Mapping Patterns for Virtual Knowledge Graphs

Diego Calvanese

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

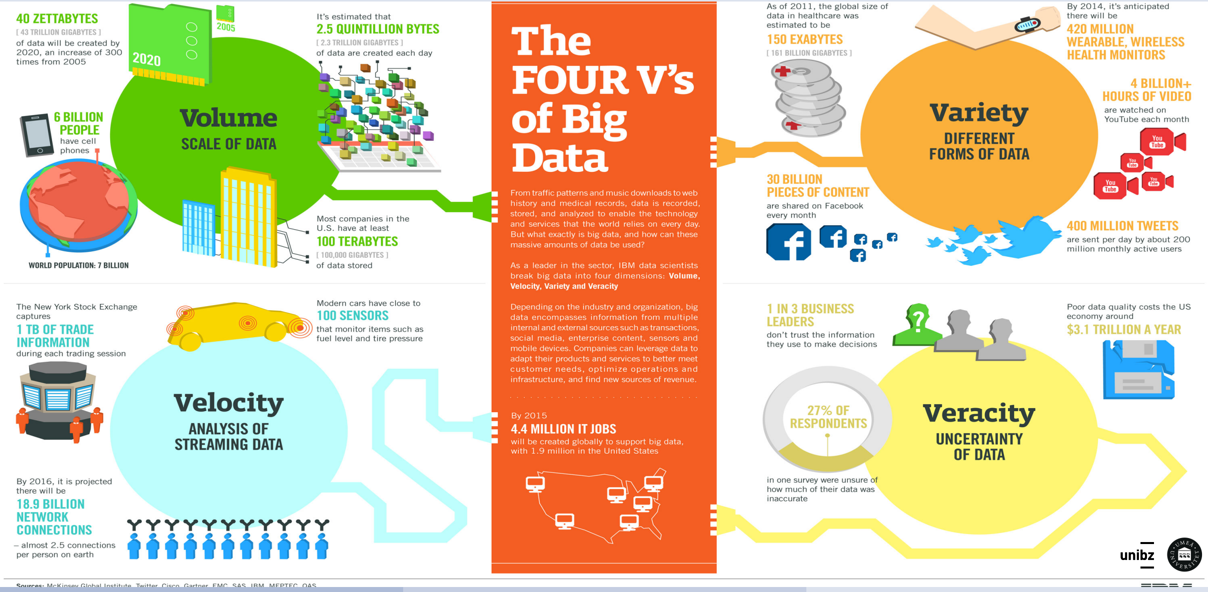
Department of Computing Science
Umeå University, Sweden

IEEE International Conference on
Artificial Intelligence & Knowledge Engineering (AIKE 2020)

9 December 2020 – Virtual

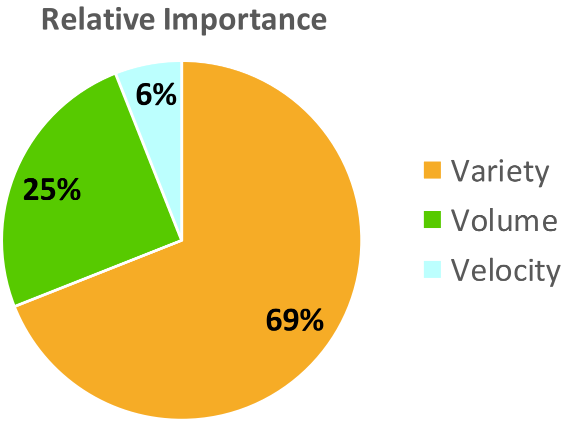
Outline of the presentation

Challenges in the Big Data era



Variety, not volume, is driving Big Data initiatives

MIT Sloan Management Review (28 March 2016)



http://sloanreview.mit.edu/article/variety-not-volume-is-driving-big-data-initiatives/

How much time is spent searching for the right data?

Important problem: searching for data and establishing its quality

Example: in oil&gas, engineers spend 30–70% of their time on this (Crompton, 2008)

Challenge: Accessing heterogeneous data

Statoil (now Equinor) Exploration

Geologists at Statoil, prior to making decisions on drilling new wellbores, need to gather relevant information about previous drillings.

Slegge relational database:

- Terabytes of relational data
- 1,545 tables and 1727 views
- each with dozens of attributes
- consulted by 900 geologists

Problem: Translating information needs

Information need expressed by geologists

In my geographical area of interest, return all pressure data tagged with key stratigraphy information with understandable quality control attributes, and suitable for further filtering.

To obtain the answer, this needs to be translated into SQL:

- Main table for wellbores has 38 columns (with cryptic names).
- To obtain pressure data requires a 4-table join with two additional filters.
- To obtain stratigraphic information requires a join with 5 more tables.

