Challenges in the Big Data era

The FOUR V’s of Big Data

Volume

- 40 Zettabytes: 43 trillion gigabytes of data will be created by 2020, an increase of 300 times from 2005
- 6 billion people have cell phones

Scale of Data

Variety

- 30 billion pieces of content are shared on Facebook every month
- 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

Velocity

- The New York Stock Exchange captures 1 TB of trade information during each trading session
- Modern cars have close to 100 sensors that monitor items such as fuel level and tire pressure
- By 2015, it is projected there will be 18.9 billion network connections – almost 2.5 connections per person on earth

Veracity

- 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

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 of 2011, the global size of data in healthcare was estimated to be 150 exabytes (161 billion gigabytes).

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, there will be 4.4 million IT jobs worldwide to support big data, with 1.9 million in the United States.

Diego Calvanese (UniBZ + UMU)
VGKs for Data Access and Integration
CSICC – 3/3/2021
Variety, not volume, is driving Big Data initiatives

MIT Sloan Management Review (28 March 2016)

Relative Importance

- Variety: 69%
- Volume: 25%
- Velocity: 6%

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 1,727 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:\(^1\):

- 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

\(^1\) SQL is the standard DB query language.
We would obtain the following SQL query:

```sql
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:

```sql
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'
```

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.

This is also very costly!

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

1 Motivations

2 Virtual Knowledge Graphs (VKGs) for data access

3 The Ontop system and the Ontopic spinoff

4 Ongoing and planned developments

5 Conclusions
Outline

1 Motivations

2 Virtual Knowledge Graphs (VKGs) for data access

3 The Ontop system and the Ontopic spinoff

4 Ongoing and planned developments

5 Conclusions
Greatly simplifies the access to information, and frees end-users from the need to know the precise structure of information sources.
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** $O$: expressed in **OWL 2 QL**  
   [W3C Rec. 2012]

3. **Mapping** $M$: 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.
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 \((A_1 \subseteq A_2)\)
Example: :MovieActor rdfs:subClassOf :Actor.
Inference: \(<\text{person/2}> \text{ rdf:type } :\text{MovieActor} .\)
\[\Rightarrow <\text{person/2}> \text{ rdf:type } :\text{Actor} .\]

Domain of properties: rdfs:domain \((\exists P \subseteq A)\)
Example: :playsIn rdfs:domain :MovieActor.
Inference: \(<\text{person/2}> :\text{playsIn} <\text{movie/3}> .\)
\[\Rightarrow <\text{person/2}> \text{ rdf:type } :\text{MovieActor} .\]

Range of properties: rdfs:range \((\exists P^- \subseteq A)\)
Example: :playsIn rdfs:range :Movie.
Inference: \(<\text{person/2}> :\text{playsIn} <\text{movie/3}> .\)
\[\Rightarrow <\text{movie/3}> \text{ rdf:type } :\text{Movie} .\]
Other constructs of OWL 2 QL

- Class disjointness
- Inverse properties
- 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. & De Giacomo 2005; Bergamaschi & Sartori 1992; Borgida 1995; C., Lenzerini & Nardi 1999; Lenzerini & Nobili 1990; Queralt et al. 2012].

```
SeriesActor ⊑ Actor
SeriesActor ⊑ ¬MovieActor
∃actsIn ⊑ Actor
∃actsIn ⊑ Play
MovieActor ⊑ ∃playsIn
playsIn ⊑ actsIn
...
```

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**.

```sparql
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.

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

SQL Query $\mapsto$ Class  
SQL Query $\mapsto$ Property

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

In the right-hand side of the mapping, we make use of **iri-templates**, which transform database values into object identifiers (IRIs).
Mapping language – Example

Ontology $O$:

Database $D$:

The mapping $M$ applied to database $D$ generates the virtual knowledge graph $V = M(D)$:
**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:**

A first-order interpretation 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**.
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 $\langle P, D \rangle$.

**Certain answers**

Given a VKG instance $\langle P, D \rangle$ and a query $q$ over it, the certain answers to $q$ are those answers that hold in all models of $\langle P, D \rangle$. 
**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 $\mathcal{P} = \langle O, M, S \rangle$.

