Mapping Patterns for Virtual Knowledge Graphs

Diego Calvanese

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

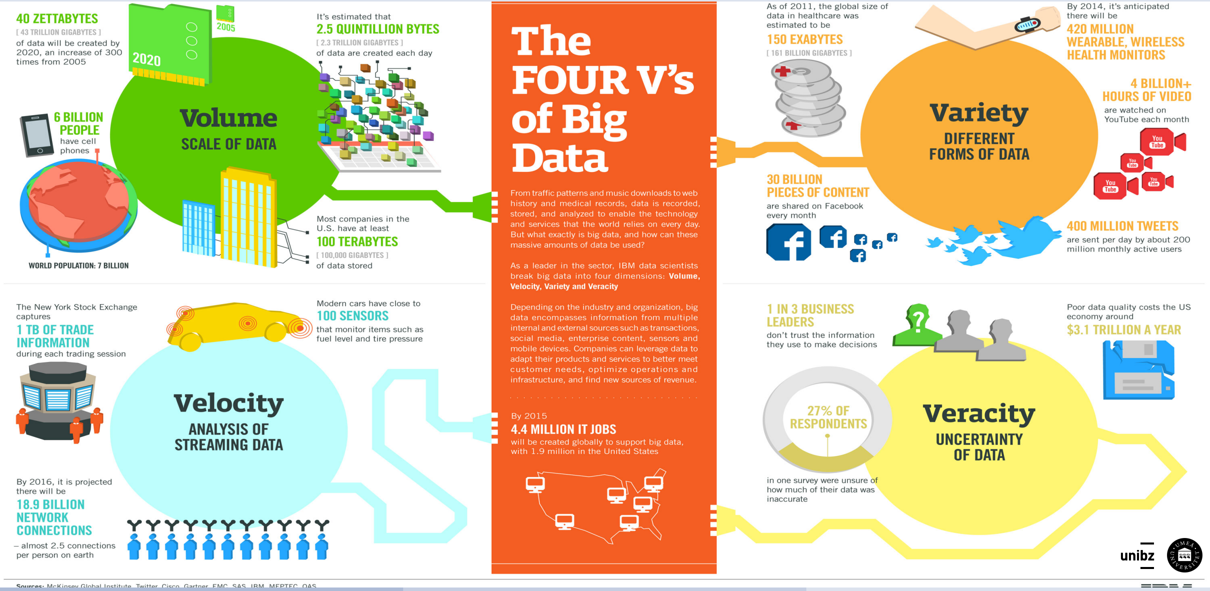
Department of Computing Science
Umeå University, Sweden

International Conference on Conceptual Structures (ICCS 2021)

22 September 2021 – Virtual

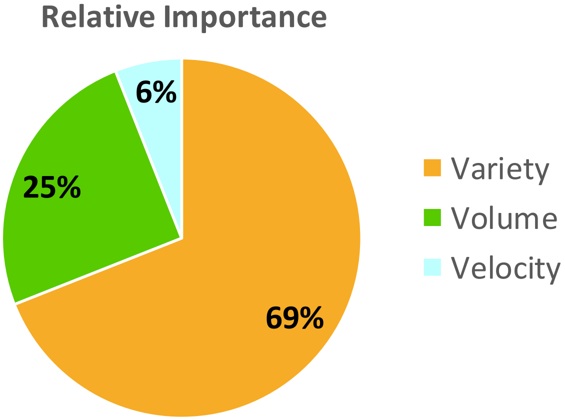
Outline of the presentation

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/

The problem of accessing relevant data!

- Every day, **huge amounts of data** are produced by various actors.
- **Effective access to such data** is crucial to make value out of them.

Key issue

Queries are hard to formulate and navigating data through conventional tools is becoming an increasingly hard, and mostly ineffective, challenge.

Accessing heterogeneous data at Statoil

(Use case from FP7 IP Optique)

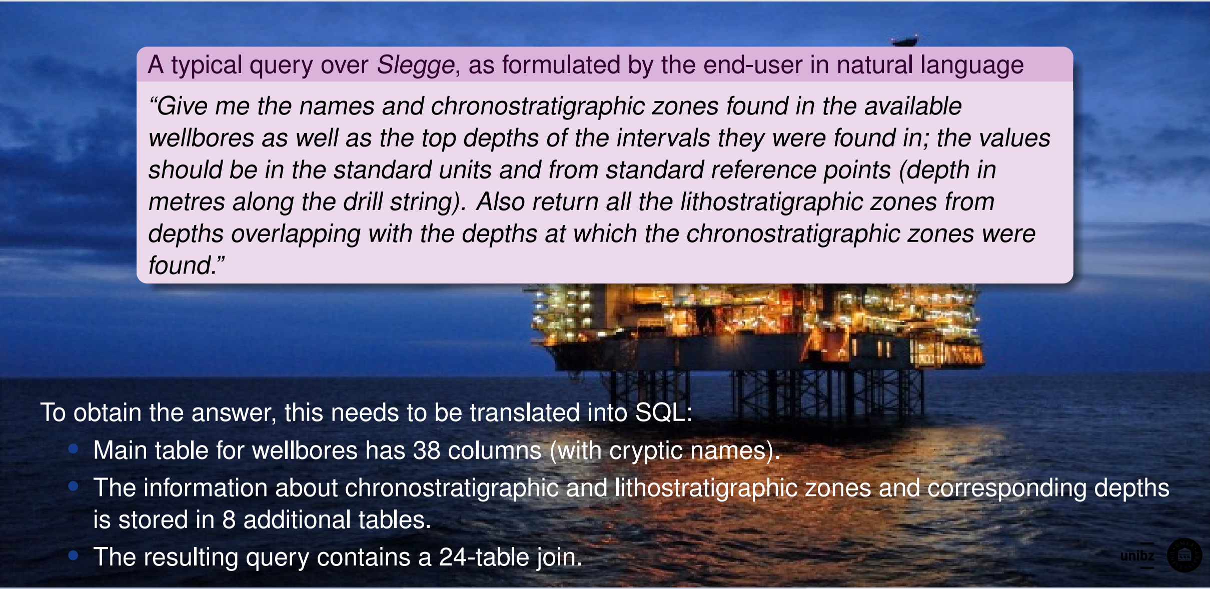
Statoil (now Equinor) Exploration

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

The humongous queries over *Slegge*



A typical query over *Slegge*, as formulated by the end-user in natural language

"Give me the names and chronostratigraphic zones found in the available wellbores as well as the top depths of the intervals they were found in; the values should be in the standard units and from standard reference points (depth in metres along the drill string). Also return all the lithostratigraphic zones from depths overlapping with the depths at which the chronostratigraphic zones were found."

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

- Main table for wellbores has 38 columns (with cryptic names).
- The information about chronostratigraphic and lithostratigraphic zones and corresponding depths is stored in 8 additional tables.
- The resulting query contains a 24-table join.

