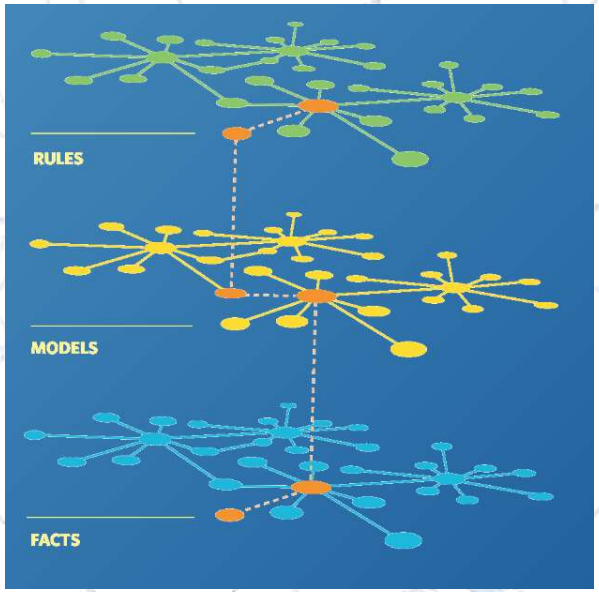
TopQuadrant™

Getting Started with Knowledge Graphs: Main Challenges and How to Overcome Them



TopBraid EDG™
Enterprise Data Governance

TOPQUADRANT COMPANY

FOUNDATION

- TopQuadrant was founded in 2001
- Strong commitment to standards-based approaches to data semantics

MISSION

- Empower people and drive results — by making enterprise information meaningful

FOCUS

- Provide comprehensive data governance solutions using knowledge graph technologies

CUSTOMERS

- Over 120 active customer organizations

Today's Agenda

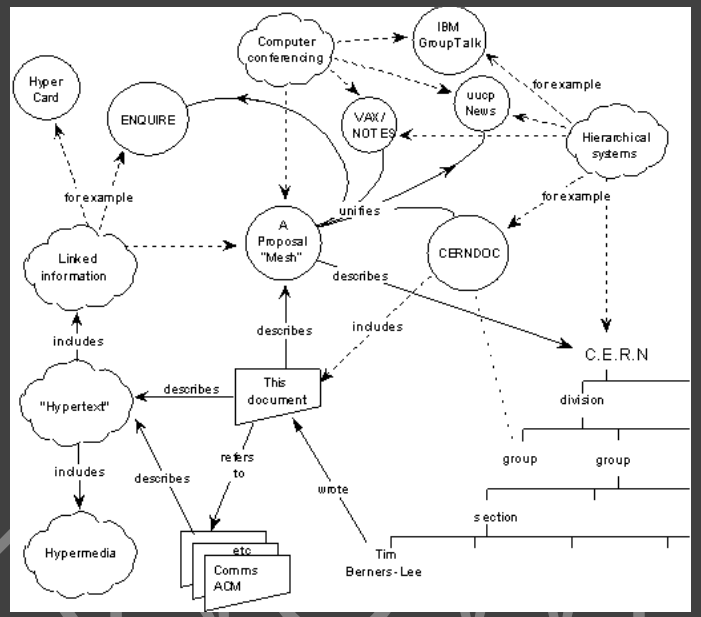
✓ ***Knowledge Graphs are growing in importance and adoption***

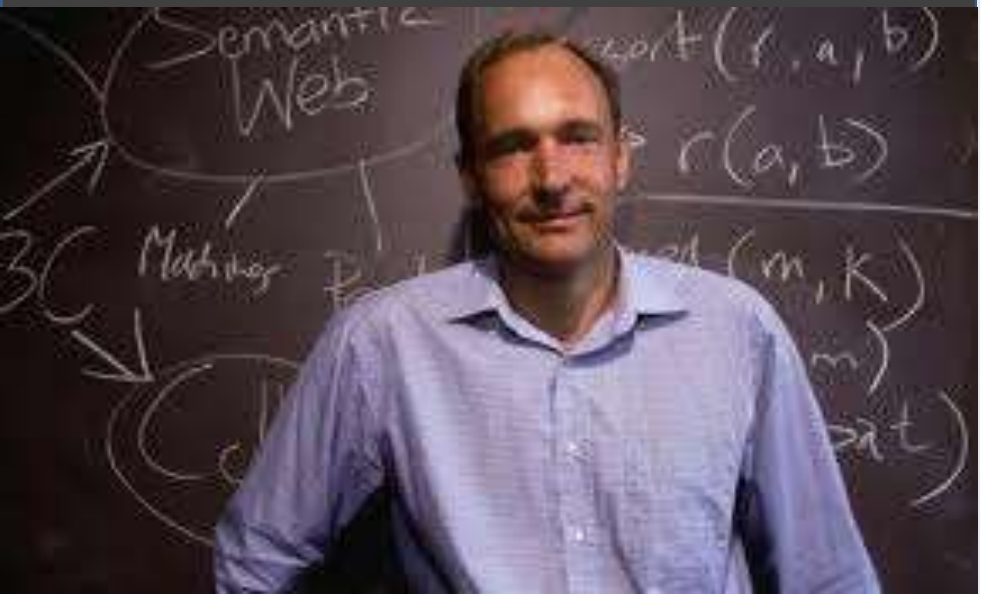
What are some key challenges and best practices to using them?

