# FHIR RDF Data Transformation and Validation Framework and Clinical Knowledge Graphs: Towards Explainable AI in Healthcare

Part II: Expose OMOP data sets as FHIR-compliant Clinical Knowledge Graphs

Guohui Xiao

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

Ontopic S.r.l.
Bolzano, Italy

13th International SWAT4HCLS Conference
Semantic Web Applications and Tools for Health Care and Life Sciences

10 January 2022 – Online

# Outline

# Outline

# Data integration

Databases are great!

They let us manage efficiently huge amounts of data . . .

. . . assuming you have put all data into your schema.

# Data integration

Databases are great!
They let us manage efficiently huge amounts of data . . .
. . . assuming you have put all data into your schema.

However, the reality is much more complicated and **heterogeneous**:

- Data sets were created independently.
- Data are often stored across different sources.
- Data sources are controlled by different people / organizations.

# Data integration

Databases are great!
They let us manage efficiently huge amounts of data . . .
. . . assuming you have put all data into your schema.

However, the reality is much more complicated and **heterogeneous**:

- Data sets were created independently.
- Data are often stored across different sources.
- Data sources are controlled by different people / organizations.

**Goal of data integration**

To put together different data sources,
created for different purposes,
and controlled by different people,
making them accessible in a uniform way.

unibz ON TOP IC

# Why heterogeneity?

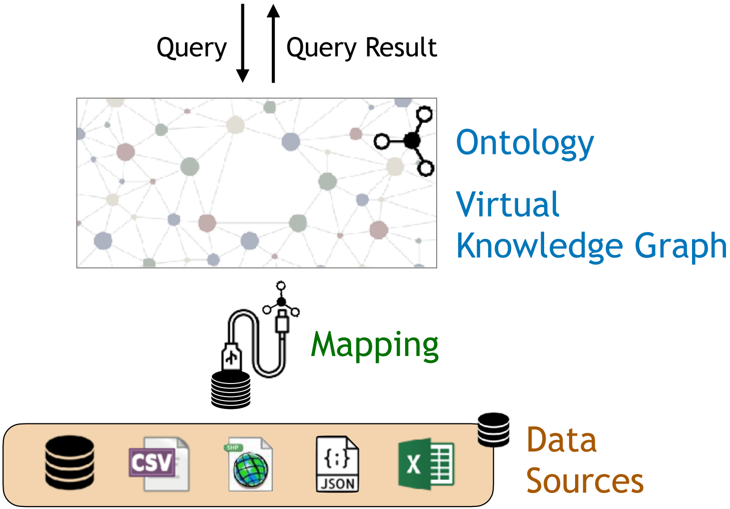
- **Data model heterogeneity**: Relational data, graph data, xml, json, csv, text files, . . .
- **System heterogeneity**: Even when systems adopt the same data model, they are not always fully compatible.
- **Schema heterogeneity**: Different people see things differently, and design schemas differently!
- **Data-level heterogeneity**: e.g., ‘IBM’ vs. ‘Int. Business Machines’ vs. ‘International Business Machines’.

# How to address heterogeneity?

We combine three key ideas:

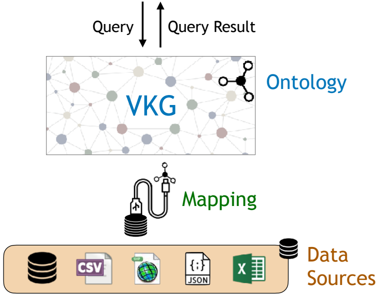
1. Use a global (or integrated) schema and **map the data sources to the global schema**.
2. Adopt a very flexible data model for the global schema
   $\leadsto$ **Knowledge Graph** whose vocabulary is expressed in an **ontology**.
3. Exploit **virtualization**, i.e., the KG is not materialized, but kept virtual.

# Virtual Knowledge Graph (VKG) architecture



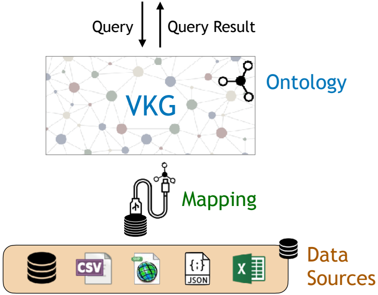
# Why an ontology?

An ontology is a structured formal representation of concepts and their relationships that are relevant for the domain of interest.



# Why an ontology?

An ontology is a structured formal representation of concepts and their relationships that are relevant for the domain of interest.

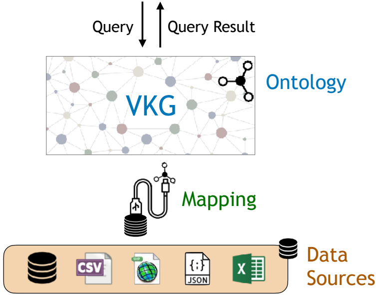



- In the VKG setting, the ontology has a twofold purpose:
  - It defines a vocabulary of terms to denote classes and properties that are familiar to the user.
  - It extends the data in the sources with background knowledge about the domain of interest, and this knowledge is machine processable.
- One can make use of custom-built domain ontologies.
- In addition, one can rely on standard ontologies, which are available for many domains.

# Why a Knowledge Graph for the global schema?

The traditional approach to data integration adopts a relational global schema.

# Why a Knowledge Graph for the global schema?




The traditional approach to data integration adopts a relational global schema.

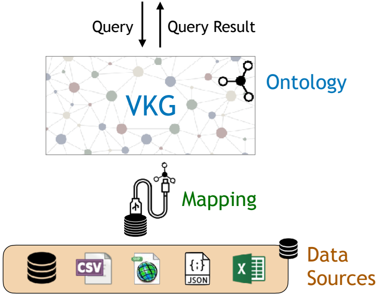
A Knowledge Graph, instead:

- Does not require to commit early on to a specific structure.
- Can better accommodate heterogeneity.
- Can better deal with missing / incomplete information.
- Does not require complex restructuring operations to accommodate new information or new data sources.

# Why mappings?

The traditional approach to data integration relies on mediators, which are specified through complex code.

# Why mappings?




The traditional approach to data integration relies on mediators, which are specified through complex code.

Mappings, instead:

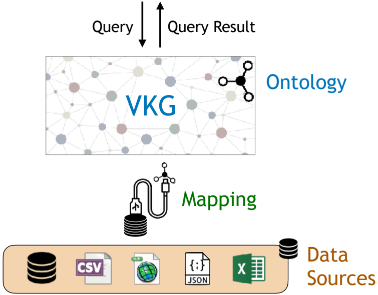
- Provide a declarative specification, and not code.
- Are easier to understand, and hence to design and to maintain.
- Support an incremental approach to integration.
- Are machine processable, hence can be used for query optimization.

# Why virtualization?

Materialized data integration relies on extract-transform-load (ETL) operations, to load data from the sources into an integrated data store / data warehouse / materialized KG.

# Why virtualization?

Materialized data integration relies on extract-transform-load (ETL) operations, to load data from the sources into an integrated data store / data warehouse / materialized KG.




In the virtual approach, instead:

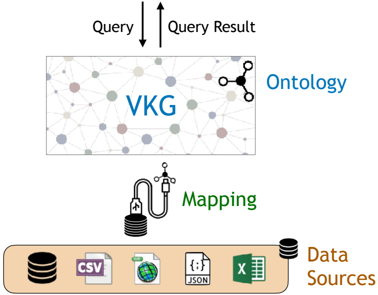
- The data stays in the sources and is only accessed at query time.
- No need to construct a large and potentially costly materialized data store and keep it up-to-date.
- Hence the data is always fresh wrt the latest updates at the sources.
- One can rely on the existing data infrastructure and expertise.
- There is better support for an incremental approach to integration.

# Outline

# Components of the VKG framework

We consider now the main components that make up the VKG framework, and the languages used to specify them.

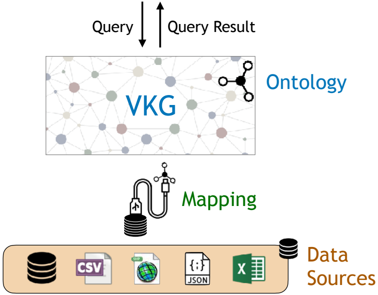
In defining such languages, we need to consider the **tradeoff between expressive power and efficiency**, where the key point is efficiency with respect to the data.



# Components of the VKG framework

We consider now the main components that make up the VKG framework, and the languages used to specify them.