Problem: Translating information needs

We would obtain the following SQL query:

```
SELECT WELLBORE.IDENTIFIER, PTY_PRESSURE.PTY_PRESSURE_S,
       STRATIGRAPHIC_ZONE.STRAT_COLUMN_IDENTIFIER, STRATIGRAPHIC_ZONE.STRAT_UNIT_IDENTIFIER
FROM WELLBORE,
     PTY_PRESSURE,
     ACTIVITY FP_DEPTH_DATA
        LEFT JOIN (PTY_LOCATION_1D FP_DEPTH_PT1_LOC
           INNER JOIN PICKED_STRATIGRAPHIC_ZONES ZS
              ON ZS.STRAT_ZONE_ENTRY_MD <= FP_DEPTH_PT1_LOC.DATA_VALUE_1_O AND
                 ZS.STRAT_ZONE_EXIT_MD >= FP_DEPTH_PT1_LOC.DATA_VALUE_1_O AND
                 ZS.STRAT_ZONE_DEPTH_UOM = FP_DEPTH_PT1_LOC.DATA_VALUE_1_OU
           INNER JOIN STRATIGRAPHIC_ZONE
              ON   ZS.WELLBORE = STRATIGRAPHIC_ZONE.WELLBORE AND
                 ZS.STRAT_COLUMN_IDENTIFIER = STRATIGRAPHIC_ZONE.STRAT_COLUMN_IDENTIFIER AND
                 ZS.STRAT_INTERP_VERSION = STRATIGRAPHIC_ZONE.STRAT_INTERP_VERSION   AND
                 ZS.STRAT_ZONE_IDENTIFIER = STRATIGRAPHIC_ZONE.STRAT_ZONE_IDENTIFIER)
           ON FP_DEPTH_DATA.FACILITY_S = ZS.WELLBORE AND
              FP_DEPTH_DATA.ACTIVITY_S  = FP_DEPTH_PT1_LOC.ACTIVITY_S,
     ACTIVITY_CLASS FORM_PRESSURE_CLASS
WHERE WELLBORE.WELLBORE_S = FP_DEPTH_DATA.FACILITY_S AND
      FP_DEPTH_DATA.ACTIVITY_S = PTY_PRESSURE.ACTIVITY_S AND
      FP_DEPTH_DATA.KIND_S = FORM_PRESSURE_CLASS.ACTIVITY_CLASS_S AND
      WELLBORE.REF_EXISTENCE_KIND = 'actual' AND
      FORM_PRESSURE_CLASS.NAME = 'formation pressure depth data'
```

Problem: Translating information needs

We would obtain the following SQL query:

```
SELECT WELL
       STRA
FROM WELLBO
     PTY_PR
     ACTIVI
        LEF
```

This can be very time consuming, and requires knowledge of the domain of interest, a deep understanding of the database structure, and general IT expertise.

```
           INNER JOIN STRATIGRAPHIC_ZONE
              ON    ZS.WELLBORE = STRATIGRAPHIC_ZONE.WELLBORE AND
```

This is also very costly!

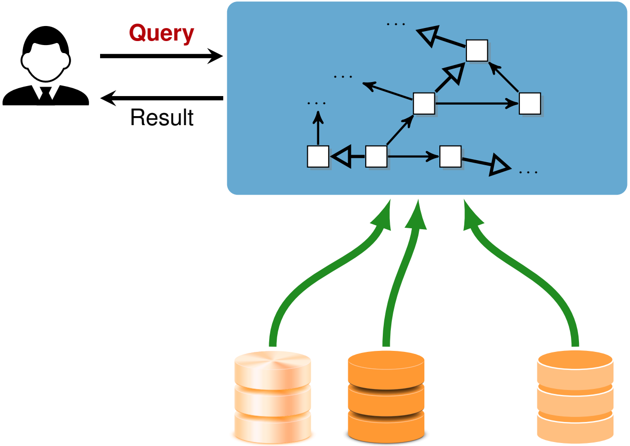
Equinor loses **50.000.000€** per year only due to this problem!!

```
     ACTIVI
WHERE WELLB
      FP_DE
      FP_DEPTH_DATA.KIND_S = FORM_PRESSURE_CLASS.ACTIVITY_CLASS_S AND
      WELLBORE.REF_EXISTENCE_KIND = 'actual' AND
      FORM_PRESSURE_CLASS.NAME = 'formation pressure depth data'
```

Outline

Outline

Solution: Virtual Knowledge Graphs (VKGs) – Also known as OBDA



Ontology $\mathcal{O}$
data is viewed as a graph, formulated in the vocabulary of the user

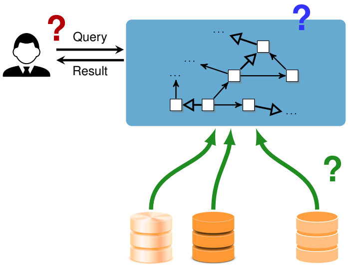
Mapping $\mathcal{M}$
specifies how to populate the ontology from the data

Data Sources $\mathcal{S}$
autonomous and heterogeneous

Greatly simplifies the access to information, and frees end-users from the need to know the precise structure of information sources.

VKG framework – Which languages to use?

The choice of the right languages needs to take into account the tradeoff between expressive power and efficiency of query answering.

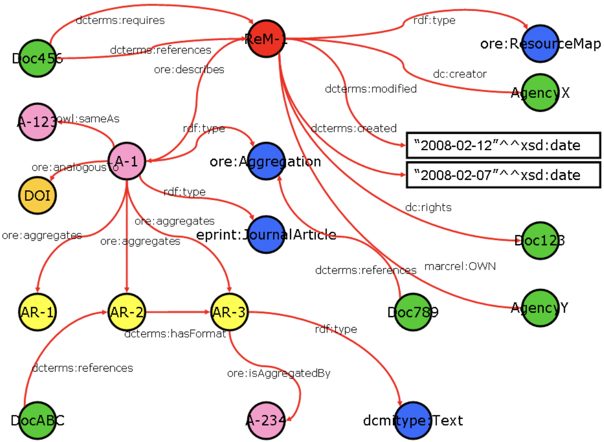


The W3C has standardized languages that are suitable for VKGs:

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**:



Object property:

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

Data property:

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

Class membership:

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

The OWL 2 QL ontology language

- **OWL 2 QL** is one of the three standard profiles of OWL 2. [W3C Rec. 2012]
- Is considered a lightweight ontology language:
  - controlled expressive power
  - efficient inference
- Optimized for accessing large amounts of data [C., De Giacomo, et al. 2007]
  - Queries over the ontology can be rewritten into SQL queries over the underlying relational database (First-order rewritability).
  - Consistency of ontology and data can also be checked by executing SQL queries.

