

Ontop: the Virtual Knowledge Graph System

Guohui Xiao

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

Ontopic S.r.l., Bolzano, Italy



KRDB Summer Online Seminars 2020

10 July 2020

Challenges in the Big Data era

40 ZETTABYTES

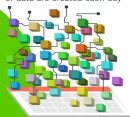
[43 TRILLION GIGABYTES]
of data will be created by 2020, an increase of 300 times from 2005



Volume
SCALE OF DATA

It's estimated that
2.5 QUINTILLION BYTES

[2.3 TRILLION GIGABYTES]
of data are created each day



Most companies in the U.S. have at least
100 TERABYTES
[100,000 GIGABYTES]
of data stored



The New York Stock Exchange captures

1 TB OF TRADE INFORMATION
during each trading session



By 2016, it is projected there will be

18.9 BILLION NETWORK CONNECTIONS

— almost 2.5 connections per person on earth



Modern cars have close to
100 SENSORS
that monitor items such as fuel level and tire pressure

Velocity
ANALYSIS OF
STREAMING DATA



The FOUR V's of Big Data

From traffic patterns and music downloads to web history and medical records, data is recorded, stored, and analyzed to enable the technology and services that the world relies on every day. But what exactly is big data, and how can these massive amounts of data be used?

As a leader in the sector, IBM data scientists break big data into four dimensions: **Volume, Velocity, Variety and Veracity**

Depending on the industry and organization, big data encompasses information from multiple internal and external sources such as transactions, social media, enterprise content, sensors and mobile devices. Companies can leverage data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue.

By 2015
4.4 MILLION IT JOBS
will be created globally to support big data, with 1.9 million in the United States



As of 2011, the global size of data in healthcare was estimated to be

150 EXABYTES
[161 BILLION GIGABYTES]



30 BILLION PIECES OF CONTENT
are shared on Facebook every month



Variety
DIFFERENT
FORMS OF DATA



By 2014, it's anticipated there will be
420 MILLION WEARABLE, WIRELESS HEALTH MONITORS

4 BILLION+ HOURS OF VIDEO
are watched on YouTube each month



400 MILLION TWEETS
are sent per day by about 200 million monthly active users



1 IN 3 BUSINESS LEADERS

don't trust the information they use to make decisions



Poor data quality costs the US economy around
\$3.1 TRILLION A YEAR



27% OF RESPONDENTS

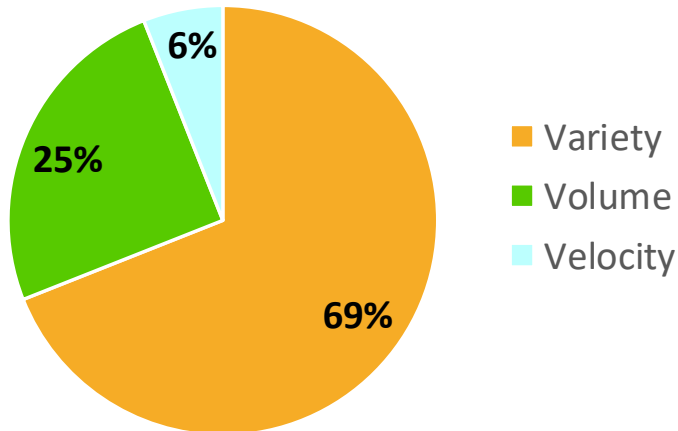
in one survey were unsure of how much of their data was inaccurate

Veracity
UNCERTAINTY
OF DATA

Variety, not volume, is driving Big Data initiatives

MIT Sloan Management Review (28 March 2016)

Relative Importance



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.



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:

- TBs 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 (the standard DB query language):

- 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_0 AND  
        ZS.STRAT_ZONE_EXIT_MD $>=$ FP_DEPTH_PT1_LOC.DATA_VALUE_1_0 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 WELLBORE_IDENTIFIER, PTY_PRESSURE, PTY_PRESSURE_CLASS_S  
FROM STRATIGRAPHIC_ZONE  
FROM WELLBORE  
FROM PTY_PRESSURE  
FROM PTY_PRESSURE_CLASS  
FROM ACTIVITY  
FROM LEI
```

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  
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 WELLBORE_IDENTITY, PTY, PRESSURE, PTY, PRESSURE, C  
      STRA  
FROM WELLBO  
      PTY_P  
      ACTIV  
      LEI
```

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.

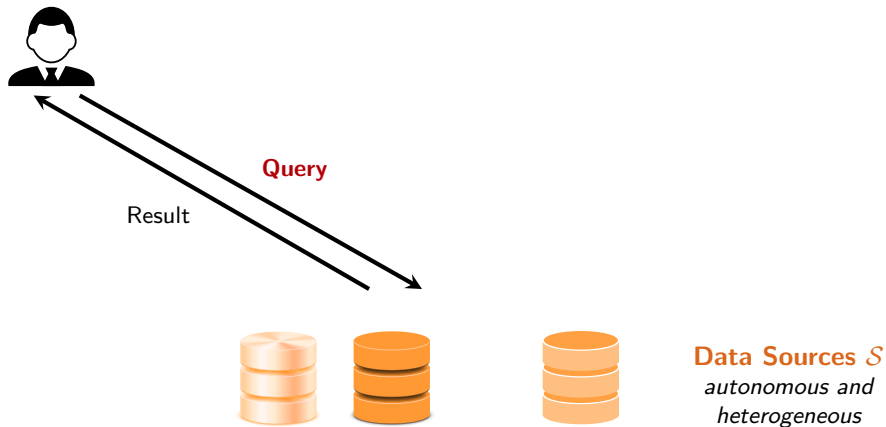
```
INNER JOIN STRATIGRAPHIC_ZONE  
      ON      ZS.WELLBORE = STRATIGRAPHIC_ZONE.WELLBORE AND
```

This is also very costly!

