Knowledge Graphs from an Industrial Perspective

Diego Calvanese

Free University of Bozen-Bolzano
Research Centre for Knowledge-based AI (KRDB)



OLYMPIA 2026 AI Data Infrastructure Summit
4–5 November 2026 – Heraklion, Crete (Greece)

The problem of data access – As faced by humans, LLMs, agents, . . .

In many contexts data management is a complex challenge:

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

unibz

The problem of data access – As faced by humans, LLMs, agents, ...

In many contexts data management is a complex challenge:

- Many different data sets are created independently.
- The data is heterogeneous in the way it is managed, 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

Challenges in data access

- **Ambiguous meaning:** terms and identifiers can have different meanings across sources.
- **Fragmented knowledge:** answering a question may require connecting information scattered across heterogeneous sources.
- **Complex information needs:** answering a question may require joins, complex conditions, or aggregation.
- **Implicit knowledge:** relevant facts may follow from domain knowledge without being explicitly stated in the retrieved information.
- **Unverifiable answers:** generated claims may lack identifiable supporting evidence.

unibz

Challenges in data access $\rightsquigarrow$ Ontologies and KGs to the rescue

- **Ambiguous meaning:** terms and identifiers can have different meanings across sources.
  $\rightsquigarrow$ Ontologies provide a shared vocabulary; KGs associate facts with identified entities.
- **Fragmented knowledge:** answering a question may require connecting information scattered across heterogeneous sources.
  $\rightsquigarrow$ A shared ontology and aligned identifiers support an integrated graph of connected entities.
- **Complex information needs:** answering a question may require joins, complex conditions, or aggregation.
  $\rightsquigarrow$ Graph queries express these requirements and delegate their evaluation to a query engine.
- **Implicit knowledge:** relevant facts may follow from domain knowledge without being explicitly stated in the retrieved information.
  $\rightsquigarrow$ Ontology-based reasoning derives answers using explicitly represented domain axioms.
- **Unverifiable answers:** generated claims may lack identifiable supporting evidence.
  $\rightsquigarrow$ Explicit facts and declarative queries allow for grounding and checking generated answers.

unibz

Extra challenges due to data dynamicity

- **Outdated information:** model knowledge and datasets may lag behind source changes.
- **Costly replication:** additional data copies require storage, transformation, and synchronization.
- **Dependence on physical schemas:** accessing databases requires understanding source-specific tables, columns, and encodings.
- **Data volumes exceeding context capacity:** transferring large datasets to an LLM for processing is expensive and often impractical.
- **Changing source structures:** schema changes can break queries and source-specific access logic used by agents.

unibz

How can we address these challenges?

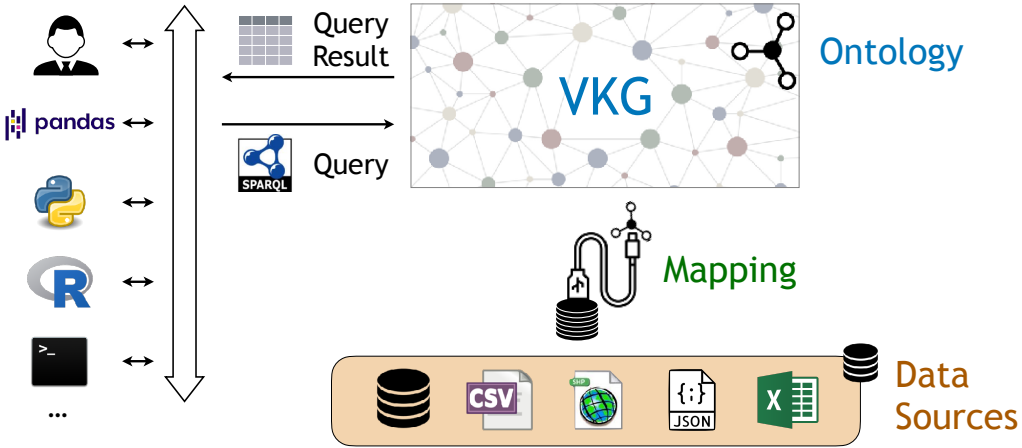
We add to KGs some additional elements:

1. We explicitly rely on an **ontology** to represent not only classes and properties of the KG, but also **domain knowledge**; hence the ontology acts as **global schema** for the data and information to manage.
2. We **map the domain ontology to the data sources**, so that they provide the data for the KG.
3. Exploit **virtualization**, i.e., the KG is not materialized, but kept virtual.

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

unibz

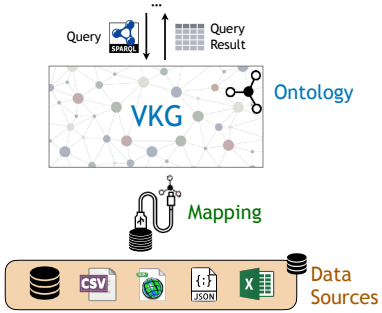
Virtual Knowledge Graph (VKG) architecture



unibz

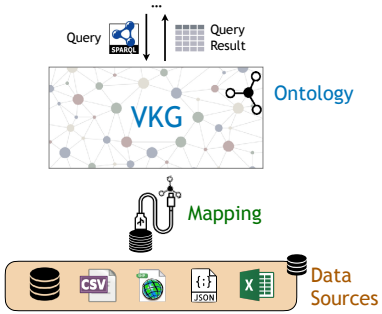
Why an ontology?

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



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.

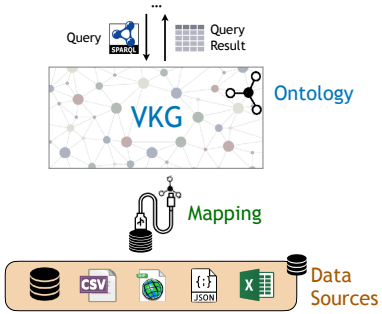
unibz

Why a Knowledge Graph for the global schema?

Traditional approaches to data management rely on the relational model.

Why a Knowledge Graph for the global schema?

Traditional approaches to data management rely on the relational model.



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 changes or new information.

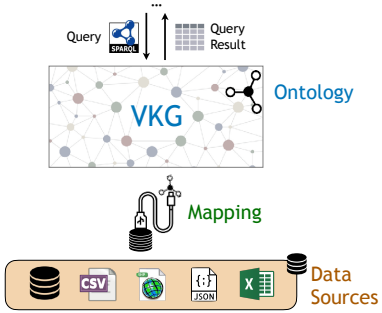
unibz

Why mappings?

Traditional approaches to data access/integration rely on mediators, which are specified through complex code.

