Graph-based Virtual Data Integration

Oscar Romero

Facultat d’Informàtica de Barcelona

Universitat Politècnica de Catalunya

Data Integration

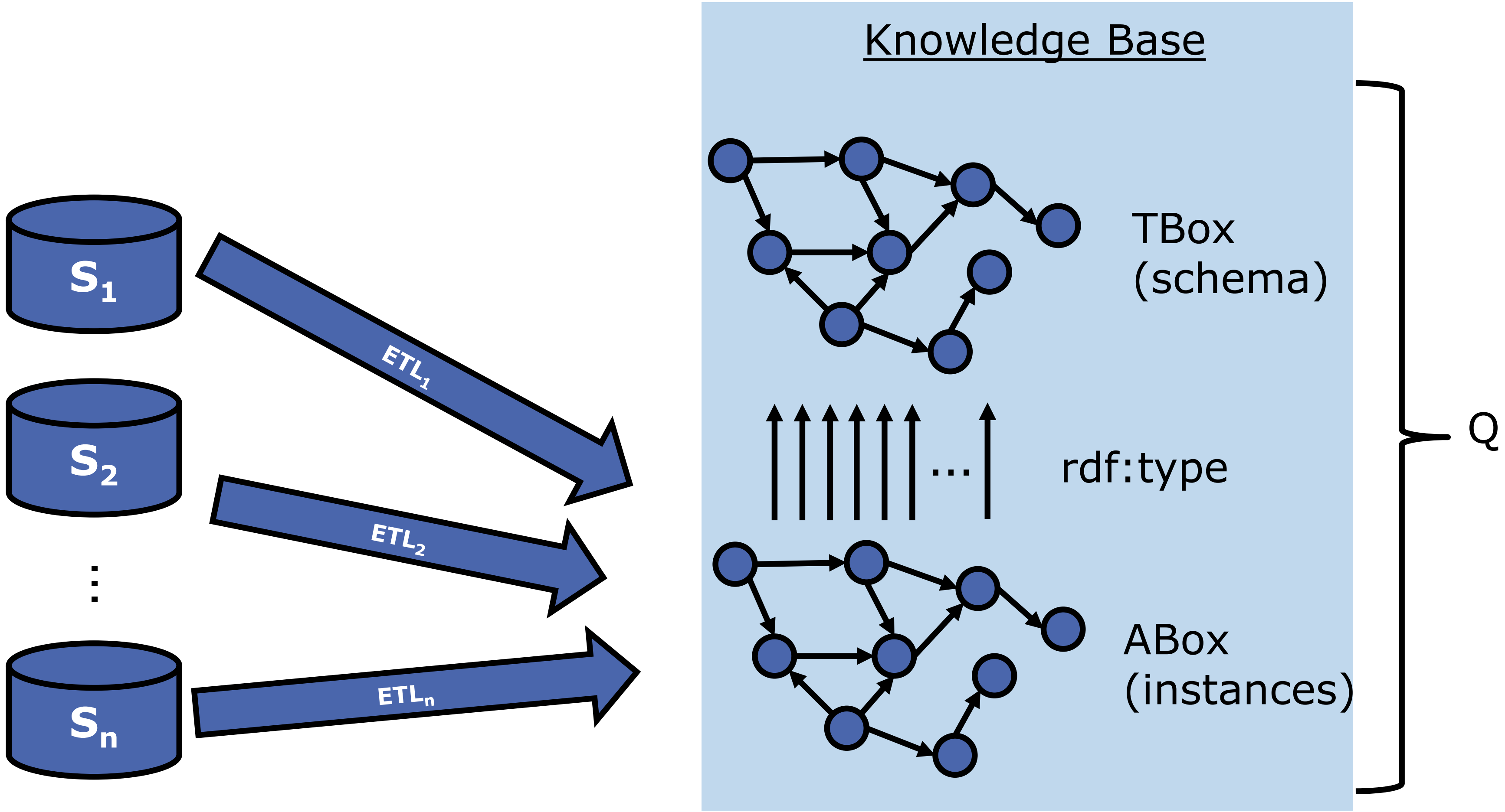
Data integration is an area of study within data management aimed at facilitating **transparent access** to a **variety of heterogeneous data sources**

- Two main options to perform data integration:
  - Physical data integration
  - Virtual data integration

In this course we will focus on using graphs to solve data integration

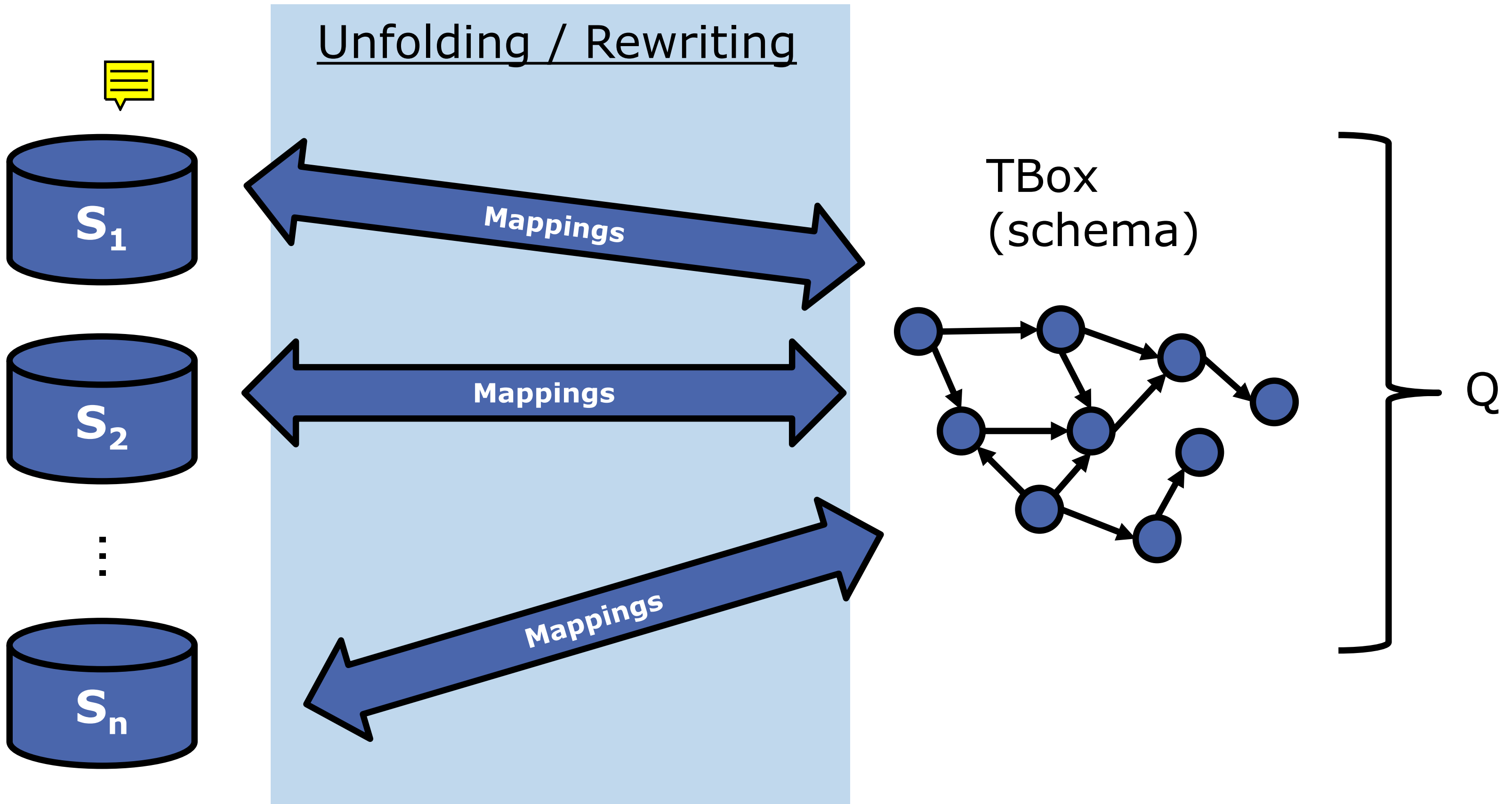
Graph-Based Data Integration at a Glance

Option 1: Physical Data Integration



Graph-Based Data Integration at a Glance

Option 2: Virtual Data Integration



Why Virtual Data Integration?

- When the data sources are not under your control and owners require a federation (e.g., data exchange between companies)
  - E.g., Data Portability
- When we do not want to move the data from where it resides
  - For example, key-based models are more performant than graph models for table scans
- When data freshness is crucial
  - ETLs run from time to time and the period between updates is called the *update window*
- Virtual data integration is simpler to maintain (most of the work resides on the rewriting algorithm)

Why Virtual Data Integration?

- Virtual Integration is very trendy in Data Science because we can use graph-based models to perform rich and complex data integration while benefitting from sequential reads when querying the data (which can reside in the most appropriate data storage)
- Many Big Data Integration systems work under this assumption
  - Data Tamer (https://www.tamr.com/)
  - The BigDAWG Polystore System (https://dl.acm.org/citation.cfm?id=3226620)
  - Ontop (http://ontop.inf.unibz.it/)
  - ODIN (http://www.essi.upc.edu/~snadal/odin.html)

Two Main Approaches

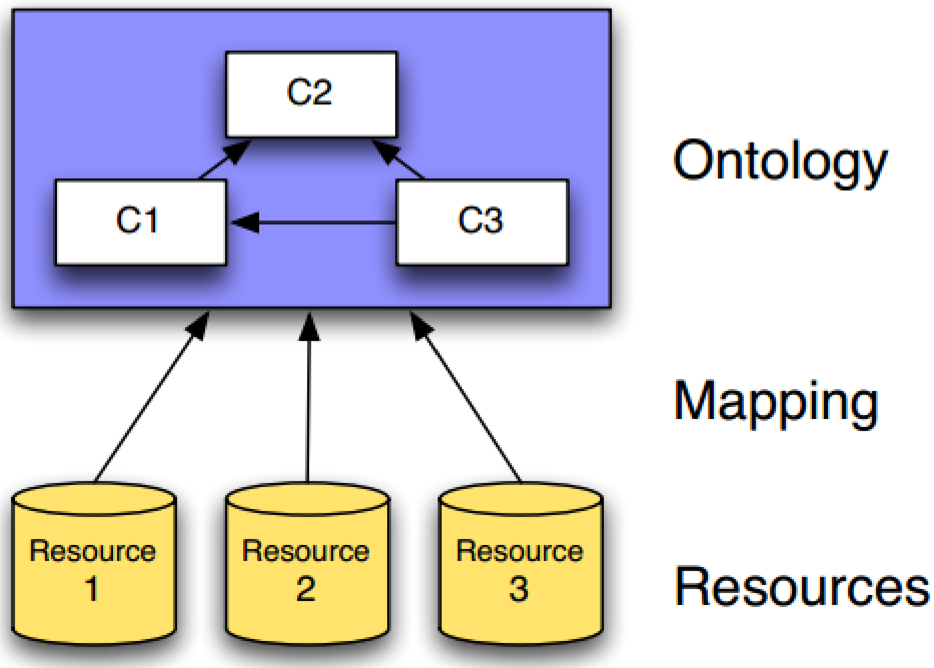
- Ontology-based data access
  - Monolithic approach
  - The TBox is directly related to the sources via mappings
- Ontology-mediated queries
  - Relies on the concept of wrapper
    - Thus, we can select a subset of the data source to be exposed to the whole integration System
      - Security
      - Modularity
  - It allows pay-as-you-go data Integration
    - The integrated schema is built incrementally as new data sources arrive