Irene Polikoff

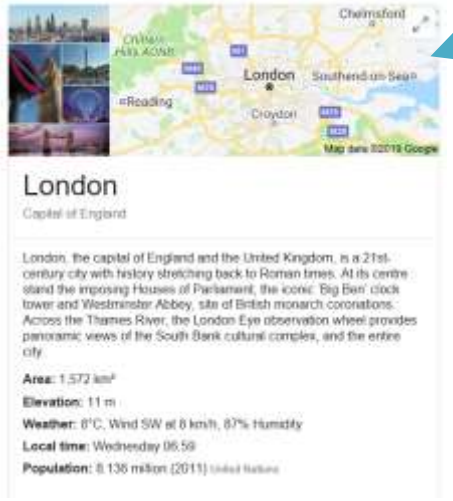
1. Define URIs – URI design
2. Create, define, use ontology models
3. Ensure that developers appreciate / like to use RDF
4. Capture statements about statements
5. Take advantage of effective ways of doing reasoning or inferencing
6. Use task appropriate visualizations

2001
Tim Berners-Lee





2012
Google



2019-2020

Knowledge Graphs in the news

> Google on AI and knowledge graphs in ZDNET

> Data governance 2.0 on Dataversity

> Conferences & workshops

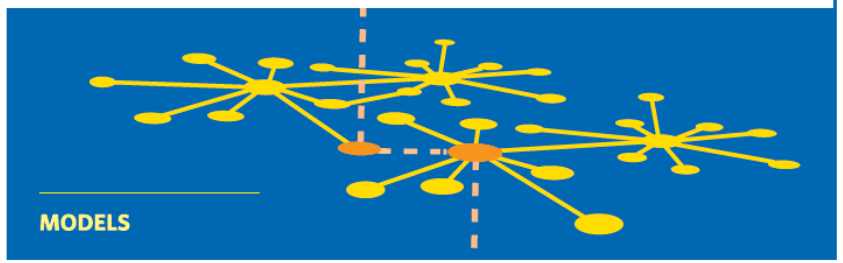
> Technology trends 2019

> Forbes article by Kurt Cagle

Enter Knowledge Graphs

- A Knowledge Graph represents a knowledge domain
- It represents knowledge as a graph
  - A network of nodes and links
  - Not tables of rows and columns
- It represents facts (data) and models (metadata) in the same way
  - Rich rules and inferencing
- It is based on open standards, from top to bottom
  - Readily connects to knowledge in private and public clouds

RULES: If both of a person's parents have blue eyes, they will also have blue eyes



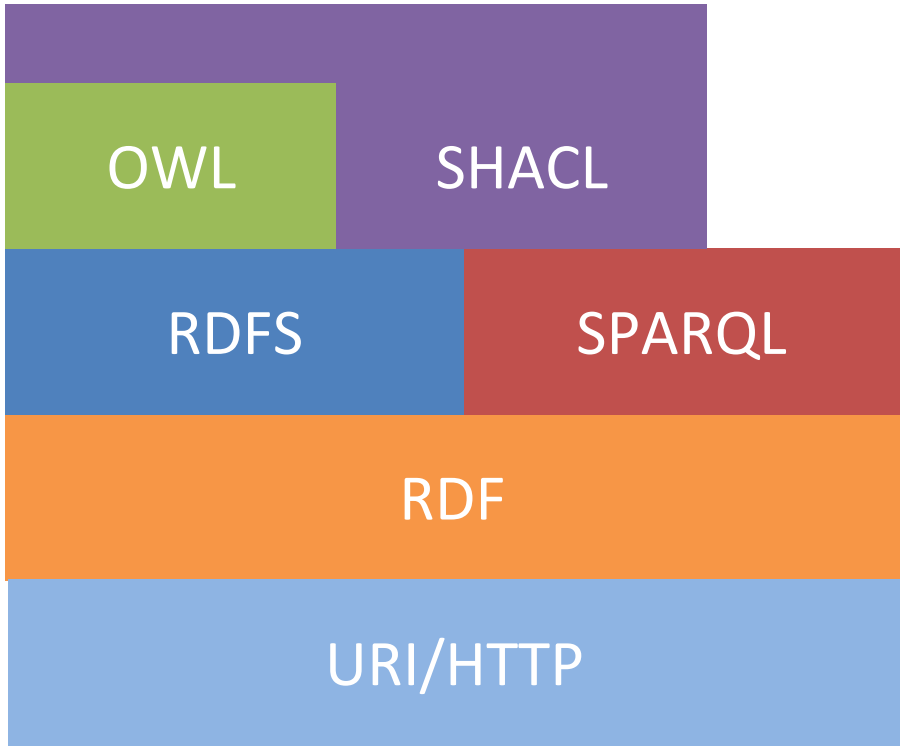
MODELS : A person has eye color. A person has two parents.
A person's father is also a person and he is male.

FACTS: James has blue eyes. James' father is Andrew. James is a person.