Why mappings?

Traditional approaches to data access/integration rely 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.

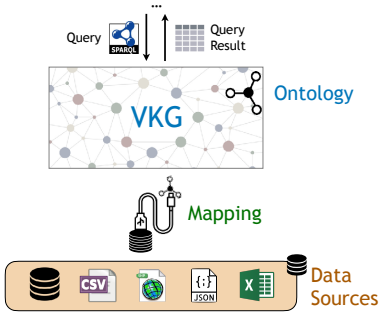
unibz

Why virtualization?

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

Why virtualization?

Materialized data access / integration relies on extract-transform-load (ETL) operations, to load data 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 existing data infrastructure and expertise.

unibz

Extra challenges due to data dynamicity

- **Outdated information:** model knowledge and datasets may lag behind source changes.
- **Costly replication:** additional data copies require storage, transformation, and synchronization.
- **Dependence on physical schemas:** accessing databases requires understanding source-specific tables, columns, and encodings.
- **Data volumes exceeding context capacity:** transferring large datasets to an LLM for processing is expensive and often impractical.
- **Changing source structures:** schema changes can break queries and source-specific access logic used by agents.

unibz

Extra challenges due to data dynamicity ⇝ Virtual KGs to the rescue

- **Outdated information:** model knowledge and datasets may lag behind source changes.
  ⇝ VKGs query the underlying sources directly, reflecting their current data at query time.
- **Costly replication:** additional data copies require storage, transformation, and synchronization.
  ⇝ A virtual graph exposes source data without requiring a separately materialized KG.
- **Dependence on physical schemas:** accessing databases requires understanding source-specific tables, columns, and encodings.
  ⇝ The VKG approach OBDA enables queries at the conceptual level.
- **Data volumes exceeding context capacity:** transferring large datasets to an LLM for processing is expensive and often impractical.
  ⇝ Query rewriting pushes supported operations to the DBMS, returning the computed results.
- **Changing source structures:** schema changes can break queries and source-specific access logic used by agents.
  ⇝ Mappings isolate sources: updating them preserves a stable ontology-level interface.

unibz

Outline

1. Virtual Knowledge Graphs (VKGs) for Data Access
2. Components of the VKG Architecture
3. Query Answering in VKGs
4. The Ontop System
5. Designing a VKG System
6. Conclusions

unibz

Outline

1 Virtual Knowledge Graphs (VKGs) for Data Access

2 Components of the VKG Architecture

3 Query Answering in VKGs

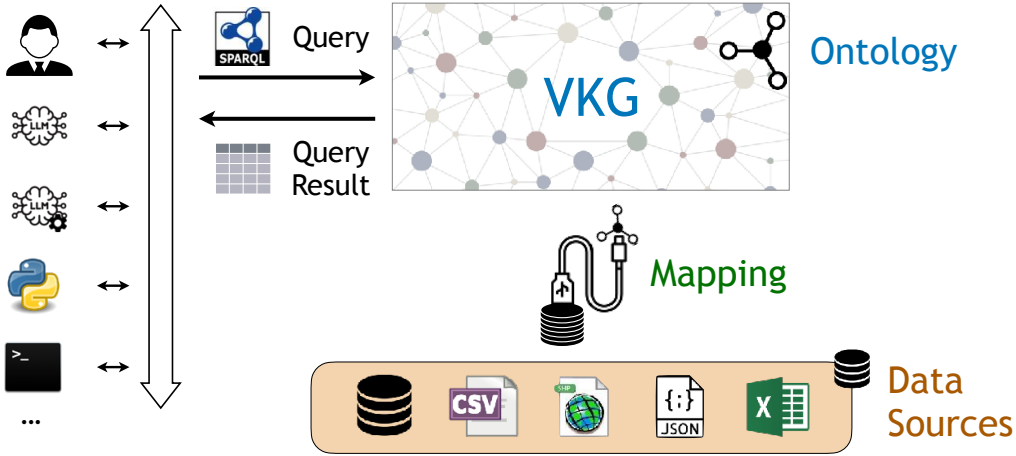
4 The Ontop System

5 Designing a VKG System

6 Conclusions

unibz

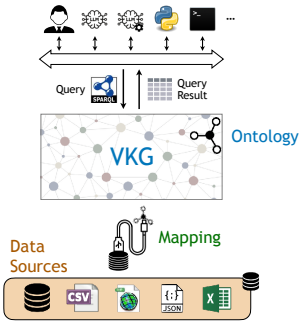
Virtual Knowledge Graph architecture



Engineering a VKG solution – Which languages?

Which are the right languages for the components of the VKG framework?

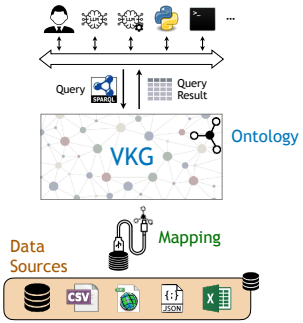
We need to consider the **tradeoff between expressive power and efficiency**, where efficiency with respect to the data is the key aspect to consider.



Engineering a VKG solution – Which languages?

Which are the right languages for the components of the VKG framework?

We need to consider the **tradeoff between expressive power and efficiency**, where efficiency with respect to the data is the key aspect to consider.



The W3C has standardized languages suitable for VKGs:

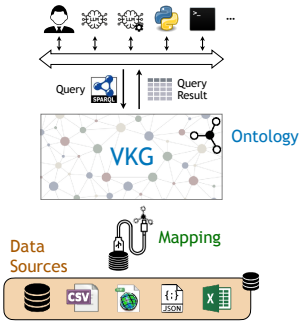
1. Knowledge graph: expressed in **RDF** [v1.1, W3C Rec. 2014]
2. Ontology $\mathcal{O}$: expressed in **OWL 2 QL** [W3C Rec. 2012]
3. Mapping $\mathcal{M}$: expressed in **R2RML** [W3C Rec. 2012]
4. Query: expressed in **SPARQL** [v1.1, W3C Rec. 2013]

unibz

Engineering a VKG solution – Which languages?

Which are the right languages for the components of the VKG framework?

We need to consider the **tradeoff between expressive power and efficiency**, where efficiency with respect to the data is the key aspect to consider.