GAV Data Integration

ONTOLOGY-BASED DATA ACCESS

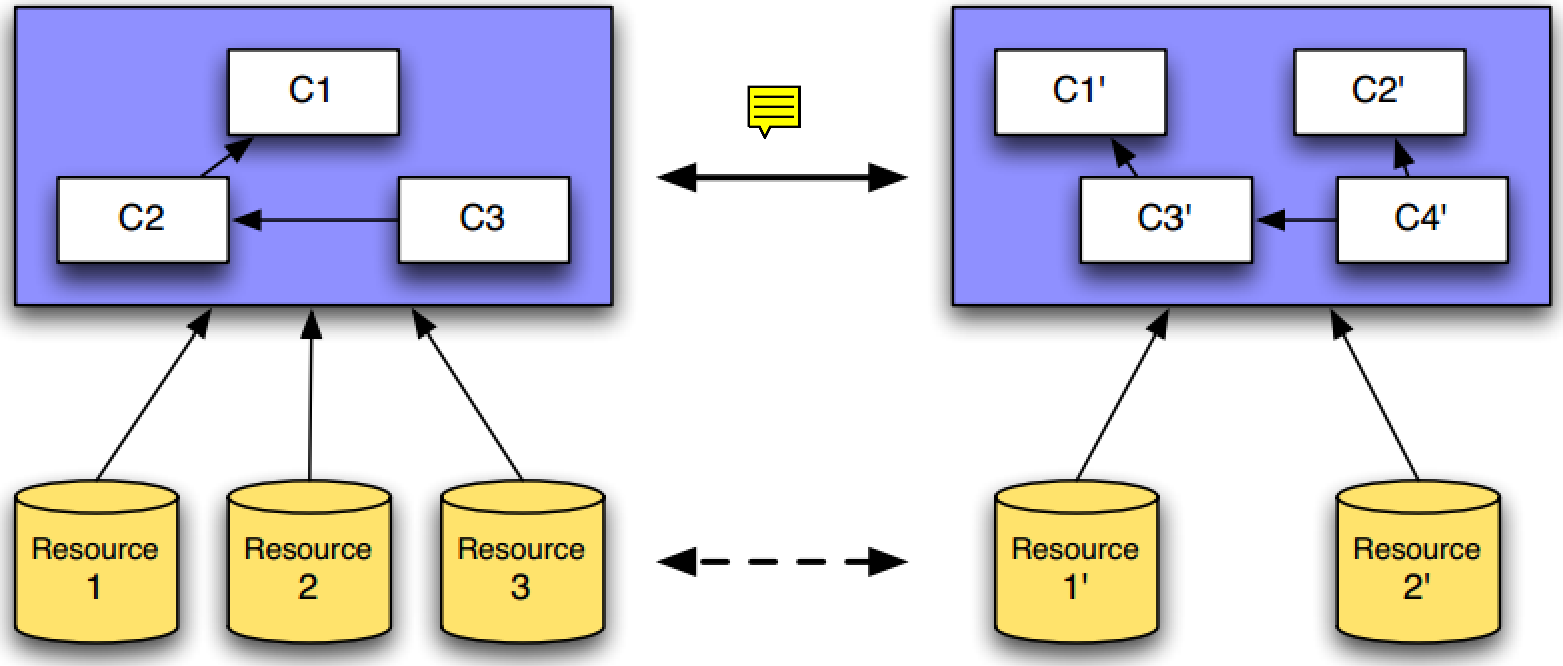
Ontology-Based Data Access

- Ontology-mediated data access



Ontology-Based Data Access

- An Approach for Data Integration



The DL-Lite Family

- Right trade-off between expressiveness and data complexity query answering
  - PTime in the size of the TBOX
  - LogSpace in the size of the ABOX
- Two maximal DLs satisfying this trade-off
  - DL-Lite$_F$
  - DL-Lite$_R$

Remember the DL-Lite family maps to **<u>OWL 2 QL</u>**

DL-Lite$_F$

TBox assertions:

- Concept inclusion assertions: $Cl \sqsubseteq Cr$, with:

$$\begin{array}{lcl} Cl & \longrightarrow & A \mid \exists Q \\ Cr & \longrightarrow & A \mid \exists Q \mid \neg A \mid \neg \exists Q \\ Q & \longrightarrow & P \mid P^{-} \end{array}$$

- Functionality assertions: $(\textbf{funct}\ Q)$

ABox assertions: $A(c)$, $P(c_1, c_2)$, with c_1, c_2 constants

Observations:

- Captures all the basic constructs of UML Class Diagrams and ER
- Notable exception: covering constraints in generalizations.

Semantics of DL-Lite

- It basically captures the expressivity of a UML class diagram

ISA between classes	$A_1 \sqsubseteq A_2$
Disjointness between classes	$A_1 \sqsubseteq \neg A_2$
Domain and range of relations	$\exists P \sqsubseteq A_1$ $\quad \exists P^- \sqsubseteq A_2$
Mandatory participation	$A_1 \sqsubseteq \exists P$ $\quad A_2 \sqsubseteq \exists P^-$
Functionality of relations (in *DL-Lite*$_\mathcal{F}$)	$(\textbf{funct}\ P)$ $\quad (\textbf{funct}\ P^-)$
ISA between relations (in *DL-Lite*$_\mathcal{R}$)	$Q_1 \sqsubseteq Q_2$
Disjointness between relations (in *DL-Lite*$_\mathcal{R}$)	$Q \sqsubseteq \neg Q$

Semantics of DL-Lite

Construct	Syntax	Example	Semantics
atomic conc.	A	Doctor	$A^\mathcal{I} \subseteq \Delta^\mathcal{I}$
exist. restr.	$\exists Q$	$\exists$child$^-$	$\{d \mid \exists e. (d,e) \in Q^\mathcal{I}\}$
at. conc. neg.	$\neg A$	$\neg$Doctor	$\Delta^\mathcal{I} \setminus A^\mathcal{I}$
conc. neg.	$\neg\exists Q$	$\neg\exists$child	$\Delta^\mathcal{I} \setminus (\exists Q)^\mathcal{I}$
atomic role	P	child	$P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
inverse role	P^-	child$^-$	$\{(o,o') \mid (o',o) \in P^\mathcal{I}\}$
role negation	$\neg Q$	$\neg$manages	$(\Delta_O^\mathcal{I} \times \Delta_O^\mathcal{I}) \setminus Q^\mathcal{I}$
conc. incl.	$Cl \sqsubseteq Cr$	Father $\sqsubseteq \exists$child	$Cl^\mathcal{I} \subseteq Cr^\mathcal{I}$
role incl.	$Q \sqsubseteq R$	hasFather $\sqsubseteq$ child$^-$	$Q^\mathcal{I} \subseteq R^\mathcal{I}$
funct. asser.	$(\textbf{funct}\ Q)$	(**funct** succ)	$\forall d,e,e'.(d,e) \in Q^\mathcal{I} \wedge (d,e') \in Q^\mathcal{I} \rightarrow e = e'$
mem. asser.	$A(c)$	Father(bob)	$c^\mathcal{I} \in A^\mathcal{I}$
mem. asser.	$P(c_1, c_2)$	child(bob, ann)	$(c_1^\mathcal{I}, c_2^\mathcal{I}) \in P^\mathcal{I}$

Linking Data to Ontologies

- The ABOX is stored in a relational database
  - OBDA has recently been extended to other kind of sources, like document-stores
- Direct mappings between the TBOX and DB
  - Query answering is reformulated in terms of the TBOX, a set of mappings and a RDBMS

Theorem

Query answering in a *DL-Lite*$_{\mathcal{A}}$ OBDM system $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, \mathcal{D} \rangle$ is

1. NP-complete in the size of the query.
2. PTIME in the size of the TBox $\mathcal{T}$ and the mappings $\mathcal{M}$.
3. LOGSPACE in the size of the database $\mathcal{D}$.

Mappings

- OBDA works with GAV mappings
- Typically, they use RDF-based mapping languages to express them
  - R2RML (a language to express mappings from global concepts to relational databases)

```
mappingId       Actor
target          imdb:name/{person_id} a dbpedia:Actor .
source          select person_id from cast_info where cast_info.role_id = 1
```

  - RML is a generalisation to map to any kind of source (http://rml.io/)

A Tool for OBDA

- Ontop: http://ontop.inf.unibz.it/
  - OWL 2 QL
  - RDFS

- Code, examples and more:

  https://github.com/ontop/ontop

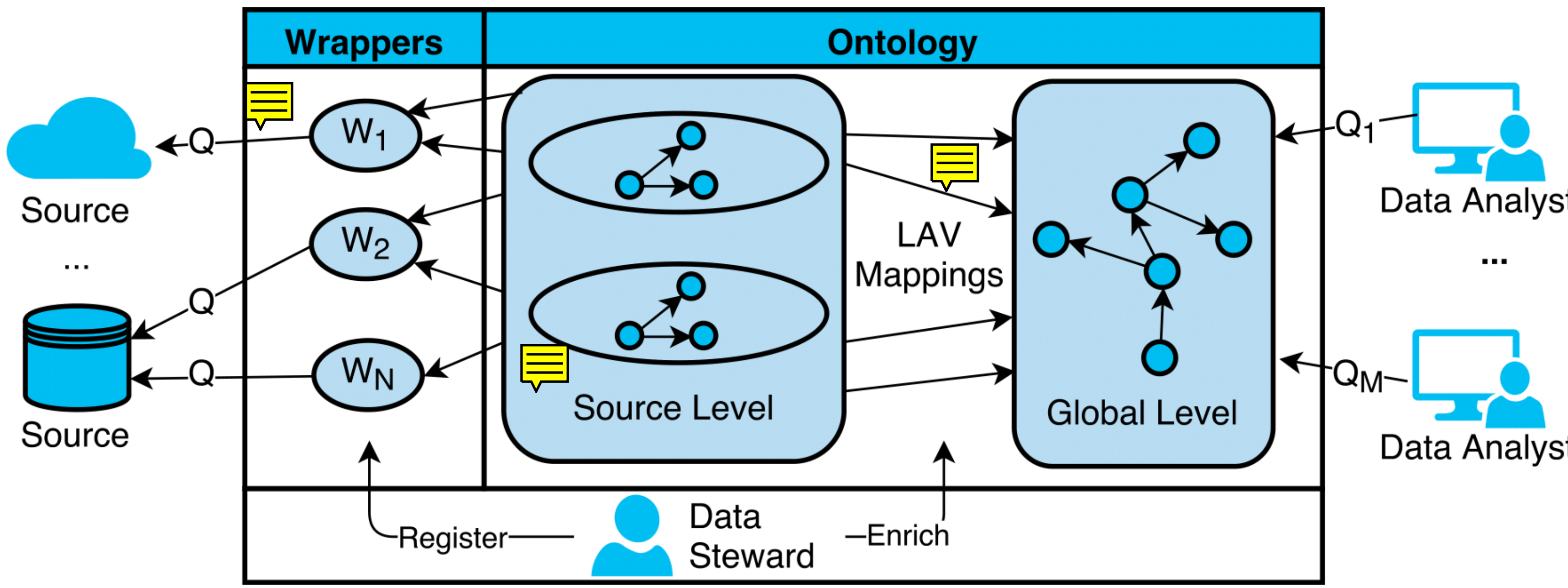
LAV Data Integration

ONTOLOGY-MEDIATED QUERIES

OMQ

- It is a family of systems performing graph-based data integration with LAV
  - Conceptually, GAV is also possible
- **Based on the well-known wrapper-mediator architecture**
- To make the querying rewriting feasible, they adopt several measures:
  - Exact mappings (i.e., Closed-World assumption)
  - Very basic reasoning capabilities (taxonomies and domain / range inference)

Ontology-mediated Query



Virtual integration with LAV mappings

Big Data Integration Ontology

- We revisit the Data Integration framework and construct an ontology as follows:
  - Global level (*G*) – Integrated view
  - Source levels (*S*) – Views on the data sources (wrappers)
  - Mappings (*M*) – LAV mappings between *G* and *S*
- Example:
  - Cross-domain queries on:
    - Monitored data on video players (lag ratio, etc.)
    - Tweets in English gathered through a feedback gathering tool

Wrappers

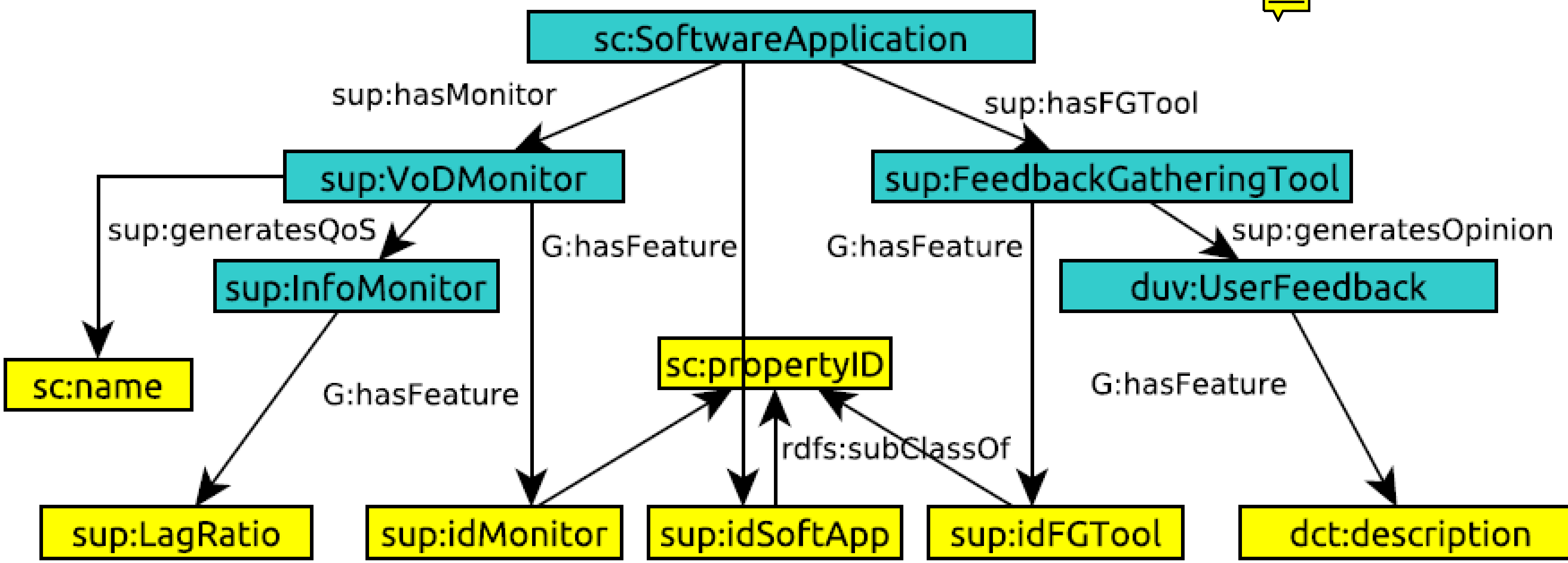
- They represent a view on the source
- You can think of a **<u>named</u>** query over the source. For example:

  W1: `SELECT a, b, c FROM T`

- Typical assumptions made by wrappers:
  - They expose the source in **<u>tabular format</u>** (1NF)
    - Thus, Cypher, SPARQL or MongoDB's aggregation framework would also meet the requirements
    - In general, most query languages produce tabular format
  - A data source may generate several wrappers
  - Typically, new versions of data are considered new wrappers

Global Level

- Green: concepts
- Yellow: attributes



Source Level

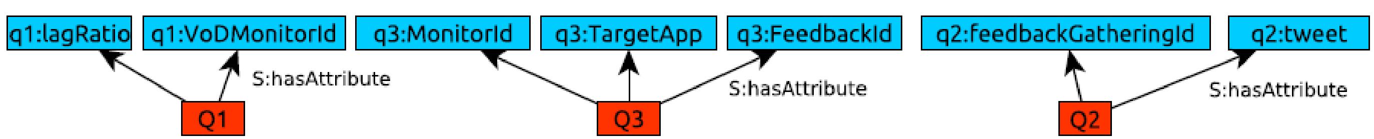
- Sources are exposed by means of wrappers
  - We automatically bootstrap the attributes projected by the wrappers

Q_1: ID and compute the lag ratio

```
db.getCollection('vod').aggregate([
  {$project: {"VoDmonitorId":true, "lagRatio": {$divide : ["$waitTime","$watchTime"]}}}
])
```

Q_2 - all attributes for tweets in english.