There can be different types and instances of Knowledge Graphs …

Standards-Based

Domain Specific Models and Rules aka Ontologies



Query

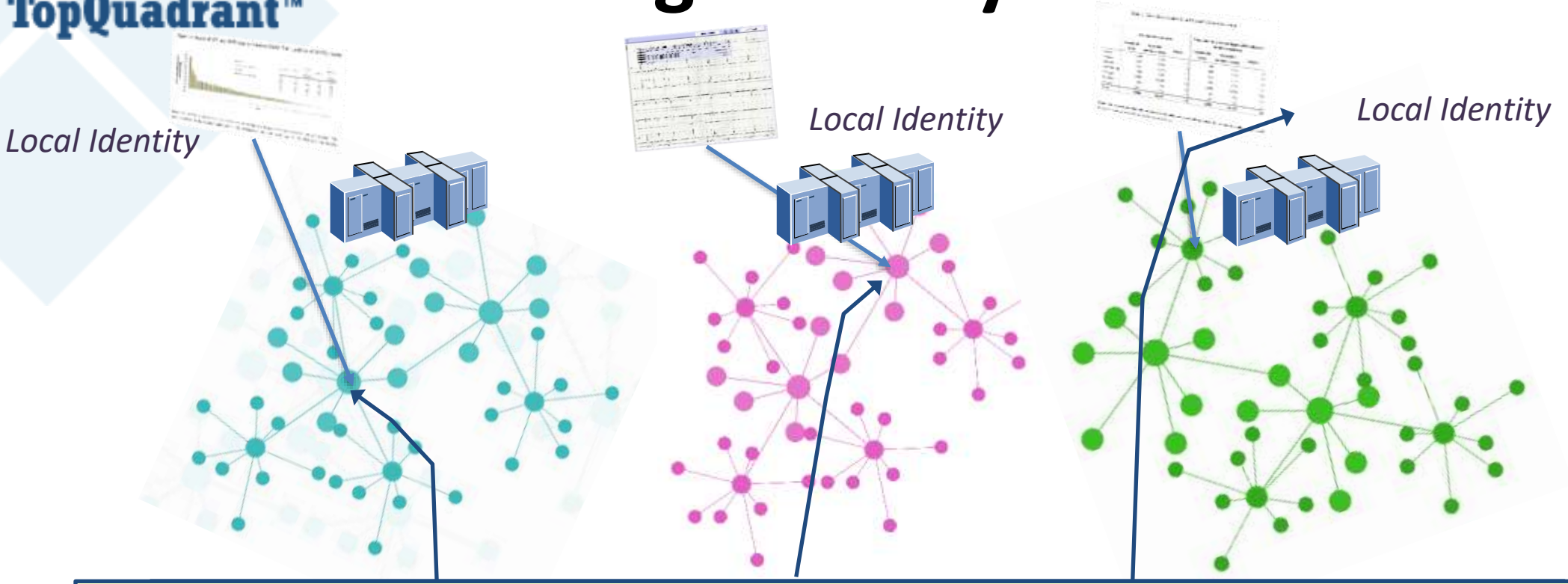
Graph Data Model

Identity

Reflecting on the Past

- Some RDF-based standards and technologies have been available for over 20 years now
  - RDF, RDFS, SPARQL
- During this period, the standards evolved and matured
  - as did best practices
  - as did our understanding of what works and what does not
- This presentation is informed by working with many industries and customers over the last 20 years
  - as they learned about ways to model with **fitness-for-purpose** and applied knowledge graph technologies in **hybrid technology stacks**

Challenge 1: Why URIs are Valuable



"Identity Matters"

For information assets as well as classes, attributes, relationships and rules that describe them

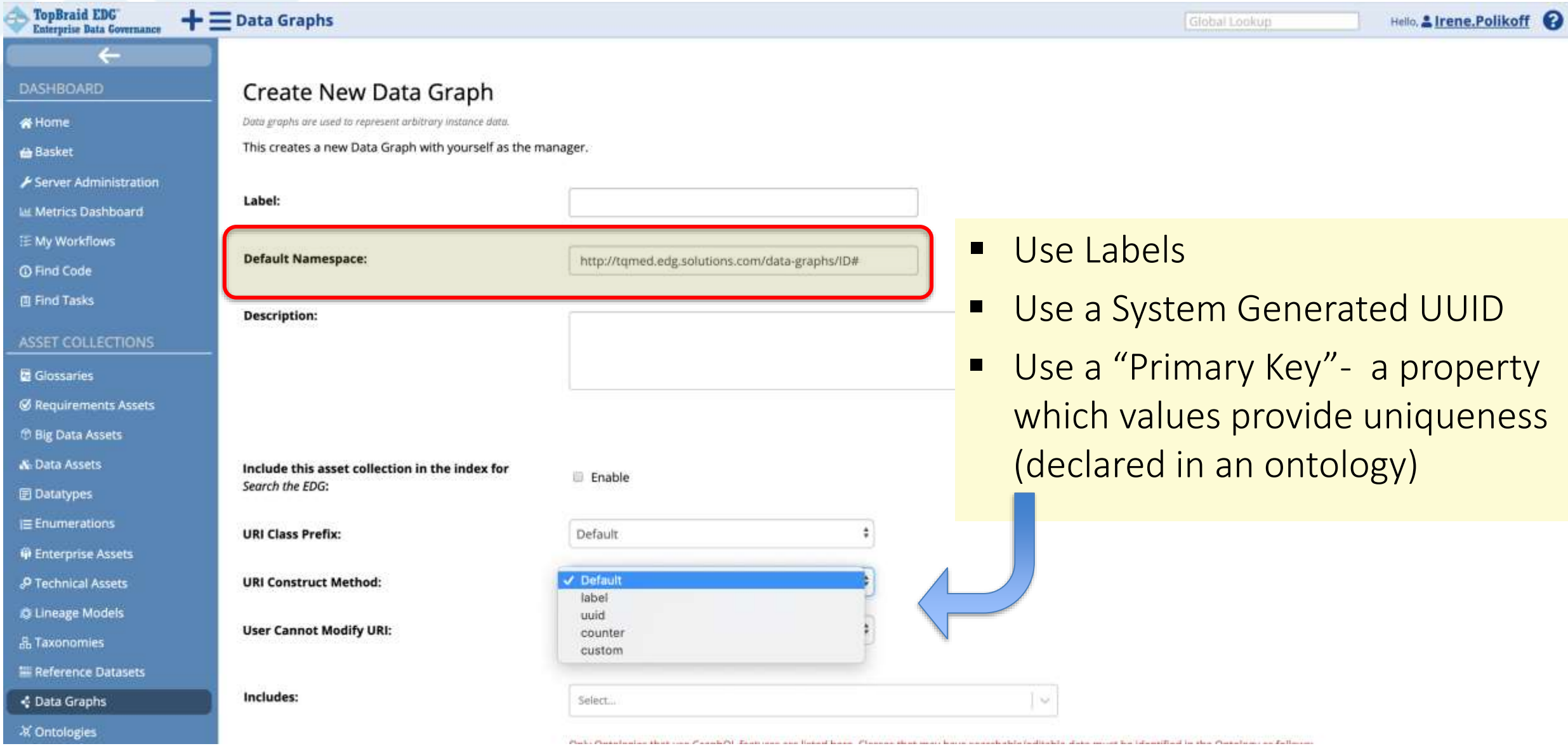
RDF – the "escape route" for Assets locked in silo repositories