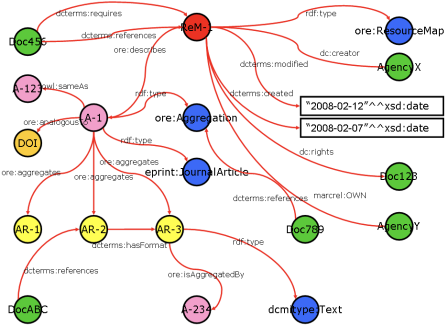
The W3C has standardized languages suitable for VKGs:

1. Knowledge graph: expressed in **RDF** [v1.1, W3C Rec. 2014] $\rightsquigarrow$ [v1.2, W3C Group Note Draft]
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** [v1.1, W3C Rec. 2013] $\rightsquigarrow$ [v1.2, W3C Working Draft]

unibz

RDF – Data is represented as a graph

The graph consists of a set of **subject-predicate-object triples** relating objects to other objects or to values, or stating membership of objects to classes.



Object property:

```
<A-1> ore:describes <ReM-1> .
```

Data property:

```
<ReM-1> :created "2008-02-07" .
```

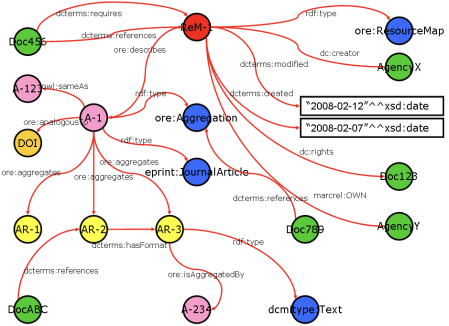
Class membership:

```
<A-1> rdf:type eprint:JournalArticle .
```

unibz

RDF – Data is represented as a graph

The graph consists of a set of **subject-predicate-object triples** relating objects to other objects or to values, or stating membership of objects to classes.



Object property:

```
<A-1> ore:describes <ReM-1> .
```

Data property:

```
<ReM-1> :created "2008-02-07" .
```

Class membership:

```
<A-1> rdf:type eprint:JournalArticle .
```

RDF 1.2 adds:

- new serialization formats (besides XML)
- nested triples, where the object of a triple is itself a triple

unibz

SPARQL query language

- Is the standard query language for RDF data. [W3C Rec. 2008 v1.0, 2013 v1.1]

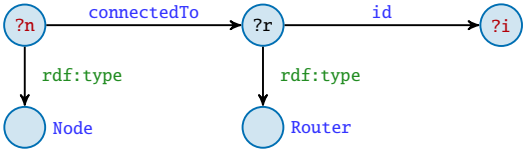
```
SELECT ?n ?i
WHERE { ?n rdf:type Node .
        ?n connectedTo ?r .
        ?r rdf:type Router .
        ?r id ?i .
      }
```

unibz

SPARQL query language

- Is the standard query language for RDF data. [W3C Rec. 2008 v1.0, 2013 v1.1]
- Core query mechanism is based on **graph matching**.

```
SELECT ?n ?i
WHERE { ?n rdf:type Node .
        ?n connectedTo ?r .
        ?r rdf:type Router .
        ?r id ?i .
      }
```

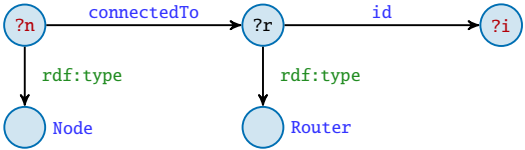


unibz

SPARQL query language

- Is the standard query language for RDF data. [W3C Rec. 2008 v1.0, 2013 v1.1]
- Core query mechanism is based on **graph matching**.

```
SELECT ?n ?i
WHERE { ?n rdf:type Node .
        ?n connectedTo ?r .
        ?r rdf:type Router .
        ?r id ?i .
      }
```



Additional language features (SPARQL 1.0):

- complex `FILTER` conditions
- `OPTIONAL`: produces a match even when part of the pattern is missing
- `UNION`: matches one of alternative graph patterns

unibz

SPARQL Evolution: v1.0 [2008] → v1.1 [2013] → v1.2 [2026]

Version	Core query capabilities	Main new language constructs
SPARQL 1.0	Basic RDF graph querying	Basic graph patterns, `FILTER`, `OPTIONAL`, `UNION`, `GRAPH`, `SELECT`, `ASK`, `CONSTRUCT`, `DESCRIBE`
SPARQL 1.1	All of 1.0, plus richer algebra and expressions	Aggregates, `GROUP BY`, `HAVING`, Property paths, Subqueries, `BIND`, `VALUES`, `MINUS`, `EXISTS` / `NOT EXISTS`, Expressions in `SELECT`, Federation via `SERVICE`
SPARQL 1.2	All of 1.1, extended to align with the RDF 1.2 data model	Nested triple patterns / triple terms, Reification and annotation syntax, `TRIPLE`, `isTRIPLE`, `SUBJECT`, `PREDICATE`, `OBJECT`, Directional language-tagged strings, `LANGDIR`, ... Further semantic / grammar refinements

unibz

SPARQL entailment regimes

Idea

Standard SPARQL evaluates BGPs by matching them against explicit triples.
An **entailment regime** extends matching so that triples entailed by the RDF graph can also contribute to query answers.

Standardized regimes

- Simple entailment (default)
- RDF entailment
- RDFS entailment
- D-entailment (datatypes)
- OWL 2 Direct Semantics
- ...

unibz

SPARQL entailment regimes

Idea

Standard SPARQL evaluates BGPs by matching them against explicit triples.
An **entailment regime** extends matching so that triples entailed by the RDF graph can also contribute to query answers.

Standardized regimes

- Simple entailment (default)
- RDF entailment
- RDFS entailment
- D-entailment (datatypes)
- OWL 2 Direct Semantics
- ...

Example

```
:SFMovie rdfs:subClassOf :Movie
:bladerunner rdf:type :SFMovie
```

Query:

```
SELECT ?x
WHERE { ?x rdf:type :Movie . }
```

Simple entailment: no answer
RDFS entailment: `?x = :bladerunner`

unibz

SPARQL entailment regimes

Idea

Standard SPARQL evaluates BGPs by matching them against explicit triples.
An **entailment regime** extends matching so that triples entailed by the RDF graph can also contribute to query answers.

Standardized regimes

- Simple entailment (default)
- RDF entailment
- RDFS entailment
- D-entailment (datatypes)
- OWL 2 Direct Semantics
- ...

Example

```
:SFMovie rdfs:subClassOf :Movie
:bladerunner rdf:type :SFMovie
```

Query:

```
SELECT ?x
WHERE { ?x rdf:type :Movie . }
```

Simple entailment: no answer
RDFS entailment: `?x = :bladerunner`

same SPARQL query + same explicit graph
different entailment regimes
$\rightsquigarrow$ **different answers**

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.

unibz

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.