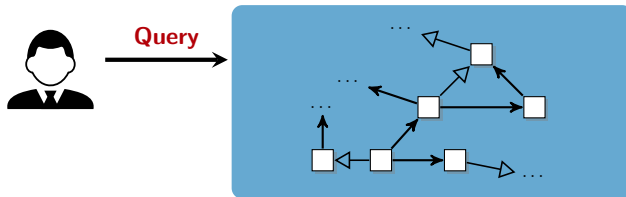
Statoil loses 50.000.000€ per year only due to this problem!!

```
ACTIV  
WHERE WELLBO  
      FP_DI  
      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'
```

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



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

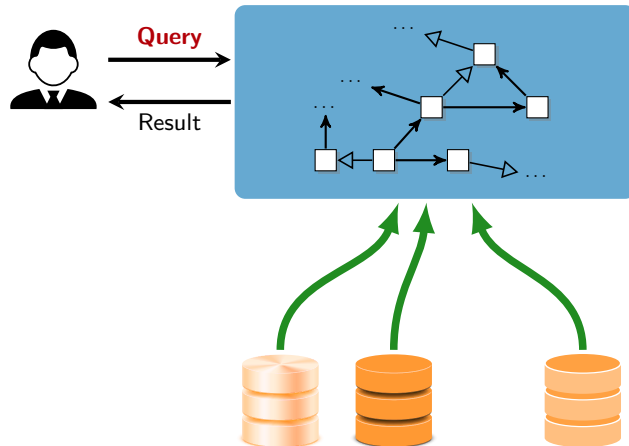


Ontology \mathcal{O}
*data is viewed as a graph,
vocabulary of the user*



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

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

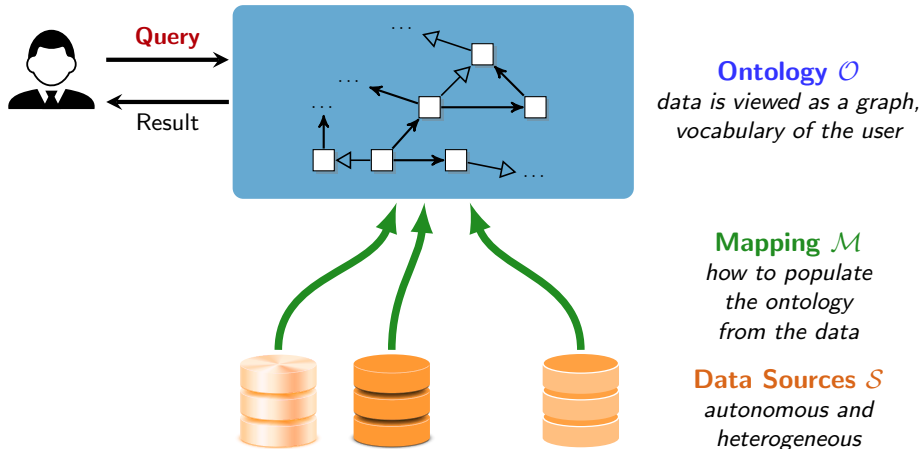


Ontology \mathcal{O}
*data is viewed as a graph,
vocabulary of the user*

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

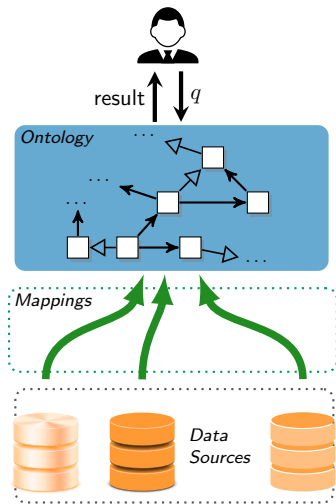
Data Sources \mathcal{S}
*autonomous and
heterogeneous*

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

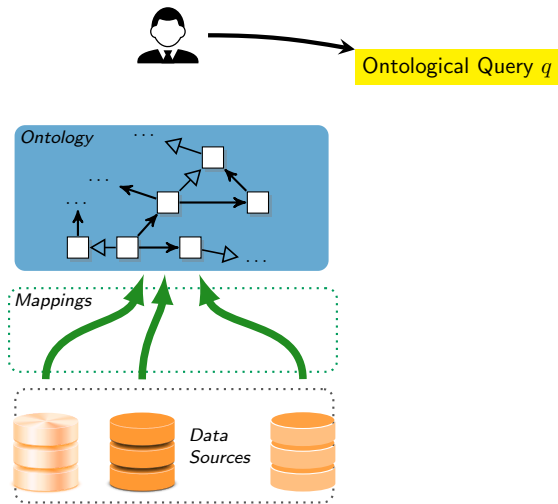


Greatly simplifies the access to information, and frees end-users from the need to know the precise structure of information sources.

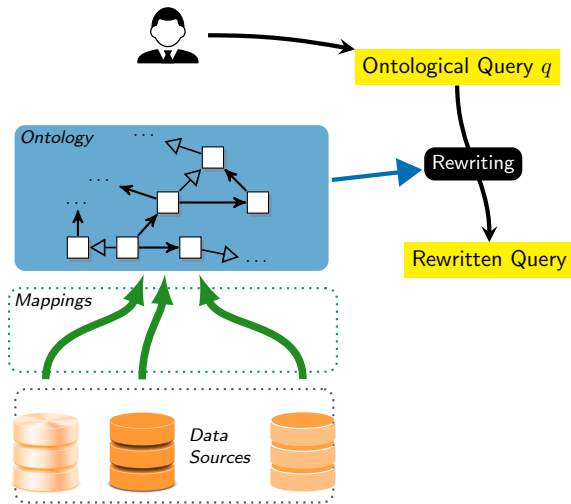
Query answering by rewriting (conceptual framework)