Q_3 - association target app $\rightarrow$ monitor, feedback gathering tool

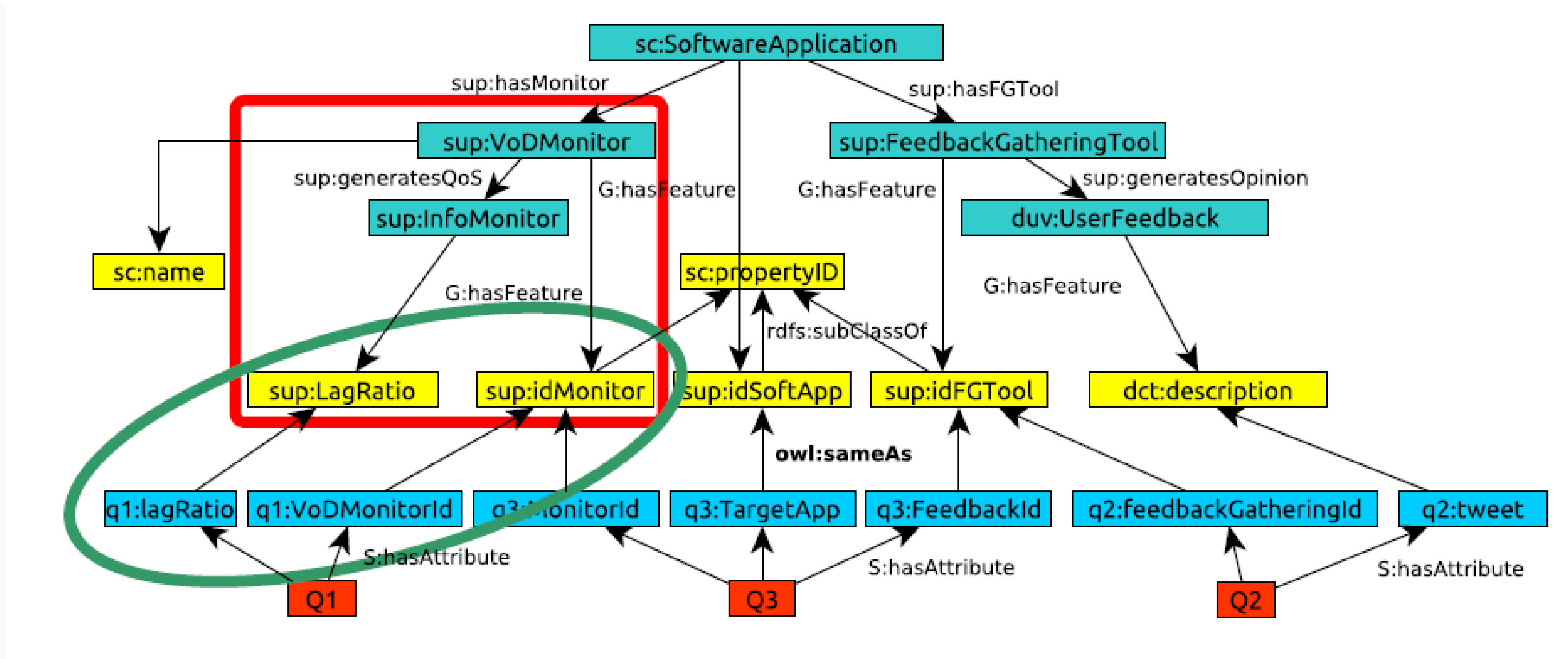


Red: Wrappers; Blue: Wrapper attributes

Mappings

- A LAV mapping for a wrapper *Q* is defined as: *M*=<*G*,*S*> where:
  - *G* is a named graph
  - *S* is a set of triples of the form:
    - <x, owl:sameAs, y>, where
    - <x, rdf:type, S:Attribute> and
    - <y, rdf:type, G:Feature>

LAV Mapping Example



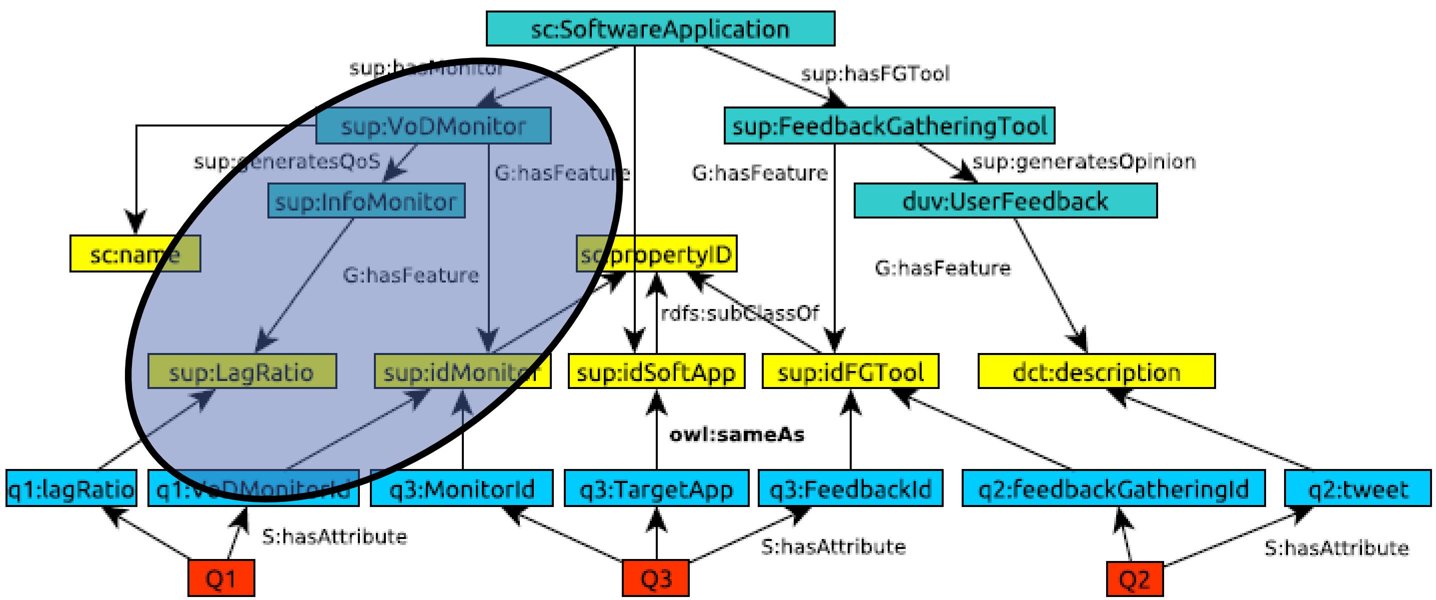
G (named graph):

```
Q1 S:provides { sup:InfoMonitor G:hasFeature sup:lagRatio .
sup:VoDMonitor sup:generatesQoS sup:InfoMonitor .
sup:VoDMonitor G:hasFeature sup:idMonitor }}
```

S (“*same as*” triples):

```
q1:lagRatio owl:sameAs sup:lagRatio
q1:VoDMonitorId owl:sameAs sup:idMonitor
```

LAV Mappings (Q1)

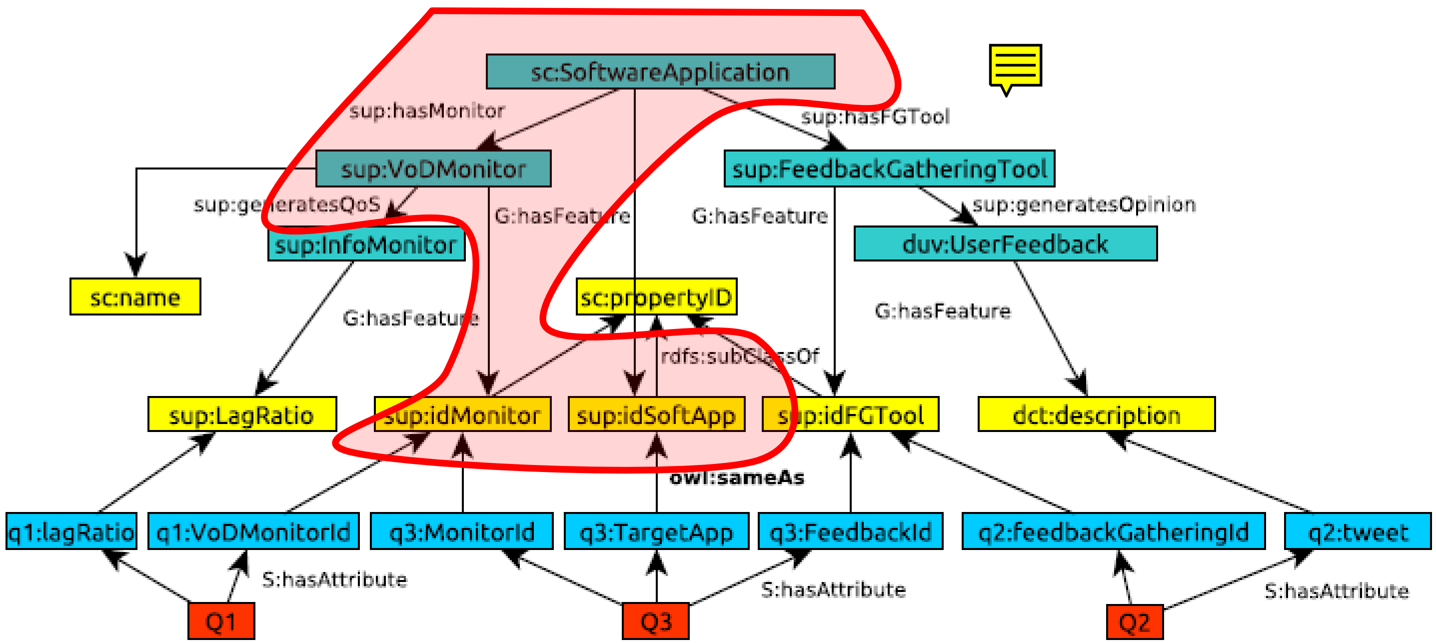



```
InfoMonitor hasF lagRatio
VoDMonitor QoS InfoMonitor
VoDMonitor hasF idMonitor
```

```
SoftwareApp hasF idSoftApp
SoftwareApp hasMon VoDMonitor
VoDMonitor hasF sup:idMonitor
```

```
UserFeedback hasF desc
FGTool opinion UserFeedback
FGTool hasF idFGTool
```

LAV Mappings (Q3)