In defining such languages, we need to consider the **tradeoff between expressive power and efficiency**, where the key point is efficiency with respect to the data.

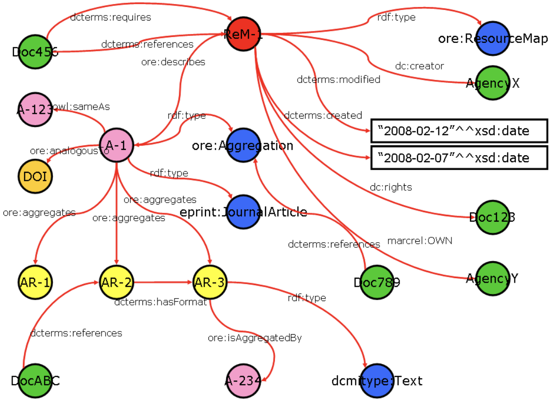



The W3C has standardized languages that are suitable for VKGs:

1. Knowledge graph: expressed in **RDF** [W3C Rec. 2014] (v1.1)
2. Ontology $\mathcal{O}$: expressed in **OWL 2 QL** [W3C Rec. 2012]
3. Mapping $\mathcal{M}$: expressed in **R2RML** [W3C Rec. 2012]
4. Query: expressed in **SPARQL** [W3C Rec. 2013] (v1.1)

# RDF – Data is represented as a graph

The graph consists of a set of **subject-predicate-object triples**:




Object property:

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

Data property:

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

Class membership:

<ReM-1> rdf:type ore:ResourceMap .

# SPARQL query language

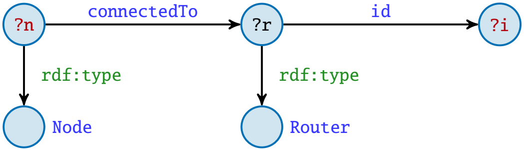
- Is the standard query language for RDF data. [W3C Rec. 2008, 2013]

```
SELECT ?n ?i
WHERE { ?n rdf:type Node .
        ?n connectedTo ?r .
        ?r rdf:type Router .
        ?r id ?i .
      }
```

# SPARQL query language

- Is the standard query language for RDF data. [W3C Rec. 2008, 2013]
- Core query mechanism is based on **graph matching**.

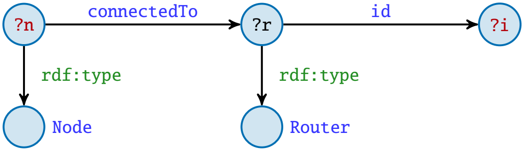
```
SELECT ?n ?i
WHERE { ?n rdf:type Node .
        ?n connectedTo ?r .
        ?r rdf:type Router .
        ?r id ?i .
      }
```



# SPARQL query language

- Is the standard query language for RDF data. [W3C Rec. 2008, 2013]
- Core query mechanism is based on **graph matching**.

```
SELECT ?n ?i
WHERE { ?n rdf:type Node .
        ?n connectedTo ?r .
        ?r rdf:type Router .
        ?r id ?i .
      }
```




Additional language features (SPARQL 1.1):

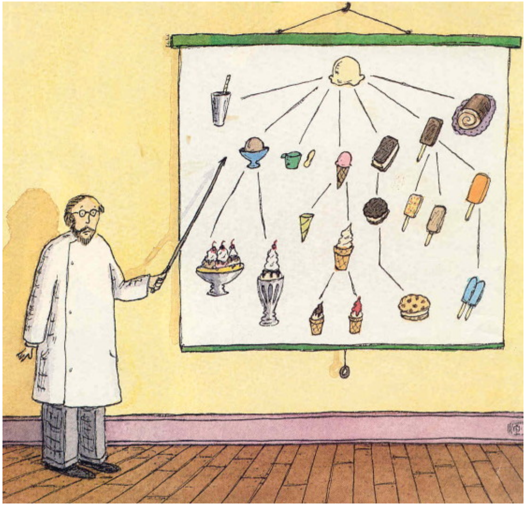
- `UNION`: matches one of alternative graph patterns
- `OPTIONAL`: produces a match even when part of the pattern is missing
- complex `FILTER` conditions
- `GROUP BY`, to express aggregations
- `MINUS`, to remove possible solutions
- property paths (regular expressions)

# What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **concepts/classes**, (binary) **relations**, and their **properties**.

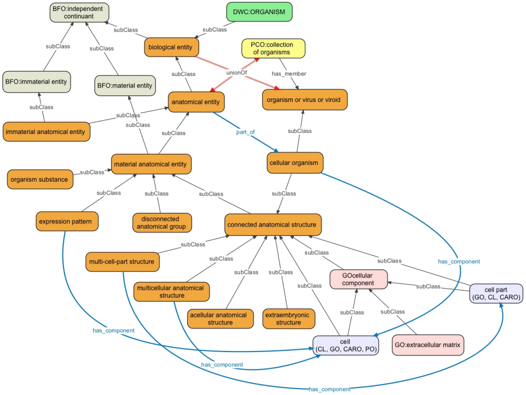
# What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts in a hierarchical structure.



# What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts in a hierarchical structure.
- Ontologies are often represented as graphs.



# What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts in a hierarchical structure.
- Ontologies are often represented as graphs.
- However, an ontology is actually a **logical theory**, expressed in a suitable fragment of first-order logic

$\forall x.\,\text{Pressure}(x) \rightarrow \text{Measurement}(x)$
$\forall x.\,\text{Porosity}(x) \rightarrow \text{Measurement}(x)$
$\forall x.\,\text{Permeability}(x) \rightarrow \text{Measurement}(x)$
$\forall x.\,\text{Temperature}(x) \rightarrow \text{Measurement}(x)$
$\forall x.\,\text{Pressure}(x) \rightarrow \neg\text{Porosity}(x) \land \neg\text{Permeability}(x) \land \neg\text{Temperature}(x)$
$\forall x.\,\text{Porosity}(x) \rightarrow \neg\text{Permeability}(x) \land \neg\text{Temperature}(x)$
$\forall x.\,\text{Permeability}(x) \rightarrow \neg\text{Temperature}(x)$

$\forall x.\,\text{HydrostaticPressure}(x) \rightarrow \text{Pressure}(x)$
$\forall x.\,\text{FormationPressure}(x) \rightarrow \text{Pressure}(x)$
$\forall x.\,\text{PorePressure}(x) \rightarrow \text{Pressure}(x)$
$\forall x.\,\text{HydrostaticPressure}(x) \rightarrow \neg\text{FormationPressure}(x) \land \neg\text{PorePressure}(x)$
$\forall x.\,\text{FormationPressure}(x) \rightarrow \neg\text{PorePressure}(x)$

$\forall x, y.\,\text{hasFormationPressure}(x, y) \rightarrow \text{Wellbore}(x) \land \text{FormationPressure}(y)$
$\forall x, y.\,\text{hasDepth}(x, y) \rightarrow \text{FormationPressure}(x) \land \text{Depth}(y)$
$\forall x.\,\text{FormationPressure}(x) \rightarrow \exists y.\,\text{hasDepth}(x, y)$

$\forall x, y.\,\text{hasFormationPressure}(x, y) \rightarrow \text{hasMeasurement}(x, y)$

$\forall x, y.\,\text{completionDate}(x, y) \rightarrow \text{Wellbore}(x) \land \text{xsd:dateTime}(y)$
$\forall x.\,\text{Wellbore}(x) \rightarrow (\sharp\{y \mid \text{completionDate}_{\text{wb}}(x, y)\} \leq 1)$
$\forall x, y.\,\text{wellboreTrack}_{\text{wb}}(x, y) \rightarrow \text{Wellbore}(x) \land \text{xsd:string}(y)$
$\forall x.\,\text{Wellbore}(x) \rightarrow (\sharp\{y \mid \text{wellboreTrack}_{\text{wb}}(x, y)\} \leq 1)$

$\forall x, y.\,\text{hasCoreSample}(x, y) \rightarrow \text{Core}(x) \land \text{CoreSample}(y)$
$\forall x.\,\text{CoreSample}(x) \rightarrow \exists y.\,\text{hasCoreSample}(y, x) \land \text{Core}(y)$
...