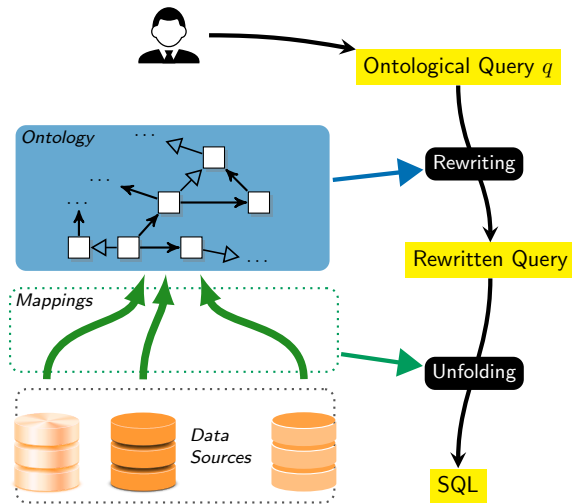
Query answering by rewriting (conceptual framework)



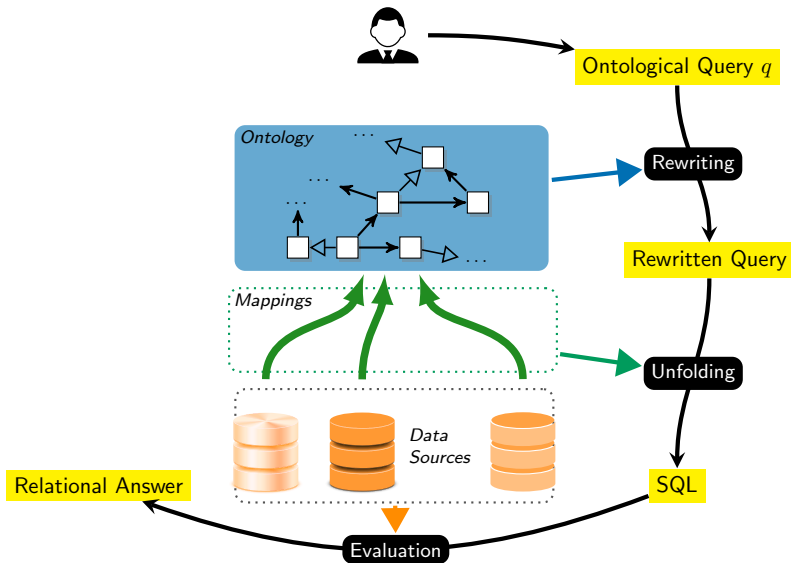
Query answering by rewriting (conceptual framework)



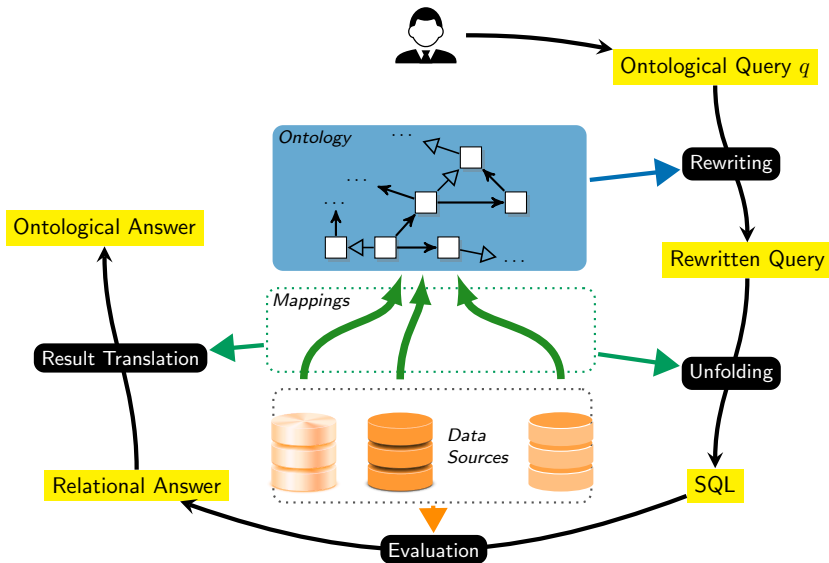
Query answering by rewriting (conceptual framework)



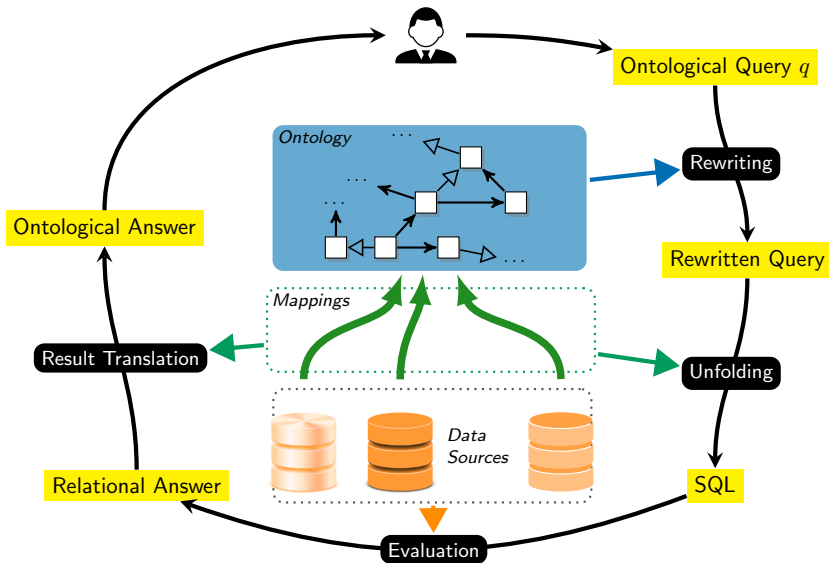
Query answering by rewriting (conceptual framework)