The humongous queries over *Slegge*

```
SELECT
 QVIEW1."IDENTIFIER" AS "wellbore",
 QVIEW2."STRAT_UNIT_IDENTIFIER" AS "chronostrat_unit",
 QVIEW15."STRAT_UNIT_IDENTIFIER" AS "lithostrat_unit",
 QVIEW2."STRAT_ZONE_ENTRY_MD" AS "top_md_m"
FROM
 "WELLBORE" QVIEW1,
 "STRATIGRAPHIC_ZONE" QVIEW2,
 "ROCK_FEATURE" QVIEW3,
 "COMPONENT_MATERIAL" QVIEW4,
 "DATA_COLLECTION" QVIEW5,
 "DATA_COLLECTION_CONTENT" QVIEW6,
 "ROCK_FEATURE" QVIEW7,
 "MATERIAL_CLASS" QVIEW8,
 "CLASSIFICATION_SYSTEM" QVIEW9,
 "DATA_COLLECTION_CONTENT" QVIEW10,
 "MATERIAL_CLASSIFICATION" QVIEW11,
 "STRATIGRAPHIC_ZONE" QVIEW15,
 "ROCK_FEATURE" QVIEW16,
 "COMPONENT_MATERIAL" QVIEW17,
 "DATA_COLLECTION" QVIEW18,
 "DATA_COLLECTION_CONTENT" QVIEW19,
 "ROCK_FEATURE" QVIEW20,
 "MATERIAL_CLASS" QVIEW21,
 "CLASSIFICATION_SYSTEM" QVIEW22,
 "DATA_COLLECTION_CONTENT" QVIEW23,
 "MATERIAL_CLASSIFICATION" QVIEW24
WHERE
 QVIEW1."REF_EXISTENCE_KIND" = 'actual' AND
 QVIEW1."IDENTIFIER" IS NOT NULL AND
 QVIEW1."IDENTIFIER" = QVIEW2."WELLBORE" AND
 QVIEW2."STRAT_ZONE_DEPTH_UOM" = 'm' AND
 QVIEW2."STRAT_COLUMN_IDENTIFIER" IS NOT NULL AND
 QVIEW2."STRAT_INTERP_VERSION" IS NOT NULL AND
 QVIEW2."STRAT_ZONE_IDENTIFIER" IS NOT NULL AND
 QVIEW2."STRAT_UNIT_IDENTIFIER" IS NOT NULL AND
 QVIEW2."STRAT_UNIT_IDENTIFIER" = QVIEW3."DESCRIPTION" AND
 QVIEW4."ENTITY_TYPE_NAME" = 'COMPONENT_MATERIAL' AND
 QVIEW3."ROCK_FEATURE_S" = QVIEW4."INCORPORATE_S" AND
QVIEW3."ROCK_FEATURE_S" = QVIEW6."COLLECTION_PART_S" AND
QVIEW5."DATA_COLLECTION_S" = QVIEW6."PART_OF_S" AND
QVIEW2."STRAT_COLUMN_IDENTIFIER" = QVIEW5."NAME" AND
QVIEW5."REF_DATA_COLLECTION_TYPE" = 'stratigraphic hierarchy' AND
QVIEW9."KIND" = 'chronostratigraphy' AND
QVIEW8."CLASSIFICATION_SYSTEM" = QVIEW9."NAME" AND
QVIEW7."ROCK_FEATURE_S" = QVIEW10."COLLECTION_PART_S" AND
QVIEW5."DATA_COLLECTION_S" = QVIEW10."PART_OF_S" AND
QVIEW7."ROCK_FEATURE_S" = QVIEW11."MATERIAL_S" AND
QVIEW8."MATERIAL_CLASS_S" = QVIEW11."MATERIAL_CLASS_S" AND
QVIEW2."STRAT_ZONE_ENTRY_MD" IS NOT NULL AND
QVIEW1."IDENTIFIER" = QVIEW15."WELLBORE" AND
QVIEW15."STRAT_ZONE_DEPTH_UOM" = 'm' AND
(((QVIEW15."STRAT_ZONE_EXIT_MD" >=
    QVIEW2."STRAT_ZONE_ENTRY_MD") AND
  (QVIEW15."STRAT_ZONE_ENTRY_MD" <=
    QVIEW2."STRAT_ZONE_EXIT_MD")) OR
 ((QVIEW2."STRAT_ZONE_EXIT_MD" >=
    QVIEW15."STRAT_ZONE_ENTRY_MD") AND
  (QVIEW2."STRAT_ZONE_ENTRY_MD" <=
    QVIEW15."STRAT_ZONE_EXIT_MD")
  )) AND
QVIEW15."STRAT_COLUMN_IDENTIFIER" IS NOT NULL AND
QVIEW15."STRAT_INTERP_VERSION" IS NOT NULL AND
QVIEW15."STRAT_ZONE_IDENTIFIER" IS NOT NULL AND
QVIEW15."STRAT_UNIT_IDENTIFIER" IS NOT NULL AND
QVIEW15."STRAT_UNIT_IDENTIFIER" = QVIEW16."DESCRIPTION" AND
QVIEW17."ENTITY_TYPE_NAME" = 'COMPONENT_MATERIAL' AND
QVIEW16."ROCK_FEATURE_S" = QVIEW17."INCORPORATE_S" AND
QVIEW15."STRAT_COLUMN_IDENTIFIER" = QVIEW18."NAME" AND
QVIEW18."REF_DATA_COLLECTION_TYPE" = 'stratigraphic hierarchy' AND
QVIEW16."ROCK_FEATURE_S" = QVIEW19."COLLECTION_PART_S" AND
QVIEW18."DATA_COLLECTION_S" = QVIEW19."PART_OF_S" AND
QVIEW22."KIND" = 'lithostratigraphy' AND
QVIEW21."CLASSIFICATION_SYSTEM" = QVIEW22."NAME" AND
QVIEW20."ROCK_FEATURE_S" = QVIEW23."COLLECTION_PART_S" AND
QVIEW18."DATA_COLLECTION_S" = QVIEW23."PART_OF_S" AND
QVIEW20."ROCK_FEATURE_S" = QVIEW24."MATERIAL_S" AND
QVIEW21."MATERIAL_CLASS_S" = QVIEW24."MATERIAL_CLASS_S"
QVIEW21."MATERIAL_CLASS_S" = QVIEW24."MATERIAL_CLASS_S"
```

The humongous queries over *Slegge*

```
SELECT
 QVIEW1."IDENTIFIER" AS "wellbore",
 QVIEW2."STRAT_UNIT_IDENTIFIER" AS "chronostrat_unit",
 QVIEW15."STRAT_UNIT_IDENTIFIER" AS "lithostrat_unit",
 QVIEW2."STRAT_ZONE_ENTRY_MD" AS "top_md_m"
FROM
 "WELLBORE" QVIEW1,
 "STRATIGRAPHIC_ZONE" QVIEW2,
```

```
QVIEW3."ROCK_FEATURE_S" = QVIEW6."COLLECTION_PART_S" AND
QVIEW5."DATA_COLLECTION_S" = QVIEW6."PART_OF_S" AND
QVIEW2."STRAT_COLUMN_IDENTIFIER" = QVIEW5."NAME" AND
QVIEW5."REF_DATA_COLLECTION_TYPE" = 'stratigraphic hierarchy' AND
QVIEW9."KIND" = 'chronostratigraphy' AND
QVIEW8."CLASSIFICATION_SYSTEM" = QVIEW9."NAME" AND
QVIEW7."ROCK_FEATURE_S" = QVIEW10."COLLECTION_PART_S" AND
QVIEW5."DATA_COLLECTION_S" = QVIEW10."PART_OF_S" AND
```

The formulation of such a query can require days of interaction between the IT personnel (database experts) and the end-users (domain experts).

```
 "DATA_COLLECTION" QVIEW18,
 "DATA_COLLECTION_CONTENT" QVIEW19,
 "ROCK_FEATURE" QVIEW20,
```

```
    QVIEW15."STRAT_ZONE_EXIT_MD")
  )) AND
QVIEW15."STRAT_COLUMN_IDENTIFIER" IS NOT NULL AND
```

This is also very costly!