# What is an ontology?

- An ontology conceptualizes a domain of interest in terms of **concepts/classes**, (binary) **relations**, and their **properties**.
- It typically organizes the concepts in a hierarchical structure.
- Ontologies are often represented as graphs.
- However, an ontology is actually a **logical theory**, expressed in a suitable fragment of first-order logic, or better, in **description logics**.

$$
\begin{aligned}
\mathsf{Pressure} &\sqsubseteq \mathsf{Measurement}\\
\mathsf{Porosity} &\sqsubseteq \mathsf{Measurement}\\
\mathsf{Permeability} &\sqsubseteq \mathsf{Measurement}\\
\mathsf{Temperature} &\sqsubseteq \mathsf{Measurement}\\
\mathsf{Pressure} &\sqsubseteq \neg\mathsf{Porosity} \sqcap \neg\mathsf{Permeability} \sqcap \neg\mathsf{Temperature}\\
\mathsf{Porosity} &\sqsubseteq \neg\mathsf{Permeability} \sqcap \neg\mathsf{Temperature}\\
\mathsf{Permeability} &\sqsubseteq \neg\mathsf{Temperature}\\
\mathsf{HydrostaticPressure} &\sqsubseteq \mathsf{Pressure}\\
\mathsf{FormationPressure} &\sqsubseteq \mathsf{Pressure}\\
\mathsf{PorePressure} &\sqsubseteq \mathsf{Pressure}\\
\mathsf{HydrostaticPressure} &\sqsubseteq \neg\mathsf{FormationPressure} \sqcap \neg\mathsf{PorePressure}\\
\mathsf{FormationPressure} &\sqsubseteq \neg\mathsf{PorePressure}\\
\exists\mathsf{hasFormationPressure} &\sqsubseteq \mathsf{Wellbore}\\
\exists\mathsf{hasFormationPressure}^- &\sqsubseteq \mathsf{FormationPressure}\\
\exists\mathsf{hasDepth} &\sqsubseteq \mathsf{FormationPressure}\\
\exists\mathsf{hasDepth}^- &\sqsubseteq \mathsf{Depth}\\
\mathsf{FormationPressure} &\sqsubseteq \exists\mathsf{hasDepth}\\
\mathsf{hasFormationPressure} &\sqsubseteq \mathsf{hasMeasurement}\\
\exists\mathsf{completionDate}_{\mathsf{wb}} &\sqsubseteq \mathsf{Wellbore}\\
\exists\mathsf{completionDate}^-_{\mathsf{wb}} &\sqsubseteq \mathsf{xsd{:}dateTime}\\
\mathsf{Wellbore} &\sqsubseteq (\leq 1\, \mathsf{completionDate}_{\mathsf{wb}})\\
\exists\mathsf{wellboreTrack}_{\mathsf{wb}} &\sqsubseteq \mathsf{Wellbore}\\
&\cdots
\end{aligned}
$$

# 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 | Router `rdfs:subClassOf` NetworkNode |
| • Class disjointness | NetworkNode `owl:disjointWith` User |
| • Domain of a property | connectedTo `rdfs:domain` User |
| • Range of a property | connectedTo `rdfs:range` NetworkNode |
| • Subproperty assertions | sendsTo `rdfs:subPropertyOf` connectedTo |
| • Inverse properties | accesses `owl:inverseOf` isAccessedBy |
| • Mandatory participation to a property | ... `owl:someValuesFrom` ... |

# Use of mappings

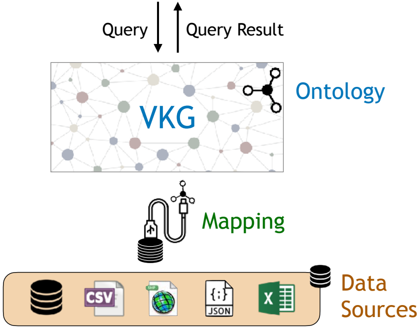
In the VKG framework, the mapping encodes how the data in the sources should be used to create the Virtual Knowledge Graph, which is formulated in the vocabulary of the ontology.

# Use of mappings

In the VKG 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.

**VKG defined from the mapping and the data.**

- Queries are answered with respect to the ontology and the data of the VKG.
- The data of the VKG is not materialized (it is virtual!).
- Instead, the information in the ontology and the mapping is used to translate queries over the ontology into queries formulated over the sources.




Note: The graph is **always up to date** wrt the data sources.

# Mapping language

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

SQL Query $\rightsquigarrow$ Class membership assertion

SQL Query $\rightsquigarrow$ Property membership assertion

# Mapping language

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

SQL Query ⇝ Class membership assertion

SQL Query ⇝ Property membership assertion

**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.

*Note:* The mapping contains also a mechanism to transform values retrieved from the database into objects of the VKG (thus solving the so-called impedance mismatch).

# Mapping language – Example

Ontology $\mathcal{O}$:

```
:actsIn rdfs:domain :MovieActor .
:actsIn rdfs:range :Movie .
:title rdfs:domain :Movie .
:title rdfs:range xsd:string .
```

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 | ··· |

# Mapping language – Example

Ontology $\mathcal{O}$:

```
:actsIn rdfs:domain :MovieActor .
:actsIn rdfs:range :Movie .
:title rdfs:domain :Movie .
:title rdfs:range xsd:string .
```

Mapping $\mathcal{M}$:

$m_1$:
```
SELECT mcode, mtitle FROM MOVIE
WHERE type = "m"
  ⇝ :m/{mcode} rdf:type :Movie .
    :m/{mcode} :title {mtitle} .
```

$m_2$:
```
SELECT M.mcode, A.acode FROM MOVIE M, ACTOR A
WHERE M.mcode = A.pcode AND M.type = "m"
  ⇝ :a/{acode} :actsIn :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 | ··· |

# Mapping language – Example

Ontology $\mathcal{O}$:

```
:actsIn rdfs:domain :MovieActor .
:actsIn rdfs:range :Movie .
:title rdfs:domain :Movie .
:title rdfs:range xsd:string .
```

Mapping $\mathcal{M}$:

$m_1$:
```
SELECT mcode, mtitle FROM MOVIE
WHERE type = "m"
  ⇝ :m/{mcode} rdf:type :Movie .
    :m/{mcode} :title {mtitle} .
```

$m_2$:
```
SELECT M.mcode, A.acode FROM MOVIE M, ACTOR A
WHERE M.mcode = A.pcode AND M.type = "m"
  ⇝ :a/{acode} :actsIn :m/{mcode} .
```

Database $\mathcal{D}$:

| MOVIE | | | | |
|---|---|---|---|---|
| *mcode* | *mtitle* | *myear* | *type* | ··· |
| 5118 | The Matrix | 1999 | m | ··· |
| 8234 | Altered Carbon | 2018 | s | ··· |
| 2281 | Blade Runner | 1982 | m | ··· |

| ACTOR | | | |
|---|---|---|---|
| *pcode* | *acode* | *aname* | ··· |
| 5118 | 438 | K. Reeves | ··· |
| 5118 | 572 | C.A. Moss | ··· |
| 2281 | 271 | H. Ford | ··· |

The mapping $\mathcal{M}$ applied to database $\mathcal{D}$ generates the (virtual) knowledge graph $\mathcal{V} = \mathcal{M}(\mathcal{D})$:

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

unibz ONTOPIC

# Query answering in VKGs

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

- The ontology contains **domain knowledge** that can be used to enrich answers.

  Example: Suppose that our data contains LJ-2025 among the Printers, and that the ontology states that each Printer is a NetworkDevice.
  If we ask for all NetworkDevices, we should return also LJ 2025, considering both the data and the knowledge in the ontology.

- The **mapping** encodes the information of how to translate a query over the ontology into a query over the **database**.

A VKG query answering engine has to take into account all these types of information.

# Query answering in VKGs

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

- The ontology contains **domain knowledge** that can be used to enrich answers.

  Example: Suppose that our data contains LJ-2025 among the Printers, and that the ontology states that each Printer is a NetworkDevice.
  If we ask for all NetworkDevices, we should return also LJ 2025, considering both the data and the knowledge in the ontology.

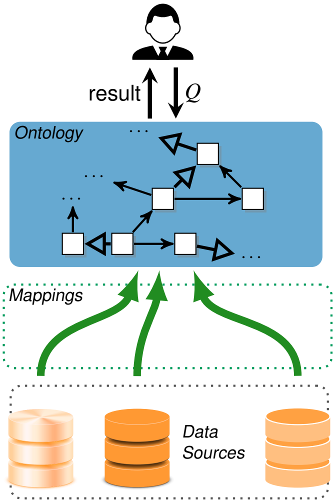
- The **mapping** encodes the information of how to translate a query over the ontology into a query over the **database**.

A VKG query answering engine has to take into account all these types of information.

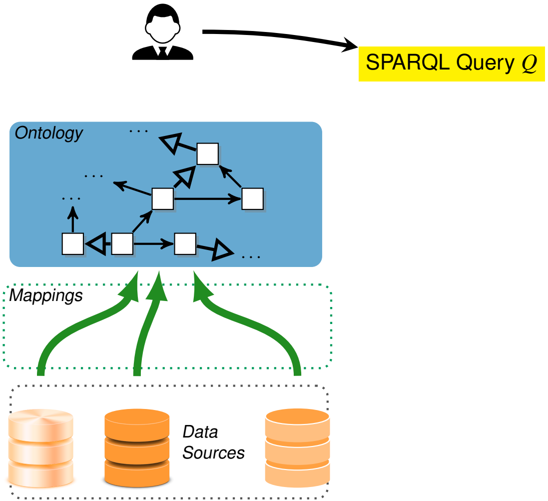
**Query answering by query rewriting**

unibz ONTOPIC

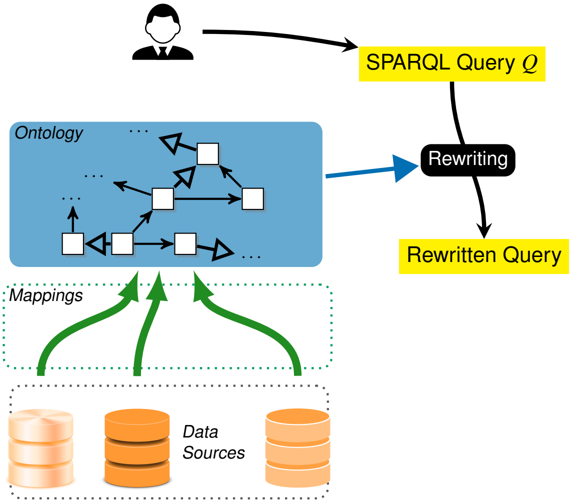
# Query answering by query rewriting



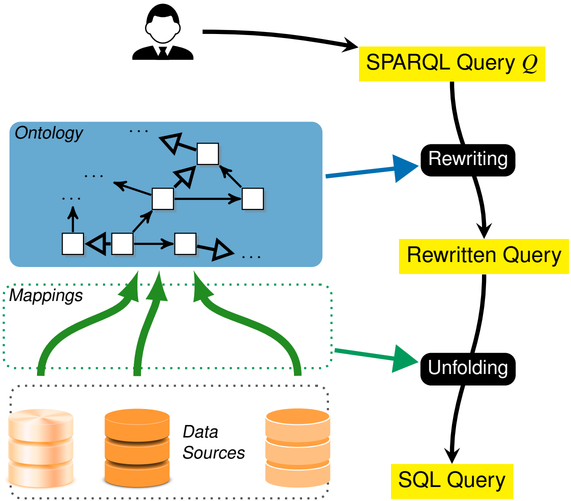
# Query answering by query rewriting



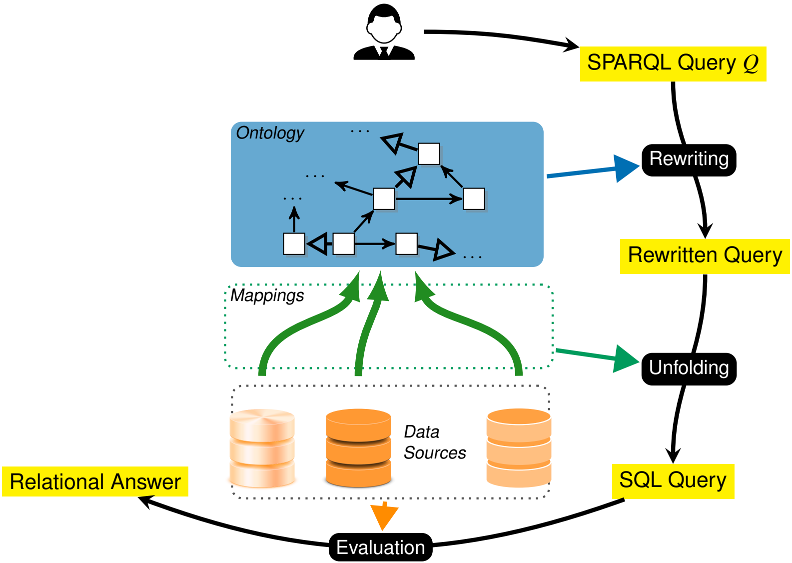
# Query answering by query rewriting



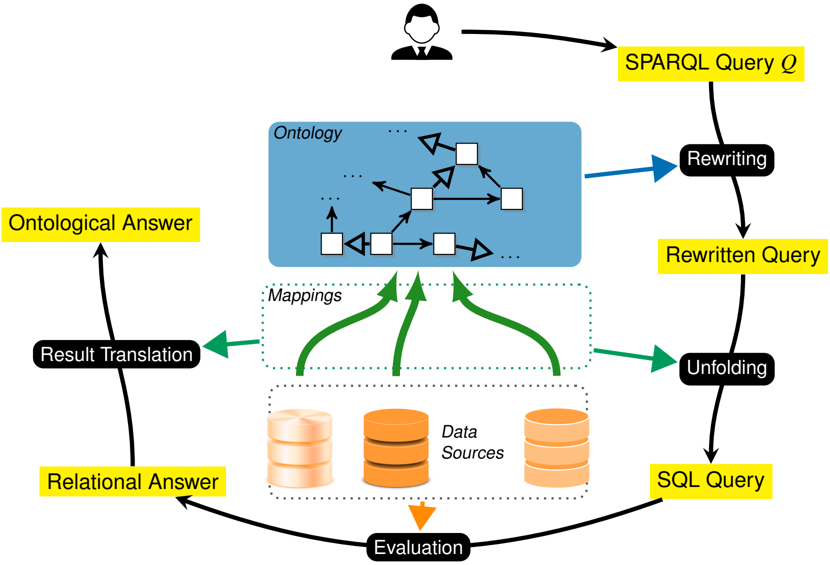
# Query answering by query rewriting



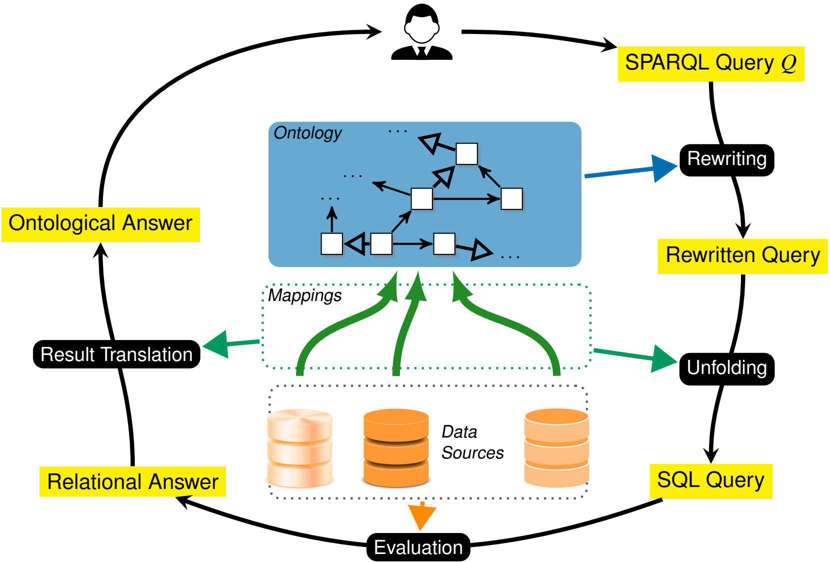
# Query answering by query rewriting



# Query answering by query rewriting



# Query answering by query rewriting



# Outline

# The *Ontop* system

https://ontop-vkg.org/

- State-of-the-art VKG system.
- Addresses the key challenges in query answering of scalability and performance.
- Compliant with all relevant Semantic Web standards:

  RDF, RDFS, OWL 2 QL, R2RML, SPARQL, and GeoSPARQL.
- Supports all major relational DBMSs:

  Oracle, DB2, MS SQL Server, Postgres, MySQL, Teiid, Dremio, Denodo, etc.
- Open-source and released under Apache 2 license.

# Developer community

unibz

ON TOP IC

Birkbeck UNIVERSITY OF LONDON

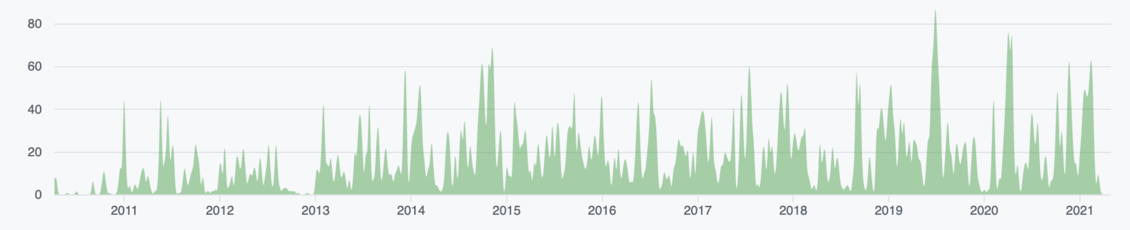
UiO : University of Oslo

HELLENIC REPUBLIC
National and Kapodistrian
University of Athens

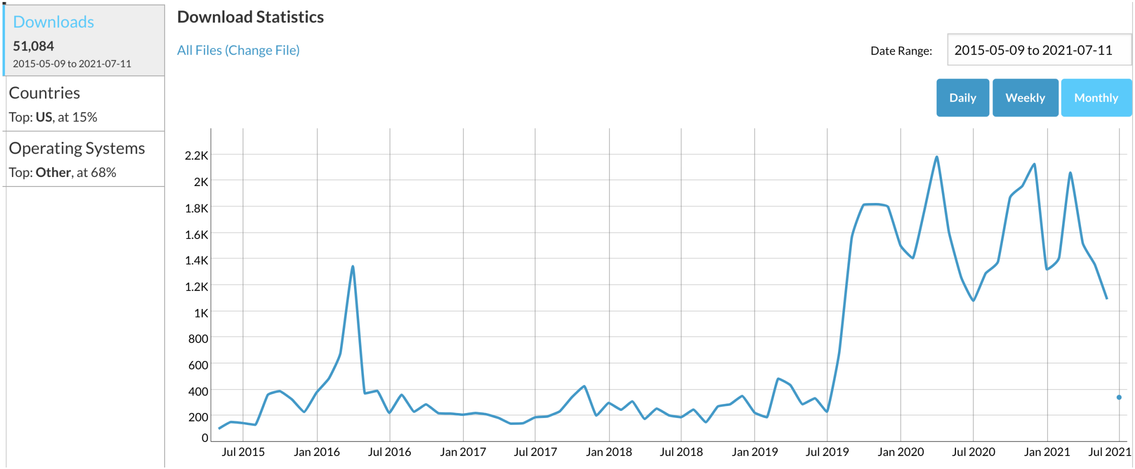
UNIVERSITÄT LEIPZIG

ontotext

POLITECNICO MILANO 1863



# Ontop downloads

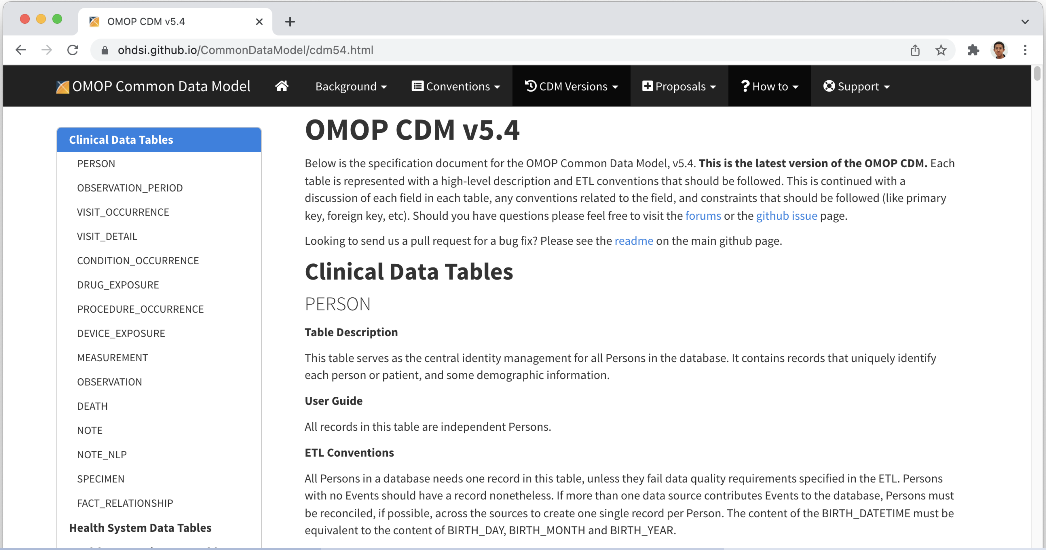


# Outline

# OMOP Common Data Model

- OMOP Common Data Model. https://ohdsi.github.io/CommonDataModel/.
- The *Observational Medical Outcomes Partnership* (OMOP) *Common Data Model* (CDM) is an open community data standard, designed to standardize the structure and content of observational data and to enable efficient analyses that can produce reliable evidence.