Query answering by rewriting (conceptual framework)



Query answering by rewriting (conceptual framework)



VKGs are by now a mature technology

Ontology-based querying of relational data sources is supported by several systems, both **open-source** and **commercial**:

- **Ontop**¹ – Free University of Bozen-Bolzano, Italy
- **Morph**² – Technical University of Madrid, Spain
- **Mastro**³ – Sapienza Università di Roma & OBDA systems SRL, Italy
- **Stardog**⁴ – Stardog Union, USA
- **Ultrawrap**⁵ – Capsenta, USA
- **Oracle Spatial and Graph**⁶ – Oracle, USA

¹<http://ontop.inf.unibz.it>

²<https://github.com/oeg-upm/morph-rdb>

³<http://www.obdasystems.com/it/mastro>

⁴<http://www.stardog.com>

⁵<https://capsenta.com/ultrawrap>

⁶<http://www.oracle.com/technetwork/database/options/spatialandgraph>



<http://ontop-vkg.org/>

- State-of-the-art VKG system developed at the Free University of Bozen-Bolzano.
- Compliant with all relevant Semantic Web standards:
RDF, RDFS, OWL 2 QL, R2RML, and SPARQL
- Supports all major relational DBs:
Oracle, DB2, MS SQL Server, Postgres, MySQL, Teiid, Dremio, Denodo, etc.
- **Open-source** and released under Apache 2 license.

- The development of *Ontop* has spanned the past decade.
- Developing such a system is highly non-trivial and requires both a theoretical investigation of the semantics and strong engineering efforts to implement all the required features.
- *Ontop* started as the PhD work of Mariano Rodriguez-Muro in 2009,
- SPARQL 1.0: 2008 & OWL 2 QL and R2RML: 2012.
- At that time, the VKG research focused on union of conjunctive queries (UCQs) as a query language.
- *Ontop* v1 relied on non-recursive Datalog as its core data structure because it perfectly fit the UCQ-based setting.

- The development of *Ontop* was boosted by the EU FP7 project Optique (2013–2016),
- during which the compliance with the relevant W3C recommendations became a priority, and significant progress was made in this direction.
- The last release of *Ontop* v1 was v1.18 in 2016
- 4.6K git commits

- A natural requirement that emerged during the Optique project were aggregates introduced in SPARQL 1.1.
- The *Ontop* development team spent a major effort, internally called *Ontop v2*, on implementing this query language feature.
- However, it became exceedingly clear that the Datalog representation was not well suited for this implementation.
- Some prototypes of *Ontop v2* were used in the Optique project for internal purposes, but never reached the level of a public release.

During the development of Ontop v2, as *Ontop* moved towards supporting the W3C recommendations for SPARQL and R2RML, we have identified the following new challenges, which turned out to be difficult to tackle in the Datalog setting:

- SPARQL is based on a rich algebra, which goes beyond the expressivity of CQs.
- Non-monotonic features like `OPTIONAL` and `MINUS`, cardinality-sensitive query modifiers (`DISTINCT`) and aggregation (`GROUP BY` with functions such as `SUM`, `AVG`, `COUNT`) are difficult to model even in extensions of Datalog.
- Even without SPARQL aggregation, cardinalities have to be treated carefully.

- SQL is *statically typed* in the sense that all values in a given relation column (both in the database and in the result of a query) have the same datatype.

SPARQL is *dynamically typed*: a variable can have values of different datatypes in different solution mappings.

- SQL queries with values of unexpected datatypes (e.g., a string as an argument for '+') are simply deemed incorrect.

In contrast, SPARQL treats such *type errors* as legitimate and handles them similarly to NULLs in SQL. For example,

```
?s ?p ?o FILTER (?o < 4)
```

retrieves all triples

- with a numerical object whose value is below 4
- (but *ignores* all triples with strings or IRIs, for example).

- SQL is *statically typed* in the sense that all values in a given relation column (both in the database and in the result of a query) have the same datatype.

SPARQL is *dynamically typed*: a variable can have values of different datatypes in different solution mappings.

- SQL queries with values of unexpected datatypes (e.g., a string as an argument for '+') are simply deemed incorrect.

In contrast, SPARQL treats such *type errors* as legitimate and handles them similarly to NULLs in SQL. For example,

```
?s ?p ?o FILTER (?o < 4)
```

retrieves all triples

- with a numerical object whose value is below 4
- (but *ignores* all triples with strings or IRIs, for example).

- the output datatype of a SPARQL function depends on the types or language tags of its arguments
 - e.g., if both arguments of '+' are `xsd:integer`, then so is the output,
 - if both arguments are `xsd:decimal`, then so is the output.
- To determine the output datatype of an *aggregate function* in SPARQL, one has to look at the datatypes of values in the group, which can vary from one group to another.

New design in Ontop v3 and v4: IQ

- *Ontop* v4 uses a variant of relational algebra tailored to encode SPARQL queries along the lines of the language described in [Xiao et al. 18]⁷.
- The language, called *Intermediate Query*, or IQ, is a uniform representation both for user SPARQL queries and for SQL queries from the mapping.
- When the query transformation (rewriting and unfolding) is complete, the IQ expression is converted into SQL, and executed by the underlying relational DBMS.

⁷Guohui Xiao, Roman Kontchakov, Benjamin Cogrel, Diego Calvanese, and Elena Botoeva. Efficient handling of SPARQL optional for OBDA. In ISWC, pages 354–373, 2018.