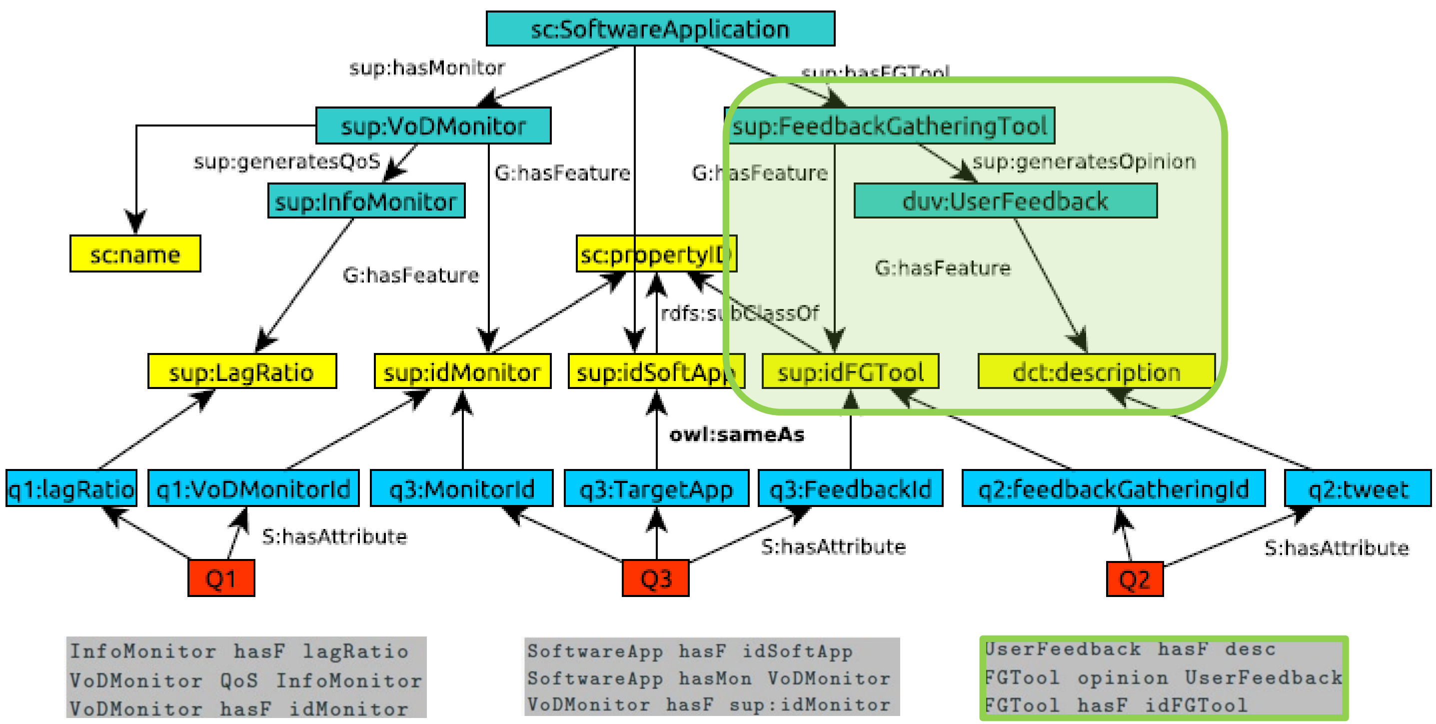



```
InfoMonitor hasF lagRatio
VoDMonitor QoS InfoMonitor
VoDMonitor hasF idMonitor
```

```
SoftwareApp hasF idSoftApp
SoftwareApp hasMon VoDMonitor
VoDMonitor hasF sup:idMonitor
```

```
UserFeedback hasF desc
FGTool opinion UserFeedback
FGTool hasF idFGTool
```

LAV Mappings (Q2)



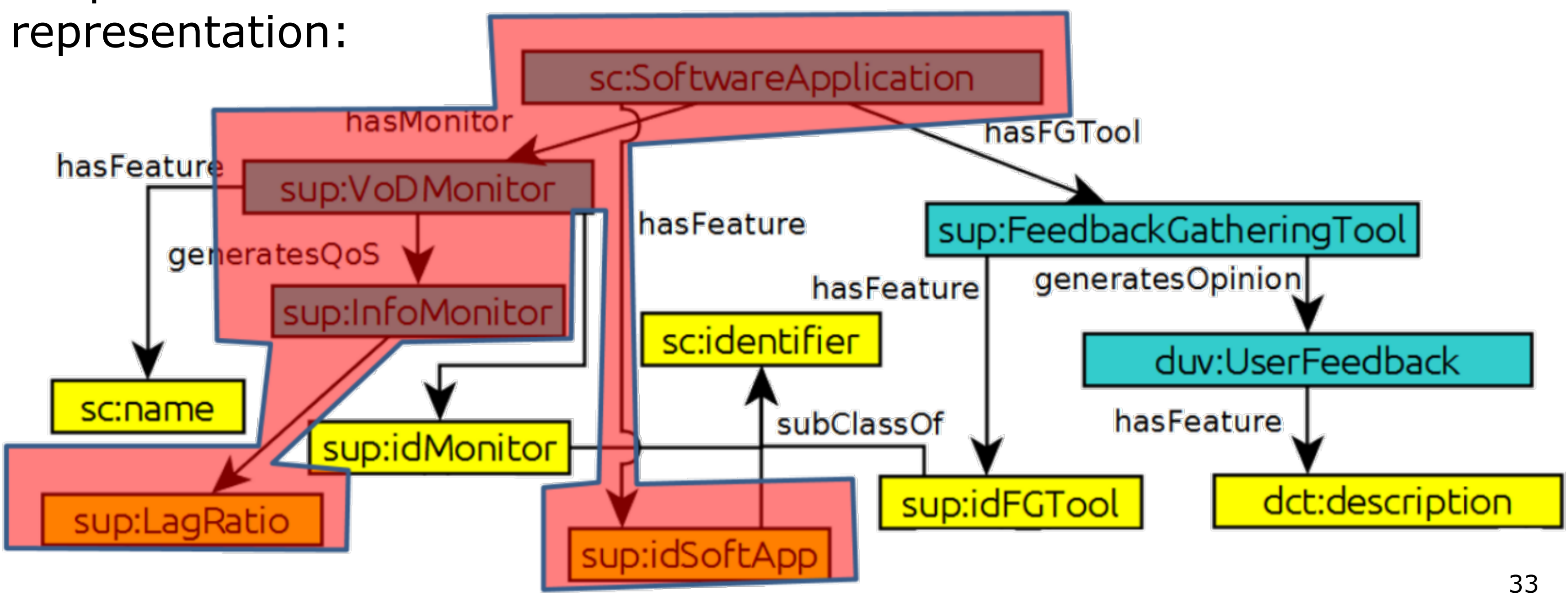
Query Answering – Rewriting Algorithm

- **Any SPARQL query on the global graph must be rewritten as a query in terms of the wrappers**
- Example of query over *G*:

SPARQL Query:

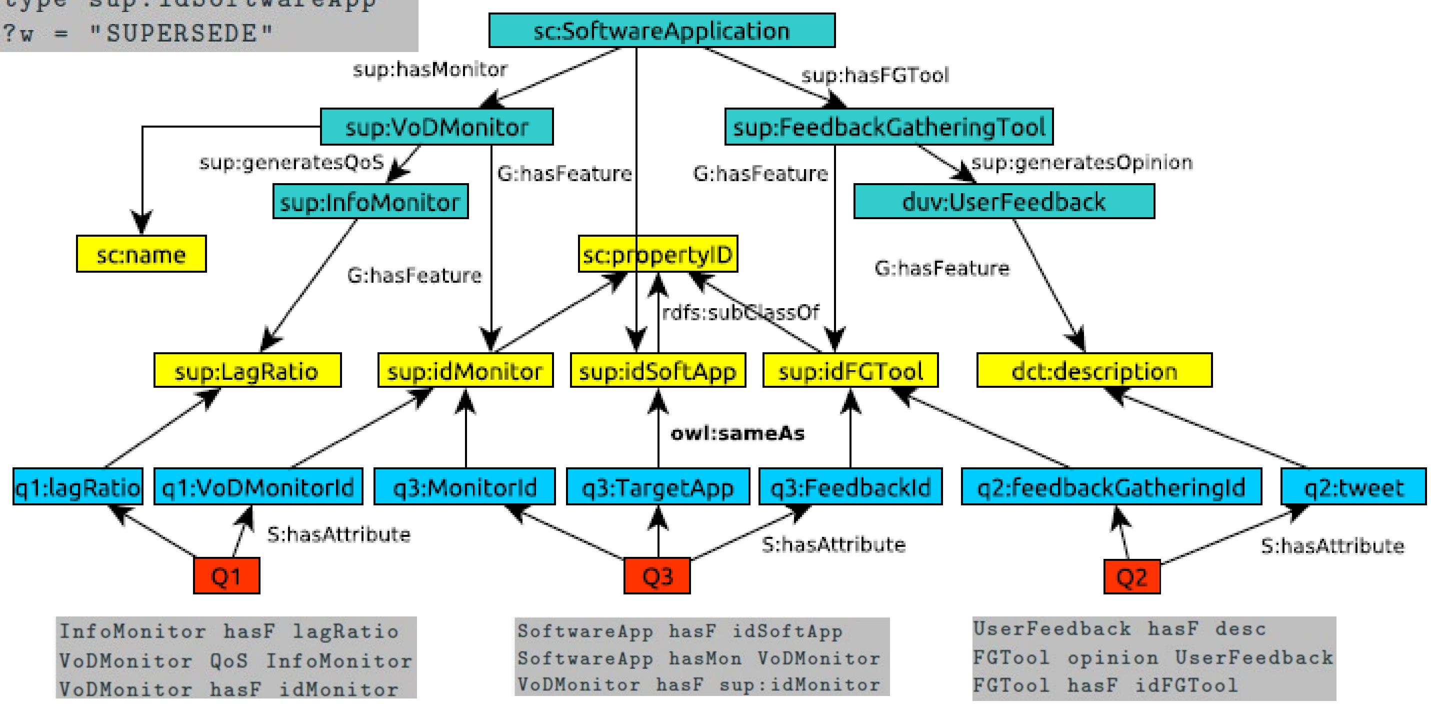
```
SELECT ?w,?t WHERE
  ?t rdf:type sup:lagRatio
  ?x G:hasFeature ?t
  ?x rdf:type sup:InfoMonitor
  ?y sup:generatesQoS ?x
  ?y rdf:type sup:VoDMonitor
  ?z sup:hasMonitor ?y
  ?z rdf:type sc:SoftwareApp
  ?z G:hasFeature ?w
  ?w rdf:type sup:idSoftwareApp
  FILTER ?w = "SUPERSEDE"
```

Graph representation:



Notions on the Query Rewriting Alg.

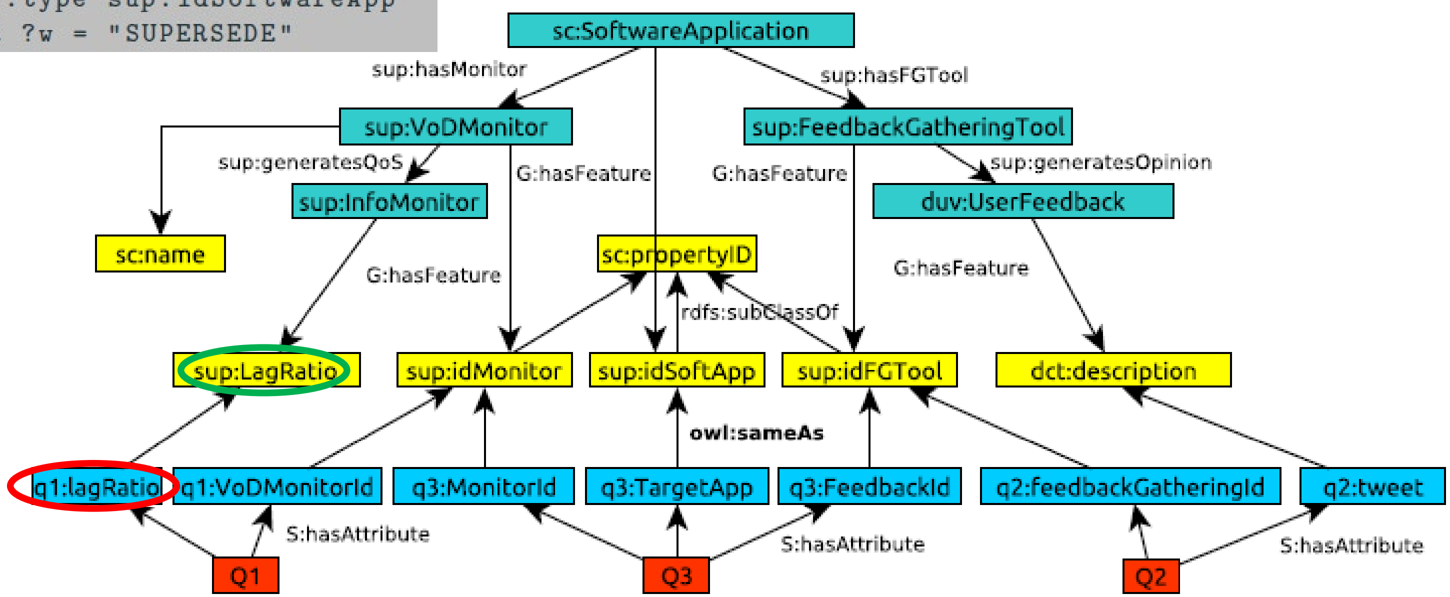
```
SELECT ?w,?t WHERE
  ?t rdf:type sup:lagRatio
  ?x G:hasFeature ?t
  ?x rdf:type sup:InfoMonitor
  ?y sup:generatesQoS ?x
  ?y rdf:type sup:VoDMonitor
  ?z sup:hasMonitor ?y
  ?z rdf:type sc:SoftwareApp
  ?z G:hasFeature ?w
  ?w rdf:type sup:idSoftwareApp
  FILTER ?w = "SUPERSEDE"
```