Assets in the enterprise intranet

Challenge 1: Defining URIs

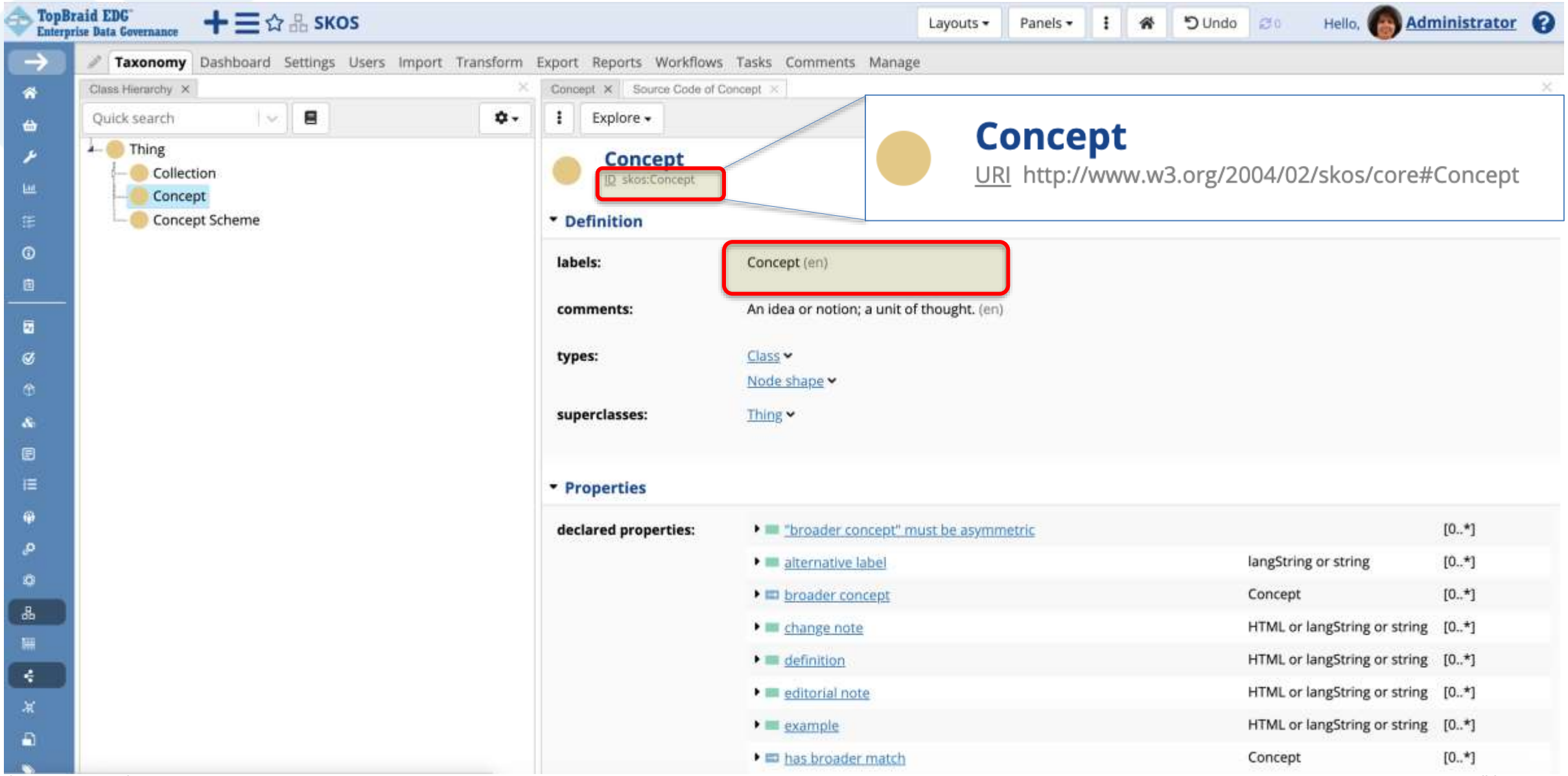
- URI Practices Converged On:
  - Use Labels
    - Most suitable for models – classes, properties
    - Can be suitable for controlled vocabularies
  - Use a System Generated UUID
    - Best for data in general
    - Numeric UUID are often required for integration
  - Use a “Primary Key” – a property with unique values
    - Best for reference data
    - Ensures integrity

