, Güzel Kalayci, et al. 2018; Güzel Kalayci et al. 2019]

Temporal data plays an important role in many scenarios.

- Example 1: find all drillings using the same equipment that are in two different locations with a distance longer than 200 km and within 2 months.
- Example 2: find all customers with at least 3 temporal overlapping loans within the last 5 years.

Ontop-temporal

- A prototype extension of *Ontop* for accessing temporal data.
- Can express complex temporal patterns.
- Use cases: turbine diagnoses, medical records.

We just started an FWF-Bolzano project in collaboration with *TU Wien* and *VirtualVehicles* (Graz).

unibz