Start from a Terminal Feature

```
SELECT ?w,?t WHERE
 ?t rdf:type sup:lagRatio
 ?x G:hasFeature ?t
 ?x rdf:type sup:InfoMonitor
 ?y sup:generatesQoS ?x
 ?y rdf:type sup:VoDMonitor
 ?z sup:hasMonitor ?y
 ?z rdf:type sc:SoftwareApp
 ?z G:hasFeature ?w
 ?w rdf:type sup:idSoftwareApp
 FILTER ?w = "SUPERSEDE"
```

$$\Pi_t(\rho_{Q_1.lagRatio \to t}(Q_1))$$




```
InfoMonitor hasF lagRatio
VoDMonitor QoS InfoMonitor
VoDMonitor hasF idMonitor
```

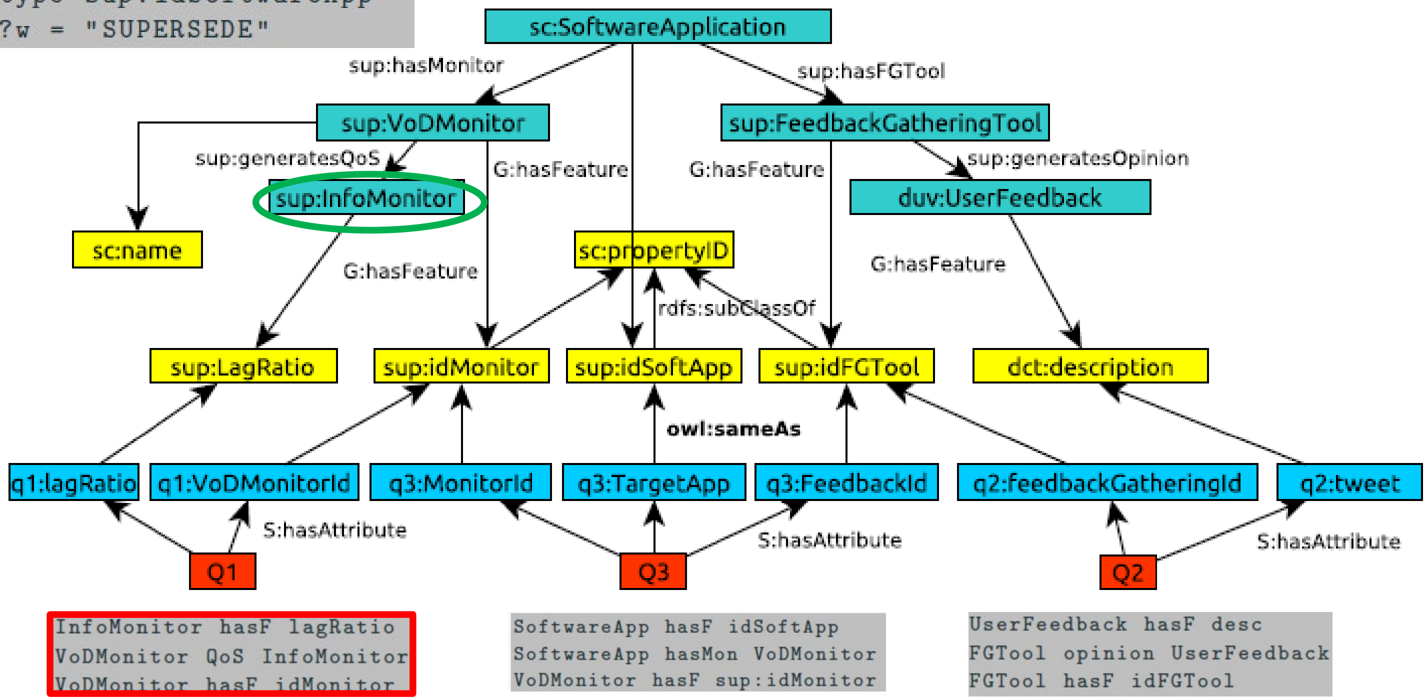
```
SoftwareApp hasF idSoftApp
SoftwareApp hasMon VoDMonitor
VoDMonitor hasF sup:idMonitor
```

```
UserFeedback hasF desc
FGTool opinion UserFeedback
FGTool hasF idFGTool
```

Navigate *G* from the Feature

```
SELECT ?w,?t WHERE
  ?t rdf:type sup:lagRatio
  ?x G:hasFeature ?t
  ?x rdf:type sup:InfoMonitor
  ?y sup:generatesQoS ?x
  ?y rdf:type sup:VoDMonitor
  ?z sup:hasMonitor ?y
  ?z rdf:type sc:SoftwareApp
  ?z G:hasFeature ?w
  ?w rdf:type sup:idSoftwareApp
  FILTER ?w = "SUPERSEDE"
```

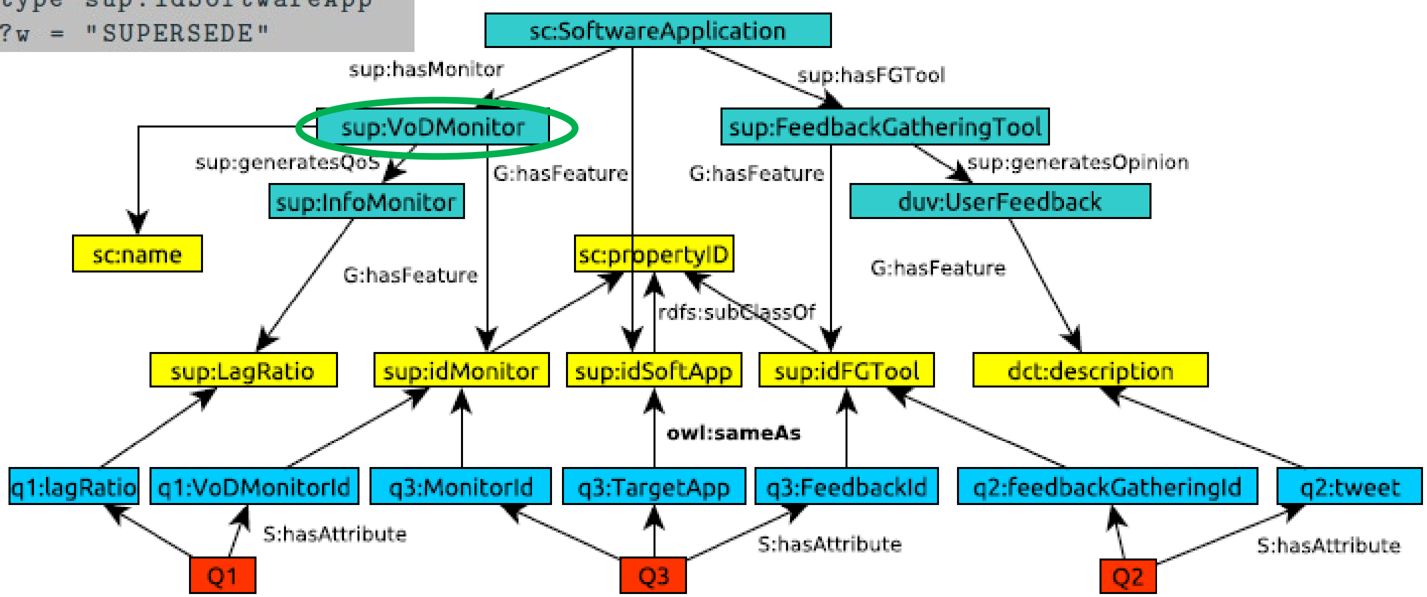
$$\Pi_{t}(\rho_{Q_1.lagRatio \rightarrow t}(Q_1))$$



Navigate *G* from the Feature

```
SELECT ?w,?t WHERE
  ?t rdf:type sup:lagRatio
  ?x G:hasFeature ?t
  ?x rdf:type sup:InfoMonitor
  ?y sup:generatesQoS ?x
  ?y rdf:type sup:VoDMonitor
  ?z sup:hasMonitor ?y
  ?z rdf:type sc:SoftwareApp
  ?z G:hasFeature ?w
  ?w rdf:type sup:idSoftwareApp
  FILTER ?w = "SUPERSEDE"
```

$$\Pi_{t}(\rho_{Q_1.lagRatio \rightarrow t}(Q_1))$$




```
InfoMonitor hasF lagRatio
VoDMonitor QoS InfoMonitor
VoDMonitor hasF idMonitor
```

```
SoftwareApp hasF idSoftApp
SoftwareApp hasMon VoDMonitor
VoDMonitor hasF sup:idMonitor
```

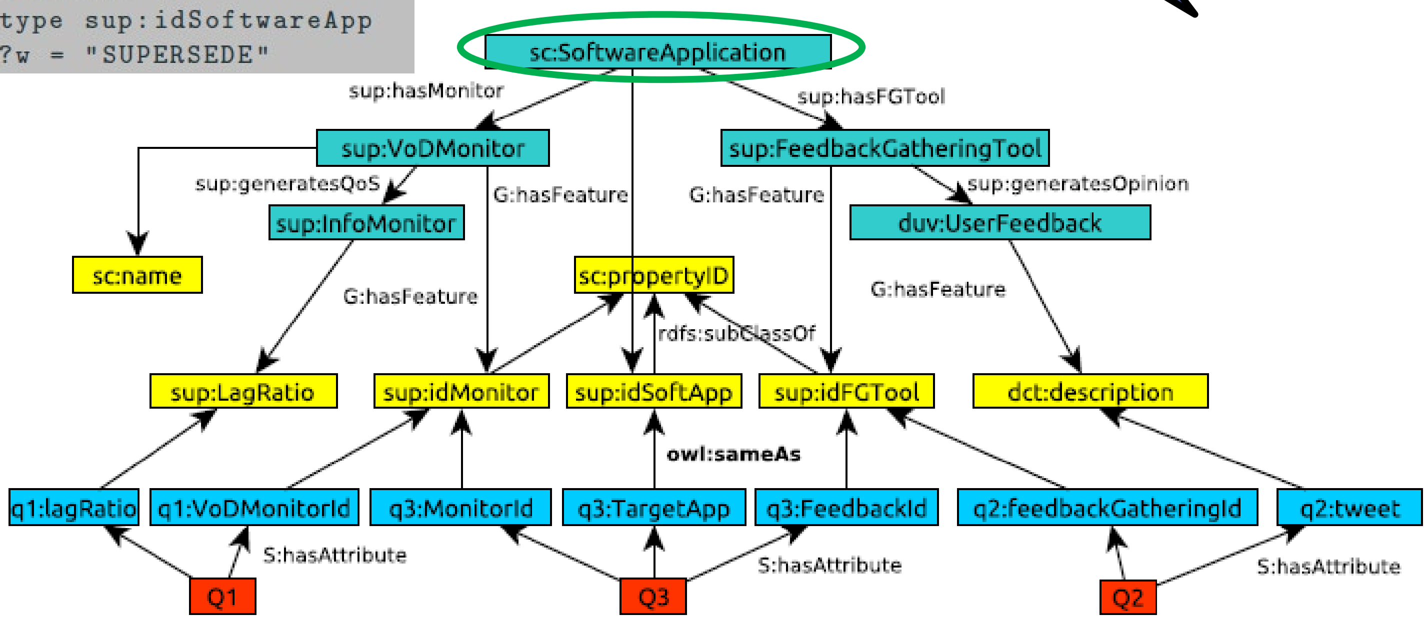
```
UserFeedback hasF desc
FGTool opinion UserFeedback
FGTool hasF idFGTool
```

Navigate G from the Feature

```
SELECT ?w,?t WHERE
  ?t rdf:type sup:lagRatio
  ?x G:hasFeature ?t
  ?x rdf:type sup:InfoMonitor
  ?y sup:generatesQoS ?x
  ?y rdf:type sup:VoDMonitor
  ?z sup:hasMonitor ?y
  ?z rdf:type sc:SoftwareApp
  ?z G:hasFeature ?w
  ?w rdf:type sup:idSoftwareApp
  FILTER ?w = "SUPERSEDE"
```

$$\Pi_{t}(\rho_{Q_1.lagRatio \to t}(Q_1))$$

Subpath not cointained in Q1!!




```
InfoMonitor hasF lagRatio
VoDMonitor QoS InfoMonitor
VoDMonitor hasF idMonitor
```

```
SoftwareApp hasF idSoftApp
SoftwareApp hasMon VoDMonitor
VoDMonitor hasF sup:idMonitor
```

```
UserFeedback hasF desc
FGTool opinion UserFeedback
FGTool hasF idFGTool
```