Main constructs of OWL 2 QL

Class hierarchy: `rdfs:subClassOf` $(A_1 \sqsubseteq A_2)$

Example: `:MovieActor rdfs:subClassOf :Actor .`

Inference: `<person/2> rdf:type :MovieActor .`

$\Longrightarrow$ `<person/2> rdf:type :Actor .`

Domain of properties: `rdfs:domain` $(\exists P \sqsubseteq A)$

Example: `:playsIn rdfs:domain :MovieActor .`

Inference: `<person/2> :playsIn <movie/3> .`

$\Longrightarrow$ `<person/2> rdf:type :MovieActor .`

Range of properties: `rdfs:range` $(\exists P^{-} \sqsubseteq A)$

Example: `:playsIn rdfs:range :Movie .`

Inference: `<person/2> :playsIn <movie/3> .`

$\Longrightarrow$ `<movie/3> rdf:type :Movie .`

...

Other constructs of OWL 2 QL

Class disjointness: `owl:disjointWith` $(A_1 \sqsubseteq \neg A_2)$

Example: `:Actor owl:disjointWith :Movie .`

Inference: `<person/2> rdf:type :Actor .`
`<person/2> rdf:type :Movie .`
$\Longrightarrow$ RDF graph inconsistent with the ontology

Inverse properties: `owl:inverseOf` $(P_1 \sqsubseteq P_2^-$ and $P_2 \sqsubseteq P_1^-)$

Example: `:actsIn owl:inverseOf :hasActor .`

Inference: `<person/2> :actsIn <movie/3> .`
$\Longrightarrow$ `<movie/3> :hasActor <person/2> .`

Property hierarchy

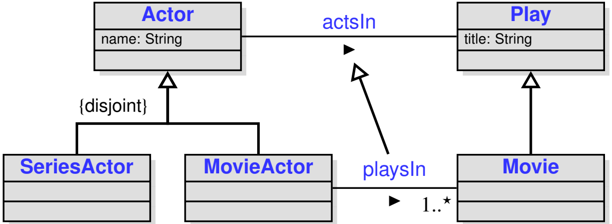
Property disjointness

Mandatory participation

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., and De Giacomo 2005; Bergamaschi and Sartori 1992; Borgida 1995; C., Lenzerini, and Nardi 1999; Lenzerini and Nobili 1990; Queralt et al. 2012].

$\text{SeriesActor} \sqsubseteq \text{Actor}$	`rdfs:subClassOf`	subclass
$\text{SeriesActor} \sqsubseteq \neg\text{MovieActor}$	`owl:disjointWith`	disjointness
$\exists\text{actsIn} \sqsubseteq \text{Actor}$	`rdfs:domain`	domain
$\exists\text{actsIn}^{-} \sqsubseteq \text{Play}$	`rdfs:range`	range
$\text{MovieActor} \sqsubseteq \exists\text{playsIn}$	`owl:someValuesFrom`	mandatory participation
$\text{playsIn} \sqsubseteq \text{actsIn}$	`rdfs:subPropertyOf`	sub-association
$\dots$		

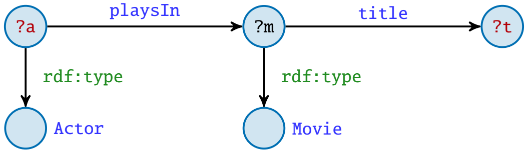


In fact, to visualize an OWL 2 QL ontology, we can use standard UML class diagrams.

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):

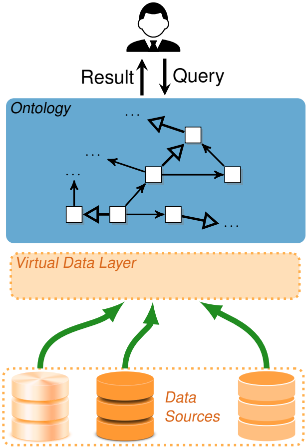
- `UNION`: matches one of alternative graph patterns
- `OPTIONAL`: produces a match even when part of the pattern is missing
- complex `FILTER` conditions
- `GROUP BY`, to express aggregations
- `MINUS`, to remove possible solutions
- property paths (regular expressions)
- ...

Use of mappings

In VKGs, the **mapping** $\mathcal{M}$ encodes how the data $\mathcal{D}$ in the sources should be used to create the virtual knowledge graph.

Virtual knowledge graph $\mathcal{V}$ defined from $\mathcal{M}$ and $\mathcal{D}$

- Queries are answered with respect to $\mathcal{O}$ and $\mathcal{V}$.
- The data of $\mathcal{V}$ is not materialized (it is virtual!).
- Instead, the information in $\mathcal{O}$ and $\mathcal{M}$ is used to translate queries over $\mathcal{O}$ into queries formulated over the sources.
- Advantage, compared to materialization: the graph is **always up to date** w.r.t. data sources.