Support for URI Strategies in EDG

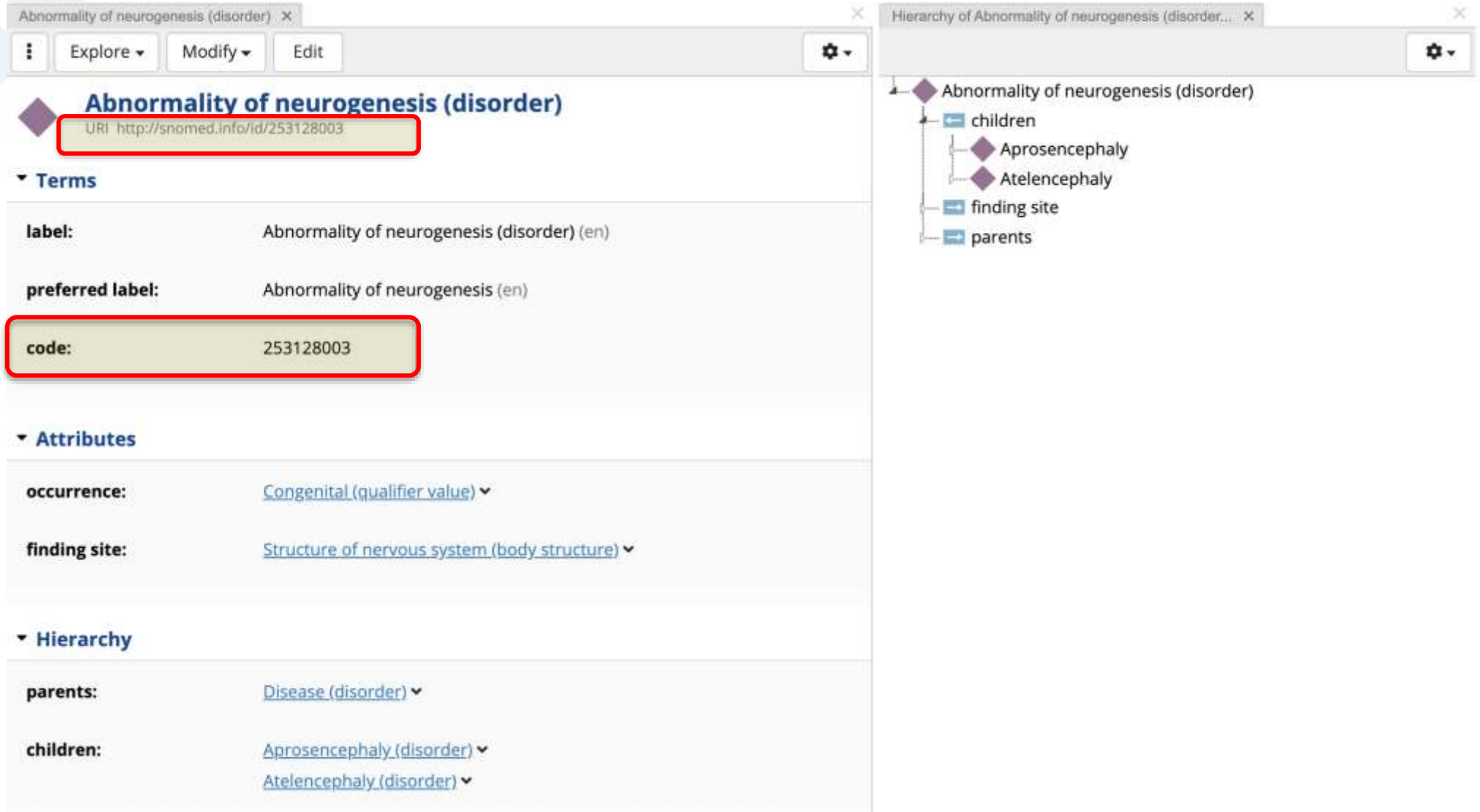


- Use Labels
- Use a System Generated UUID
- Use a “Primary Key”- a property which values provide uniqueness (declared in an ontology)

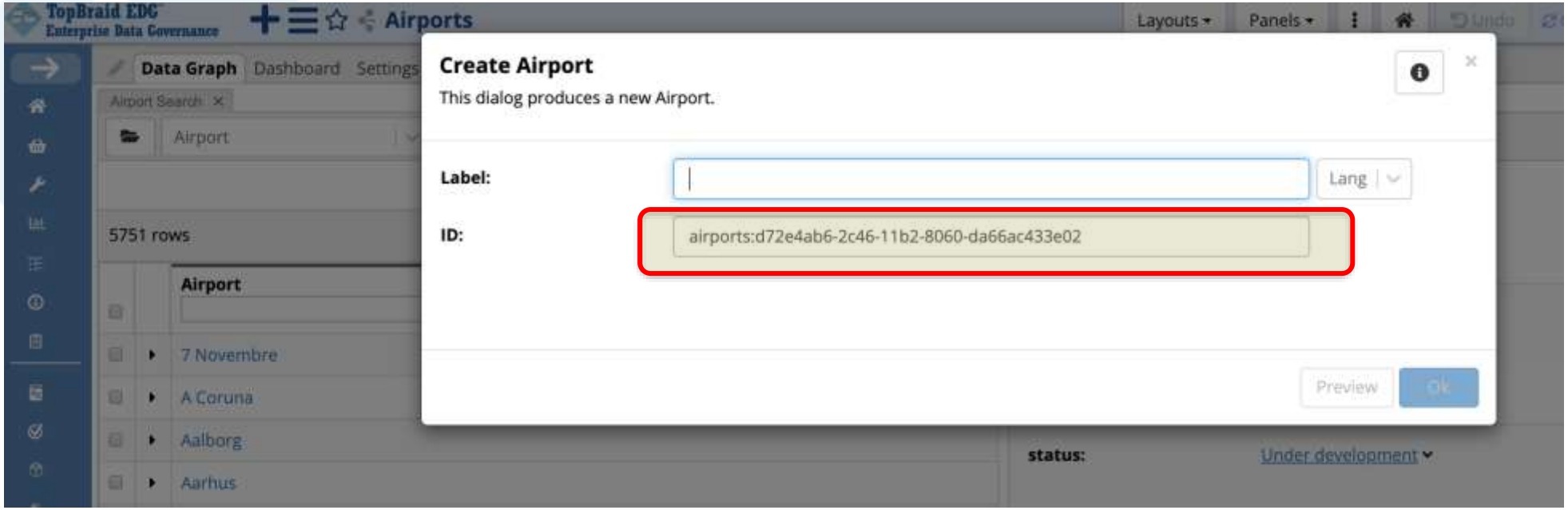
SKOS: Example of using labels to generate URIs