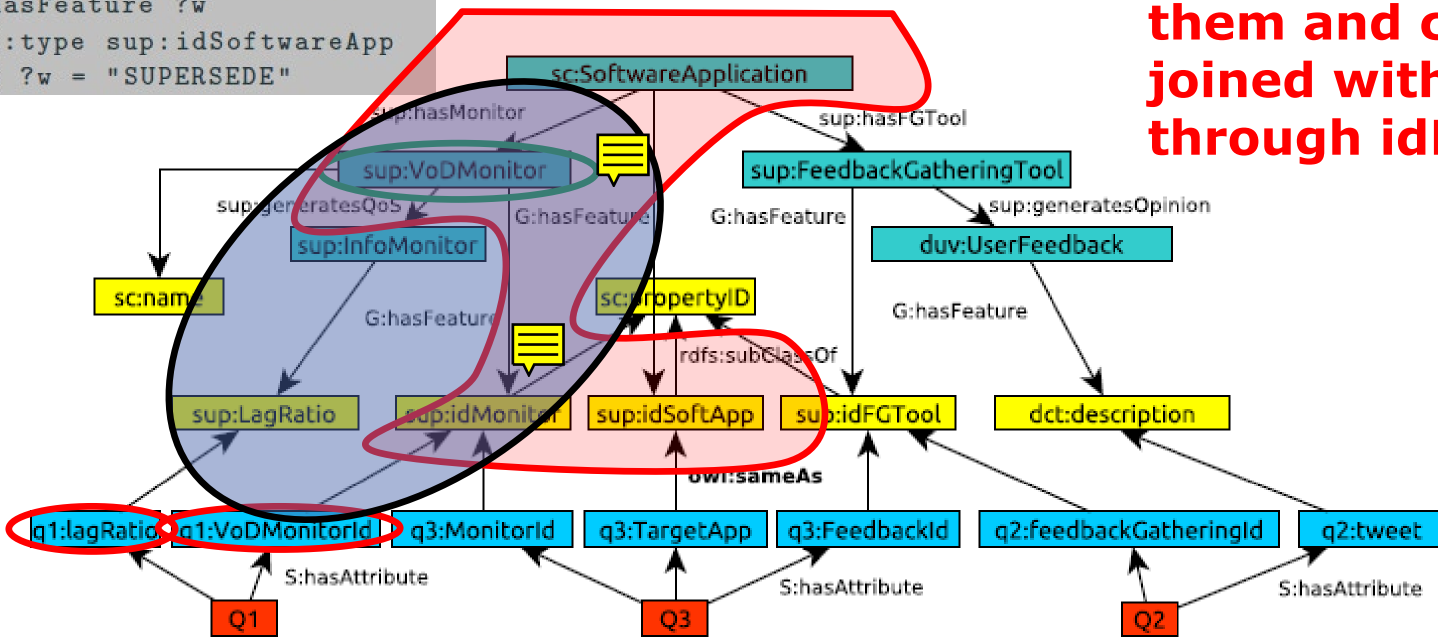
Explore Join Candidates

```
SELECT ?w,?t WHERE
  ?t rdf:type sup:lagRatio
  ?x G:hasFeature ?t
  ?x rdf:type sup:InfoMonitor
  ?y sup:generatesQoS ?x
  ?y rdf:type sup:VoDMonitor
  ?z sup:hasMonitor ?y
  ?z rdf:type sc:SoftwareApp
  ?z G:hasFeature ?w
  ?w rdf:type sup:idSoftwareApp
  FILTER ?w = "SUPERSEDE"
```

$$\Pi_{t}(\rho_{Q_1.lagRatio \to t}(Q_1))$$

Any other wrapper contains these triples?

Yes! Q3 covers them and can be joined with Q1 through idMonitor!




```
InfoMonitor hasF lagRatio
VoDMonitor QoS InfoMonitor
VoDMonitor hasF idMonitor
```

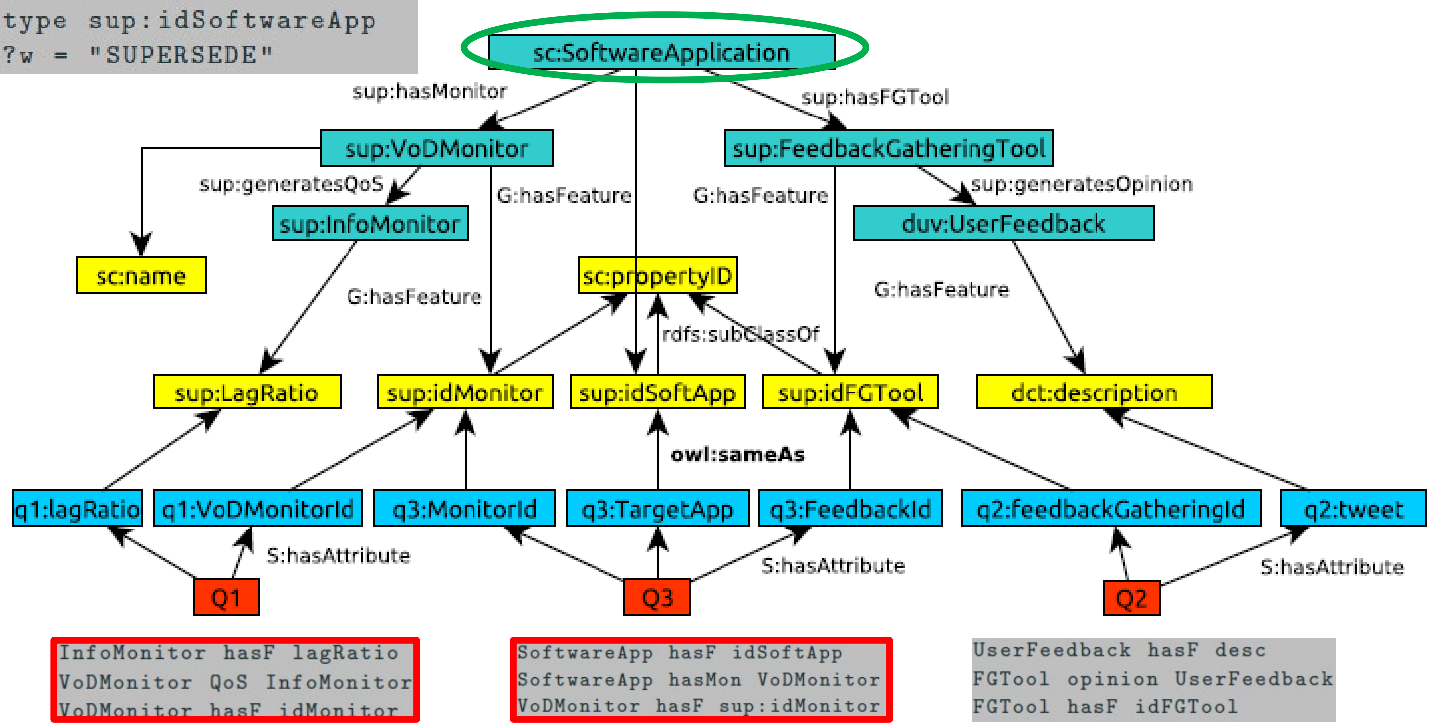
```
SoftwareApp hasF idSoftApp
SoftwareApp hasMon VoDMonitor
VoDMonitor hasF sup:idMonitor
```

```
UserFeedback hasF desc
FGTool opinion UserFeedback
FGTool hasF idFGTool
```

Join to an Alternative Wrapper

```
SELECT ?w,?t WHERE
  ?t rdf:type sup:lagRatio
  ?x G:hasFeature ?t
  ?x rdf:type sup:InfoMonitor
  ?y sup:generatesQoS ?x
  ?y rdf:type sup:VoDMonitor
  ?z sup:hasMonitor ?y
  ?z rdf:type sc:SoftwareApp
  ?z G:hasFeature ?w
  ?w rdf:type sup:idSoftwareApp
  FILTER ?w = "SUPERSEDE"
```

$$\Pi_{t}(\rho_{Q_1.lagRatio\rightarrow t}\ \sigma_{Q_1.VoDMonitorId=Q_3.MonitorId}(Q_1 \times Q_3))$$

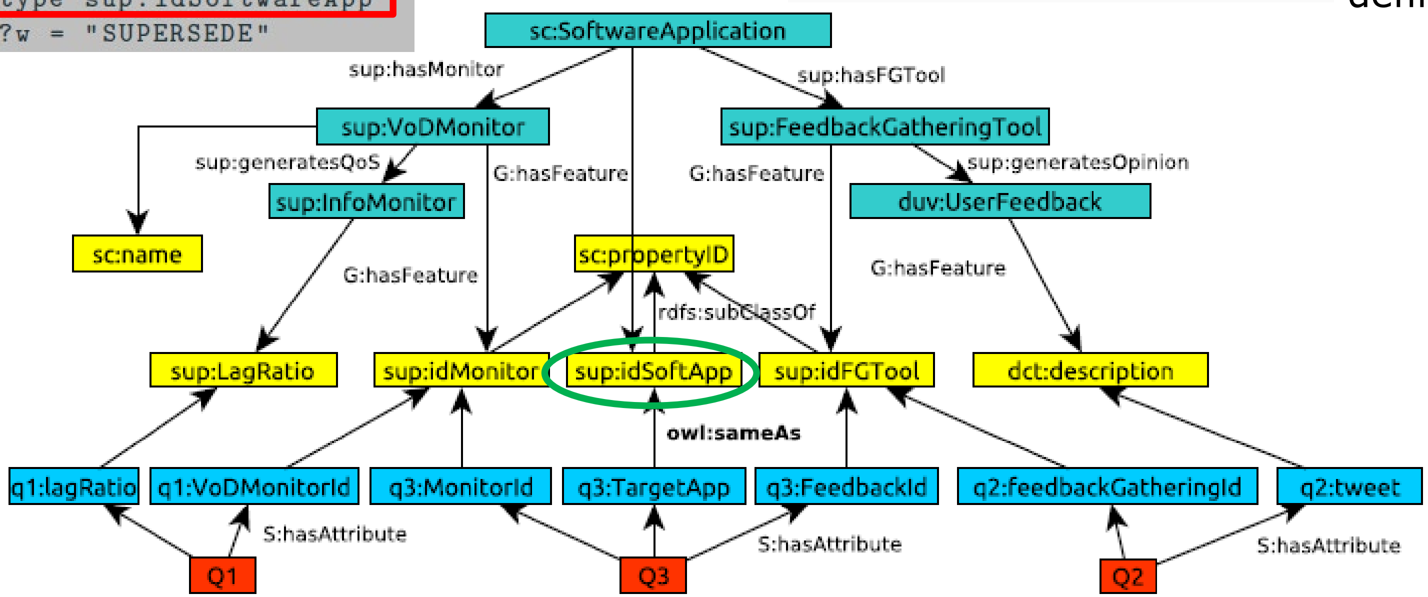


Continue Navigating G

```
SELECT ?w,?t WHERE
  ?t rdf:type sup:lagRatio
  ?x G:hasFeature ?t
  ?x rdf:type sup:InfoMonitor
  ?y sup:generatesQoS ?x
  ?y rdf:type sup:VoDMonitor
  ?z sup:hasMonitor ?y
  ?z rdf:type sc:SoftwareApp
  ?z G:hasFeature ?w
  ?w rdf:type sup:idSoftwareApp
  FILTER ?w = "SUPERSEDE"
```

$$\Pi_{w,t}(\rho_{Q_1.lagRatio \to t}\ \rho_{Q_3.TargetApp \to\ w}\ \sigma_{Q_1.VoDMonitorId = Q_3.MonitorId}\ (Q_1 \times Q_3))$$

This is a query over the wrappers! Now, each wrapper name must be replaced by its definition query




```
InfoMonitor hasF lagRatio
VoDMonitor QoS InfoMonitor
VoDMonitor hasF idMonitor
```

```
SoftwareApp hasF idSoftApp
SoftwareApp hasMon VoDMonitor
VoDMonitor hasF sup:idMonitor
```

```
UserFeedback hasF desc
FGTool opinion UserFeedback
FGTool hasF idFGTool
```

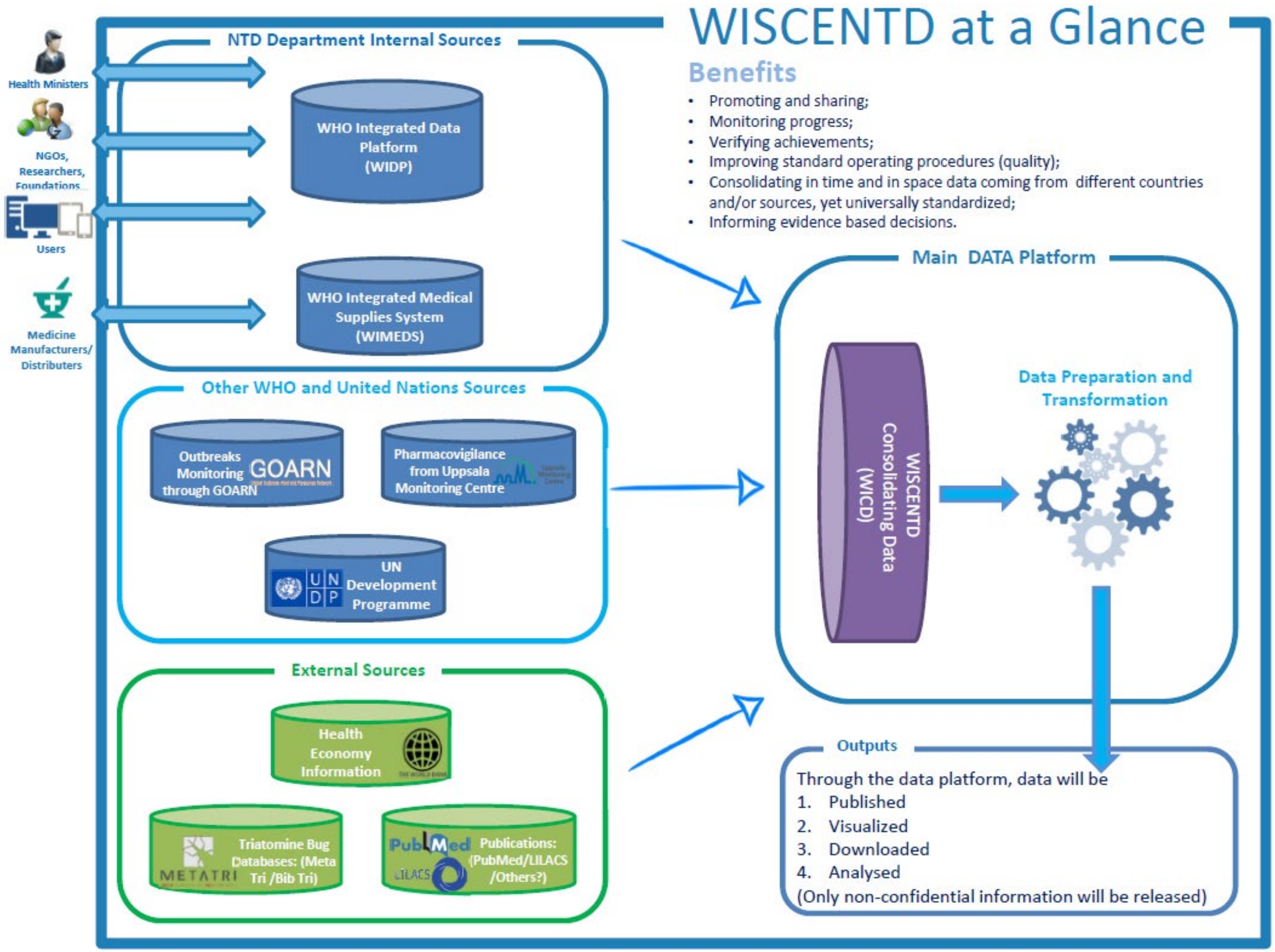
Computational Complexity

- This query rewriting algorithm is:
  - Linear in the size of the subgraph of *G* to navigate
  - Linear in the size of the wrappers mappings
  - Exponential in the number of wrappers that may join
    - Our experiments show that typically Big Data sources have few join points and therefore this exponential complexity is affordable in real cases