Mapping language

The mapping consists of a set of assertions of the form

$$\Phi(\vec{x}) \rightsquigarrow \mathbf{t}(\vec{x}) \ \texttt{rdf:type} \ C$$
$$\Phi(\vec{x}) \rightsquigarrow \mathbf{t}_1(\vec{x}) \ p \ \mathbf{t}_2(\vec{x})$$

where

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

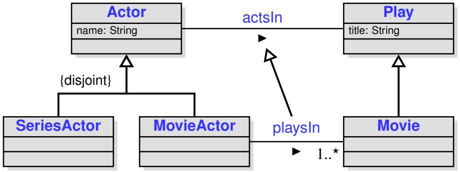
Impedance mismatch: values in the DB vs. objects in the knowledge graph

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

Note: When convenient, we group multiple mapping assertions with the same source query into a single assertion where the target is a set of triple patterns.

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} :playsIn :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 $\mathcal{V} = \mathcal{M}(\mathcal{D})$:

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

Outline

Formalizing VKGs [Poggi et al. 2008; Xiao, C., et al. 2018]

VKG specification $\mathcal{P} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$ and **VKG instance** $\langle \mathcal{P}, \mathcal{D} \rangle$

- $\mathcal{O}$ is an ontology (expressed in OWL 2 QL),
- $\mathcal{M}$ is a set of (R2RML) mapping assertions,
- $\mathcal{S}$ is a (relational) database schema with integrity constraints,
- $\mathcal{D}$ is a database conforming to $\mathcal{S}$.

Semantics is specified in terms of first-order logic interpretations of the ontology.

A FOL interpretation $\mathcal{I}$ of the ontology predicates is a **model** of $\langle \mathcal{P}, \mathcal{D} \rangle$ if

- it satisfies all axioms in $\mathcal{O}$, and
- contains all facts in $\mathcal{M}(\mathcal{D})$, i.e., retrieved through $\mathcal{M}$ from $\mathcal{D}$.

Note:

- In general, $\langle \mathcal{P}, \mathcal{D} \rangle$ has infinitely many models, and some of these might be infinite.
- However, for query answering, we do not need to compute such models.

Certain answers

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

Consider our formalization of VKG and a VKG instance $\mathcal{J} = \langle \mathcal{P}, \mathcal{D} \rangle$.

Certain answers

Given a VKG instance $\mathcal{J}$ and a query q over $\mathcal{J}$, the certain answers to q are those answers that hold in all models of $\mathcal{J}$.

First-order rewritability

To make computing certain answers viable in practice, the VKG setting relies on reducing it to evaluating SQL (i.e., first-order logic) queries over the data.

Consider a VKG specification $\mathcal{P} = \langle \mathcal{O}, \mathcal{M}, \mathcal{S} \rangle$.

First-order rewritability [Poggi et al. 2008]

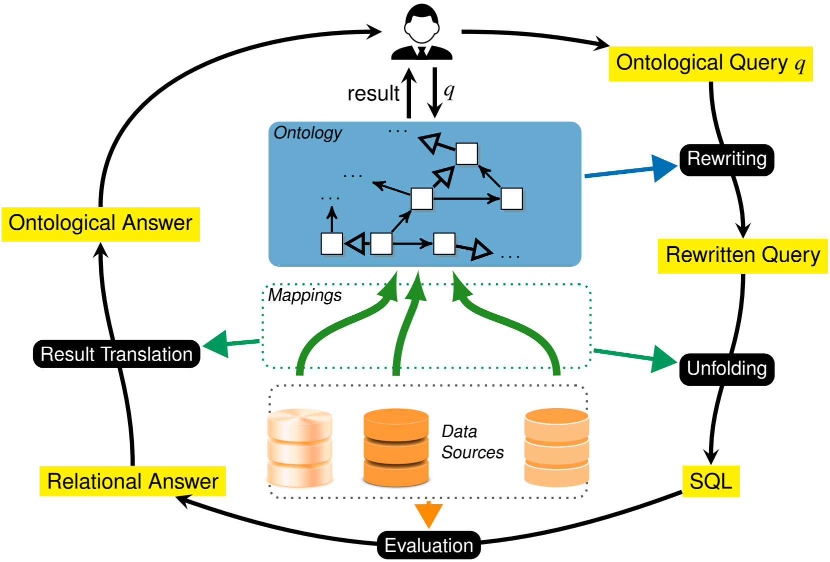
A query $r(\vec{x})$ is a **first-order rewriting** of a query $q(\vec{x})$ with respect to $\mathcal{P}$ if, for every source DB $\mathcal{D}$,

certain answers to $q(\vec{x})$ over $\langle \mathcal{P}, \mathcal{D} \rangle$ $=$ answers to $r(\vec{x})$ over $\mathcal{D}$.

For OWL 2 QL ontologies and R2RML mappings,
(core) SPARQL queries are first-order rewritable.