Example (DB and Mapping)

DB Tables (x is the primary key)

$T_1(x:\text{TEXT}, y:\text{INTEGER}), T_2(x:\text{TEXT}, y:\text{DECIMAL}), T_3(x:\text{TEXT}, y:\text{TEXT}), T_4(x:\text{TEXT}, y:\text{INTEGER})$

Mapping

$$T_1(x, y) \rightsquigarrow :b\{x\} :p \ y,$$

$$T_3(x, y) \rightsquigarrow :b\{x\} :p \ y,$$

$$T_2(x, y) \rightsquigarrow :b\{x\} :p \ y,$$

$$T_4(x, y) \rightsquigarrow :b\{x\} :q \ y,$$

Mapping in IQ

$$T_1(x, y) \rightsquigarrow \text{triple}(\text{rdf}(:b\{x\})(x), |R|, :p, \text{rdf}(\text{i2t}(y), \text{xsd:integer})),$$

$$T_2(x, y) \rightsquigarrow \text{triple}(\text{rdf}(:b\{x\})(x), |R|, :p, \text{rdf}(\text{d2t}(y), \text{xsd:decimal})),$$

$$T_3(x, y) \rightsquigarrow \text{triple}(\text{rdf}(:b\{x\})(x), |R|, :p, \text{rdf}(y, \text{xsd:string})),$$

$$T_4(x, y) \rightsquigarrow \text{triple}(\text{rdf}(:b\{x\})(x), |R|, :q, \text{rdf}(\text{i2t}(y), \text{xsd:integer})).$$

Example (DB and Mapping)

DB Tables (x is the primary key)

$T_1(x:\text{TEXT}, y:\text{INTEGER}), T_2(x:\text{TEXT}, y:\text{DECIMAL}), T_3(x:\text{TEXT}, y:\text{TEXT}), T_4(x:\text{TEXT}, y:\text{INTEGER})$

Mapping

$$T_1(x, y) \rightsquigarrow :b\{x\} :p \ y,$$

$$T_3(x, y) \rightsquigarrow :b\{x\} :p \ y,$$

$$T_2(x, y) \rightsquigarrow :b\{x\} :p \ y,$$

$$T_4(x, y) \rightsquigarrow :b\{x\} :q \ y,$$

Mapping in IQ

$$T_1(x, y) \rightsquigarrow \text{triple}(\text{rdf}(:b\{x\})(x), \text{IRI}, :p, \text{rdf}(\text{i2t}(y), \text{xsd:integer})),$$

$$T_2(x, y) \rightsquigarrow \text{triple}(\text{rdf}(:b\{x\})(x), \text{IRI}, :p, \text{rdf}(\text{d2t}(y), \text{xsd:decimal})),$$

$$T_3(x, y) \rightsquigarrow \text{triple}(\text{rdf}(:b\{x\})(x), \text{IRI}, :p, \text{rdf}(y, \text{xsd:string})),$$

$$T_4(x, y) \rightsquigarrow \text{triple}(\text{rdf}(:b\{x\})(x), \text{IRI}, :q, \text{rdf}(\text{i2t}(y), \text{xsd:integer})).$$

Example (SPARQL Query)

SPARQL

```
SELECT ?s WHERE { ?x :p ?n . ?x :q ?m .  
  BIND ((?n + ?m) AS ?s) FILTER (bound(?s)) }
```

SPARQL in IQ

```
PROJ?s?s/numeric-add(?n,?m)  
JOIN¬isNull(numeric-add(?n,?m))(triple(?x, :p, ?n), triple(?x, :q, ?m)),
```

Example (SPARQL Query)

SPARQL

```
SELECT ?s WHERE { ?x :p ?n . ?x :q ?m .  
  BIND ((?n + ?m) AS ?s) FILTER (bound(?s)) }
```

SPARQL in IQ

$$\text{PROJ}_{?s/\text{numeric-add}(?n,?m)}^{?s}$$
$$\text{JOIN}_{\neg \text{isNull}(\text{numeric-add}(?n,?m))}(\text{triple}(?x, :p, ?n), \text{triple}(?x, :q, ?m)),$$

Example: Unfolded Query w.r.t. Mapping

Unfolded Query

$$\begin{aligned} & \text{PROJ}_{?s/\text{numeric-add}(?n,?m)}^{?s} \text{ JOIN}_{\neg \text{isNull}(\text{numeric-add}(?n,?m))} \\ & \quad \text{UNION} \\ & \quad \text{PROJ}_{?x/\text{rdf}(:\text{b}\{ \}(y_1),\text{IRI}), ?n/\text{rdf}(\text{i2t}(z_1),\text{xsd:integer})}^{?x,?n} T_1(y_1, z_1) \\ & \quad \text{PROJ}_{?x/\text{rdf}(:\text{b}\{ \}(y_2),\text{IRI}), ?n/\text{rdf}(\text{d2t}(z_2),\text{xsd:decimal})}^{?x,?n} T_2(y_2, z_2) \\ & \quad \text{PROJ}_{?x/\text{rdf}(:\text{b}\{ \}(y_3),\text{IRI}), ?n/\text{rdf}(z_3,\text{xsd:string})}^{?x,?n} T_3(y_3, z_3) \\ & \quad \text{PROJ}_{?x/\text{rdf}(:\text{b}\{ \}(y_4),\text{IRI}), ?m/\text{rdf}(\text{i2t}(z_4),\text{xsd:integer})}^{?x,?m} T_4(y_4, z_4). \end{aligned}$$

This cannot be directly translated into SQL because of SPARQL functions (e.g. `numeric-add`). After a few steps of transformation and optimization, we obtain the following queries, which can be translated to SQL:

Final IQ

$$\begin{aligned} & \text{PROJ}_{?s/\text{rdf}(\text{IF}(p=1, \text{i2t}(\text{t2i}(v)+z_4), \text{d2t}(\text{t2d}(v)+\text{i2d}(z_4))), \text{IF}(p=1, \text{xsd:integer}, \text{xsd:decimal}))}^{?s} \\ & \quad \text{JOIN} \\ & \quad \text{UNION}(\text{PROJ}_{v/\text{i2t}(z_1), p/1}^{y,v,p} T_1(y, z_1), \text{PROJ}_{v/\text{d2t}(z_2), p/2}^{y,v,p} T_2(y, z_2)) \\ & \quad T_4(y, z_4) \end{aligned}$$

Example: Unfolded Query w.r.t. Mapping

Unfolded Query

$$\begin{aligned} & \text{PROJ}_{?s/\text{numeric-add}(?n,?m)}^{?s} \text{ JOIN } \neg \text{isNull}(\text{numeric-add}(?n,?m)) \\ & \quad \text{UNION} \\ & \quad \text{PROJ}_{?x/\text{rdf}(:\text{b}\{ \}(y_1),\text{IRI}), ?n/\text{rdf}(\text{i2t}(z_1),\text{xsd:integer})}^{?x,?n} T_1(y_1, z_1) \\ & \quad \text{PROJ}_{?x/\text{rdf}(:\text{b}\{ \}(y_2),\text{IRI}), ?n/\text{rdf}(\text{d2t}(z_2),\text{xsd:decimal})}^{?x,?n} T_2(y_2, z_2) \\ & \quad \text{PROJ}_{?x/\text{rdf}(:\text{b}\{ \}(y_3),\text{IRI}), ?n/\text{rdf}(z_3,\text{xsd:string})}^{?x,?n} T_3(y_3, z_3) \\ & \quad \text{PROJ}_{?x/\text{rdf}(:\text{b}\{ \}(y_4),\text{IRI}), ?m/\text{rdf}(\text{i2t}(z_4),\text{xsd:integer})}^{?x,?m} T_4(y_4, z_4). \end{aligned}$$

This cannot be directly translated into SQL because of SPARQL functions (e.g. `numeric-add`). After a few steps of transformation and optimization, we obtain the following queries, which can be translated to SQL:

Final IQ

$$\begin{aligned} & \text{PROJ}_{?s/\text{rdf}(\text{IF}(p=1, \text{i2t}(\text{t2i}(v)+z_4), \text{d2t}(\text{t2d}(v)+\text{i2d}(z_4))), \text{IF}(p=1, \text{xsd:integer}, \text{xsd:decimal}))}^{?s} \\ & \quad \text{JOIN} \\ & \quad \text{UNION}(\text{PROJ}_{v/\text{i2t}(z_1), p/1}^{y,v,p} T_1(y, z_1), \text{PROJ}_{v/\text{d2t}(z_2), p/2}^{y,v,p} T_2(y, z_2)) \\ & \quad T_4(y, z_4) \end{aligned}$$

SQL Dialects

- Unlike SPARQL with its standard syntax and semantics, SQL is more varied as DBMS vendors do not strictly follow the ANSI/ISO standard.
- Instead, many use specific datatypes and functions and follow different conventions, for example, for column and table identifiers and query modifiers;
- Even a common function CONCAT can behave differently: NULL-rejecting in MySQL, but not in PostgreSQL and Oracle.
- Support for the particular SQL dialect is thus essential for transforming SPARQL into SQL.

SQL Dialect Adapters in Ontop model

- their datatypes,
- their conventions in terms of attribute and table identifiers and query modifiers,
- the semantics of their functions,
- their restrictions on clauses such as WHERE and ORDER BY, and
- the structure of their data catalog.

SQL Dialects

- Unlike SPARQL with its standard syntax and semantics, SQL is more varied as DBMS vendors do not strictly follow the ANSI/ISO standard.
- Instead, many use specific datatypes and functions and follow different conventions, for example, for column and table identifiers and query modifiers;
- Even a common function CONCAT can behave differently: NULL-rejecting in MySQL, but not in PostgreSQL and Oracle.
- Support for the particular SQL dialect is thus essential for transforming SPARQL into SQL.

SQL Dialect Adapters in Ontop model

- their datatypes,
- their conventions in terms of attribute and table identifiers and query modifiers,
- the semantics of their functions,
- their restrictions on clauses such as WHERE and ORDER BY, and
- the structure of their data catalog.

Ontop v3 and v4 releases

- Using IQ, we have rewritten a large fragment of the code base of *Ontop*.
- After two beta releases in 2017 and 2018, we have released the stable version of *Ontop* v3 in 2019, which contains additional 4.5K commits with respect to *Ontop* v1.
- After *Ontop* v3, the development focussed on improving compliance and adding several major features.
- *Ontop* v4-beta-1, released in late 2019 (aggregates supports added), +1.5K commits.
- *Ontop* v4-rc-1 is released on 8 July (+0.7K git commits).
- *Ontop* v4 is planned by the end of July (11K+ git commits in total).

Ontop v3 and v4 releases

- Using IQ, we have rewritten a large fragment of the code base of *Ontop*.
- After two beta releases in 2017 and 2018, we have released the stable version of *Ontop* v3 in 2019, which contains additional 4.5K commits with respect to *Ontop* v1.
- After *Ontop* v3, the development focussed on improving compliance and adding several major features.
- *Ontop* v4-beta-1, released in late 2019 (aggregates supports added), +1.5K commits.
- *Ontop* v4-rc-1 is released on 8 July (+0.7K git commits).
- *Ontop* v4 is planned by the end of July (11K+ git commits in total).

Ontop v3 and v4 releases

- Using IQ, we have rewritten a large fragment of the code base of *Ontop*.
- After two beta releases in 2017 and 2018, we have released the stable version of *Ontop* v3 in 2019, which contains additional 4.5K commits with respect to *Ontop* v1.
- After *Ontop* v3, the development focussed on improving compliance and adding several major features.
- *Ontop* v4-beta-1, released in late 2019 (aggregates supports added), +1.5K commits.
- *Ontop* v4-rc-1 is released on 8 July (+0.7K git commits).
- *Ontop* v4 is planned by the end of July (11K+ git commits in total).