# OMOP CDM v5.4



OMOP CDM v5.4

ohdsi.github.io/CommonDataModel/cdm54.html

OMOP Common Data Model | Background | Conventions | CDM Versions | Proposals | How to | Support

**Clinical Data Tables**

- PERSON
- OBSERVATION_PERIOD
- VISIT_OCCURRENCE
- VISIT_DETAIL
- CONDITION_OCCURRENCE
- DRUG_EXPOSURE
- PROCEDURE_OCCURRENCE
- DEVICE_EXPOSURE
- MEASUREMENT
- OBSERVATION
- DEATH
- NOTE
- NOTE_NLP
- SPECIMEN
- FACT_RELATIONSHIP

**Health System Data Tables**

# OMOP CDM v5.4

Below is the specification document for the OMOP Common Data Model, v5.4. **This is the latest version of the OMOP CDM.** Each table is represented with a high-level description and ETL conventions that should be followed. This is continued with a discussion of each field in each table, any conventions related to the field, and constraints that should be followed (like primary key, foreign key, etc). Should you have questions please feel free to visit the forums or the github issue page.

Looking to send us a pull request for a bug fix? Please see the readme on the main github page.

## Clinical Data Tables

### PERSON

**Table Description**

This table serves as the central identity management for all Persons in the database. It contains records that uniquely identify each person or patient, and some demographic information.

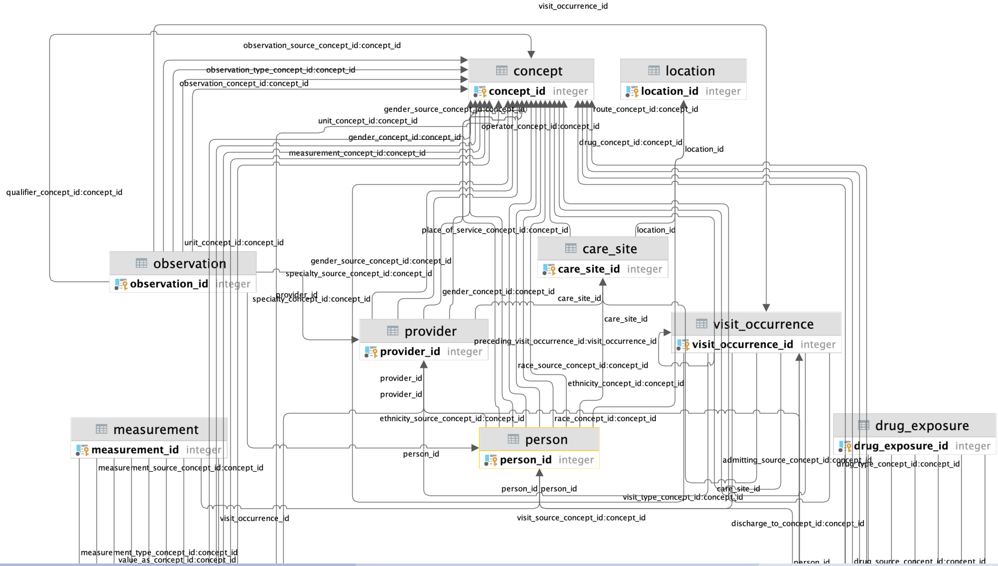
**User Guide**

All records in this table are independent Persons.

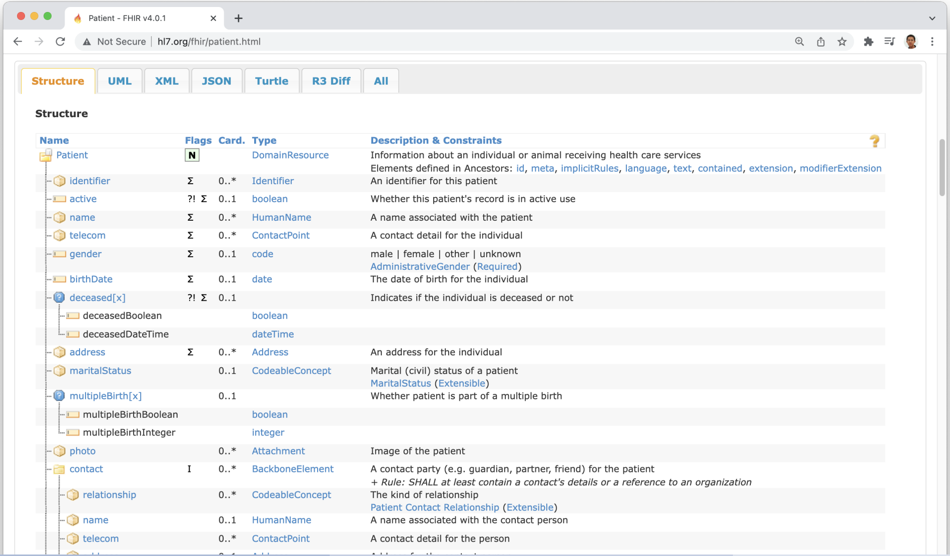
**ETL Conventions**

All Persons in a database needs one record in this table, unless they fail data quality requirements specified in the ETL. Persons with no Events should have a record nonetheless. If more than one data source contributes Events to the database, Persons must be reconciled, if possible, across the sources to create one single record per Person. The content of the BIRTH_DATETIME must be equivalent to the content of BIRTH_DAY, BIRTH_MONTH and BIRTH_YEAR.

# Source Schema: Example OMOP tables



# Target Schema: FHIR RDF




| Name | Flags | Card. | Type | Description & Constraints |
|---|---|---|---|---|
| Patient | N | | DomainResource | Information about an individual or animal receiving health care services<br>Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension |
| identifier | Σ | 0..* | Identifier | An identifier for this patient |
| active | ?! Σ | 0..1 | boolean | Whether this patient's record is in active use |
| name | Σ | 0..* | HumanName | A name associated with the patient |
| telecom | Σ | 0..* | ContactPoint | A contact detail for the individual |
| gender | Σ | 0..1 | code | male \| female \| other \| unknown<br>AdministrativeGender (Required) |
| birthDate | Σ | 0..1 | date | The date of birth for the individual |
| deceased[x] | ?! Σ | 0..1 | | Indicates if the individual is deceased or not |
| deceasedBoolean | | | boolean | |
| deceasedDateTime | | | dateTime | |
| address | Σ | 0..* | Address | An address for the individual |
| maritalStatus | | 0..1 | CodeableConcept | Marital (civil) status of a patient<br>MaritalStatus (Extensible) |
| multipleBirth[x] | | 0..1 | | Whether patient is part of a multiple birth |
| multipleBirthBoolean | | | boolean | |
| multipleBirthInteger | | | integer | |
| photo | | 0..* | Attachment | Image of the patient |
| contact | I | 0..* | BackboneElement | A contact party (e.g. guardian, partner, friend) for the patient<br>*+ Rule: SHALL at least contain a contact's details or a reference to an organization* |
| relationship | | 0..* | CodeableConcept | The kind of relationship<br>Patient Contact Relationship (Extensible) |
| name | | 0..1 | HumanName | A name associated with the contact person |
| telecom | | 0..* | ContactPoint | A contact detail for the person |

# FHIR RDF as Triple Template

Patient - FHIR v4.0.1

Not Secure | hl7.org/fhir/patient.html

Structure | UML | XML | JSON | Turtle | R3 Diff | All

**Turtle Template**

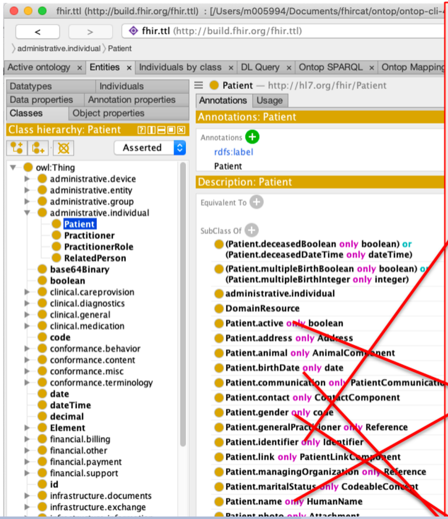
```
@prefix fhir: <http://hl7.org/fhir/> .