- Subclass assertions — SeriesActor `rdfs:subClassOf` Actor
- Class disjointness — SeriesActor `owl:disjointWith` MovieActor
- Domain of a property — playsIn `rdfs:domain` Actor
- Range of a property — playsIn `rdfs:range` Play
- Subproperty assertions — actsIn `rdfs:subPropertyOf` playsIn
- Inverse properties — actsIn `owl:inverseOf` hasActor
- Mandatory participation to a property — ... `owl:someValuesFrom` ...

unibz

OWL 2 QL ontologies and conceptual data models

There is a close correspondence between OWL 2 QL and conceptual modeling formalisms, such as UML Class Diagrams and ER Schemas.

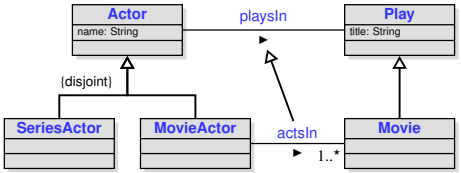
SeriesActor `rdfs:subClassOf` Actor
playsIn `rdfs:domain` Actor
playsIn `rdfs:range` Play
actsIn `rdfs:subPropertyOf` playsIn
SeriesActor `owl:disjointWith` MovieActor
... `owl:someValuesFrom` ...

unibz

OWL 2 QL ontologies and conceptual data models

There is a close correspondence between OWL 2 QL and conceptual modeling formalisms, such as UML Class Diagrams and ER Schemas.

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



In fact, to visualize an OWL 2 QL ontology, we can use standard UML Class Diagrams / ER Schemas.

unibz

The role 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.

unibz

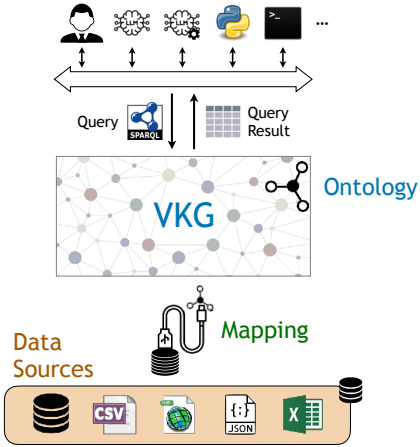
The role of mappings

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

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

- $Q_{sql}(\vec{x})$ is the source query 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 source query.

unibz

Mapping language

The **mapping** consists of a set of assertions of the form

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

- $Q_{sql}(\vec{x})$ is the source query 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 source 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.

unibz

Mapping language

The **mapping** consists of a set of assertions of the form

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

- $Q_{sql}(\vec{x})$ is the source query 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 source query.

Intuition behind the mapping

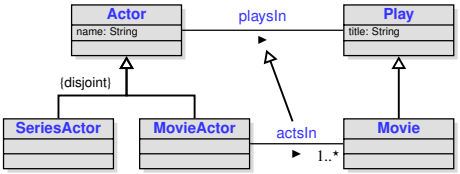
The answers returned by the SQL query in the left-hand side are used to create the objects (and values) that populate the class / property in the right-hand side.

Impedance mismatch: values in the DB vs. objects in the knowledge graph

In the target, we make use of **`iri-templates`** of the form $\mathbf{iri}(\vec{x})$, which transform database values into object identifiers (IRIs) or literals.

Mapping language – Example

Ontology $\mathcal{O}$:



Database $\mathcal{D}$:

MOVIE

mcode	*mtitle*	*myear*	*type*	···
5118	The Matrix	1999	m	···
8234	Altered Carbon	2018	s	···
2281	Blade Runner	1982	m	···

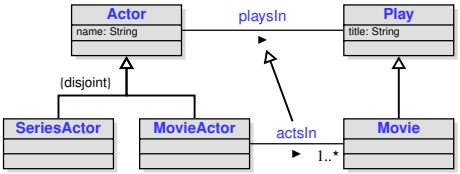
ACTOR

pcode	*acode*	*aname*	···
5118	438	K. Reeves	···
5118	572	C.A. Moss	···
2281	271	H. Ford	···

unibz

Mapping language – Example

Ontology $\mathcal{O}$:



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*	···
5118	The Matrix	1999	m	···
8234	Altered Carbon	2018	s	···
2281	Blade Runner	1982	m	···

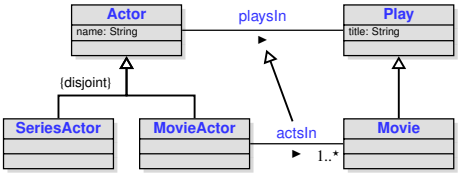
ACTOR

pcode	*acode*	*aname*	···
5118	438	K. Reeves	···
5118	572	C.A. Moss	···
2281	271	H. Ford	···

unibz

Mapping language – Example

Ontology $\mathcal{O}$:



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*	···
5118	The Matrix	1999	m	···
8234	Altered Carbon	2018	s	···
2281	Blade Runner	1982	m	···

ACTOR

pcode	*acode*	*aname*	···
5118	438	K. Reeves	···
5118	572	C.A. Moss	···
2281	271	H. Ford	···

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

:m/5118 **rdf:type** :Movie . :m/5118 :title "The Matrix" .
:m/2281 **rdf:type** :Movie . :m/2281 :title "Blade Runner" .
:a/438 :actsIn :m/5118 . :a/572 :actsIn :m/5118 . :a/271 :actsIn :m/2281 .

unibz

Outline

1 Virtual Knowledge Graphs (VKGs) for Data Access

2 Components of the VKG Architecture

3 Query Answering in VKGs

4 The Ontop System

5 Designing a VKG System

6 Conclusions

unibz

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 bladerunner among the SFMovies, and that the ontology states that each SFMovie is a Movie.
  If we ask for all Movies, we should return also bladerunner, 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.

unibz

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 bladerunner among the SFMovies, and that the ontology states that each SFMovie is a Movie.
  If we ask for all Movies, we should return also bladerunner, considering both the data and the knowledge in the ontology.

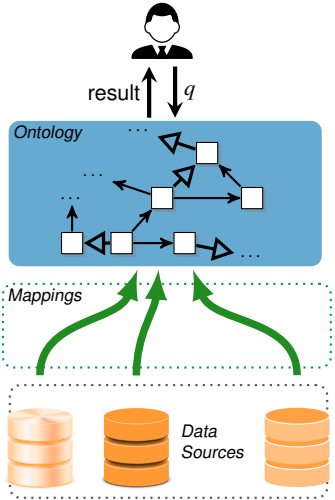
- The **mapping** encodes the information of how to translate a query over the ontology into a query over the **database**.