**First-order rewritability** [Poggi et al. 2008]

A query $R$ is a **first-order rewriting** of a query $Q$ with respect to $\mathcal{P}$ if, for every source database $D$, certain answers to $Q$ over $\langle \mathcal{P}, D \rangle$ = answers to $R$ over $D$.

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 computing its rewriting, which is a SQL query, and evaluating it over the sources.
Query answering by query rewriting

Ontology

Mappings

Data Sources

Ontological Answer

Rewritten Query

Rewriting

Evaluation

Result Translation

Relational Answer

SQL

Unfolding

Ontological Query q

Rewritten Query

Evaluation

Result Translation

Relational Answer

 Ontology

Mappings

Data Sources

Ontological Answer

Rewritten Query

Rewriting

Ontological Query q

Rewritten Query
Outline

1. Motivations
2. Virtual Knowledge Graphs (VKGs) for data access
3. The Ontop system and the Ontopic spinoff
4. Ongoing and planned developments
5. Conclusions

https://ontop-vkg.org

- State-of-the-art system for OBDA and VKGs.
- Compliant with the relevant W3C standard (RDF, OWL 2 QL, R2RML, and SPARQL).
- 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.
  - Major v4 just released.
  - Already well established:
    - +200 members in the mailing list
    - +14000 downloads in the last year.
  - Main development carried out in the context of several local, national, and EU projects, and at a university spinoff.
Architecture of **Ontop**

**Application Layer**
- Protege
- RDF4J Workbench & SPARQL Endpoint
- CLI (Docker)

**API Layer**
- OWL API
- RDF4J API

**Ontop Core**
- Ontop SPARQL Query Answering Engine
- OWL API (OWL Parser)
- R2RML API
- JDBC
- RDF4J API (SPARQL Parser)

**Inputs**
- OWL 2 QL Ontologies
- R2RML Mappings
- Relational Databases
- SPARQL Queries
Ontop plugin available from Protégé plugin repository
Mapping editor in Protégé
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., 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 currently 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

See [Xiao, Ding, et al. 2019] for a survey on VKG systems and use cases.
SIRIS Academic – UNiCS UNiversity AnalytiCS platform

http://www.sirisacademic.com/wb/siris-lab/unics/
The *Ontopic* spinoff of unibz

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

- **Ontopic Suite** currently under development.
  - Ensures scalability, reliability, and cost-efficiency at design and runtime of VKG solutions.
  - Strong focus on usability.

- **Technical services**
  - Technical support for Ontop and Ontopic Suite.
  - Customized developments.

- **Consulting** on adoption of VKG-based solutions for data access and integration.
<table>
<thead>
<tr>
<th></th>
<th>Motivations</th>
<th>Virtual Knowledge Graphs (VKGs) for data access</th>
<th>The <em>Ontop</em> system and the <em>Ontopic</em> spinoff</th>
<th>Ongoing and planned developments</th>
<th>Conclusions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Motivations</td>
<td>Virtual Knowledge Graphs (VKGs) for data access</td>
<td>The <em>Ontop</em> system and the <em>Ontopic</em> spinoff</td>
<td>Ongoing and planned developments</td>
<td>Conclusions</td>
</tr>
<tr>
<td>2</td>
<td>Virtual Knowledge Graphs (VKGs) for data access</td>
<td>The <em>Ontop</em> system and the <em>Ontopic</em> spinoff</td>
<td>Ongoing and planned developments</td>
<td>Conclusions</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>The <em>Ontop</em> system and the <em>Ontopic</em> spinoff</td>
<td>Ongoing and planned developments</td>
<td>Conclusions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Ongoing and planned developments</td>
<td>Conclusions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Conclusions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Support data analytics in VKGs

Supporting data analytics is currently a top priority for us.

Main challenges addressed in Ontop v4:

- **Semantics:** computing aggregation functions correctly, in particular those depending on cardinalities (SUM, COUNT, AVG) – bag vs. set semantics is an issue.

- **Performance:** efficient computation of aggregates, by delegating their execution to the database whenever possible.