[ a fhir:Patient;
  fhir:nodeRole fhir:treeRoot; # if this is the parser root

  # from Resource: .id, .meta, .implicitRules, and .language
  # from DomainResource: .text, .contained, .extension, and .modifierExtension
  fhir:Patient.identifier [ Identifier ], ... ; # 0..* An identifier for this patient
  fhir:Patient.active [ boolean ]; # 0..1 Whether this patient's record is in active use
  fhir:Patient.name [ HumanName ], ... ; # 0..* A name associated with the patient
  fhir:Patient.telecom [ ContactPoint ], ... ; # 0..* A contact detail for the individual
  fhir:Patient.gender [ code ]; # 0..1 male | female | other | unknown
  fhir:Patient.birthDate [ date ]; # 0..1 The date of birth for the individual
  # Patient.deceased[x] : 0..1 Indicates if the individual is deceased or not. One of these 2
    fhir:Patient.deceasedBoolean [ boolean ]
    fhir:Patient.deceasedDateTime [ dateTime ]
  fhir:Patient.address [ Address ], ... ; # 0..* An address for the individual
  fhir:Patient.maritalStatus [ CodeableConcept ]; # 0..1 Marital (civil) status of a patient
  # Patient.multipleBirth[x] : 0..1 Whether patient is part of a multiple birth. One of these 2
    fhir:Patient.multipleBirthBoolean [ boolean ]
    fhir:Patient.multipleBirthInteger [ integer ]
  fhir:Patient.photo [ Attachment ], ... ; # 0..* Image of the patient
  fhir:Patient.contact [ # 0..* A contact party (e.g. guardian, partner, friend) for the patient
    fhir:Patient.contact.relationship [ CodeableConcept ], ... ; # 0..* The kind of relationship
    fhir:Patient.contact.name [ HumanName ]; # 0..1 A name associated with the contact person
    fhir:Patient.contact.telecom [ ContactPoint ], ... ; # 0..* A contact detail for the person
    fhir:Patient.contact.address [ Address ]; # 0..1 Address for the contact person
    fhir:Patient.contact.gender [ code ]; # 0..1 male | female | other | unknown
    fhir:Patient.contact.organization [ Reference(Organization) ]; # 0..1 Organization that is a
ssociated with the contact
    fhir:Patient.contact.period [ Period ]; # 0..1 The period during which this contact person o
```

# FHIR Model Ontology and Instance data in RDF

## Patient Class in FHIR Model Ontology



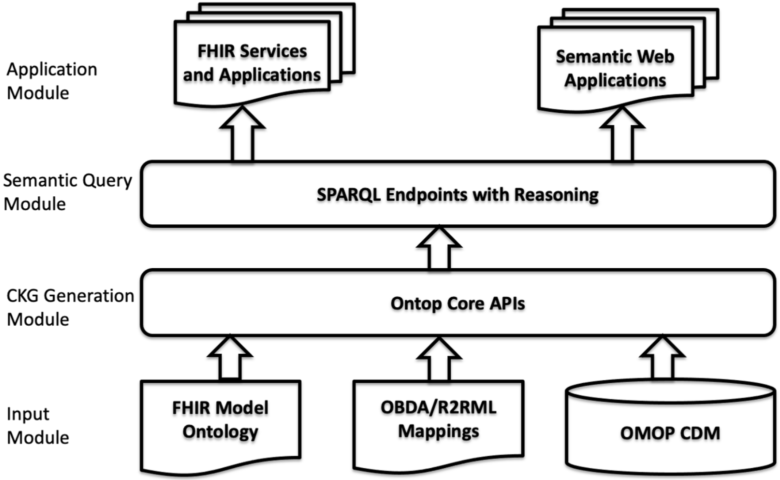

## Patient Instance Data in FHIR RDF