Equinor loses **50.000.000€** per year

only due to this problem!!

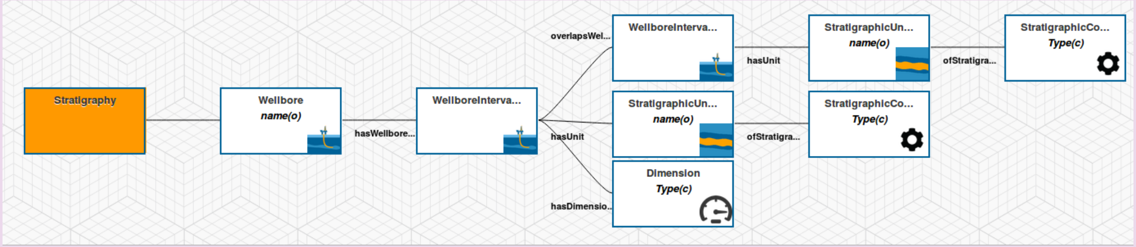
```
 QVIEW4."ENTITY_TYPE_NAME" = 'COMPONENT_MATERIAL' AND
 QVIEW3."ROCK_FEATURE_S" = QVIEW4."INCORPORATE_S" AND
```

```
QVIEW20."ROCK_FEATURE_S" = QVIEW24."MATERIAL_S" AND
QVIEW21."MATERIAL_CLASS_S" = QVIEW24."MATERIAL_CLASS_S"
QVIEW21."MATERIAL_CLASS_S" = QVIEW24."MATERIAL_CLASS_S"
```

A not-so humongous equivalent query

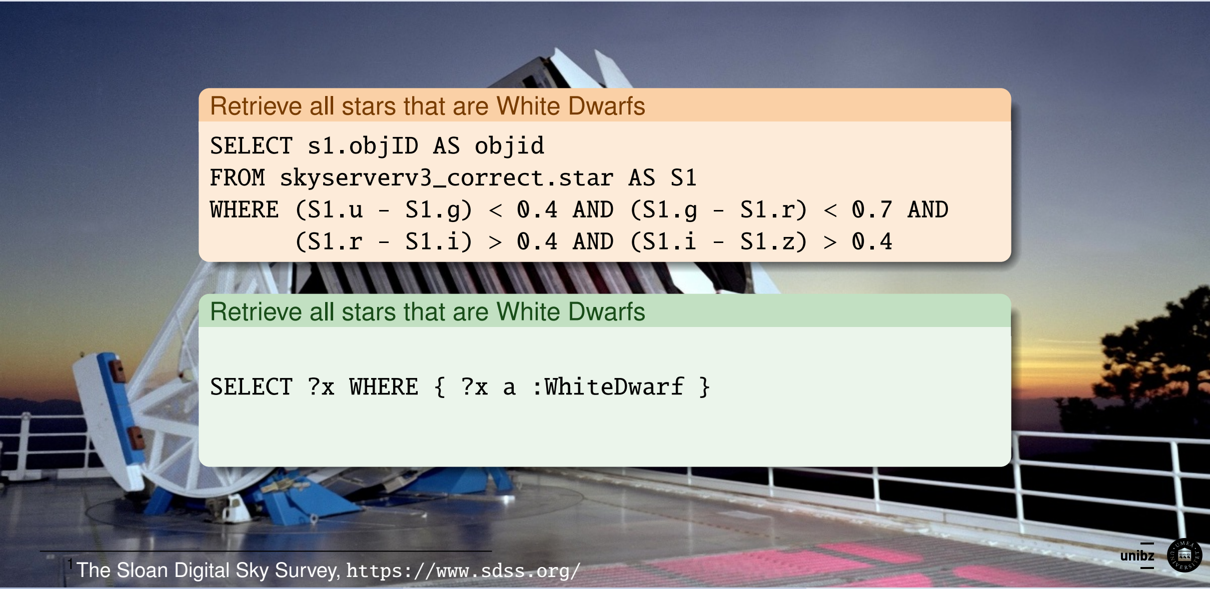
The FP7 IP Optique solution

```
SELECT ?wellbore ?chronostrat_unit ?top_md_m ?lithostrat_unit
WHERE {
  ?w a :Wellbore ;
    :name ?wellbore ;
```




```
  ?litho_intv :hasUnit ?lu .
  ?lu :name ?lithostrat_unit ;
    :ofStratigraphicColumn [ a :LithoStratigraphicColumn ] .
}
```

Uninformative names from SDSS[1] (Use case from H2020 RIA INODE)



Retrieve all stars that are White Dwarfs

```
SELECT s1.objID AS objid
FROM skyserverv3_correct.star AS S1
WHERE (S1.u - S1.g) < 0.4 AND (S1.g - S1.r) < 0.7 AND
      (S1.r - S1.i) > 0.4 AND (S1.i - S1.z) > 0.4
```

Retrieve all stars that are White Dwarfs

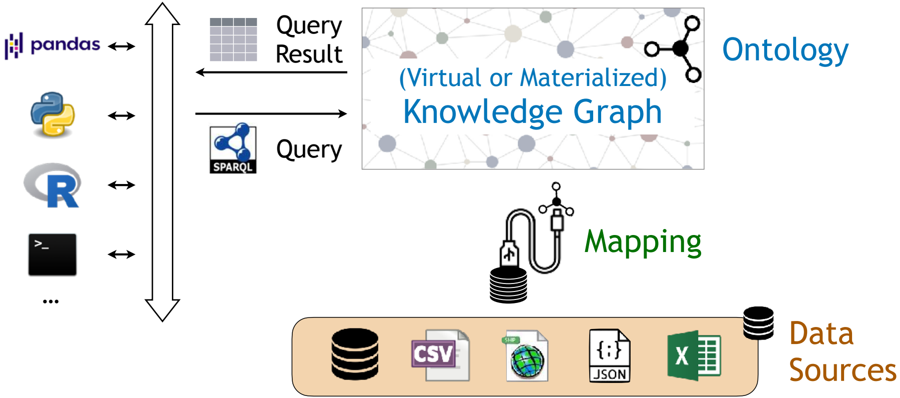
```
SELECT ?x WHERE { ?x a :WhiteDwarf }
```

