From Ontop to Ontopic: a virtual-first perspective on knowledge graph construction

Benjamin Cogrel

Knowledge Graph Construction Workshop
28 May 2023, Hersonissos, Greece

ONTOPIC

About me

- CTO and co-founder of Ontopic
- Core developer of the Virtual Knowledge Graph engine Ontop since 2014
- Research on VKG for 5 years at the Free University of Bozen-Bolzano

ONTOPIC

Ontopic started in 2019 as a spin-off of the Free University of Bozen-Bolzano (Italy), from the research group that authored Ontop, an open-source Virtual Knowledge Graph engine

ONTOPIC spinoffunibz

Founders

Peter Hopfgartner

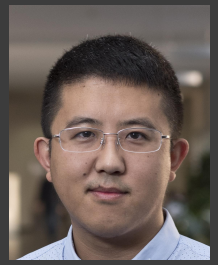
CEO

Benjamin Cogrel

CTO

Diego Calvanese

Scientific advisor of the board
Full professor at unibz
ACM Fellow



Guohui Xiao

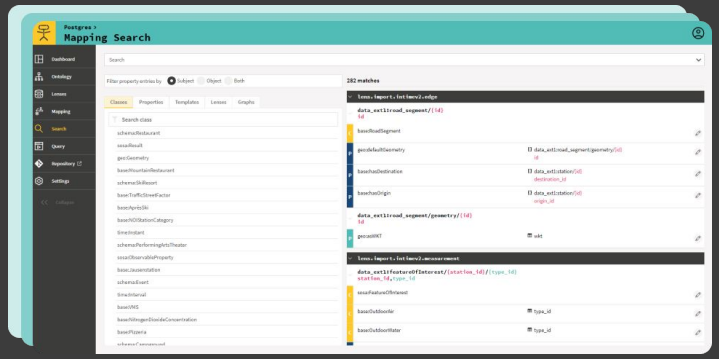
Chief scientist
Assoc. professor at University of Bergen (NO)

Marco Montali

Scientific consultant
Full professor at unibz

Our flagship product & services

ONTOPIC
Studio



Extending and adapting Ontop

Projects and consulting on data integration

Support and training for Ontopic Studio

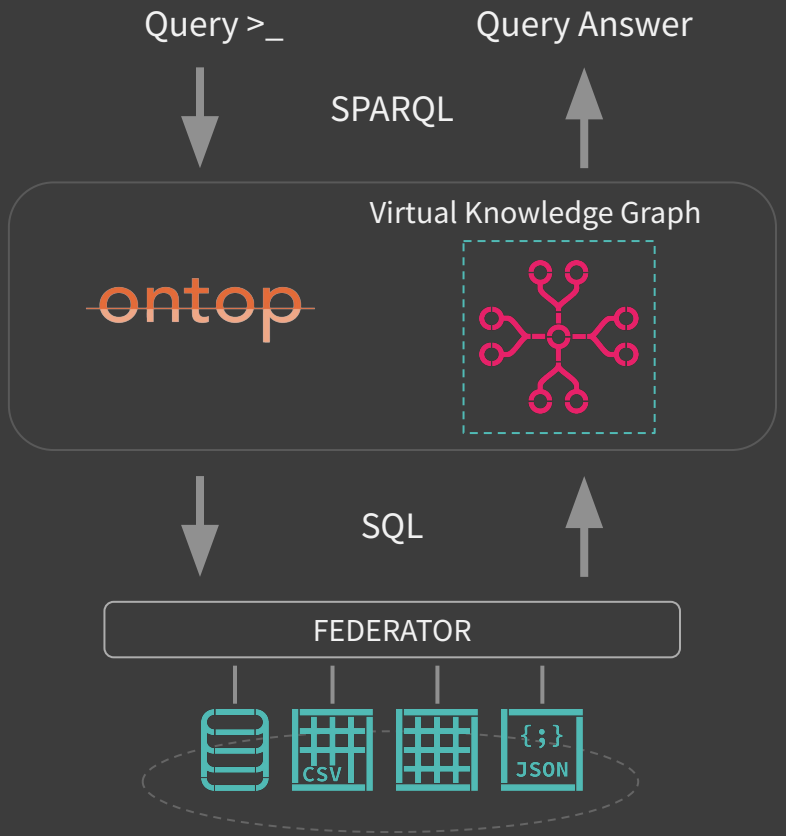
Agenda

1. What is key for virtual KGs
2. Mapping design
3. Some feedback from industry

A Virtual-first take on KG construction

- Central component of KG construction: mapping
- 2 ways to use a mapping
  a. To extract and transform data in RDF (KG materialization)
     - Always possible
  b. To reformulate SPARQL queries into source queries (KG virtualization)
     - Requires care and metadata

KG virtualization with Ontop



Specification:

- Ontology
- Mapping

Supported federators:

- Dremio
- Denodo
- Apache Spark
- Trino / Athena

What makes the virtual approach to KG attractive

- No requirement to move data from the "sources"
  - Immediate testing
  - Lightweight reasoning at almost no cost
- Flexible notion of "sources"
  - Not necessarily the primary sources (e.g. operational databases)
  - Can be a data warehouse / lakehouses / in-memory DBs (large diversity)
  - Allows for materialization to speed up queries (just not done at the RDF level)
- Enables hybrid KG solutions
  - One part in a triplestore, one part kept virtual

Common reasons for choosing the virtual approach

- You don't want to move data
  - Too large and frequently updated (sensor data)
  - Minimize the number of copies to keep access auditing simpler (sensitive data)
- You want better performance for your particular workload
  - When triplestores are not the fastest DBs for it
  - Interactive Business Intelligence analytical queries
    - OLAP-cube-like acceleration

Downside of the virtual approach

- Less "out-of-the-box" decent performance than with triplestores
  - Need to pay more attention when designing the mapping
  - Frequent need for extra metadata
- "Meta-queries" are more challenging
  - Where properties or classes are variables
  - Permanent effort to bring new optimizations in VKG engines
- Less suitable for graph analytics
- Limited reasoning capabilities

Where extra attention is needed?