```
@prefix fhir: <http://hl7.org/fhir/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# - resource -------------------------------------------------------------------

<http://hl7.org/fhir/Patient/pat3> a fhir:Patient;
  fhir:nodeRole fhir:treeRoot;
  fhir:Resource.id [ fhir:value "pat3"];
  fhir:DomainResource.text [
     fhir:Narrative.status [ fhir:value "generated" ];
     fhir:Narrative.div "<div xmlns=\"http://www.w3.org/1999/xhtml\">\n      \n      <p>Patient
Simon Notsowell @ Acme Healthcare, Inc. MR = 123457, DECEASED</p>\n    \n    </div>"
  ];
  fhir:Patient.identifier [
     fhir:index 0;
     fhir:Identifier.use [ fhir:value "usual" ];
     fhir:Identifier.type [
       fhir:CodeableConcept.coding [
         fhir:index 0;
         fhir:Coding.system [ fhir:value "http://terminology.hl7.org/CodeSystem/v2-0203" ];
         fhir:Coding.code [ fhir:value "MR" ]
       ]
     ];
     fhir:Identifier.system [ fhir:value "urn:oid:0.1.2.3.4.5.6.7" ];
     fhir:Identifier.value [ fhir:value "123457" ]
  ];
  fhir:Patient.active [ fhir:value "true"^^xsd:boolean];
  fhir:Patient.name [
     fhir:index 0;
     fhir:HumanName.use [ fhir:value "official" ];
     fhir:HumanName.family [ fhir:value "Notsowell" ];
     fhir:HumanName.given [
       fhir:value "Simon";
       fhir:index 0
     ]
  ];
```

# Expose OMOP data sets as FHIR-compliant Clinical Knowledge Graphs



# Example

## Records in OMOP Person Table:

| person_id | gender_concept_id | year_of_birth | month_of_birth | day_of_birth | birth_datetime |
|---|---|---|---|---|---|
| 392776072 | 8507 | 2138 | 7 | 17 | 2138-07-17 00:00:00 |
| 392776073 | 8507 | 2025 | 4 | 11 | 2025-04-11 00:00:00 |
| 392776074 | 8532 | 2143 | 5 | 12 | 2143-05-12 00:00:00 |
| 392776075 | 8507 | 2103 | 2 | 2 | 2103-02-02 00:00:00 |
| 392776076 | 8532 | 2109 | 6 | 21 | 2109-06-21 00:00:00 |
| 392776077 | 8532 | 2121 | 5 | 23 | 2121-05-23 00:00:00 |
| 392776078 | 8507 | 2117 | 11 | 20 | 2117-11-20 00:00:00 |

## Clinical Knowledge Graph in FHIR RDF:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX fhir: <http://hl7.org/fhir/>

<http://hl7.org/fhir/Patient/392776072>
    a fhir:Patient ;
    fhir:Resource.id [fhir:value "392776072"^^xsd:string ];
    fhir:Patient.gender [fhir:value "male"^^xsd:string ];
    fhir:Patient.birthDate [fhir:value  "2138-07-17"^^xsd:date ] .
```

# Dealing with Nested triples

- FHIR/RDF makes extensive use of "nested" triples
- E.g. `:Patient/392776072 fhir:Resource.id [ fhir:value "392776072"]`

# Dealing with Nested triples

- FHIR/RDF makes extensive use of “nested” triples
- E.g. `:Patient/392776072 fhir:Resource.id [ fhir:value "392776072"]`
- This two triples: `:Patient/392776072> fhir:Resource.id _:1 .`
  `_:1 fhir:value "392776072" .`

# Dealing with Nested triples

- FHIR/RDF makes extensive use of "nested" triples
- E.g. `:Patient/392776072 fhir:Resource.id [ fhir:value "392776072"]`
- This two triples: `:Patient/392776072> fhir:Resource.id _:1 .`
  `_:1 fhir:value "392776072" .`
- The blank node `_:1` should not be used in other places, so better write it like:
  `:Patient/392776072 fhir:Resource.id  :Patient/392776072/Resource.id .`
  `:Patient/392776072/Resource.id fhir:value "392776072" .`

# Dealing with Nested triples

- FHIR/RDF makes extensive use of “nested” triples
- E.g. `:Patient/392776072 fhir:Resource.id [ fhir:value "392776072"]`
- This two triples: `:Patient/392776072> fhir:Resource.id _:1 .`
  `_:1 fhir:value "392776072" .`
- The blank node `_:1` should not be used in other places, so better write it like:

```
:Patient/392776072 fhir:Resource.id   :Patient/392776072/Resource.id .
:Patient/392776072/Resource.id fhir:value "392776072" .
```

- The target part of the Ontop mapping looks like:

```
:Patient/{person_id} fhir:Resource.id   :Patient/{person_id}/Resource.id .
:Patient/{person_id}/Resource.id fhir:value "{person_id}" .
```

unibz ON TOP IC

# Dealing with Nested triples

- FHIR/RDF makes extensive use of "nested" triples
- E.g. `:Patient/392776072 fhir:Resource.id [ fhir:value "392776072"]`
- This two triples: `:Patient/392776072> fhir:Resource.id _:1 .`
  `_:1 fhir:value "392776072" .`
- The blank node `_:1` should not be used in other places, so better write it like:

```
:Patient/392776072 fhir:Resource.id   :Patient/392776072/Resource.id .
:Patient/392776072/Resource.id fhir:value "392776072" .
```

- The target part of the Ontop mapping looks like:

```
:Patient/{person_id} fhir:Resource.id   :Patient/{person_id}/Resource.id .
:Patient/{person_id}/Resource.id fhir:value "{person_id}" .
```

- This becomes unmanageable with multiple levels

```
fhir:Patient.identifier [
       fhir:CodeableConcept.coding [
         fhir:Coding.system [ fhir:value "http://terminology.hl7.org/CodeSystem/
         fhir:Coding.code [ fhir:value "MR" ] ]];
     fhir:Identifier.system [ fhir:value "urn:oid:0.1.2.3.4.5.6.7" ];
     fhir:Identifier.value [ fhir:value "123457" ] ];
```

# Turtle Template Mapping Language

Key ideas:

- Stay close to Triple Template in FHIR RDF (human-readable)
- Reuse (abuse) R2RML vocabulary as much as possible
- Generate Blank nodes automatically
- Translatable to Ontop Mapping/R2RML

```
[] # measurement -> :Observation
    rr:logicalTable       [ rr:sqlQuery """SELECT * FROM measurement""" ] ;
    rr:subjectMap         [ rr:template "http://hl7.org/fhir/Observation/{measurement_id}" ] ;
    rr:predicateObjectMap [ a                                  :Observation ;
                            :Resource.id                       [ :value [ rr:column "measurement_id" ] ] ;
                            :Observation.status                [ :value "final"];
                            :Observation.code                  [ rr:termType rr:IRI ;
                                                                 rr:template "http://hl7.org/fhir/CodeableConcept/{measurement_concept_id}" ] ;
                            :Observation.category              [ rr:termType rr:IRI ;
                                                                 rr:template "http://hl7.org/fhir/CodeableConcept/{measurement_type_concept_id}" ] ;
                            :Observation.subject               [ :link [ rr:termType rr:IRI ;
                                                                 rr:template "http://hl7.org/fhir/Patient/{person_id}" ] ];
                            :Observation.encounter             [ :link [ rr:termType rr:IRI ;
                                                                 rr:template "http://hl7.org/fhir/Encounter/{visit_occurrence_id}" ] ];
                            :Observation.effectiveDateTime     [ :value [ rr:column   "measurement_datetime" ;
                                                                          rr:datatype xsd:dateTime ] ] ;
                            :Observation.effectivePeriod       [ :Period.start [ :value [ rr:column   "measurement_datetime" ;
                                                                                          rr:datatype xsd:dateTime ] ] ] ;
                            :Observation.valueCodeableConcept  [ rr:termType rr:IRI ;
                                                                 rr:template "http://hl7.org/fhir/CodeableConcept/{value_as_concept_id}" ] ;
                            :Observation.valueString           [ :value [ rr:column   "value_as_number" ;
```