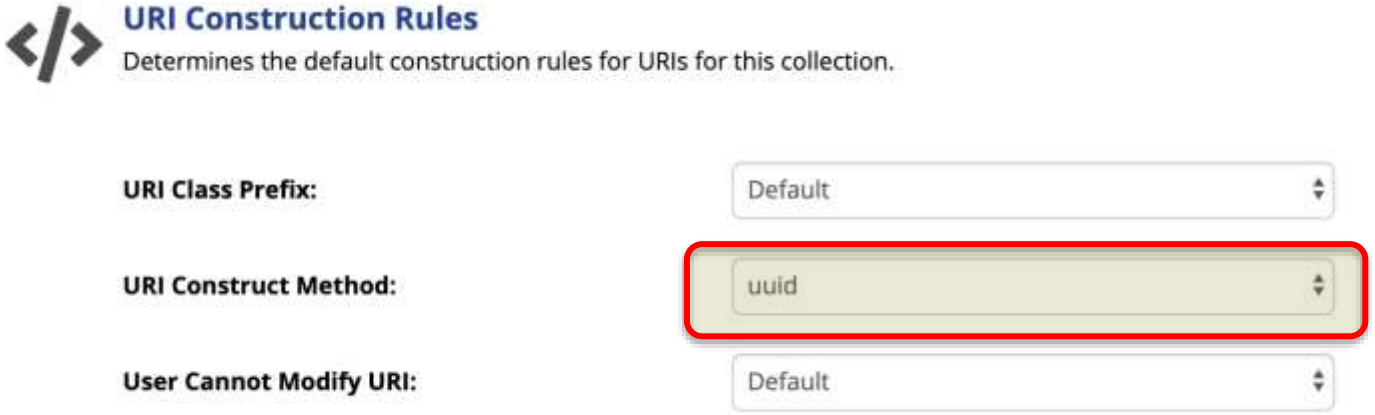


SNOMED: Example of using values of “code” property to generate URIs



URI Construction Rules and UUIDs



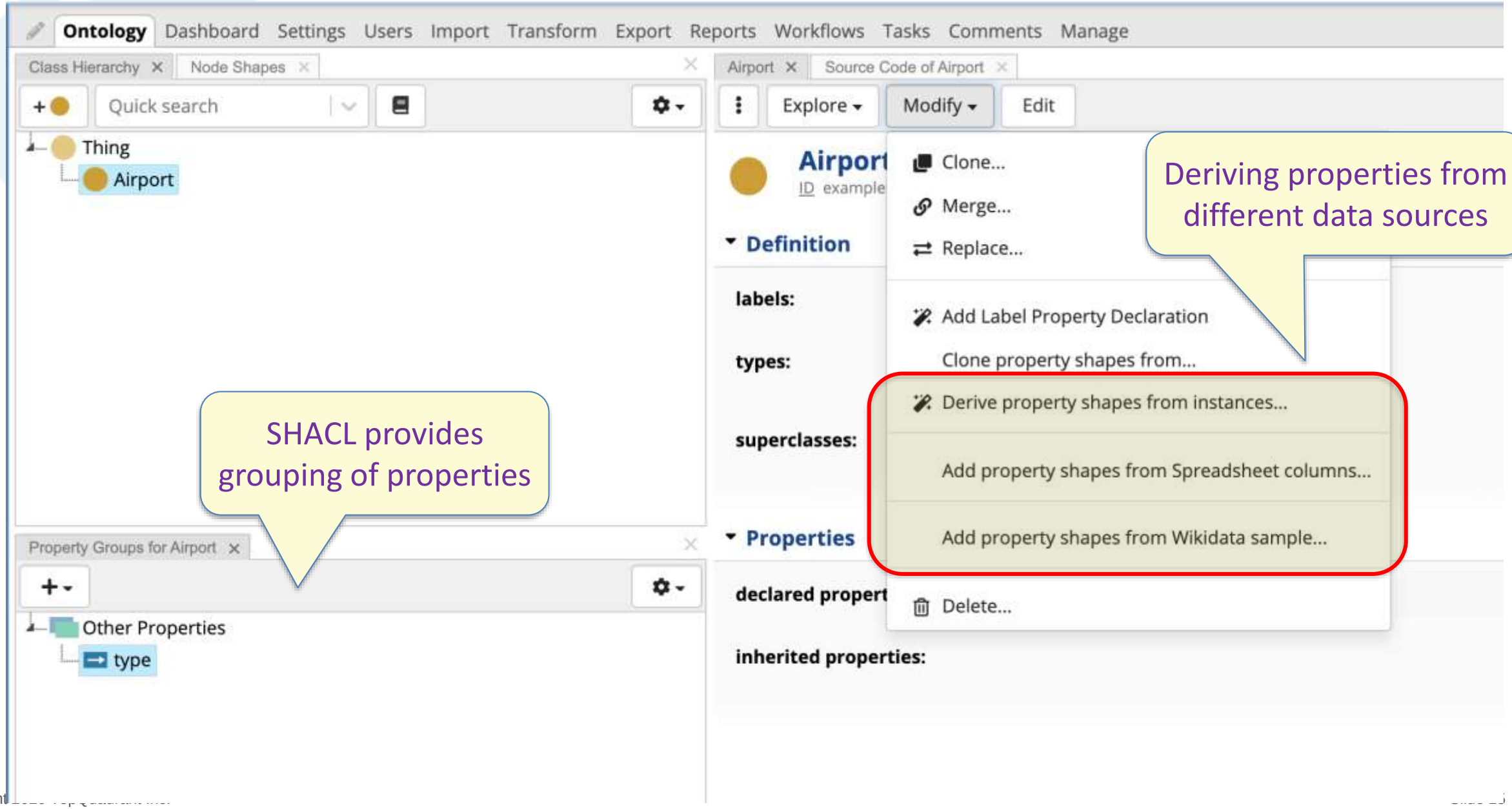


Challenge 2: Creating Ontology Models

- **Not Effective:**
  - Modeling in abstract, separate from data and how it will be used
  - Once created, users have serious issues with using these models