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

Query answering by query rewriting

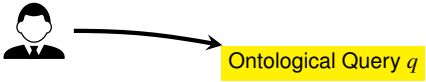
unibz

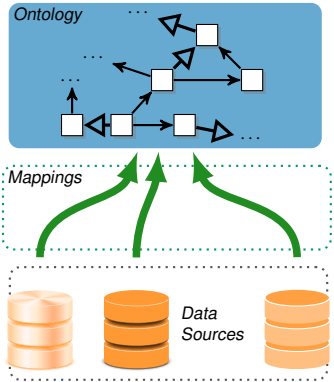
Query answering by query rewriting



unibz

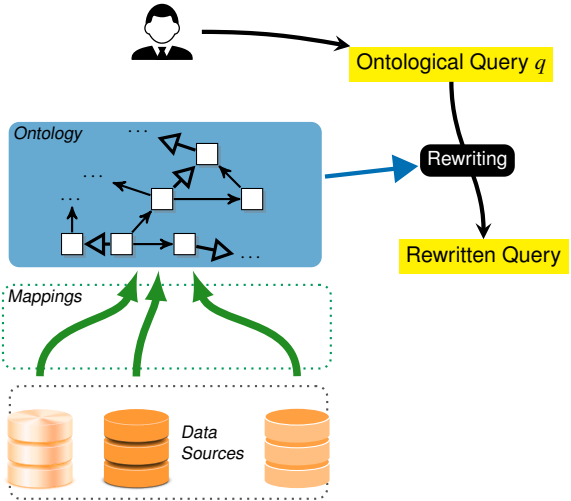
Query answering by query rewriting





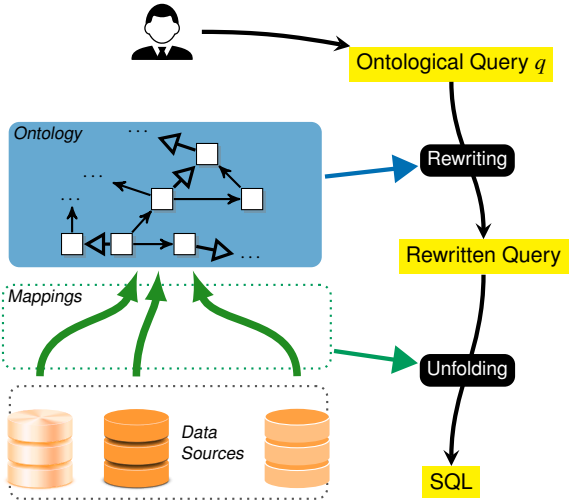
unibz

Query answering by query rewriting

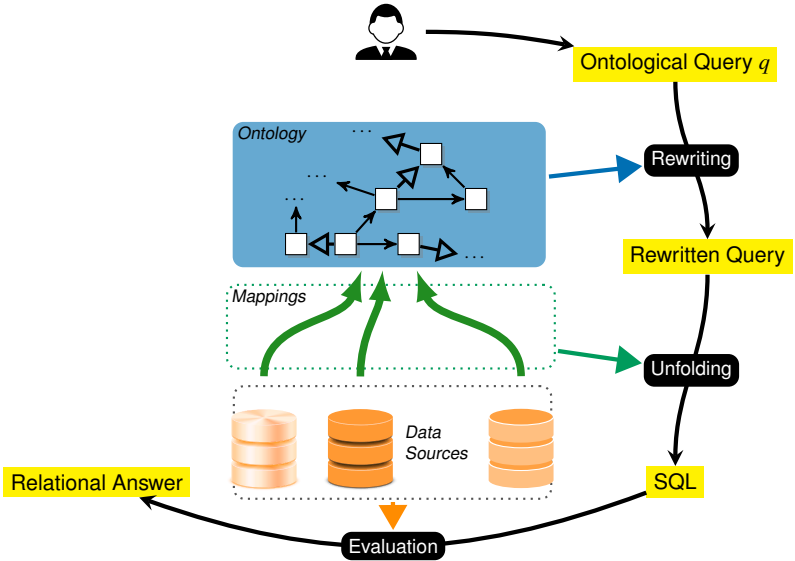


unibz

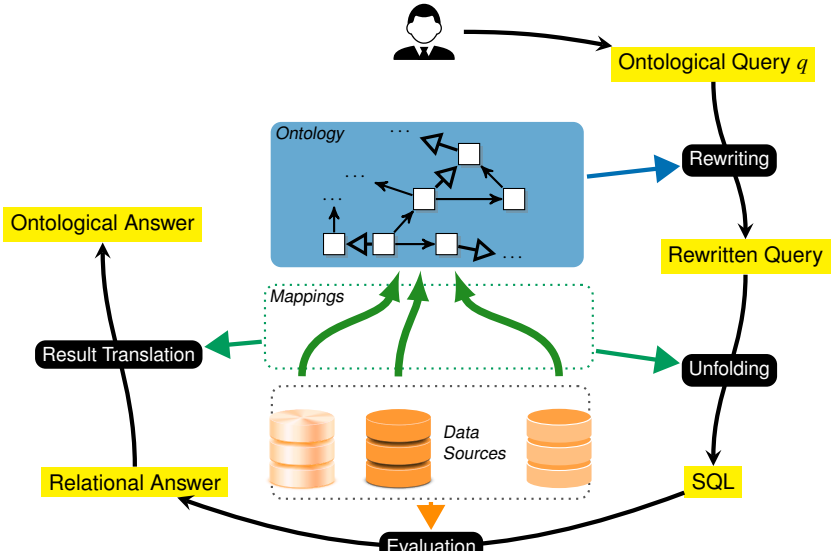
Query answering by query rewriting



Query answering by query rewriting

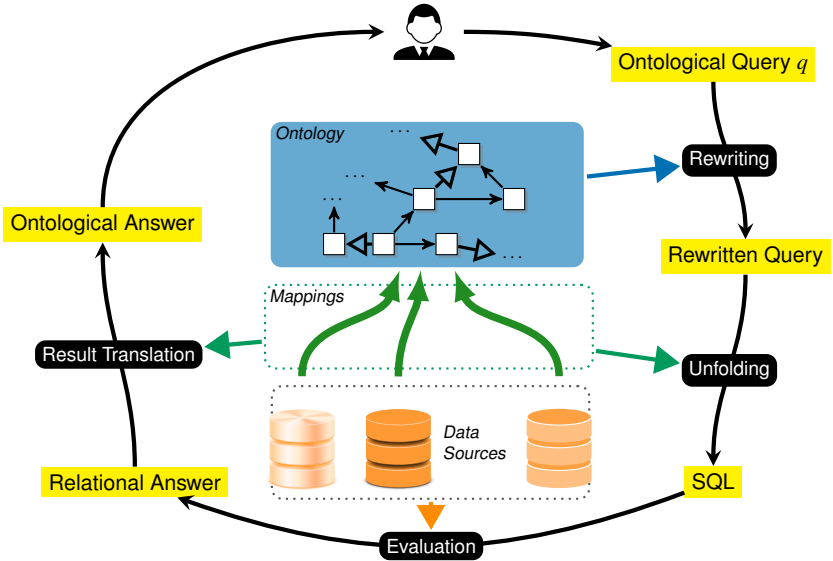


Query answering by query rewriting



unibz

Query answering by query rewriting



unibz

Outline

1 Virtual Knowledge Graphs (VKGs) for Data Access

2 Components of the VKG Architecture

3 Query Answering in VKGs

4 The Ontop System

5 Designing a VKG System

6 Conclusions

unibz

The *Ontop* system

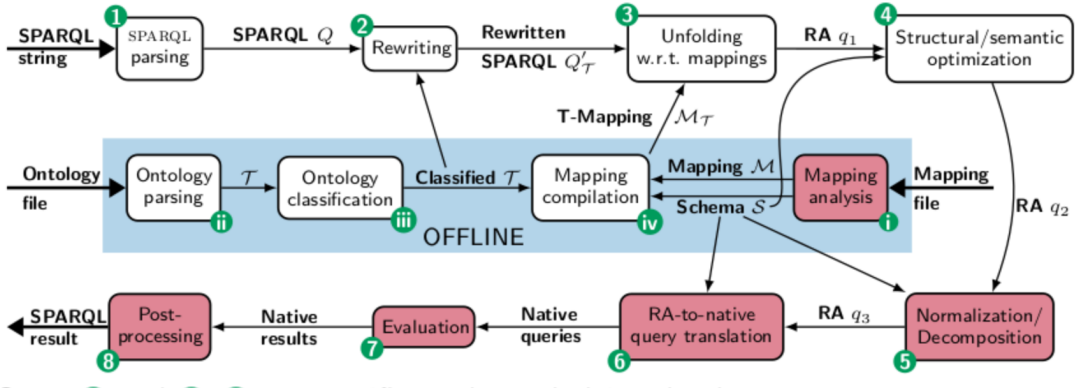
ontop

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.

unibz

Query answering in *Ontop*



Outline

1 Virtual Knowledge Graphs (VKGs) for Data Access

2 Components of the VKG Architecture

3 Query Answering in VKGs

4 The Ontop System

5 Designing a VKG System

6 Conclusions

unibz

Designing the ontology

- Ontologies might become very large and complex, hence designing an ontology is a challenging task.
- In many domains (e.g., the biomedical one) ontologies are developed independently by trained experts and already available to be re-used.
- Relying on “standardized ontologies” also enables interoperability across different domains.
- 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].
- Also, due to the correspondence with UML Class Diagrams / ER Schemas, we can rely on standard methodologies and tools adopted in software engineering or database design.

unibz

Designing VKG mappings

VKG mappings:

- Map complex queries to the target ontology.
- Overcome the abstraction mismatch between relational data and a knowledge graph.
- Are inherently more sophisticated than mappings used in other settings, such as schema matching [Rahm & Bernstein 2001] or ontology matching [Euzenat & Shvaiko 2007].

unibz

Designing VKG mappings

VKG mappings:

- Map complex queries to the target ontology.
- Overcome the abstraction mismatch between relational data and a knowledge graph.
- Are inherently more sophisticated than mappings used in other settings, such as schema matching [Rahm & Bernstein 2001] or ontology matching [Euzenat & Shvaiko 2007].

As a consequence:

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

unibz

Designing VKG mappings

VKG mappings:

- Map complex queries to the target ontology.
- Overcome the abstraction mismatch between relational data and a knowledge graph.
- Are inherently more sophisticated than mappings used in other settings, such as schema matching [Rahm & Bernstein 2001] or ontology matching [Euzenat & Shvaiko 2007].

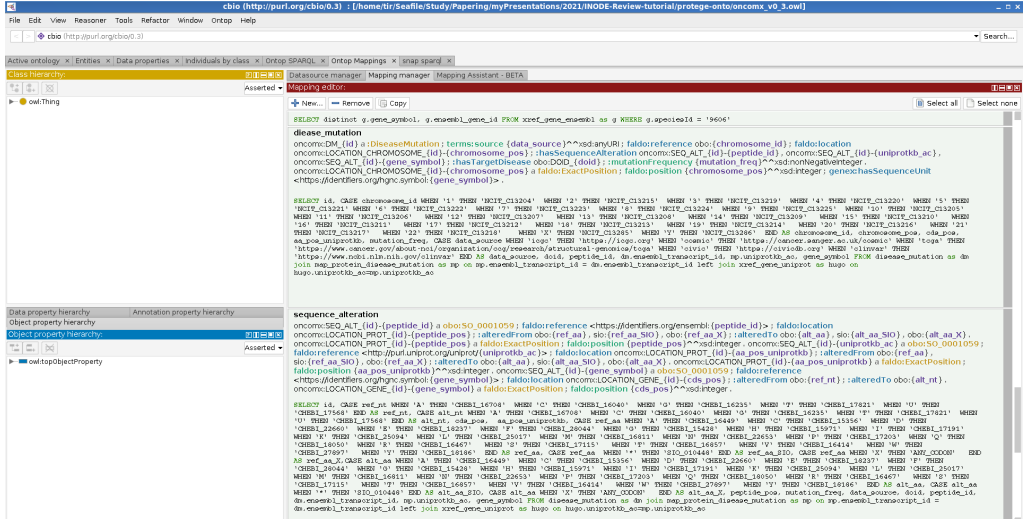
As a consequence:

- The design of VKG mappings is an effort that has been essentially **manual**, **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

Designing VKG mappings



VKG mapping patterns [C. et al. 2023]

Several approaches and tools supporting the creation of mappings have been proposed, several of them based on mapping patterns.

unibz

VKG mapping patterns [C. et al. 2023]

Several approaches and tools supporting the creation of mappings have been proposed, several of them based on 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.

unibz

VKG mapping patterns [C. et al. 2023]

Several approaches and tools supporting the creation of mappings have been proposed, several of them based on 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.