[1] The Sloan Digital Sky Survey, https://www.sdss.org/

Outline

Outline

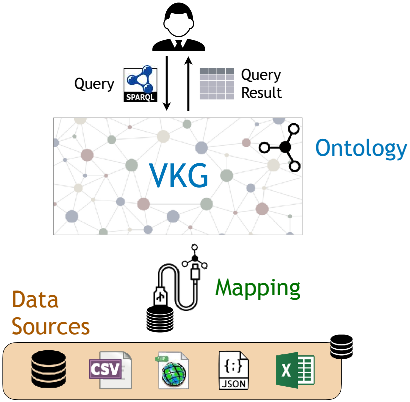
(Virtual) Knowledge Graphs for data access



Components of the VKG framework

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

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

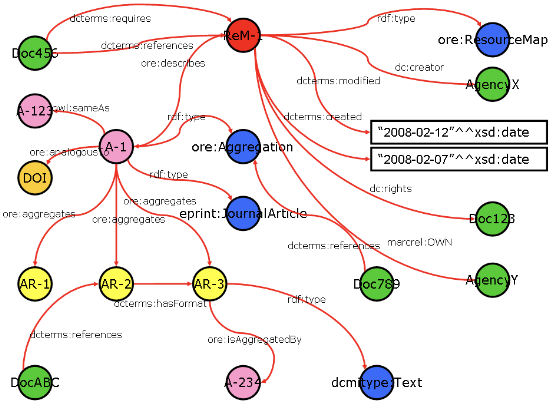


The W3C has standardized languages suitable for VKGs:

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

RDF – Data is represented as a graph

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



Class membership:

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

Data property:

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

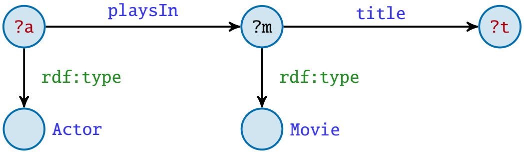
Object property:

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

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)
- ...

The OWL 2 QL ontology language

- **OWL 2 QL** is one of the three standard sub-languages of the very expressive standard ontology language OWL 2. [W3C Rec. 2012]
- It is considered a lightweight ontology language:
  - controlled expressive power
  - efficient inference
- Optimized for accessing large amounts of data
  - Queries over the ontology can be rewritten into SQL queries over the underlying relational database (First-order rewritability).
  - Logical consistency of ontology and data can also be checked by executing SQL queries over the underlying database.

Constructs of OWL 2 QL

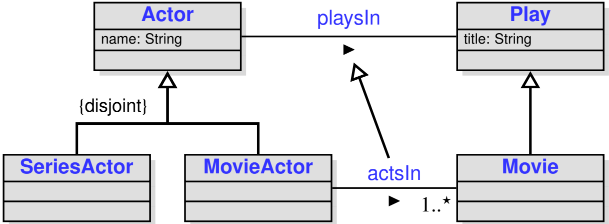
In an OWL 2 QL ontology, one can express knowledge about the classes and properties in the domain of interest by means of various types of assertions.

- Subclass assertions — $\mathsf{MovieActor} \sqsubseteq \mathsf{Actor}$
- Class disjointness — $\mathsf{Actor} \sqsubseteq \neg\mathsf{Movie}$
- Domain of a property — $\exists\mathsf{actsIn} \sqsubseteq \mathsf{MovieActor}$
- Range of a property — $\exists\mathsf{actsIn}^{-} \sqsubseteq \mathsf{Movie}$
- Mandatory participation to a property — $\mathsf{MovieActor} \sqsubseteq \exists\mathsf{actsIn}$
- Subproperty assertions — $\mathsf{actsIn} \sqsubseteq \mathsf{playsIn}$
- Inverse properties — $\mathsf{actsIn} \equiv \mathsf{hasActor}^{-}$

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].

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



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

Use of mappings

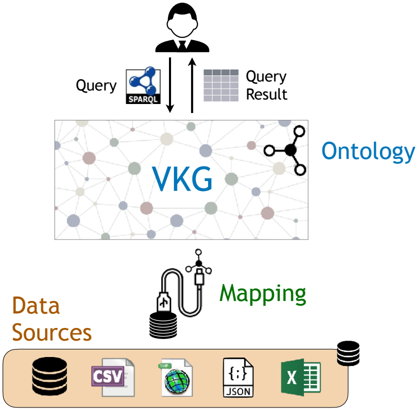
In the (V)KG framework, the mapping encodes how the data in the sources should be used to create the (virtual) knowledge graph, which is formulated in the vocabulary of the ontology.

(V)KG defined from the mapping and the data.

- Queries are answered with respect to the ontology and the data of the (V)KG.
- Such data is defined in terms of the data sources and the mapping.

In the virtual setting:

- The data of the VKG is not materialized.
- Instead, the information in the ontology and the mapping is used to translate queries over the ontology into queries formulated over the sources.
- The graph is **always up to date** wrt the data sources.



Mapping language

The mapping consists of a set of assertions of the form

$$\begin{array}{lcl} \Phi(\vec{x}) & \rightsquigarrow & C(\mathbf{iri}(\vec{x})) \\ \Phi(\vec{x}) & \rightsquigarrow & p(\mathbf{iri}_1(\vec{x}), \mathbf{iri}_2(\vec{x})) \end{array}$$

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

Intuition behind the mapping

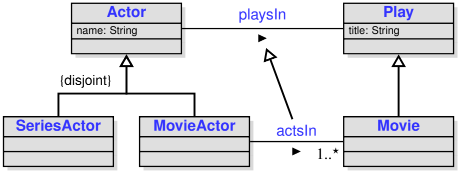
The answers returned by the SQL query in the left-hand side are used to create the objects (and values) that populate the class / property in the right-hand side.

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

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

Mapping language – Example

Ontology $\mathcal{O}$:



Mapping $\mathcal{M}$:

m_1: **SELECT** `mcode, mtitle` **FROM** `MOVIE`
WHERE `type = "m"`
$\rightsquigarrow$ Movie($\mathbf{iri}_m$(`mcode`))
title($\mathbf{iri}_m$(`mcode`), `mtitle`)

m_2: **SELECT** `M.mcode, A.acode` **FROM** `MOVIE M, ACTOR A`
WHERE `M.mcode = A.pcode` **AND** `M.type = "m"`
$\rightsquigarrow$ actsIn($\mathbf{iri}_a$(`acode`), $\mathbf{iri}_m$(`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 $\mathcal{V} = \mathcal{M}(\mathcal{D})$:

Movie($\mathbf{iri}_m$(5118)) title($\mathbf{iri}_m$(5118), "The Matrix")
Movie($\mathbf{iri}_m$(2281)) title($\mathbf{iri}_m$(2281), "Blade Runner")
actsIn($\mathbf{iri}_a$(438), $\mathbf{iri}_m$(5118)) actsIn($\mathbf{iri}_a$(572), $\mathbf{iri}_m$(5118)) actsIn($\mathbf{iri}_a$(271), $\mathbf{iri}_m$(2281))

Outline

Who provides the ontology?