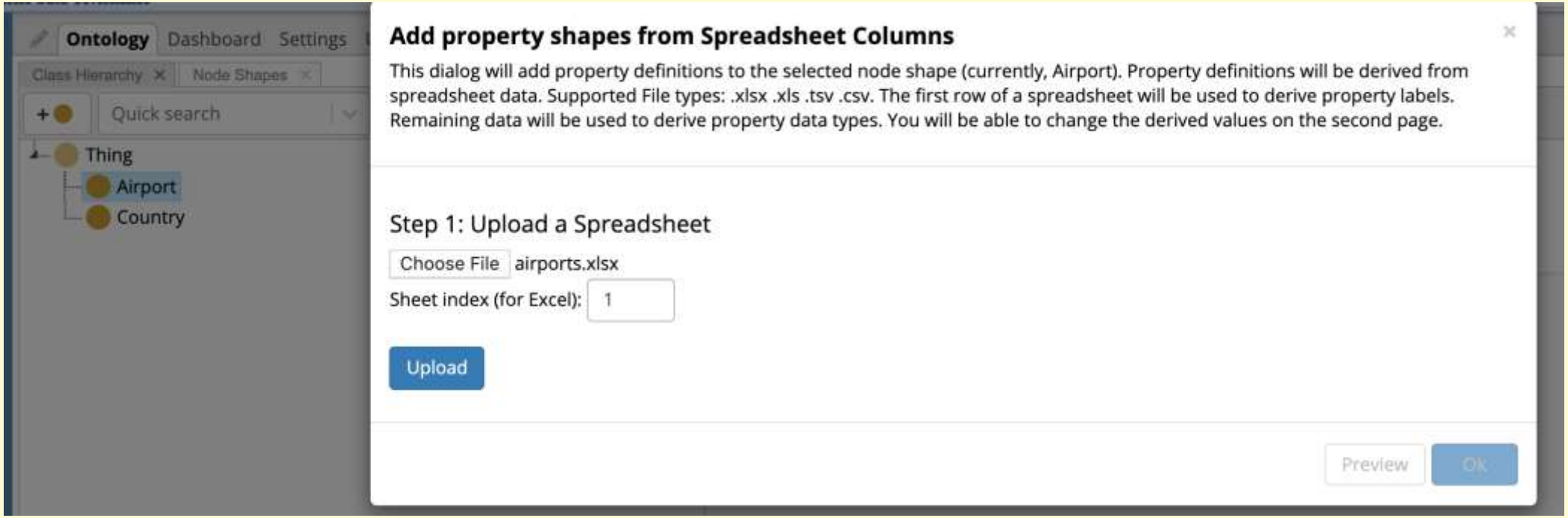
- **Best Practice**
  - Understand purposes of the ontology – they drive the design
  - Use your Data to test the salience and quality of the models
  - Harmonize, iterate and align as necessary
  - Develop approaches for reuse of models