We have defined a **catalog of VKG mapping patterns**, where for each pattern we **take into account all the available information**:

- the domain knowledge that is encoded in ontology axioms
- the relational DB schema with its constraints (keys, foreign keys, . . .)
- data stored in the DB, when available
- the conceptual schema at the basis of the relational schema

unibz

Two major groups of mapping patterns

Schema-driven patterns

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

Two major groups of mapping patterns

Schema-driven patterns

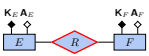
Are shaped by the structure of the DB schema and its explicit constraints.

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.

unibz

Example of schema-driven patterns

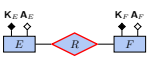
E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY
Schema Entity (SE)			
	$T_E(\underline{\mathbf{K}}, \mathbf{A})$	s: T_E t: $C_E(\mathbf{iri}_E(\mathbf{K}))$, $\{d_A(\mathbf{iri}_E(\mathbf{K}), A)\}_{A \in \mathbf{K} \cup \mathbf{A}}$	$\{\exists d_A \sqsubseteq C_E\}_{A \in \mathbf{K} \cup \mathbf{A}}$
Schema 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)**



Example of schema-driven patterns

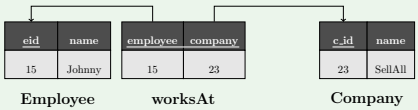
E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY
Schema Entity (SE)			
	$T_E(\underline{\mathbf{K}}, \mathbf{A})$	s: T_E t: $C_E(\mathbf{iri}_E(\mathbf{K}))$, $\{d_A(\mathbf{iri}_E(\mathbf{K}), A)\}_{A \in \mathbf{K} \cup \mathbf{A}}$	$\{\exists d_A \sqsubseteq C_E\}_{A \in \mathbf{K} \cup \mathbf{A}}$
Schema 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

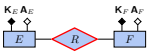


DB Schema



RDF (data only)

Example of schema-driven patterns

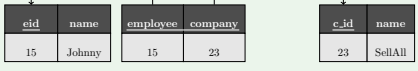
E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY
Schema Entity (SE)			
	$T_E(\underline{\mathbf{K}}, \mathbf{A})$	s: T_E t: $C_E(\mathbf{iri}_E(\mathbf{K}))$, $\{d_A(\mathbf{iri}_E(\mathbf{K}), A)\}_{A \in \mathbf{K} \cup \mathbf{A}}$	$\{\exists d_A \sqsubseteq C_E\}_{A \in \mathbf{K} \cup \mathbf{A}}$
Schema 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., $\mathbf{K}_{RF}$).

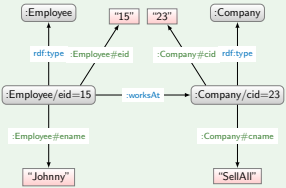
Conceptual



DB Schema



RDF (data only)



Example of a data-driven pattern

E-R diagram	DB schema	Mapping	Ontology
Clustering Entity to Class (CE2C)			
 $\mathbf{B} \subseteq \mathbf{K} \cup \mathbf{A}$, $\mathit{partition}_{\mathcal{D}}(\mathbf{B}, E)$	$T_E(\mathbf{K}, \mathbf{A})$ $\mathsf{unique}_{T_E}(\mathbf{K})$ $\mathbf{B} \subseteq \mathbf{K} \cup \mathbf{A}$ $\mathit{partition}_{\mathcal{D}}(\mathbf{B}, E)$ $\{V_{E_\mathbf{v}}(\mathbf{K}, \mathbf{A}) = \sigma_{\mathbf{B}=\mathbf{v}}(T_E)\}_{\mathbf{v} \in \pi_\mathbf{B}(T_E)}$	$\{s : \sigma_{\mathbf{B}=\mathbf{v}}(T_E)$ $t : C_E^\mathbf{v}(\mathbf{iri}_E(\mathbf{K}))\}_{\mathbf{v} \in \pi_\mathbf{B}(T_E)}$	$\{C_E^\mathbf{v} \sqsubseteq C_E\}_{\mathbf{v} \in \pi_\mathbf{B}(T_E)}$

unibz

Example of a data-driven pattern

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

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

Employee

unibz

Example of a data-driven pattern

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

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

Employee

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

Mapping-patterns for LLM-based mapping extraction [Xiao, Ren, et al. 2025]

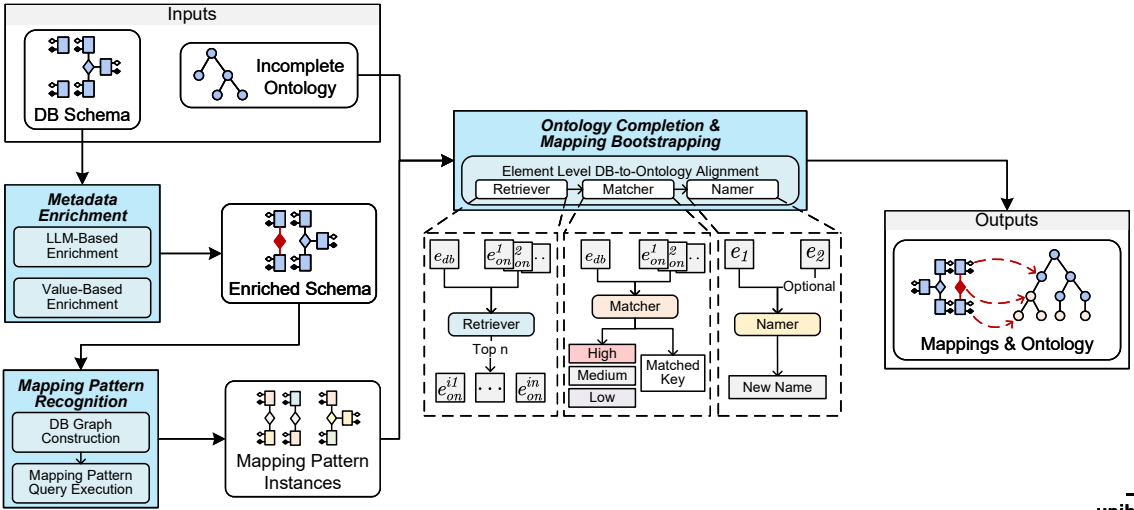
We have exploited mapping patterns in an LLM-based mapping extraction tool:

- Schema Entity
- Schema Relationship
- Schema Relationship with Merging
- Schema Hierarchy

1. We enrich the DB schema using both data profiling techniques and LLM-based enrichment.
2. The DB schema with constraints is mapped to a graph, and we use SPARQL queries to retrieve matches in this graph for the mapping patterns.
3. We exploit the extracted mapping patterns via three LLM-based modules to perform element level DB-to-ontology alignment
   - *Retriever* uses a pre-trained sentence similarity language model to retrieve top-n candidate elements for a DB schema element.
   - *Matcher* takes a DB schema element and a set of ontology elements, and uses a generative language model to generate a matching degree (with possible values High, Medium, or Low).
   - *Namer* uses a generative LLM to generates a name for a new ontology element from a DB schema element.