In other words, **in VKGs, we can compute the certain answers to a SPARQL query by evaluating over the sources its rewriting, which is a SQL query.**

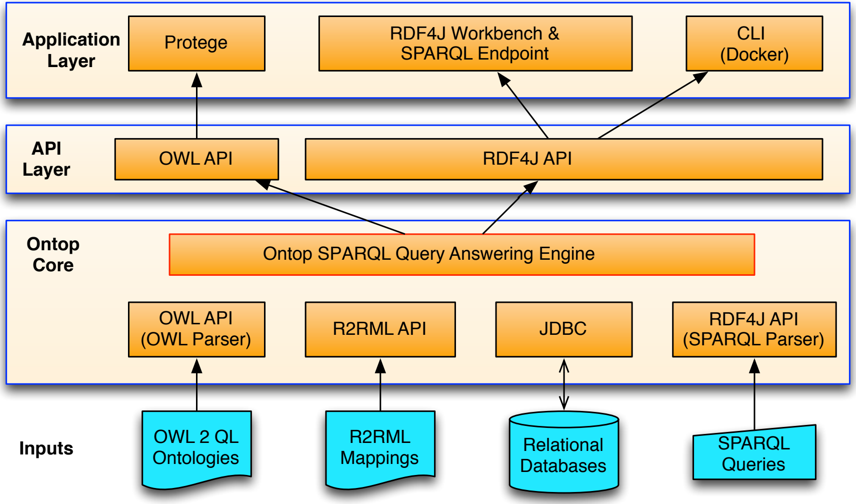
Query answering by query rewriting



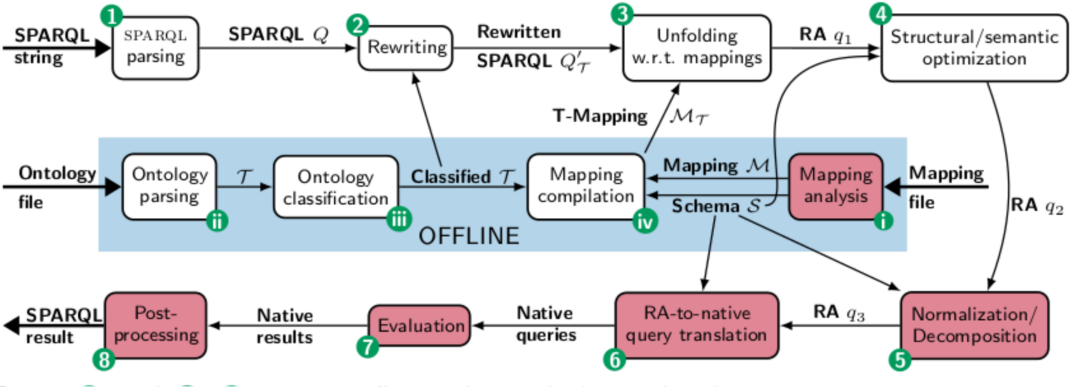
Ontop [C., Cogrel, et al. 2017]

- State-of-the-art VKG system
- Compliant with the RDFS, OWL 2 QL, R2RML, and SPARQL standards.
- Supports all major relational DBs
  - Oracle, DB2, MS SQL Server, Postgres, MySQL, Denodo, Dremio, Teiid, etc.
- Open-source and released under Apache 2 license
- Development of *Ontop*:
  - Development started in 2009
  - Already well established:
    - +200 members in the mailing list
    - +9000 downloads in the last 2 years
  - Main development carried out in the context of several local, national, and EU projects

Architecture of *Ontop*



Virtual approach for query answering in *Ontop*



Some use cases of *Ontop* – Research projects

- EU FP7 project Optique "Scalable End-user Access to Big Data" (11/2012 – 10/2016)
  - 10 Partners, including industrial partners Statoil, Siemens, DNV.
  - *Ontop* is core component of the Optique platform.
- EU project EPNet (ERC Advanced Grant) "Production and distribution of food during the Roman Empire: Economics and Political Dynamics"
  - Access to data in the cultural heritage domain .
- Euregio funded project KAOS "Knowledge-aware Operational Support" (06/2016 – 05/2019)
  - Preparation of standardized log files from timestamped log data for the purpose of process mining. [C., T. E. Kalayci, et al. 2017]
- EU H2020 project INODE "Intelligent Open Data Exploration" (11/2019 – 10/2022)
  - Development of techniques for the flexible interaction with data.

Commercial use cases of *Ontop* in which we are involved

- NOI Techpark in Bolzano – Development of knowledge graph of South Tyrol data [Ding et al. 2020]
  - Tourism data
  - Mobility data

- Collaboration with SIRIS Academic (Barcelona) – Development of data integration and dashboards for data analysis over open data from public institutions
  - Tuscany's Observatory of Research and Innovation
  - Sorbonne University

- Robert Bosch GmbH – Product quality analysis of the Surface Mounting Process pipeline [E. G. Kalayci et al. 2020]

See [Xiao, Ding, et al. 2019] for a survey on VKG systems and use cases.

Ontopic srl

ONTOPIC

- First spin-off of the Free University of Bozen-Bolzano.
- Incorporated in April 2019.
- Product: *Ontopic Suite* based on the *Ontop* engine.
- Services around *Ontop* and the *Ontopic Suite*.
- Consultancy for VKG-based data integration projects.

Outline

VKG 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 of schema matching [Rahm and Bernstein 2001] and ontology matching [Euzenat and Shvaiko 2007].

As a consequence

- Management of VKG mappings is a labor-intensive, essentially manual effort.
- Requires highly-skilled professionals [Spanos, Stavrou, and 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.