1. IRI templates: make sure they can be decomposed when taking more than one argument
    - In the source, you don't want to join over concatenated values but over indexed columns
    - Notion of safe-separator in R2RML (e.g. putting a slash between arguments)
    - Easy
2. Database constraints
    - Unique constraints, foreign keys, non-nullability, etc.
    - Reduce the number of joins and unions in source queries
    - Eliminate distincts
        - Recall that a RDF graph doesn’t contain duplicates (it is a set of triples)

Getting database constraint metadata

- Ideal case: integrity constraints enforced and metadata exposed by the source
- In practice: many analytical data sources do not expose this metadata!
  - E.g. Dremio, Databricks, DuckDB, BigQuery (until recently)
  - Missing metadata for data coming from files (e.g. CSV, JSON)
- Denormalized data needs general forms of functional and inclusion dependencies

Providing missing DB constraint metadata is key for the virtual approach!

Canonical example: self-join elimination using unique constraints

- SPARQL is full of joins
  - e.g. 4 triple patterns -> 3 joins
  - After replacing the triple patterns by their definitions (from the mapping), most of these joins are redundant
  - If not eliminated, unusual source queries
    - Most of the time not well handled by the source query processors
- You need integrity constraints to eliminate them
  - Canonical case: join the same table over a unique constraint

Canonical example: self-join elimination using unique constraints

Live demo with the Crunchbase dataset on Snowflake

```
PREFIX schema: <http://schema.org/>
SELECT * WHERE {
 ?company schema:name ?name ;
          schema:address ?address .
 ?address schema:addressLocality ?locality ;
          schema:addressRegion ?region ;
          schema:addressCountry ?country
} LIMIT 10
```

Mapping design

Ontop native mapping language (.obda)

- Historical mapping language of Ontop
  - In the Ontop-Protégé plugin
  - Regularly used in plain text editors
- Bi-directional conversion from/to R2RML
- For SQL data sources
- Turtle-like template syntax in the target
- Explicit reuse of IRI templates across mapping entries

Target:

```
:uni1/student/{s_id} a :Student ;
    foaf:firstName {first_name}^^xsd:string ;
    foaf:lastName {last_name}^^xsd:string .
```

Source:

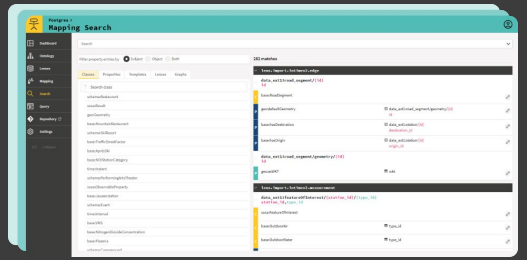
```
SELECT *
FROM "uni1"."student"
```

R2RML

- W3C standard (interoperability)
- For SQL data sources
- Well-thought for both virtualization and materialization
- Promotes reuse of triples maps rather than of IRI templates
- Often perceived as not user-friendly
  - Hard to teach (compared to Ontop's native format)
  - Most Ontop users prefer our native format

Ontopic Studio

- Both for virtualization and KG materialization
- No-code (form-based)
  - Avoid typos (e.g. column names, IRIs and IRI templates)
  - Reduce the need to write SQL queries
  - Visualize data
- Efficient search for large mappings
- UX: reuse of IRI templates, no triples maps
- Output: R2RML mapping, OWL/RDFS ontology, lenses



Lenses

- Virtual views defined outside of the data sources
- They have a name and can be referenced in the R2RML mapping
- Data "preparation" in Ontopic Studio
  - Functions (dialect-independent or dialect-specific)
- Specify missing database constraints
- Different types
  - Basic, join, SQL, union lenses
  - Flatten lenses for dealing with nested data
- Often saves people from writing SQL queries manually

Nested data

- Common with files (JSON, Parquet, but actually also CSV)
- Most SQL dialects support flattening
- Several SQL processors handle files directly (e.g. Dremio and DuckDB)
- Optimizations in Ontop (upcoming version)

Next standard after R2RML 1.0?

- Maybe a future version of RML?
- Both for virtualization and materialization
- Essential
  - Support for nested data
  - Missing database constraints
- Bonus
  - Dialect-independent functions
- Can still be treated as a "low-level" interoperable standard
  - Freedom to choose the UX concepts (lenses for us)

Some feedback from industry

- Current status-quo
  - KG market is a niche in the database/data integration world
  - Too many people are not using mappings but scripts
  - Most KG projects are more about metadata than regular data
  - Virtualization is fairly new
- No-code for mapping is getting accepted
  - Multiple vendors
  - Users asking for end-to-end experience
- Interest for standards to avoid vendor lock-in

Suggested research topics

Directly interesting for industry

- Data mesh
- Efficient composability of KGs
- Virtual and hybrid KGs

Take-home messages

- Mappings are not just about KG materialization
- KG virtualization is catching up with handling nested data
- Database constraint metadata should take part of the future mapping standard