- Designing an ontology is not an easy task.
- In many domains (e.g., the biomedical one) ontologies are developed independently by trained experts and already available to be re-used.
- Having "standardized ontologies" enables interoperability across different data sources.
- However, ontology design is a well investigated task, and methodologies and supporting tools are readily available. See, e.g.,
  - the series of *Workshops on Ontology Design Patterns* [`http://ontologydesignpatterns.org/`];
  - the OntoClean methodology for ontology analysis based on formal, domain-independent properties of classes [Guarino & Welty 2009].

Who provides the mappings?

(V)KG 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 for schema matching [Rahm & Bernstein 2001] and ontology matching [Euzenat & Shvaiko 2007].

As a consequence:

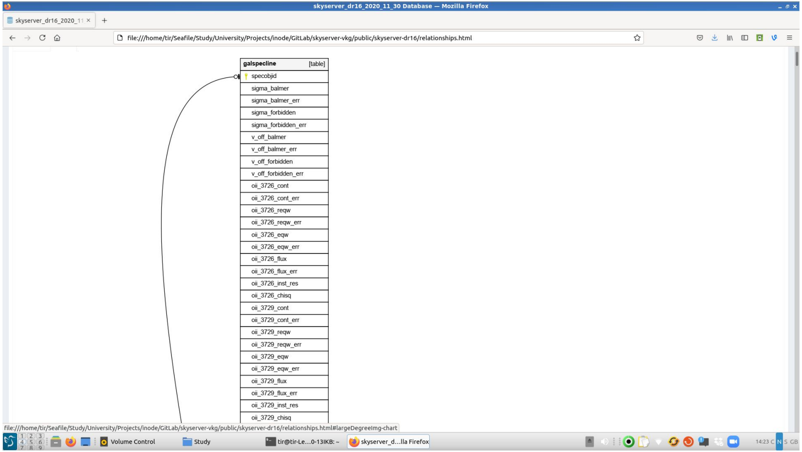
- Management of VKG mappings is an essentially manual effort that is **labor-intensive** and **error-prone**.
- Requires highly-skilled professionals [Spanos, Stavrou & Mitrou 2012].
- Writing mappings is challenging in terms of semantics, correctness, and performance.

Designing and managing mappings is the most critical bottleneck
for the adoption of the VKG approach.

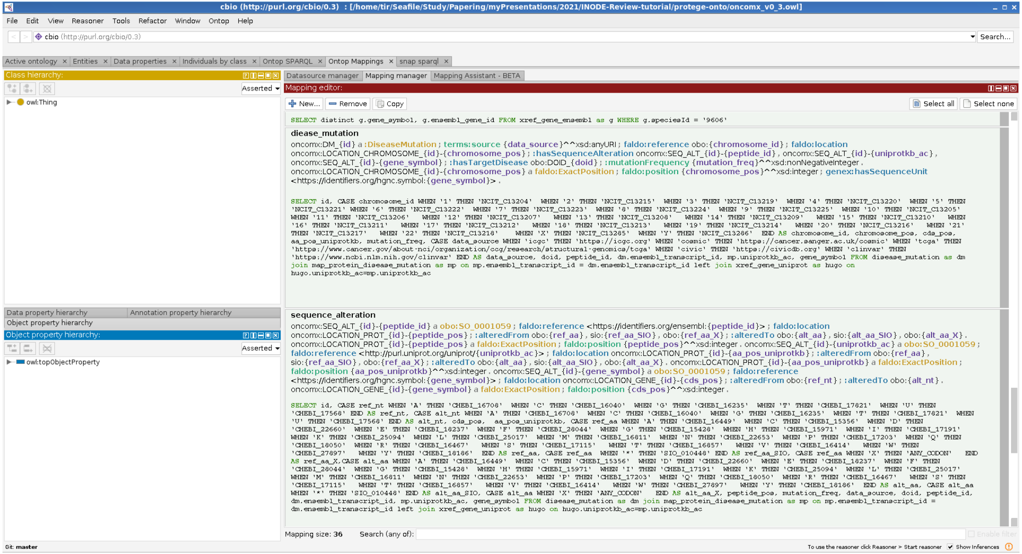
Who provides the mapping?

Writing mappings manually is a **time-consuming** and **error-prone** task.

Who provides the mapping?



Who provides the mapping?



Outline

Mapping patterns

In relational database design, **well-established conceptual modeling principles** and **methodologies** are usually employed.

- The resulting schema should suitably reflects the application domain at hand.
- This design phase relies on semantically-rich representations such as ER diagrams.
- However, these representations, typically:
  - get lost during deployment, since they are not conveyed together with the database itself, or
  - quickly get outdated due to continuous adjustments triggered by changing requirements.

Key Observation

While the relational model may be semantically-poor with respect to ontological models, the original semantically-rich design of the application domain **leaves recognizable footprints** that can be converted into ontological mapping patterns.

VKG mapping patterns

Several approaches and tools that deal with the problem of extracting a KG from a relational data source have been proposed, several of them based on mapping patterns.

However, to the best of our knowledge:

- There is no comprehensive approach for KG mapping patterns exploiting all of:
  - the relational schema with its constraints
  - extensional data stored in the DB
  - the domain knowledge that is encoded in ontology axioms
  - the conceptual schema at the basis of the relational schema
- only a few come with a systematic categorization of the mappings that they produce.
- None of them have drawn an explicit and precise connection between their outputs and conceptual modeling practices found in DB design.
- None of them attempts an analysis over real-world scenarios

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.

For the moment, 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.

Two major groups of mapping patterns

Schema-driven patterns

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

Data-driven patterns

- Consider also constraints emerging from specific configurations of the data in the DB.
- For each schema-driven pattern, we identify a data-driven version:
  The constraints over the schema are not explicitly specified, but hold in the data.
- We provide also data-driven patterns that do not have a schema-driven counterpart.

- We use also additional semantic information from the ontology $\rightsquigarrow$ **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 traditional types of DB constraints:

- **Primary key constraint**: $T(\underline{\mathbf{K}}, \mathbf{A})$
- **Key constraint**: $\text{key}_T(\mathbf{K})$
- **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:

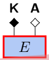
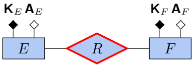
$$T_1\ (\mathbf{A}, \mathbf{B}) \qquad T_2\ (\underline{\mathbf{K}}, \mathbf{A}')$$

Note: We use normal font (e.g., A) for single attributes, and boldface for sets of attributes (e.g., $\mathbf{A}$).