VKG mapping patterns

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

But there is no comprehensive approach for VKG mapping patterns exploiting all of:

- the relational schema with its constraints
- extensional data stored in the DB
- the TBox axioms that constitute the ontology
- the conceptual schema at the basis of the relational schema

I would like to address this issue by presenting:

- a catalog of VKG mapping patterns,
- design scenarios for VKG mapping patterns, and
- an analysis of practical VKG use cases wrt the use of mapping patterns.

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.

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.

Types of mapping patterns

Patterns are organized in two major groups:

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 $\leadsto$ **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 following types of constraints:

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

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

Note: Normal math font (e.g., A) is used for single attributes, while boldface is used for sets of attributes (e.g., $\mathbf{A}$ or $\mathbf{K}$).

Types of mapping patterns

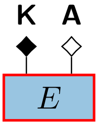
We have identified several different mapping patterns:

1. Entity (MpE)
2. Relationship (MpR)
3. Relationship with Identifier Alignment (MpRa)
4. Relationship with Merging (MpRm)
5. 1-1 Relationship with Merging (MpR11m)
6. Reified Relationship (MpRR)
7. Hierarchy (MpH)
8. Hierarchy with Identifier Alignment (MpHa)
9. Clustering Entity to Class / Data Property / Object Property (MpCE2X)

Each mapping pattern is specified through four different components that we illustrate on an example.

Note: The patterns make use of IRI-templates of the form "`:E/{`**`K`**`}`", where we assume that "`:E/`" is a prefix that is specific for the instances of a class C_E.

Mapping pattern: Entity (MpE)



Relational schema and constraints:

$T_E(\underline{\mathbf{K}}, \mathbf{A})$

Mapping assertion:

$s : T_E$

t : `:E/{`**K**`} rdf:type` C_E .

`{ :E/{`**K**`}` d_A `{`A`} . }`$_{A \in \mathbf{K} \cup \mathbf{A}}$

Ontology axioms:

$\{ \exists d_A \sqsubseteq C_E \}_{A \in \mathbf{K} \cup \mathbf{A}}$

For the application of the mapping pattern, we observe the following:

- This fundamental pattern considers a single table T_E with primary key $\underline{\mathbf{K}}$ and other relevant attributes $\mathbf{A}$.
- The pattern captures how T_E is mapped into a corresponding class C_E.
- The primary key $\underline{\mathbf{K}}$ of T_E is used to construct the objects that are instances of C_E, using a template `:E/{`**K**`}` specific for C_E.
- Each relevant attribute of T_E is mapped to a data property of C_E.

Mapping pattern: Entity (MpE) – Example

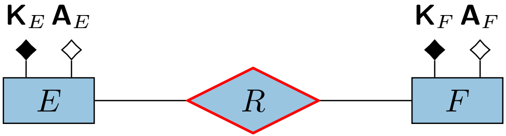
Consider a TClient table containing ssns of clients, together with name, dateOfBirth, and hobbies as additional attributes.

TClient(<u>ssn</u>,name,dateOfBirth,hobbies)

Mapping: TClient is mapped to a Client class using the attributes ssn to construct its objects. In addition, the ssn, name, and dateOfBirth are used to populate in the object position the three data properties ssn, name, and dob, respectively. The attribute hobbies is ignored.

```
mappingId MClient
source    SELECT ssn, name, dateOfBirth FROM TClient
target    :C/{ssn} rdf:type :Client ;
                   :ssn {ssn} ;
                   :name {name} ;
                   :dob {dateOfBirth} .
```

Mapping pattern: Relationship (MpR)



Relational schema and constraints:

$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $T_F(\underline{\mathbf{K}_F}, \mathbf{A}_F)$

$T_R(\underline{\mathbf{K}_{RE}, \mathbf{K}_{RF}})$

Mapping assertion:

$s : T_R$

$t :$ `:E/{`$\mathbf{K}_{RE}$`}` p_R `:F/{`$\mathbf{K}_{RF}$`}` .

Ontology axioms:

$\exists p_R \sqsubseteq C_E$

$\exists p_R^- \sqsubseteq C_F$

For the application of the mapping pattern, we observe the following:

- This pattern considers three tables T_R, T_E, and T_F.
- The primary key of T_R is partitioned into two parts $\mathbf{K}_{RE}$ and $\mathbf{K}_{RF}$ that are foreign keys to T_E and T_F, respectively.
- T_R has no additional (relevant) attributes.
- The pattern captures how T_R is mapped to an object property p_R, using the two parts $\mathbf{K}_{RE}$ and $\mathbf{K}_{RF}$ of the primary key to construct respectively the subject and the object of the triples in p_R.

Mapping pattern: Relationship (MpR) – Example

An additional TAddress table in the client registry stores the addresses at which each client can be reached, and such table has a foreign key to a table TLocation storing locations using attributes city and street.

```
TClient(ssn,name,dateOfBirth,hobbies)
TLocation(city,street)
TAddress(client,locCity,locStreet)
    FK: TAddress[client] -> Tclient[ssn]
    FK: TAddress[locCity,locStreet] -> TLocation[city,street]
```

Mapping: The TAddress table is mapped to an address object property, for which the ontology asserts that the domain is the class Person and the range an additional class Location, corresponding to the TLocation table.