- **Expressiveness:** support user-defined aggregation functions beyond the ones in SPARQL 1.1 (Ongoing).
The base version of Ontop, does not provide any information about how query answers are constructed.

In many cases, we are interested in:
- which data from which relation/source has been used to obtain an answer
- which mappings have been activated
- which ontology axioms have contributed to the answer

We have developed a framework for provenance/explanation in VKGs, building on provenance semi-rings in relational databases.

We have a prototype extension of Ontop that supports this framework.

We are currently incorporating the framework in the latest release of Ontop.
Spatial data play an important role in many scenarios.

**Geo-spatial extension on *Ontop***

- *Ontop* 4 provides full support for accessing geospatial data.
- Supports GeoSPARQL query language standardized by Open Geospatial Consortium (OGC).
- Translates GeoSPARQL functions into functions supported by PostGIS.
- Use cases: urban development, land management, disaster management.
noSQL data sources [Botoeva, C., Cogrel, Corman, et al. 2019]

Prototype extension of *Ontop* over MongoDB databases.

**MongoDB**

- Most popular noSQL DBMS.
- Stores data as collections of **JSON** documents.
- Comes with an expressive (low-level) query language: Mongo Aggregate Queries.

Benefits of virtual VKGs over MongoDB:

- **Interface**: higher-level query language (SPARQL) for the end-user.
- **Performance**: *Ontop* delegates query execution to the MongoDB engine ⇒ leverages document-based storage.
- Query translation relies on a correspondence between nested-relational algebra and Mongo Aggregate Queries [Botoeva, C., Cogrel & Xiao 2018].
Temporal data plays an important role in many scenarios.

- Example 1: find all transactions from a same account that are in two different locations with a distance longer than 1000 km and within 5 min.

- Example 2: find all customers with at least 3 temporal overlapping loans within the last 5 years.

**Ontop-temporal**

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

1 Motivations

2 Virtual Knowledge Graphs (VKGs) for data access

3 The Ontop system and the Ontopic spinoff

4 Ongoing and planned developments

5 Conclusions
Conclusions

- VKGs are by now a mature technology to address the challenges related to data access and integration.

- It has been well-investigated and applied in real-world scenarios mostly for the case of relational data sources.

- Also in that setting, performance and scalability w.r.t. larger datasets (volume), larger and more complex ontologies (variety, veracity), and multiple heterogeneous data sources (variety, volume) is a challenge.

- Recently VKGs have been investigated for alternative types of data, such as temporal data, noSQL and tree structured data, linked open data, and geo-spatial data.

- Performance and scalability are even more critical for these more complex domains.
Further research directions

Theoretical investigations:

- Dealing with data inconsistency and incompleteness – Data quality!
- Addressing privacy and security issues.
- Ontology-based update.
- Coping with evolution of data in the presence of ontological constraints.

From a practical point of view, supporting technologies need to be developed to make the VKG technology easier to adopt:

- Improving the support for multiple, heterogeneous data sources.
- Techniques for (semi-)automatic extraction/learning of ontology axioms and mapping assertions [C., Gal, et al. 2020].
- Techniques and tools for efficient management of mappings and ontology axioms, to support design, maintenance, and evolution \(\leadsto\) Ontopic Suite
- User-friendly ontology querying modalities (graphical languages, natural language queries).
Thank you!
A great thank you to all my collaborators

Elena Botoeva  Julien Cormann  Linfang Ding  Elem Güzel  Davide Lanti  Marco Montali  Alessandro Mosca  Mariano Rodriguez Muro  Guohui Xiao

Technion Haifa

Avigdor Gal  Roe Shraga

Ontopic s.r.l.

Benjamin Cogrel  Sarah Komla Ebri

U. Roma “La Sapienza”

Giuseppe De Giacomo  Domenico Lembo  Maurizio Lenzerini  Riccardo Rosati  Antonella Poggi

VKGs

Birkbeck College London

Roman Kontchakov  Vladislav RyzhikovZakharyaschev  Michael Zakharyaschev

Diego Calvanese (UniBZ + UMU)
References


References V