# Corresponding Ontop Mapping

```
mappingId	mapping12
target	<http://hl7.org/fhir/Observation/{measurement_id}> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://hl7.org/fhir/Observation> . 
<http://hl7.org/fhir/Observation/{measurement_id}> <http://hl7.org/fhir/Resource.id> <http://hl7.org/fhir/Observation/{measurement_id}/Resource.id> . 
<http://hl7.org/fhir/Observation/{measurement_id}/Resource.id> <http://hl7.org/fhir/value> "{measurement_id}"^^<http://www.w3.org/2001/XMLSchema#string>
. <http://hl7.org/fhir/Observation/{measurement_id}> <http://hl7.org/fhir/Observation.status> <http://hl7
.org/fhir/Observation/{measurement_id}/Observation.status> . <http://hl7.org/fhir/Observation/{measurement_id}/Observation.status> <http://hl7
.org/fhir/value> "final"^^<http://www.w3.org/2001/XMLSchema#string> . <http://hl7.org/fhir/Observation/{measurement_id}> <http://hl7.org/fhir/Observation
.code> <http://hl7.org/fhir/CodeableConcept/{measurement_concept_id}> . <http://hl7.org/fhir/Observation/{measurement_id}> <http://hl7
.org/fhir/Observation.category> <http://hl7.org/fhir/CodeableConcept/{measurement_type_concept_id}> . <http://hl7.org/fhir/Observation/{measurement_id}>
<http://hl7.org/fhir/Observation.subject> <http://hl7.org/fhir/Observation/{measurement_id}/Observation.subject> . <http://hl7
.org/fhir/Observation/{measurement_id}/Observation.subject> <http://hl7.org/fhir/link> <http://hl7.org/fhir/Patient/{person_id}> . <http://hl7
.org/fhir/Observation/{measurement_id}> <http://hl7.org/fhir/Observation.encounter> <http://hl7.org/fhir/Observation/{measurement_id}/Observation
.encounter> . <http://hl7.org/fhir/Observation/{measurement_id}/Observation.encounter> <http://hl7.org/fhir/link> <http://hl7
.org/fhir/Encounter/{visit_occurrence_id}> . <http://hl7.org/fhir/Observation/{measurement_id}> <http://hl7.org/fhir/Observation.effectiveDateTime> 
<http://hl7.org/fhir/Observation/{measurement_id}/Observation.effectiveDateTime> . <http://hl7.org/fhir/Observation/{measurement_id}/Observation
.effectiveDateTime> <http://hl7.org/fhir/value> "{measurement_datetime}"^^<http://www.w3.org/2001/XMLSchema#dateTime> . <http://hl7
.org/fhir/Observation/{measurement_id}> <http://hl7.org/fhir/Observation.effectivePeriod> <http://hl7.org/fhir/Observation/{measurement_id}/Observation
.effectivePeriod> . <http://hl7.org/fhir/Observation/{measurement_id}/Observation.effectivePeriod> <http://hl7.org/fhir/Period.start> <http://hl7
.org/fhir/Observation/{measurement_id}/Observation.effectivePeriod/Period.start> . <http://hl7.org/fhir/Observation/{measurement_id}/Observation
.effectivePeriod/Period.start> <http://hl7.org/fhir/value> "{measurement_datetime}"^^<http://www.w3.org/2001/XMLSchema#dateTime> . <http://hl7
.org/fhir/Observation/{measurement_id}> <http://hl7.org/fhir/Observation.valueCodeableConcept> <http://hl7
.org/fhir/CodeableConcept/{value_as_concept_id}> . <http://hl7.org/fhir/Observation/{measurement_id}> <http://hl7.org/fhir/Observation.valueString> 
<http://hl7.org/fhir/Observation/{measurement_id}/Observation.valueString> . <http://hl7.org/fhir/Observation/{measurement_id}/Observation.valueString> 
<http://hl7.org/fhir/value> "{value_as_number}"^^<http://www.w3.org/2001/XMLSchema#string> . <http://hl7.org/fhir/Observation/{measurement_id}> 
<http://hl7.org/fhir/Observation.referenceRange> <http://hl7.org/fhir/Observation/{measurement_id}/Observation.referenceRange> . <http://hl7
.org/fhir/Observation/{measurement_id}/Observation.referenceRange> <http://hl7.org/fhir/Observation.referenceRange.low> <http://hl7
.org/fhir/Observation/{measurement_id}/Observation.referenceRange/Observation.referenceRange.low> . <http://hl7
.org/fhir/Observation/{measurement_id}/Observation.referenceRange/Observation.referenceRange.low> <http://hl7.org/fhir/value> "{range_low}"^^<http://www
.w3.org/2001/XMLSchema#string> . <http://hl7.org/fhir/Observation/{measurement_id}/Observation.referenceRange> <http://hl7.org/fhir/Observation
.referenceRange.high> <http://hl7.org/fhir/Observation/{measurement_id}/Observation.referenceRange/Observation.referenceRange.high> . <http://hl7
.org/fhir/Observation/{measurement_id}/Observation.referenceRange/Observation.referenceRange.high> <http://hl7.org/fhir/value> 
```

# SPARQL Endpoint

Ontop SPARQL endpoint v4.2.1

localhost:8081

**FHIR/OMOP VKG Demo**

endpoint address: http://localhost:8081/sparql | ontop v4.2.1-SNAPSHOT

Playground | Example Queries | Evaluation Queries

QUERY 1 | QUERY 2 | QUERY 3 | QUERY 4 | QUERY 5

```
#--Query result: Male patients with inpatient admissions lasting >= 5 days

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX fhir: <http://hl7.org/fhir/>
PREFIX w5: <http://hl7.org/fhir/w5#>
PREFIX ofn:<http://www.ontotext.com/sparql/functions/>

SELECT  distinct ?patient ?gender ?birthDate ?type ?start ?end ?diff
WHERE {
  ?e a fhir:Encounter .
  ?e fhir:Encounter.period  [
        fhir:Period.start [ fhir:value ?start ]  ; fhir:Period.end   [ fhir:value ?end ] ] .
  ?e fhir:Encounter.subject [ fhir:link ?patient ] .
  ?e fhir:Encounter.class [fhir:CodeableConcept.coding [fhir:Coding.code [fhir:value ?type] ] ].
  ?patient a fhir:Patient ;
     fhir:Patient.gender    [   fhir:value ?gender      ] ;
     fhir:Patient.birthDate [   fhir:value ?birthDate   ] .
  FILTER (?gender = 'male')
  BIND (ofn:daysBetween(?start, ?end) as ?diff)
  FILTER (?diff >= "5"^^xsd:integer)
}
LIMIT 100
```

Table | Response | Pivot Table | Google Chart | Geo

Showing 1 to 50 of 100 entries (in 1.462 seconds)

Search: Show 50 entries

| | patient | gender | birthDate | type | start | end | diff |
|---|---|---|---|---|---|---|---|
| 1 | http://hl7.org/fhir/Patient/392775853 | male | "2078-03-06"^^xsd:date | OMOP generated | "2133-03-31T03:58:00"^^xsd:dateTime | "2133-04-23T15:00:00"^^xsd:dateTime | "23"^^xsd:long |
| 2 | http://hl7.org/fhir/Patient/392775856 | male | "2086-07-31"^^xsd:date | OMOP generated | "2163-07-26T10:30:00"^^xsd:dateTime | "2163-08-02T14:14:00"^^xsd:dateTime | "7"^^xsd:long |
| 3 | http://hl7.org/fhir/Patient/392775856 | male | "2086-07-31"^^xsd:date | OMOP generated | "2166-07-21T14:34:00"^^xsd:dateTime | "2166-07-28T12:45:00"^^xsd:dateTime | "6"^^xsd:long |
| 4 | http://hl7.org/fhir/Patient/392775856 | male | "2086-07-31"^^xsd:date | OMOP generated | "2170-06-15T00:25:00"^^xsd:dateTime | "2170-06-27T16:17:00"^^xsd:dateTime | "12"^^xsd:long |

# Outline

# Demo

- Setting up the VKG
  - Instruction: https://github.com/fhircat/FHIROntopOMOP
  - Requirement: connection to an OMOP database
- Validation:
  - https://github.com/shexjs/shex.js
  - Example:

```
shex-validate -x ../patient.shex -d omop.392775850.ttl
-s http://hl7.org/fhir/shape/Patient
-n http://hl7.org/fhir/Patient/392775850
```

# Outline

1. Ontology-Based Data Integration
2. The VKG Framework
3. The Ontop System
4. FHIR-Ontop-OMOP
5. Demo
6. 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 many different scenarios mostly for the case of relational data sources.
- The technology is general purpose, but the bio-medical domain is very well suited for its application.
- Ontop can be used to bridge OMOP and FHIR/RDF!

# Thank you!

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

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