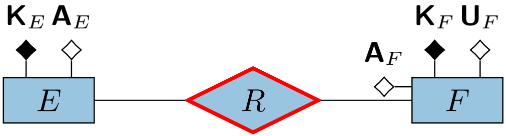
```
mappingId MAddress
source    SELECT client, locCity, locStreet FROM TAddress
target    :C/{client} :address :L/{locCity}/{locStreet} .
```

Mapping pattern: Relationship with Identifier Alignment (MpRa)

Relational schema and constraints:

$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $\qquad$ $T_F(\underline{\mathbf{K}_F}, \mathbf{U}_F, \mathbf{A}_F)$

$T_R(\underline{\mathbf{K}_{RE}, \mathbf{U}_{RF}})$ $\qquad$ $key_{R_F}(\mathbf{U}_F)$



Mapping assertion:

$s : T_R \bowtie_{\mathbf{U}_{RF} = \mathbf{U}_F} T_F$

$t : \texttt{:E/\{}\mathbf{K}_{RE}\texttt{\}}\ p_R\ \texttt{:F/\{}\mathbf{K}_F\texttt{\}}$.

Ontology axioms:

$\exists p_R \sqsubseteq C_E$

$\exists p_R^- \sqsubseteq C_F$

For the application of the mapping pattern, we observe the following:

- Such pattern is a variation of pattern **MpR**, in which the foreign key in T_R does not point to the primary key $\mathbf{K}_F$ of T_F, but to an additional key $\mathbf{U}_F$.
- Since the instances of class C_F corresponding to T_F are constructed using the primary key $\mathbf{K}_F$ of T_F (cf. pattern **MpE**), also the pairs that populate p_R should refer in their object position to $\mathbf{K}_F$.
- Note that $\mathbf{K}_F$ can only be retrieved by a join between T_R and T_F on the additional key $\mathbf{U}_F$.

Mapping pattern: Rel. with Identifier Alignment (MpRa) – Example

The primary key of the table TLocationCoord is now not given by the city and street, which are used in the table TAddress that relates clients to their addresses, but is given by the latitude and longitude of locations.

```
TClient(ssn,name,dateOfBirth,hobbies)
TLocationCoord(latitude,longitude,city,street)     key[TLocation]: city,street
TAddress(client,locCity,locStreet)
    FK: TAddress[client] -> Tclient[ssn]
    FK: TAddress[locCity,locStreet] -> TLocationCoord[city,street]
```

Mapping: The Address table is mapped to an address object property, for which the ontology asserts that the domain is the class Person and the range an additional class Location, corresponding to the Location table.

```
mappingId MAddressCoord
source    SELECT client, latitude, longitude
          FROM TAddress JOIN TLocationCoord ON locCity = city AND locStreet = street
target    :C/{client} :address :LC/{latitude}/{longitude} .
```

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.

Analysis of VKG use cases

We have analyzed the concrete mapping strategies in a number of VKG use cases:

- Understand how patterns occur in practice.
- Understand with which frequency patterns are used.

We have considered 6 different scenarios:

- Berlin Sparql Benchmark [Bizer and Schultz 2009]
- NPD Benchmark [Lanti et al. 2015]
- University Ontology Benchmark [Zhou et al. 2013]
- Südtirol OpenData `https://github.com/dinglinfang/suedTirolOpenDataOBDA`
- Open Data Hub VKG `https://sparql.opendatahub.bz.it/`
- Cordis `https://www.sirisacademic.com/wb/`

We have (manually) classified 1582 mapping assertions, falling in 367 pattern applications.

Occurrences of mapping patterns over the scenarios

	BSBM		NPD		UOBM		ODH		ST-OD		Cordis		Total	
SE	8	52	57	505	8	16	10	43	8	37	11	56	102	709
SR	–	–	–	–	2	2	–	–	–	–	3	3	5	5
SRm	8	8	45	45	5	5	–	–	7	7	7	7	72	72
SRR	–	–	1	12	–	–	–	–	–	–	1	16	2	28
SH	–	–	3	132	–	–	–	–	–	–	–	–	3	132
DE	–	–	–	–	–	–	–	–	3	7	4	9	7	16
DRm	5	5	17	17	36	36	2	2	1	1	2	2	63	63
DH	–	–	–	–	5	9	–	–	–	–	–	–	5	9
DRR	2	2	–	–	–	–	–	–	–	–	–	–	2	2
DR1Nm	4	4	19	54	–	–	–	–	9	15	1	1	33	74
D11Rm	–	–	–	–	–	–	6	78	5	14	–	–	11	92
DH0N	–	–	–	–	–	–	–	–	–	–	1	2	1	2
CE2C	–	–	11	82	6	19	5	23	–	–	1	12	23	136
CE2D	–	–	23	49	–	–	–	–	–	–	–	–	23	49
CE2O	–	–	13	148	–	–	–	–	–	–	–	–	13	148
CR2O	–	–	–	–	1	12	–	–	–	–	1	1	1	12
unknown	–	–	3	6	1	1	1	4	1	12	4	9	10	32

Left number: number of applications of a pattern in a scenario
Right number: number of mappings involved

Outline

Conclusions

- VKGs are by now a mature technology to address the data access and integration problems.
- This paradigm has been well-investigated and applied in real-world scenarios (mostly for the case of relational data sources).
- Performance and scalability of query answering for data access w.r.t. large datasets (volume), large and complex ontologies (variety, veracity), and multiple heterogeneous data sources (variety, volume) are a challenge, but sophisticated techniques and tools are available that achieve a good performance.
- However, the **design of** the ontology and notably **the mapping layer** is still a largely manual, labor-intensive, and error-prone activity.
- This represents a major bottleneck for a wider adoption of the VKG approach in real-world scenarios.
- Principled solutions based on well-established conceptual modeling principles, e.g., realized through **mapping patterns** look promising to address this challenge.