Fragment of schema-driven patterns from [C., Gal, Lanti, et al. 2020]

Primary Key

Source part of the mapping assertion

E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY
Schema Entity (SE)			
	$T_E(\underline{\mathbf{K}}, \mathbf{A})$	s: T_E t: $C_E(\mathbf{iri}_E(\mathbf{K}))$, $\{d_A(\mathbf{iri}_E(\mathbf{K}), A)\}_{A \in \mathbf{K} \cup \mathbf{A}}$	$\{\exists d_A \sqsubseteq C_E\}_{A \in \mathbf{K} \cup \mathbf{A}}$
Schema Relationship (SR)			
	$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $T_F(\underline{\mathbf{K}_F}, \mathbf{A}_F)$ $T_R(\underline{\mathbf{K}_{RE}, \mathbf{K}_{RF}})$	s: T_R t: $p_R(\mathbf{iri}_E(\mathbf{K}_{RE}), \mathbf{iri}_F(\mathbf{K}_{RF}))$	$\exists p_R \sqsubseteq C_E$ $\exists p_R^- \sqsubseteq C_F$

Target part of the mapping assertion

Domain Axiom

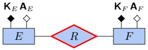
In case of $(\_, 1)$ cardinality on role R_E (resp., R_F), the primary key for T_R is restricted to the attributes $\mathbf{K}_{RE}$ (resp., $\mathbf{K}_{RF}$).

Foreign Key

iri Template Function

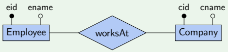
Range Axiom

Example 1: Schema Relationship Pattern

E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY
Schema Entity (SE)			
	$T_E(\underline{\mathbf{K}}, \mathbf{A})$	s: T_E t: $C_E(\mathbf{iri}_E(\mathbf{K}))$, $\{d_A(\mathbf{iri}_E(\mathbf{K}), A)\}_{A \in \mathbf{K} \cup \mathbf{A}}$	$\{\exists d_A \sqsubseteq C_E\}_{A \in \mathbf{K} \cup \mathbf{A}}$
Schema Relationship (SR)			
	$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $T_F(\underline{\mathbf{K}_F}, \mathbf{A}_F)$ $T_R(\underline{\mathbf{K}_{RE}, \mathbf{K}_{RF}})$	s: T_R t: $p_R(\mathbf{iri}_E(\mathbf{K}_{RE}), \mathbf{iri}_F(\mathbf{K}_{RF}))$	$\exists p_R \sqsubseteq C_E$ $\exists p_R^- \sqsubseteq C_F$

In case of $(\_, 1)$ cardinality on role R_E (resp., R_F), the primary key for T_R is restricted to the attributes $\mathbf{K}_{RE}$ (resp., $\mathbf{K}_{RF}$).

Conceptual



DB Schema

eid	name
1	Ginetto

Employee

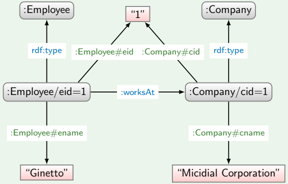
employee	company
1	1

worksAt

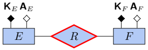
c_id	name
1	Micidial Corporation

Company

RDF (data only)

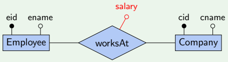


Example 2

E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY
Schema Entity (SE)			
	$T_E(\underline{\mathbf{K}}, \mathbf{A})$	s: T_E t: $C_E(\mathbf{iri}_E(\mathbf{K}))$, $\{d_A(\mathbf{iri}_E(\mathbf{K}), A)\}_{A \in \mathbf{K} \cup \mathbf{A}}$	$\{\exists d_A \sqsubseteq C_E\}_{A \in \mathbf{K} \cup \mathbf{A}}$
Schema Relationship (SR)			
	$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $T_F(\underline{\mathbf{K}_F}, \mathbf{A}_F)$ $T_R(\underline{\mathbf{K}_{RE}, \mathbf{K}_{RF}})$	s: T_R t: $p_R(\mathbf{iri}_E(\mathbf{K}_{RE}), \mathbf{iri}_F(\mathbf{K}_{RF}))$	$\exists p_R \sqsubseteq C_E$ $\exists p_R^- \sqsubseteq C_F$

In case of $(\_, 1)$ cardinality on role R_E (resp., R_F), the primary key for T_R is restricted to the attributes $\mathbf{K}_{RE}$ (resp., $\mathbf{K}_{RF}$).

Conceptual



DB Schema

eid	name
1	Ginetto

Employee

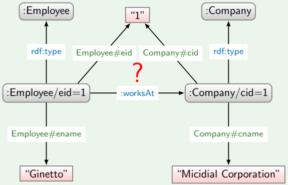
employee	company	salary
1	1	2000

worksAt

c_id	name
1	Micidial Corporation

Company

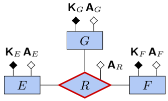
RDF (data only)



Example 2 – Revised

E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY

Schema Reified Relationship (SRR)



$$\mathbf{K}_R := \mathbf{K}_{RE} \cup \mathbf{K}_{RF} \cup \mathbf{K}_{RG}$$

$$T_G(\underline{\mathbf{K}_G}, \mathbf{A}_G)$$

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

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

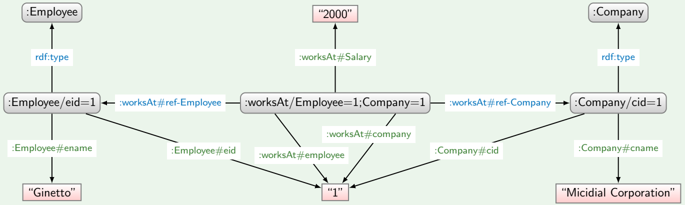
s: T_R

t: $C_R(\mathbf{iri}_R(\mathbf{K}_R))$,
$\{d_A(\mathbf{iri}_R(\mathbf{K}_R), A)\}_{A \in \mathbf{K}_R \cup \mathbf{A}_R}$,
$p_{RE}(\mathbf{iri}_R(\mathbf{K}_R), \mathbf{iri}_E(\mathbf{K}_{RE}))$,
$p_{RF}(\mathbf{iri}_R(\mathbf{K}_R), \mathbf{iri}_F(\mathbf{K}_{RF}))$,
$p_{RG}(\mathbf{iri}_R(\mathbf{K}_R), \mathbf{iri}_G(\mathbf{K}_{RG}))$

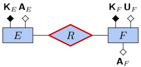
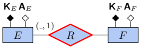
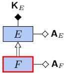
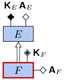
$\exists p_{RE} \sqsubseteq C_R$
$\exists p_{RE}^- \sqsubseteq C_E$
$\exists p_{RF} \sqsubseteq C_R$
$\exists p_{RF}^- \sqsubseteq C_F$
$\exists p_{RG} \sqsubseteq C_R$
$\exists p_{RG}^- \sqsubseteq C_G$
$\{\exists d_A \sqsubseteq C_R\}_{A \in \mathbf{K}_R \cup \mathbf{A}_R}$

SRR applies whenever there are three or more participating roles, or when the relationship has attributes. . . .