unibz

Architecture of the LLM-based mapping extraction tool



Evaluation of the LLM-based mapping extraction tool

DB	Tradition						Vanilla LLM			SAGE (ours)			
	D2RQ	MIRR	Ontop	COMA	IncM.	B.OX	DS-V3	o1	G4o	G4o	G5.4-m	G4o-m	Q-7b
Conference domain, adjusted naming													
CMT	0.31	0.28	0.28	0.48	0.45	0.76	0.43	0.30	0.07	**0.86**	0.79	0.79	0.79
Conference	0.26	0.27	0.26	0.36	0.53	0.51	0.50	0.36	0.10	**0.92**	0.85	0.90	**0.92**
SIGKDD	0.38	0.30	0.38	0.66	0.76	0.86	0.66	0.45	0.10	**0.93**	**0.93**	0.86	0.88
Conference domain, restructured													
CMT	0.14	0.17	0.14	0.38	0.44	0.41	0.45	0.48	0.13	**0.55**	0.52	0.52	0.40
Conference	0.21	0.23	0.13	0.31	0.41	0.41	0.42	0.50	0.12	0.59	**0.63**	0.59	0.58
SIGKDD	0.28	0.11	0.21	0.41	0.38	0.52	0.42	0.40	0.11	**0.71**	0.59	**0.71**	0.52
Conference domain, combined case													
SIGKDD	0.21	0.11	0.21	0.28	0.38	0.48	0.31	0.51	0.14	**0.72**	0.68	0.69	0.68
Conference domain, missing FKs													
Conference	0.18	0.17	–	0.21	0.41	0.33	0.43	0.38	0.15	**0.51**	0.49	0.44	0.43
Conference domain, denormalized													
CMT	0.20	0.22	0.20	–	0.40	0.44	0.44	0.48	0.16	**0.52**	**0.52**	**0.52**	0.34
Geodata domain													
Classic Rel.	0.06	–	–	–	0.08	0.13	–	–	–	0.18	0.16	**0.24**	0.11
Oil & gas domain													
Atomic	0.08	0.00	0.10	0.02	0.12	0.14	–	–	–	0.19	**0.20**	0.14	0.15
Average (methods with complete data)													
Average	0.21	–	–	–	0.40	0.45	–	–	–	**0.61**	0.58	0.58	0.53

Systems: MIRR = MIRROR, IncM. = IncMap, B.OX = BootOX, DS-V3 = DeepSeek-V3. **SAGE backbones:** G4o = GPT-4o, G5.4-m = GPT-5.4-mini, G4o-m = GPT-4o-mini, Q-7b = Qwen2.5-7B. Per row, the best is in **bold**, the second best is underlined, and the highest among the comparison methods (non-SAGE columns) is in blue. "–" denotes no result (system unsupported or input exceeds the LLM context window). The "missing FKs" scenario is the original RODI scenario of that name and should not be confused with RODI-MF, the controlled all-FK-removed variant used in Section VI-B.

F1 scores on RODI benchmark [Pinkel et al. 2018].

unibz

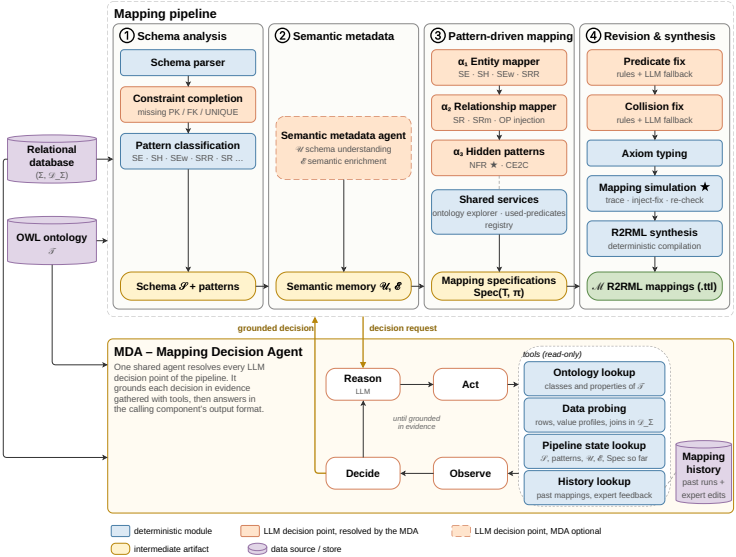
Multi-agent-based mapping extraction

We are currently working on a multi-agent-based refined architecture for mapping extraction that exploits additional mapping patterns:

- Schema Weak Entity
- Schema Reified Relationship
- Clustering Entity to Class
- Nullable Foreign Key Role Inference

unibz

Architecture of multi-agent based mapping extraction tool



Outline

1 Virtual Knowledge Graphs (VKGs) for Data Access

2 Components of the VKG Architecture

3 Query Answering in VKGs

4 The Ontop System

5 Designing a VKG System

6 Conclusions

unibz

Conclusions

- The VKG approach provides semantically grounded data access to LLMs and agents.
- The design of VKG mappings is a complex task, that currently is the major bottleneck in the wider adoption of the VKG paradigm.
- Mapping patterns and their exploitation in LLM and agent-based tools are a promising approach for simplifying this complex task.
- Work is needed to understand how to use VKGs to provide support for LLMs and agentic systems.

unibz

Thank you!

- E: calvanese@inf.unibz.it
- H: http://www.inf.unibz.it/~calvanese/





- *Ontop* website: https://ontop-vkg.org/
- Github: https://github.com/ontop/ontop/
- Facebook: https://www.facebook.com/obdaontop/
- Twitter: @ontop4obda
- *Ontopic* website: https://ontopic.ai/

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] 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.

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

[4] 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.

[5] Christoph Pinkel et al. "RODI: Benchmarking Relational-to-Ontology Mapping Generation Quality". In: *Semantic Web J.* 9.1 (2018), pp. 25–52. DOI: 10.3233/SW-170268.

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

unibz

References II

[7] 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.

[8] 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.

[9] Guohui Xiao, Lin Ren, Guilin Qi, Haohan Xue, Marco Di Panfilo & Davide Lanti. “LLM4VKG: Leveraging Large Language Models for Virtual Knowledge Graph Construction”. In: *Proc. of IJCAI 2025*. 2025, pp. 4715–4723. DOI: 10.24963/ijcai.2025/525.

unibz