Ongoing work

We are currently developing technologies to make the VKG paradigm easier to adopt:

- Mapping patterns are at the basis of techniques for (semi-)automatic extraction/learning of ontology axioms and mapping assertions.
- We are working on automated support for the efficient management of mappings and ontology axioms, to support design, maintenance, and evolution. Ontopic is developing specific tools for this.
- In addition, we are working on improving the support for multiple, heterogeneous data sources.

In parallel, we are working on foundational and applied issues to enrich the VKG setting:

- More expressive queries, supporting analytical tasks.
- Support for additional types of data sources, such as graph data, geospatial data, temporal data.
- Dealing with data provenance and explanation.
- Dealing with data inconsistency and incompleteness – Data quality!
- Ontology-based update.

Conclusions and ongoing work

- We have identified a catalog of mapping patterns for the VKG framework.
- We have validated the patterns on a set of real-world use cases.
- We are currently working on using our patterns for mapping (and ontology) bootstrapping and generation.

Thank you!

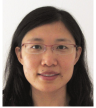
- E: calvanese@inf.unibz.it
- H: `http://www.inf.unibz.it/~calvanese/`

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

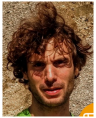
A great thank you to all my collaborators


Elena Botoeva

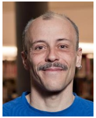

Julien Corman


Linfang Ding


Elem Güzel


Davide Lanti


Marco Montali


Alessandro Mosca


Mariano Rodriguez Muro


Guohui Xiao

Technion Haifa


Avigdor Gal


Roee Shraga

Birkbeck College London

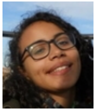

Roman Kontchakov


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

References I

[1] Marcelo Arenas, Alexandre Bertails, Eric Prud'hommeaux, and 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] Daniela Berardi, Diego C., and Giuseppe De Giacomo. "Reasoning on UML Class Diagrams". In: *Artificial Intelligence* 168.1–2 (2005), pp. 70–118.

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

[4] Christian Bizer and Andreas Schultz. "The Berlin SPARQL Benchmark". In: *Int. J. on Semantic Web and Information Systems* 5.2 (2009), pp. 1–24.

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

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

References II

[7] Diego C., Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. "Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family". In: *J. of Automated Reasoning* 39.3 (2007), pp. 385–429.

[8] Diego C., Tahir Emre Kalayci, Marco Montali, and Ario Santoso. "OBDA for Log Extraction in Process Mining". In: *Reasoning Web: Semantic Interoperability on the Web – 13th Int. Summer School Tutorial Lectures (RW 2017)*. Vol. 10370. Lecture Notes in Computer Science. Springer, 2017, pp. 292–345. DOI: 10.1007/978-3-319-61033-7_9.

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

[10] Diego C., Pietro Liuzzo, Alessandro Mosca, Jose Remesal, Martin Rezk, and Guillem Rull. "Ontology-Based Data Integration in EPNet: Production and Distribution of Food During the Roman Empire". In: *Engineering Applications of Artificial Intelligence* 51 (2016), pp. 212–229. DOI: 10.1016/j.engappai.2016.01.005.

[11] Linfang Ding, Guohui Xiao, Diego C., and Liqiu Meng. "A Framework Uniting Ontology-based Geodata Integration and Geovisual Analytics". In: *Int. J. of Geo-Information* (2020). To appear.

References III

[12] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, 2007.

[13] Elem Güzel Kalayci, Irlan Grangel González, Felix Lösch, Guohui Xiao, Anees ul-Mehdi, Evgeny Kharlamov, and Diego C. "Semantic Integration of Bosch Manufacturing Data Using Virtual Knowledge Graphs". In: *Proc. of the 19th Int. Semantic Web Conf. (ISWC)*. Vol. 12507. Lecture Notes in Computer Science. Springer, 2020, pp. 464–481. DOI: 10.1007/978-3-030-62466-8_29.

[14] Davide Lanti, Martin Rezk, Guohui Xiao, and Diego C. "The NPD Benchmark: Reality Check for OBDA Systems". In: *Proc. of the 18th Int. Conf. on Extending Database Technology (EDBT)*. OpenProceedings.org, 2015, pp. 617–628. DOI: 10.5441/002/edbt.2015.62.

[15] 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.

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

References IV

[17] Antonella Poggi, Domenico Lembo, Diego C., Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. "Linking Data to Ontologies". In: *J. on Data Semantics* 10 (2008), pp. 133–173. DOI: 10.1007/978-3-540-77688-8_5.

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

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

[20] Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. "Bringing relational databases into the Semantic Web: A survey". In: *Semantic Web J.* 3.2 (2012), pp. 169–209.

[21] Guohui Xiao, Diego C., Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and 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.

References V

[22] Guohui Xiao, Linfang Ding, Benjamin Cogrel, and 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.

[23] Yujiao Zhou, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, and Jay Banerjee. "Making the Most of your Triple Store: Query Answering in OWL 2 Using an RL Reasoner". In: *Proc. of the 22nd Int. World Wide Web Conf. (WWW)*. ACM, 2013, pp. 1569–1580.