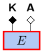
RDF (data only)



The other schema-driven patterns [C., Gal, Lanti, et al. 2020]

E-R DIAGRAM	DB SCHEMA	MAPPING	ONTOLOGY
		Schema Relationship with Identifier Alignment (SRa)	
	$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $T_F(\underline{\mathbf{K}_F}, \mathbf{U}_F, \mathbf{A}_F)$ $T_R(\underline{\mathbf{K}_{RE}, \mathbf{U}_{RF}})$ key$_{R_F}(\mathbf{U}_F)$	s: $T_R \bowtie_{\mathbf{U}_{RF}=\mathbf{U}_F} T_F$ t: $p_R(\mathbf{iri}_E(\mathbf{K}_{RE}), \mathbf{iri}_F(\mathbf{K}_F))$	$\exists p_R \sqsubseteq C_E$ $\exists p_R^- \sqsubseteq C_F$
In case of $(\_, 1)$ cardinality on role R_E (resp., R_F), the primary key for T_R is restricted to the attributes $\mathbf{K}_{RE}$ (resp., $\mathbf{U}_{RF}$).			
		Schema Relationship with Merging (SRm)	
	$T_F(\underline{\mathbf{K}_F}, \mathbf{A}_F)$ $T_E(\underline{\mathbf{K}_E}, \mathbf{K}_{EF}, \mathbf{A}_E)$	s: T_E t: $p_{EF}(\mathbf{iri}_E(\mathbf{K}_E), \mathbf{iri}_F(\mathbf{K}_{EF}))$	$\exists p_{EF} \sqsubseteq C_E$ $\exists p_{EF}^- \sqsubseteq C_F$
		Schema Hierarchy (SH)	
	$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ $T_F(\underline{\mathbf{K}_{FE}}, \mathbf{A}_F)$	s: T_F t: $C_F(\mathbf{iri}_E(\mathbf{K}_{FE}))$, $\{d_A(\mathbf{iri}_E(\mathbf{K}_{FE}), A)\}_{A \in \mathbf{A}_F}$	$C_F \sqsubseteq C_E$ $\{\exists d_A \sqsubseteq C_F\}_{A \in \mathbf{A}_F}$
		Schema Hierarchy with Identifier Alignment (SHa)	
	$T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ key$_{T_F}(\mathbf{U}_F)$ $T_F(\underline{\mathbf{K}_F}, \mathbf{U}_F, \mathbf{A}_F)$ ——— $T_E(\underline{\mathbf{K}_E}, \mathbf{A}_E)$ key$_{V_F}(\mathbf{K}_F)$ $V_F(\mathbf{K}_F, \underline{\mathbf{U}_F}, \mathbf{A}_F) = T_F$	s: T_F t: $C_F(\mathbf{iri}_E(\mathbf{U}_F))$, $\{d_A(\mathbf{iri}_E(\mathbf{U}_F), A)\}_{A \in \mathbf{K}_F \cup \mathbf{A}_F}$	$C_F \sqsubseteq C_E$ $\{\exists d_A \sqsubseteq C_F\}_{A \in \mathbf{K}_F \cup \mathbf{A}_F}$
In this pattern, the “alignment” is meant to align the primary identifier used in the child entity to the primary identifier used in the parent entity.			

A “data”-driven pattern [C., Gal, Lanti, et al. 2020]

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

eid	name	gender	
1	Ginetto	M	Male
2	Ginetta	F	Female
3	Anna	F	Female
4	Paolo	M	Male

Employee

Design scenarios for VKG mapping patterns

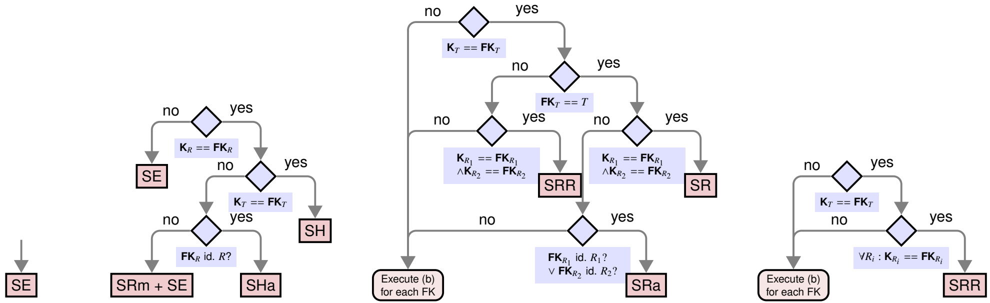
Depending on what information is available, we can consider different design scenarios where the patterns can be applied:

1. **Debugging of a VKG specification** that is already in place.
2. **Conceptual schema reverse engineering** for a DB that represents the domain of interest by using a given full VKG specification.
3. **Mapping bootstrapping** for a given DB and ontology that miss the mappings relating them.
4. **Ontology + mapping bootstrapping** from a given DB with constraints, and possibly a conceptual schema.
5. **VKG bootstrapping**, where the goal is to set up a full VKG specification from a conceptual schema of the domain.

Outline

The ADaMaP algorithm [C., Gal, Haba, et al. 2021]

Idea: Classify each table according to a pattern



(a) $|fkeys| = 0$ **(b)** $|fkeys| = 1$ **(c)** $|fkeys| = 2$ **(d)** $|fkeys| \geq 3$

- R: Referenced Table
- T: Current Table
- id: “identifies”

- $\mathbf{K}_T$: Key of current table
- $\mathbf{FK}_T$: Considered foreign key of current table

Evaluation setting

To assess the applicability of our approach, we rely on two **non-trivial** and **real-world** scenarios:

- **NPD**:
  - Scenario built around the domain of oil and gas extraction.
  - Presents a **high number of mappings** (>1k).
  - **Most mappings** are **automatically generated** (Direct Mapping).
  - **Several** complex **manually-written** mappings as well.
- **Cordis**:
  - Domain of competitive research projects, provided by SIRIS Academic S.L., a consultancy company specializing in higher education and research.
  - The mappings were **manually-written**, and they amount to 120.

Coverage Analysis

In this analysis, we check how many mappings in the analyzed scenarios can be explained through mapping patterns.

Pattern	#usages	#mappings
SE	13	60
SR	3	3
SRm	3	3
SRR	1	16
Covered Mappings: 89 (out of 120)		

CORDIS Coverage

Pattern	#usages	#mappings
SE	61	454
SRm	74	74
SRR	1	12
SH	3	132
Covered Mappings: 672 (out of 1173)		

NPD Coverage

Mismatches analysis

- We compare the classification returned by ADaMAP to a classification manually verified by a human expert.
- M: the output of ADaMAP
- M^*: the manually verified classification

Precision

$$P_{M^*}(M) = \frac{|\, M \cap M^* \,|}{|\, M \,|}$$

Recall

$$R_{M^*}(M) = \frac{|\, M \cap M^* \,|}{|\, M^* \,|}$$

F1-measure

The harmonic mean of $P_{M^*}(M)$ and $R_{M^*}(M)$

Mismatch analysis (Cordis)

	SE	SR	SRa	SRm	SRR	SH	SHa
topics	CP						
subject_areas	CP						
projects	CP			CP			
project_topics		CP					
project_subject_areas		CP					
project_programmes		CP					
project_members	DP			DP	DN		
project_member_roles	CP						
project_erc_panels		DN		DP		DP	
programmes	CP			CP			
people	CP						
institutions	CP			CP			
funding_schemes	CP						
eu_territorial_units	DP						
erc_research_domains	CP						
erc_panels	CP			CP			
ec_framework_programs	CP						
countries	CP						
activity_types	CP						

- **DP/DN**: Discoordinated Positive/Negative
- **CP/CN**: Coordinated Positive/Negative
- **SE**: Schema Entity
- **SR/SRa/SRm/SRR**: Schema Relationship/with alignment/with merging/with reification
- **SH/SHa**: Schema Hierarchy/with alignment

$$P_{M^*}(M) = R_{M^*}(M) = F_{M^*}(M) = 0.8$$

- Overall, ADaMAP and the manual extraction have 20% of disagreements.
- All but one disagreement stem from the fact that multiple conceptual schemata can correspond to the same database schema.
- The algorithm cannot determine which of these equally valid choices is actually the one that was adopted by the human designer.