How to obtain Ontop v4?

- The command line interface and Protege plugin are at SourceForge⁸ and Github⁹.
- the Docker image for the SPARQL endpoint with the latest v4 development is available at Docker Hub¹⁰.
- Source code from Github¹¹.
- The documentation, including tutorials, is provided at the official website¹².

⁸<http://sourceforge.net/projects/ontop4obda/files/>

⁹<https://github.com/ontop/ontop/releases>

¹⁰<https://hub.docker.com/r/ontop/ontop-endpoint>

¹¹<https://github.com/ontop/ontop>

¹²<https://ontop-vkg.org/>

SPARQL 1.1 Compliance in Ontop v4

Section in SPARQL 1.1	Features	Coverage
5–7. Graph Patterns, etc.	BGP, FILTER, OPTIONAL, UNION	4/4
8. Negation	MINUS, FILTER [NOT] EXISTS	1/2
9. Property Paths	PredicatePath, InversePath, ZeroOrMorePath, ...	0
10. Assignment	BIND, VALUES	2/2
11. Aggregates	COUNT, SUM, MIN, MAX, AVG, GROUP_CONCAT, SAMPLE	6/6
12. Subqueries	Subqueries	1/1
13. RDF Dataset	GRAPH, FROM [NAMED]	1/2
14. Basic Federated Query	SERVICE	0
15. Solution Seqs. & Mods.	ORDER BY, SELECT, DISTINCT, REDUCED, OFFSET, LIMIT	6/6
16. Query Forms	SELECT, CONSTRUCT, ASK, DESCRIBE	4/4

unsupported features are ~~crossed out~~


SPARQL 1.1 Compliance in Ontop v4: Functions

Section in SPARQL 1.1	Features	Coverage
17.4.1. Functional Forms	BOUND, IF, COALESCE, [NOT] EXISTS, , &&, =, sameTerm, [NOT] IN	6/10
17.4.2. Fns. on RDF Terms	isIRI, isBlank, isLiteral, isNumeric, str, lang, datatype, IRI, BNODE, STRDT, STRLANG, UUID, STRUUID	9/13
17.4.3. Fns. on Strings	STRLEN, SUBSTR, UCASE, LCASE, STRSTARTS, STREND, STREND, CONTAINS, STRBEFORE, STRAFTER, ENCODE_FOR_URI, CONCAT, langMatches, REGEX, REPLACE	14/14
17.4.4. Fns. on Numerics	abs, round, ceil, floor, RAND	5/5
17.4.5. Fns. on Dates&Times	now, year, month, day, hours, minutes, seconds, timezone, tz	8/9
17.4.6. Hash Fns.	MD5, SHA1, SHA256, SHA384, SHA512	5/5
17.5. XPath Constructor Fns.	casting	0
17.6. Extensible Value Testing	user defined functions	0

* Regular expressions and Hash functions are partially supported because they have to be delegated to the database

- *Ontop* v4 has greatly improved its compliance with relevant W3C recommendations and provides good performance in query answering.
- It supports almost all the features of SPARQL 1.1, R2RML, OWL 2 QL, and SPARQL entailment regime, and the SPARQL 1.1 HTTP Protocol.
- Two recent independent evaluations of VKG systems have confirmed the robust performance of *Ontop*^{13,14}.
- When considering all the perspectives, like usability, completeness, and soundness, *Ontop* clearly stands out among the open-source systems.

¹³M. Chaloupka and M. Necasky. Using Berlin SPARQL benchmark to evaluate relational database virtual SPARQL endpoints. Submitted to SWJ, 2020.

¹⁴M. Namici and G. De Giacomo. Comparing query answering in OBDA tools over W3C-compliant specifications. In  Proc. DL, volume 2211. CEUR-WS.org, 2018.

Ontop Downloads

Downloads

32,627

2015-05-03 to 2020-07-05

Countries

Operating Systems

Download Statistics

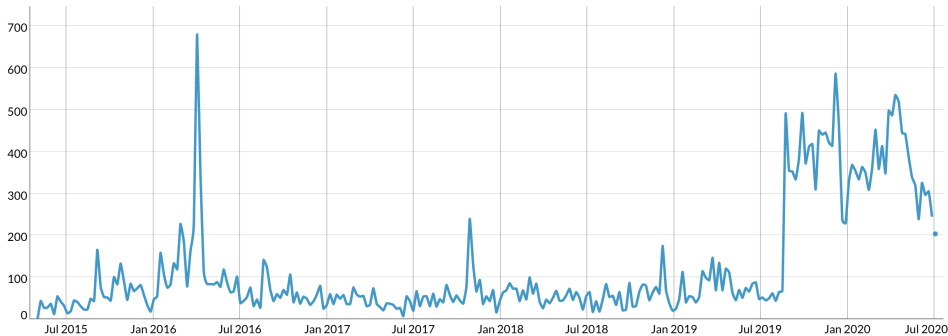
[All Files \(Change File\)](#)

Date Range: 2015-05-03 to 2020-07-05

Daily

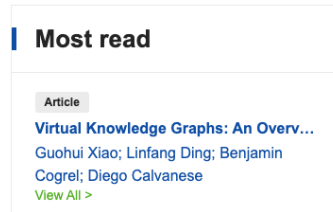
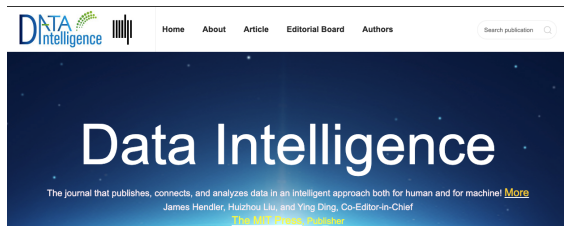
Weekly

Monthly



Use Cases of Ontop (and VKG in general)

- *Ontop* has been adopted in many academic and industrial use cases.
- However, due to its liberal Apache 2 license, it is essentially impossible to obtain a complete picture of all use cases and adoptions.
- Nevertheless, a few significant use cases have been summarized in a recent survey paper¹⁵.