WISCENTD

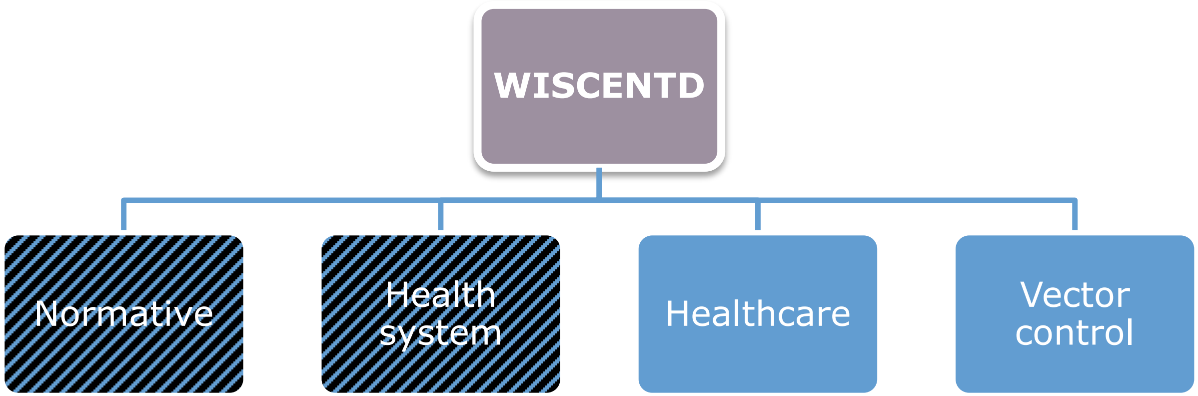
WHO Information System *to* Control/Eliminate NTDs

Example of application: The World Health Organisation


WISCENTD at a Glance
Benefits
Promoting and sharing;
Monitoring progress;
Verifying achievements;
Improving standard operating procedures (quality);
Consolidating in time and in space data coming from different countries and/or sources, yet universally standardized;
Informing evidence based decisions.
NTD Department Internal Sources
Health Ministers
NGOs, Researchers, Foundations
Users
Medicine Manufacturers/ Distributers
WHO Integrated Data Platform (WIDP)
WHO Integrated Medical Supplies System (WIMEDS)
Other WHO and United Nations Sources
Outbreaks Monitoring through GOARN
GOARN
Pharmacovigilance from Uppsala Monitoring Centre
UN Development Programme
UNDP
External Sources
Health Economy Information
Triatomine Bug Databases: (Meta Tri /Bib Tri)
METATRI
Publications: (PubMed/LILACS /Others?)
PubMed
LILACS
Main DATA Platform
WISCENTD Consolidating Data (WICD)
Data Preparation and Transformation
Outputs
Through the data platform, data will be
1. Published
2. Visualized
3. Downloaded
4. Analysed
(Only non-confidential information will be released)

Standardisation

- Data has been organized into 4 packages
  - **Healthcare**: to collect patient data
  - **Vector control**: to collect data on vector control activities
  - **Health system**: to collect general information on how NTDs are included in the national health systems
  - **Normative**: to collect information about regulations implemented in to country in order to control and eliminate NTDs

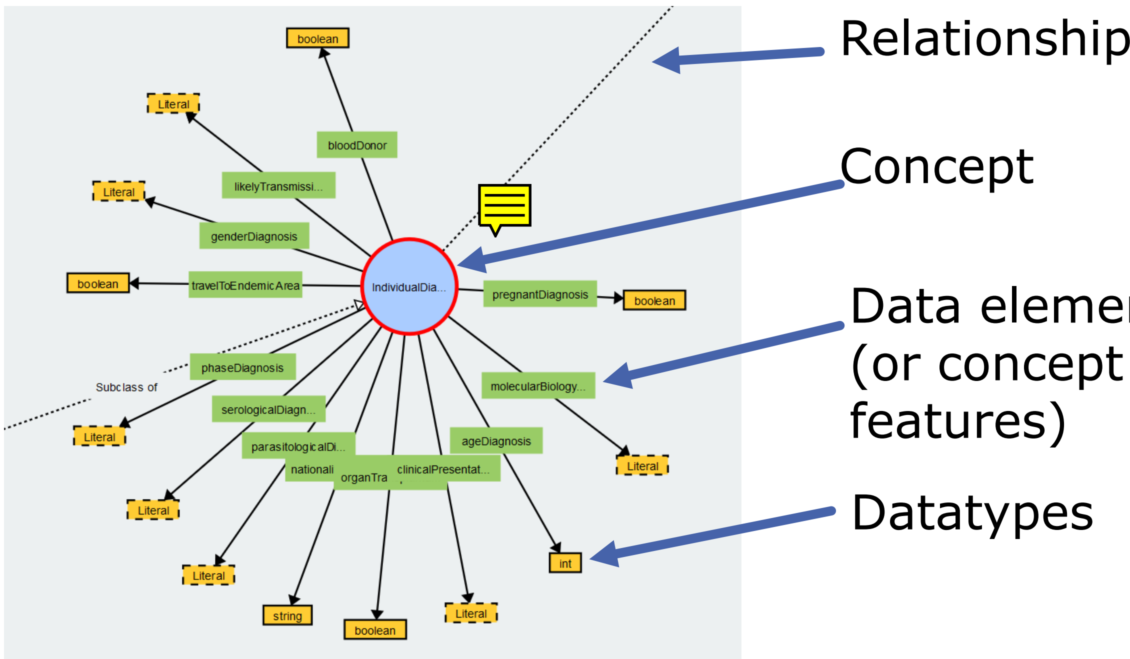


Standardisation

- WISCENTD provides a single **standardised** view of the whole domain

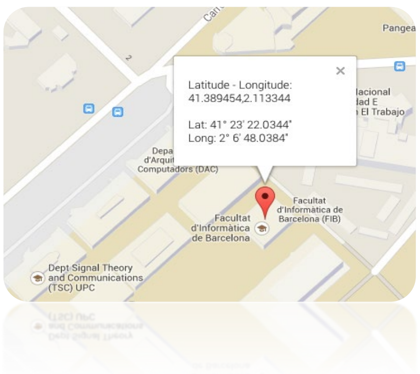
WISCENTD provides a graph-based metaphor representing the domain:

- Concepts
- Data elements of each concept (and their datatypes)
- Relationships between concepts