Mismatch analysis (NPD)

wlbpoint	DN			DN		CP	
wellbore_shallow_all						CP	
wellbore_oil_sample	CP			CP			
wellbore_npdid_overview	CP						
wellbore_mud	CP			CP			
wellbore_formation_top	CP			CP			
wellbore_exploration_all				DP		DN	DP
wellbore_dst	CP			CP			
wellbore_document	CP			DP			
wellbore_development_all	DN			CP		DN	DP
wellbore_core_photo	CP			CP			
wellbore_core	CP			DP			
wellbore_coordinates	DN			DN		CP	
wellbore_casing_and_lot	CP			CP			
tuf_petreg_message	CP			CP			
tuf_petreg_licence_oper	DN			DP		DP	
tuf_petreg_licence_licencee					CP		
tuf_petreg_licence	CP						
tuf_owner_hst	CP			CP			
tuf_operator_hst	CP			CP			
strat_litho_wellbore_core	CP			CP			
strat_litho_wellbore	CP			CP			
seis_acquisition_progress	CP			DN			
seis_acquisition_coordinates_inc_turnarea	CP			DN			
seis_acquisition	CP			DN			
seamultiline	CP			DN		DN	
seaarea	CP			DN			
prlareasplitbyblock	CP			CP			
prlarea	CP			CP			
pipline	CP			CP			
licence_transfer_hst	CP			CP			
licence_task	CP			CP			
licence_phase_hst	CP			CP			
licence_petreg_message	CP			CP			
licence_petreg_licence_oper	DN			CP		CP	
licence_petreg_licence_licencee					CP		
licence_petreg_licence						CP	

	SE	SR	SRa	SRm	SRR	SH	SHa
licence_oper_hst	CP			CP			
licence_licensee_hst	CP			CP			
licence_area_poly_hst	CP			CP			
licence	CP						
fldarea	CP			CP			
field_reserves	DN			DN		CP	
field_production_yearly	CP						
field_production_totalt_ncs_year	DP						
field_production_totalt_ncs_month	CP						
field_production_monthly	CP						
field_owner_hst	CP			CP			
field_operator_hst	CP			CP			
field_licensee_hst	CP			CP			
field_investment_yearly	CP			CP			
field_description	DP			DP			
field_activity_status_hst	CP			CP			
field	CP			DP			
fclpoint	DN			CP		DP	
facility_moveable	CP			CP			
facility_fixed	CP					DN	
dscarea	CP			CP			
discovery_reserves	CP			CP			
discovery	CP			CP			
company_reserves					DP		
company	DP			DN			
bsns_arr_area_transfer_hst	CP			CP			
bsns_arr_area_operator	DN			CP		CP	
bsns_arr_area_licensee_hst	CP			CP			
bsns_arr_area_poly_hst	CP			CP			
bsns_arr_area	CP						
baaarea	CP			CP			
apaareanet	CP						
apaareagross	CP						

The algorithm and the manual analysis disagree on 35 instances, with 14 discoordinated positives and 21 discoordinated negatives.

In terms of precision (P), recall (R), and F1-measure (F), ADaMAP obtains the following results:

$$P_{M^*}(M) = 0.88, R_{M^*}(M) = 0.82, F_{M^*}(M) = 0.85$$

Outline

Conclusions

- The design of mappings is a complex task, that currently is the major bottleneck in the wider adoption of the VKG paradigm for data access and integration.
- Mapping patterns are a promising approach for simplifying this complex task.
- In our work, we have identified a catalog of mapping patterns for the VKG framework.
- We have introduced ADaMAP, an algorithmic technique that extracts semantics from a relational data source, by automatically identifying how ontology mapping patterns are applied to fragments of its schema
- ADaMAP can be used to support the automatic generation of ontologies and mappings
- The patterns identified by ADaMAP provide a solid basis that can be manually improved by human experts
- The validation of ADaMAP in two significant real-world case studies confirms that the identified patterns by-and-large agree with those detected by a human expert

Future Work

ADaMAP comes with some limitations that should be tackled:

- For a given relational schema there are in general many possible combinations of mapping patterns that are, in principle, equally valid . . .
  - while ADaMAP only returns the "most typical" one.
- ADaMAP ignores data, however "data"-driven patterns[2] are also important . . .
  - especially in those scenarios where the DB schema is poorly structured or denormalized.

[2] As in C., Gal, Lanti, et al. 2020

Thank you!

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

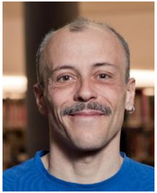
- *Ontop* website: `https://ontop-vkg.org/`
- Github: `http://github.com/ontop/ontop/`
- Facebook: `https://www.facebook.com/obdaontop/`
- Twitter: @ontop4obda
- *Ontopic* website: `https://ontopic.biz/`

A great thank you to all collaborators

Unibz

Davide Lanti

Marco Montali



Alessandro Mosca

Technion
Haifa

Avigdor Gal

Roee Shraga

References I

[1] Marcelo Arenas, Alexandre Bertails, Eric Prud'hommeaux & Juan Sequeda. *A Direct Mapping of Relational Data to RDF*. W3C Recommendation. Available at http://www.w3.org/TR/rdb-direct-mapping/. World Wide Web Consortium, Sept. 2012.

[2] Daniela Berardi, Diego C. & Giuseppe De Giacomo. "Reasoning on UML Class Diagrams". In: *Artificial Intelligence* 168.1–2 (2005), pp. 70–118.

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

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

[5] Diego C., Avigdor Gal, Naor Haba, Davide Lanti, Marco Montali, Alessandro Mosca & Roee Shraga. "ADaMaP: Automatic Alignment of Data Sources using Mapping Patterns". In: *Proc. of the 33rd Int. Conf. on Advanced Information Systems Engineering (CAiSE 2021)*. Vol. 12751. Lecture Notes in Computer Science. Springer, 2021, pp. 193–209. DOI: 10.1007/978-3-030-79382-1_12.

References II

[6] Diego C., Avigdor Gal, Davide Lanti, Marco Montali, Alessandro Mosca & Roee Shraga. *Mapping Patterns for Virtual Knowledge Graphs*. CoRR Technical Report arXiv:2012.01917. arXiv.org e-Print archive, 2020.

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

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

[9] Nicola Guarino & Christopher A. Welty. "An Overview of OntoClean". In: *Handbook on Ontologies*. Ed. by Steffen Staab & Rudi Studer. International Handbooks on Information Systems. Springer, 2009, pp. 201–220. DOI: 10.1007/978-3-540-92673-3_9.

[10] 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.

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

References III

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

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

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