¹⁵Guohui Xiao, Linfang Ding, Benjamin Cogrel, and Diego Calvanese. Virtual knowledge graphs: An overview of systems and use cases. *Data Intelligence*, 1:201–223, 2019.

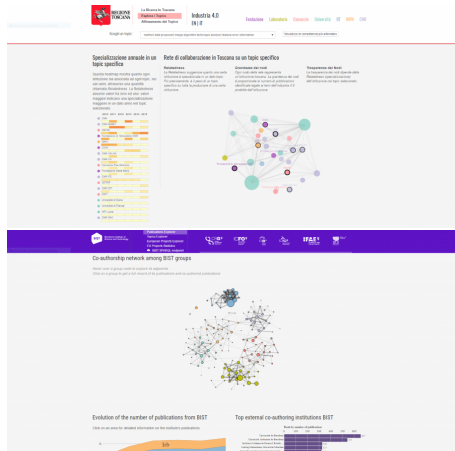
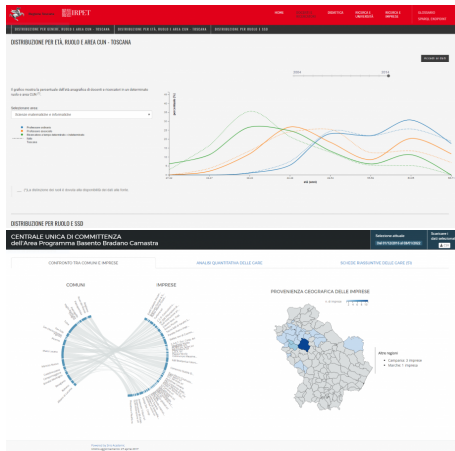
Some use cases of *Ontop* – Research projects

- 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
- EU H2020 project **INODE** “Intelligent Open Data Exploration” (11/2019 – 10/2022)
 - Development of techniques for the flexible interaction with data

Some use cases of *Ontop* – Commercial adoption

- **NOI Techpark in Bolzano** – Development of knowledge graph of South Tyrol tourism data
- **SIRIS Academic** (Barcelona) – Development of data integration and dashboards for data analysis over open data from public institutions
- **Siemens Corporate Technologies** (Munich) – Access to temporal and streaming data
- **Robert Bosch GmbH** (Stuttgart) – Analysis of log data for manufacturing processes
- **Metaphacts** (Germany) – Adoption of *Ontop* in their semantic data management platform
- **Fluxicon** (Milano)
- **Isagog** (Rome)

UNiCS UNiversity AnalytiCS platform, powered by Ontop(ic)



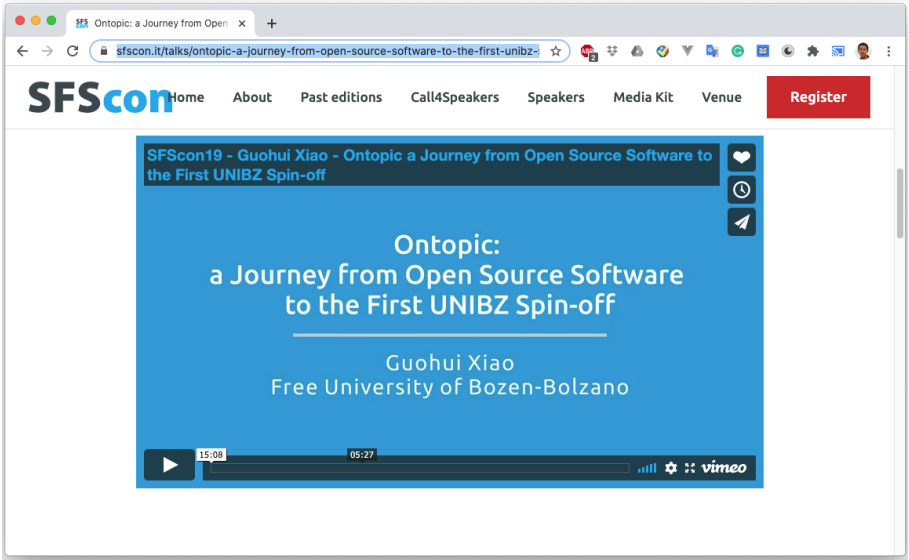
See the talk “UNiCS: The open data platform for Research and Innovation” by Alessandro Mosca (<https://www.inf.unibz.it/krd/sos-2020/>)



<https://ontopic.biz/>

Funded in April 2019 as the first spinoff of unibz.
Incubated at NOI Techpark.

- **Ontopic Suite** currently under development – Will be licenced
 - Ensures scalability, reliability, and cost-effectiveness at design/runtime of VKG solutions
 - Strong focus on usability
- **Technical services**
 - Technical support for Ontop and Ontopic suite
 - Customized developments
- **Consulting** on adoption of VKG solutions for data access and integration



`https://www.sfscon.it/talks/
ontopic-a-journey-from-open-source-software-to-the-first-unibz-spin-off/`

Thank you!

- E: xiao@inf.unibz.it
- H: <http://www.ghxiao.org>



- *Ontop* website: <http://ontop-vkg.org/>
- Github: <http://github.com/ontop/ontop/>
- Facebook: <https://www.facebook.com/obdaontop/>
- Twitter: @ontop4obda