Master Data: Geographic and Temporal Components

- Coordinates



- Polygons

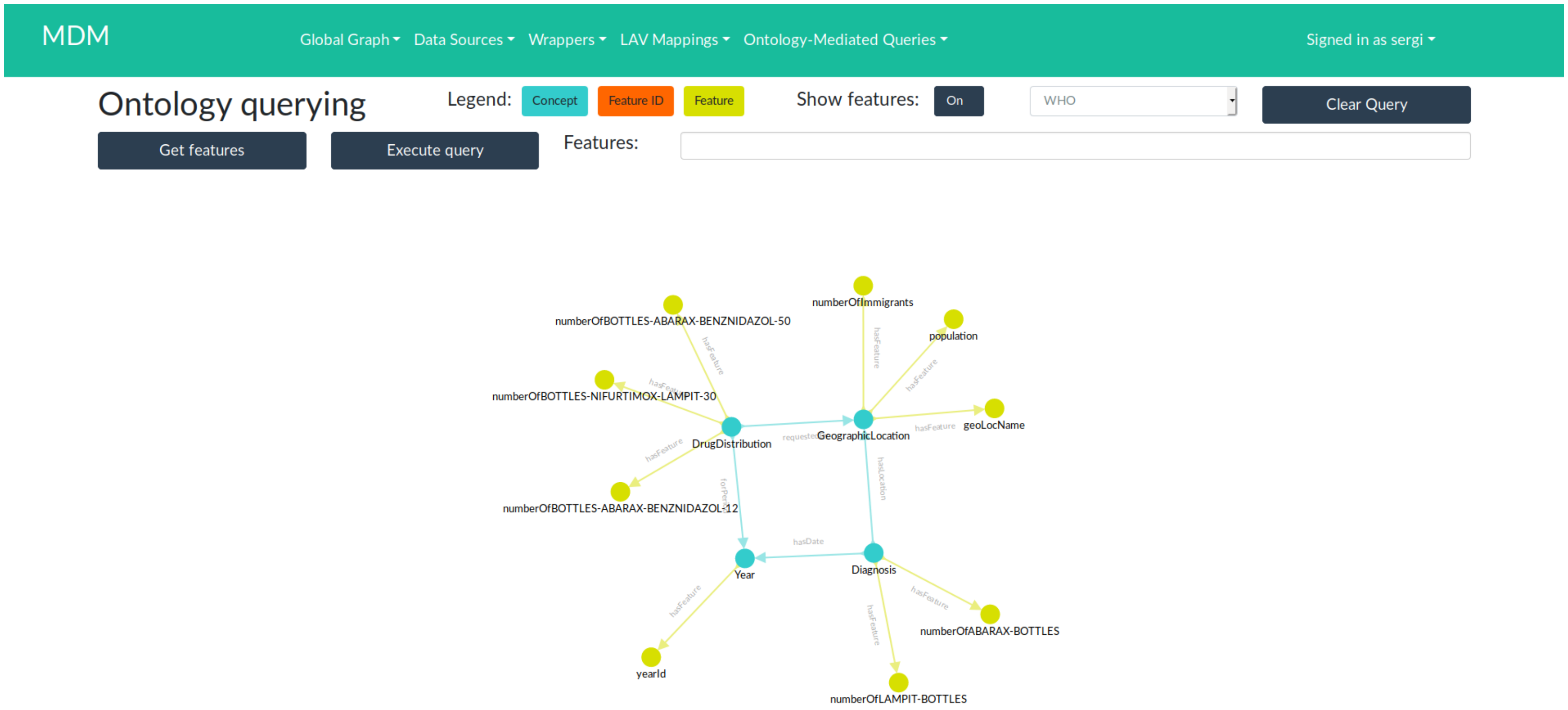
LOCATION

- Timestamps

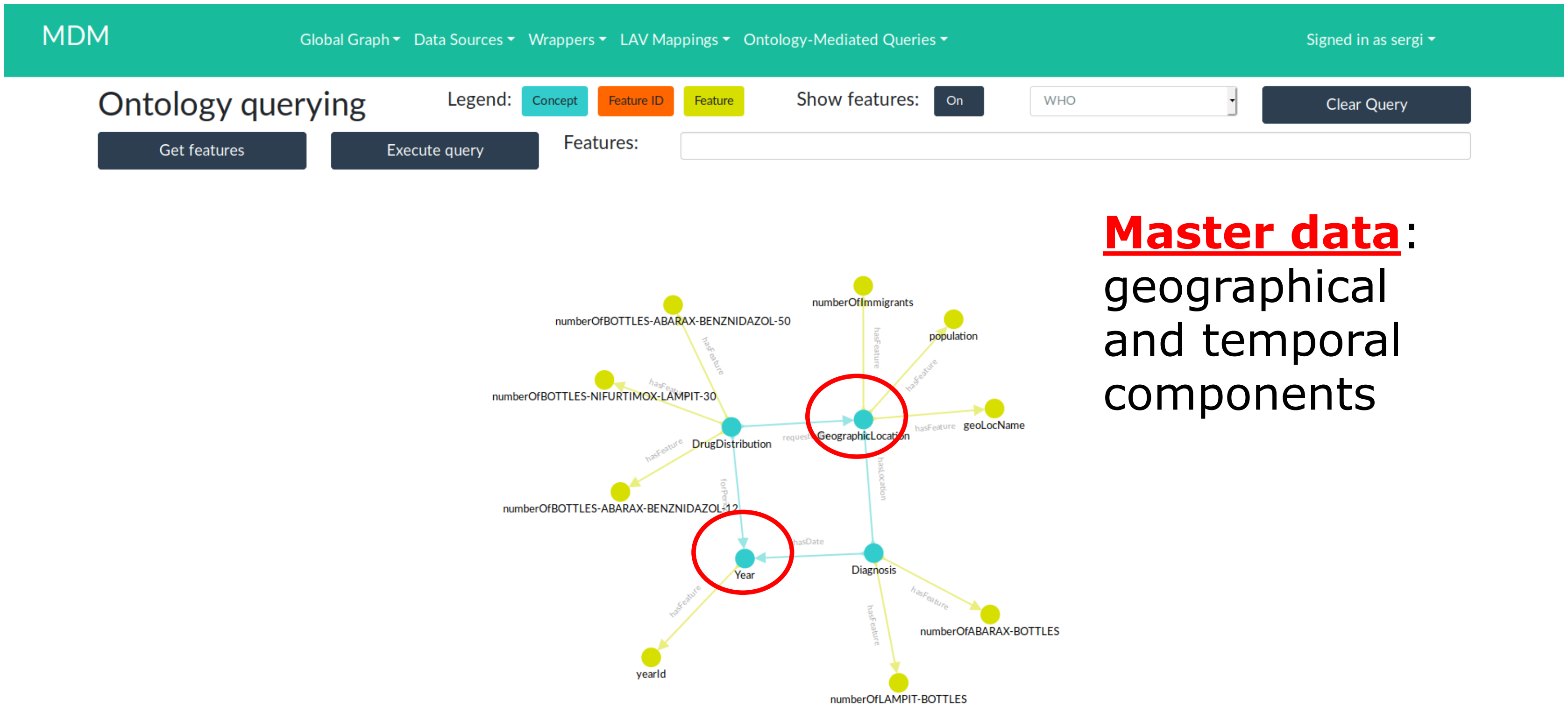
- Time periods

TIME

Data Analysis

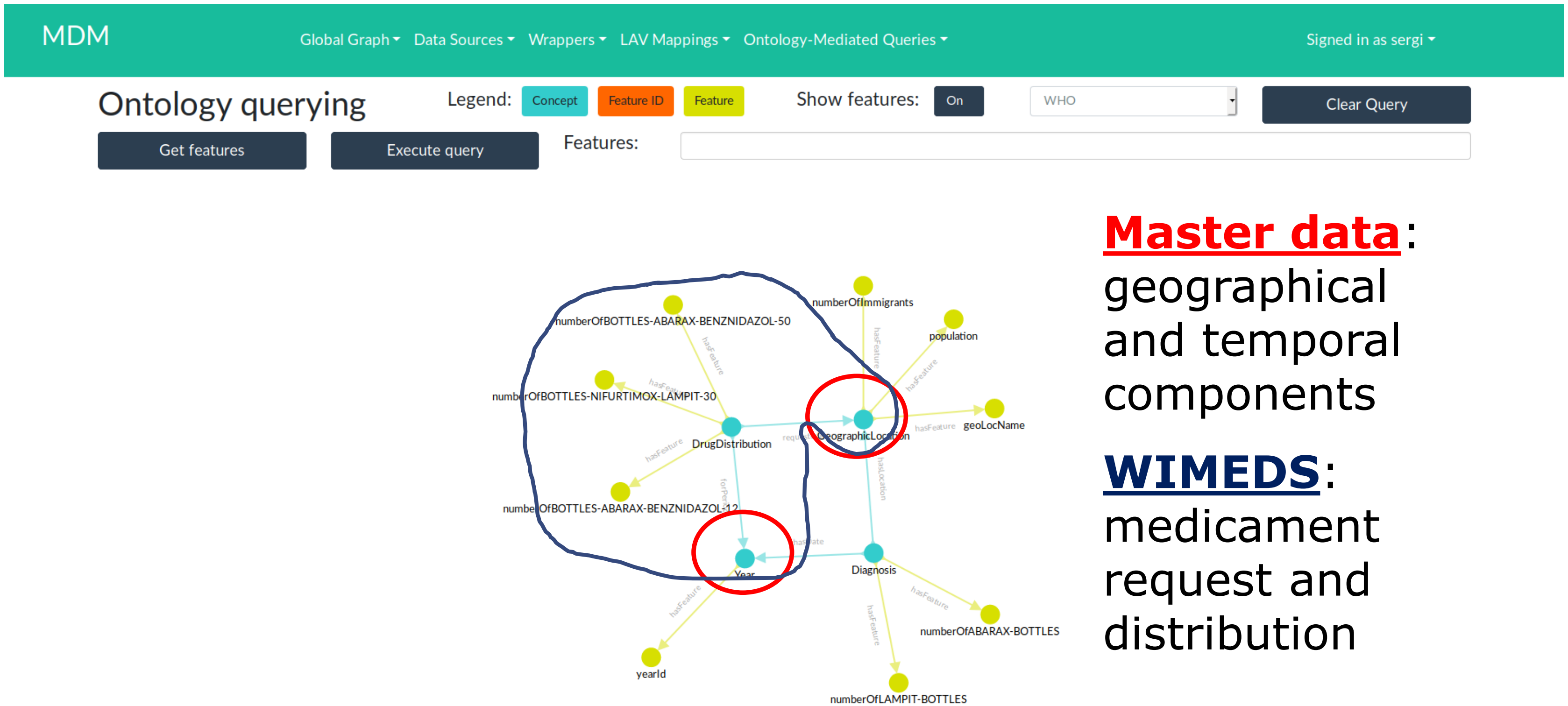


Data Analysis



Master data: geographical and temporal components

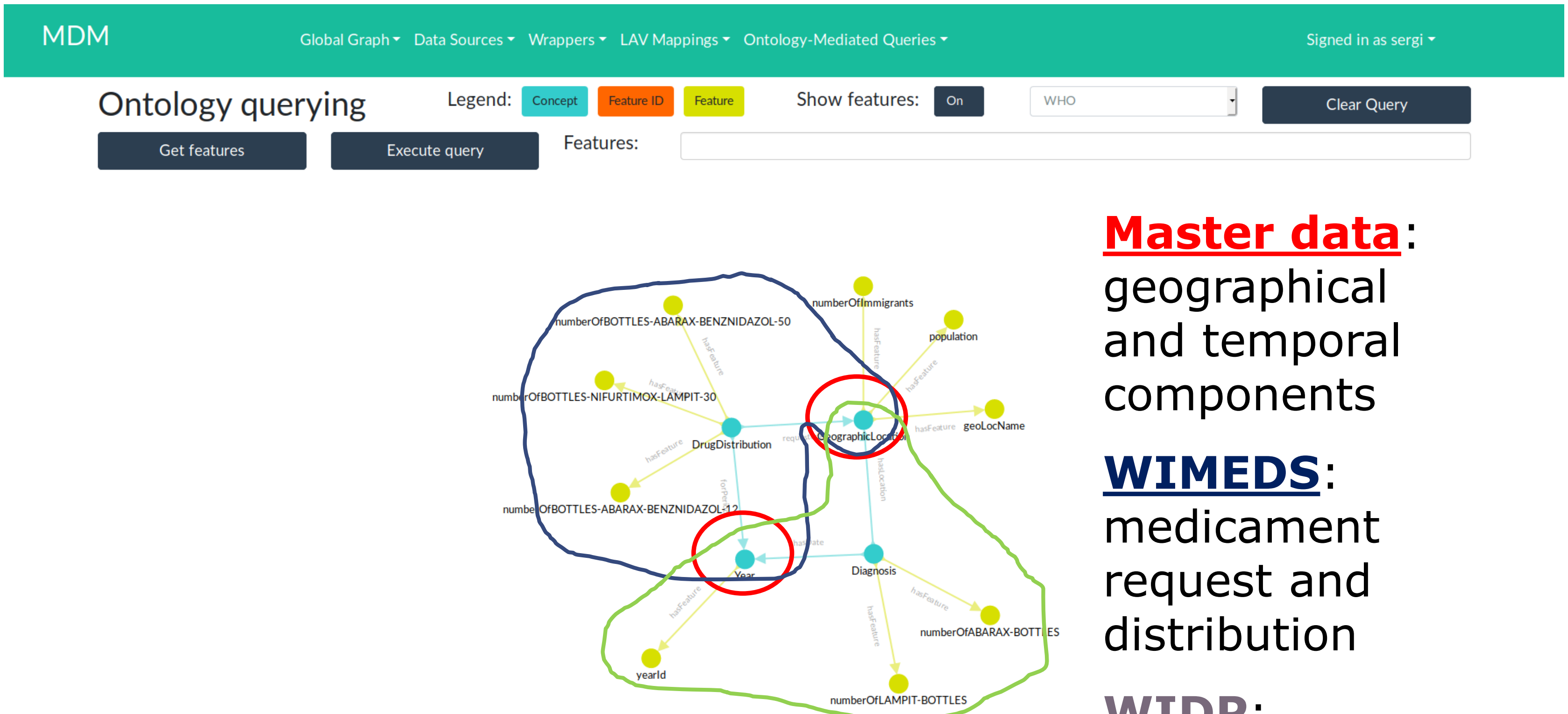
Data Analysis



Master data: geographical and temporal components

WIMEDS: medicament request and distribution

Data Analysis

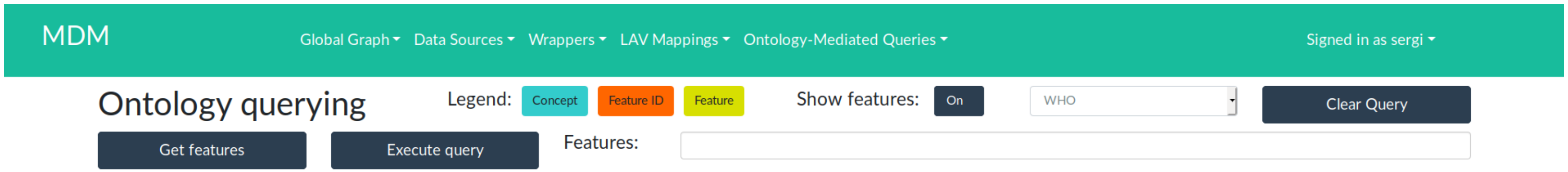


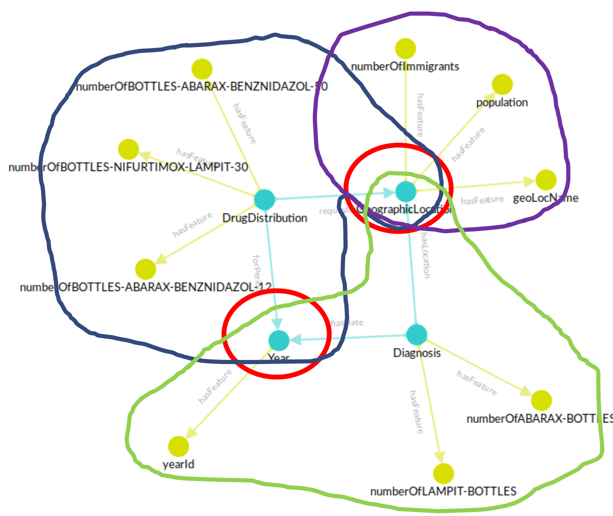
Master data: geographical and temporal components

WIMEDS: medicament request and distribution

WIDP: diagnosis and treatment

Data Analysis





UN Data:
Health economics (two sources: population and immigration data)

Master data:
geographical and temporal components

WIMEDS:
medicament request and distribution

WIDP:
diagnosis and treatment

Data Analysis

"I would like to correlate the number of treatments with the population and number of immigrants of a specific greographical area per year"

Data Analysis

Find this video in Learn-SQL (video 1)

Data Analysis

"I would like to correlate the number of treatments with the number of medicines distributed in a specific greographical area per year.

This information should also include the population and the number of immigrants of that area"

Data Analysis

Find this video in Learn-SQL (video 2)

Management: Extending the Ontology

- *My new data source contains data elements not covered by the attributes of the standardised model. How do I extend it?*

Find this video in Learn-SQL (video 3)

Management: Registering a Source

- *My data source contains data that is not covered in the attributes of the standardised model. How do I extend it?*
- *Great! Now, I want to register a new source providing such data*

Find this video in Learn-SQL (video 4)

Management: Querying a New Source

- *My data source contains data that is not covered in the attributes of the standardised model. How do I extend it?*
- *Great! Now, I want to register a new source providing such data*
- *Good! Now "I would like to see the medicine distribution per medicine sender, per year and geographical area"*

Find this video in Learn-SQL (video 5)

A Tool for OMQ

- ODIN: http://www.essi.upc.edu/~snadal/odin.html
  - RDFS / OWL (but limited reasoning)
  - LAV mappings
  - Pay-as-you-go data integration

Summary

- Graph-based Virtual Data Integration
- Ontology-based Data Access
  - DL-Lite
  - GAV mappings
  - Linking Data to Ontologies
- Ontology-mediated Queries
  - RDFS
  - LAV mappings
  - Sources exposed as wrappers