Example of Lifting a Model from Data



TopBraid EDG Derives Schema from Spreadsheets - 1

1. Select a spreadsheet

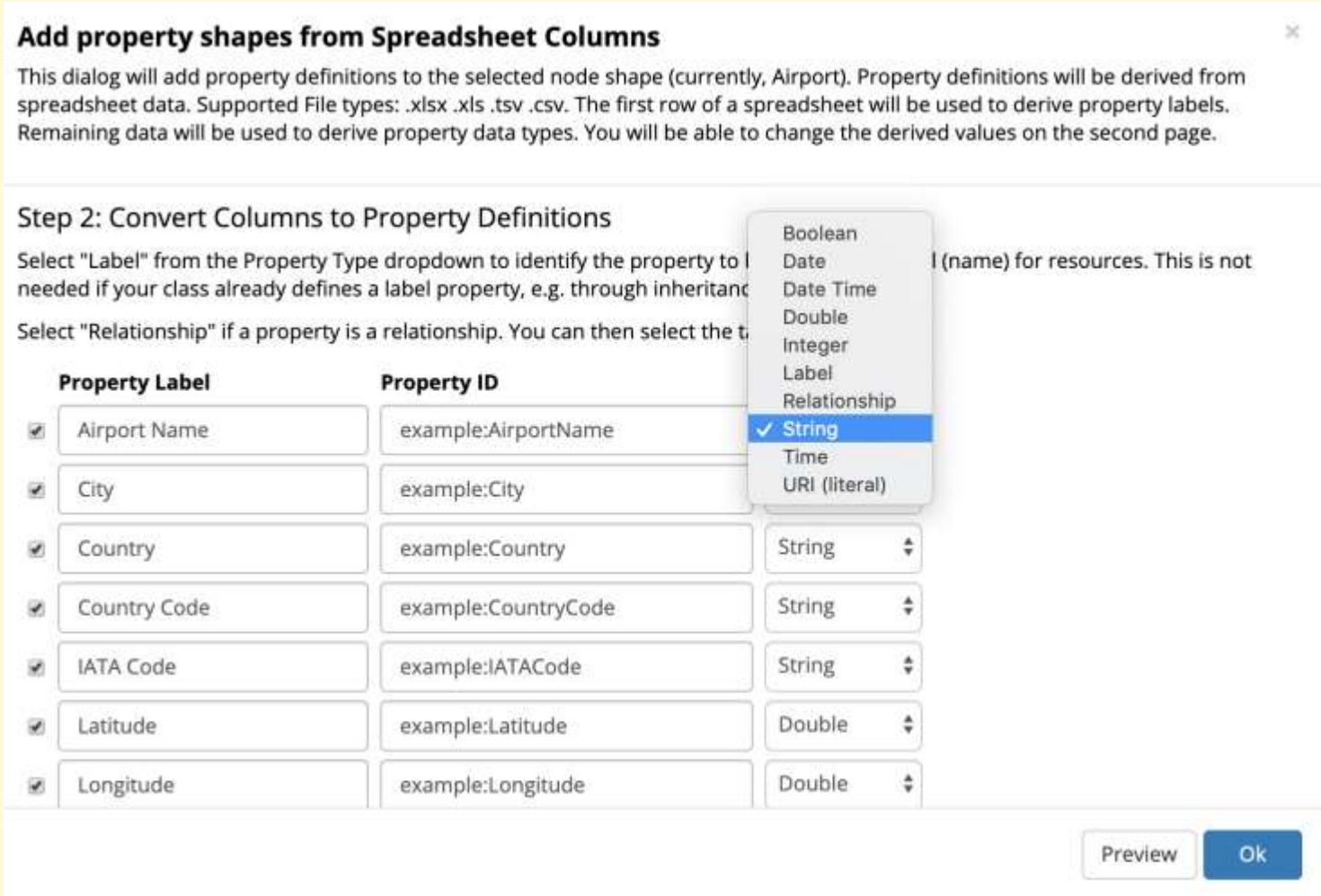


2. Select, modify, map as needed
3. Airport's properties are added

TopBraid EDG Derives Schema from Spreadsheets - 2

2

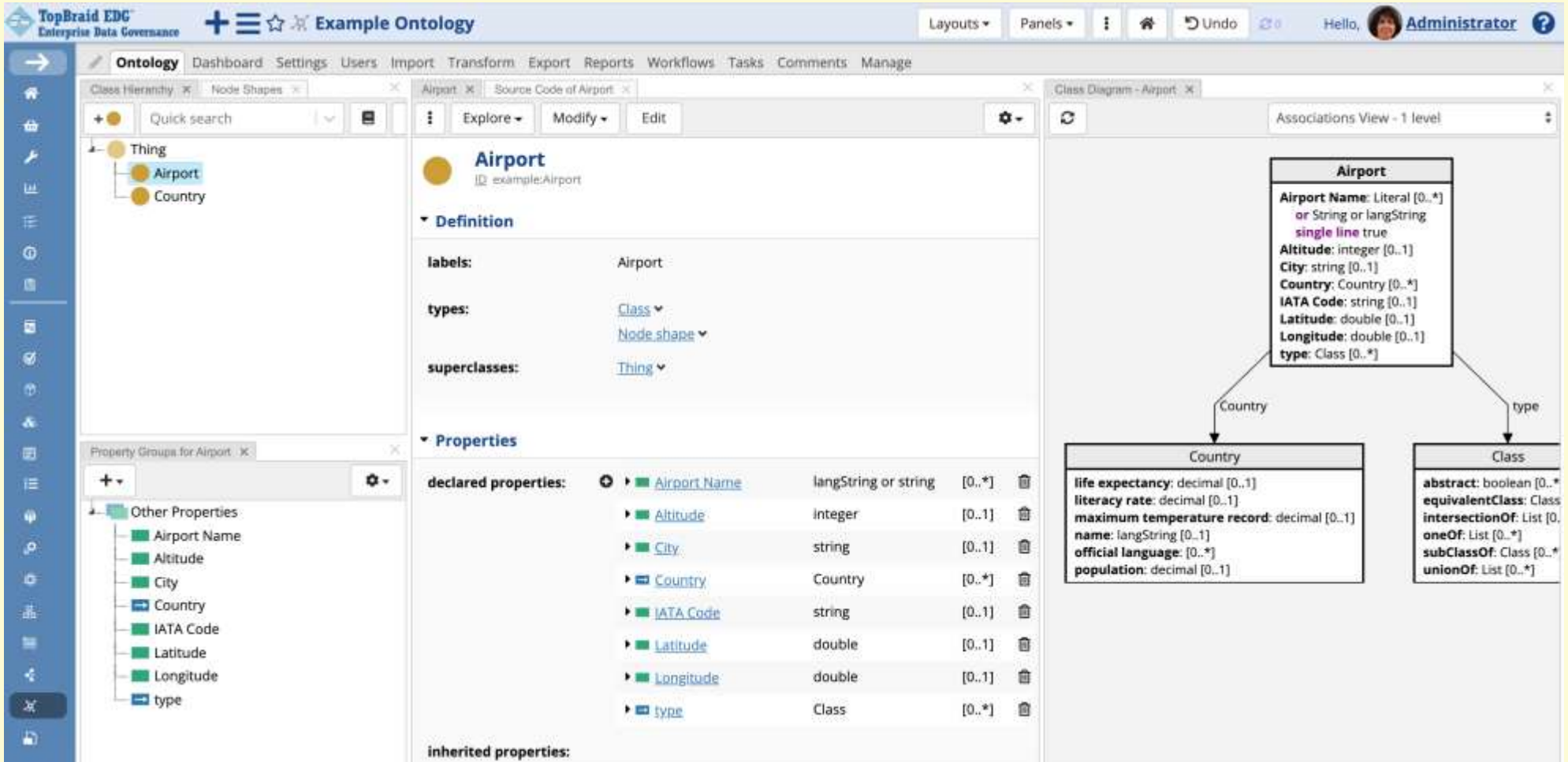
1. Select a spreadsheet
2. Select, modify, map as needed



TopBraid EDG Derives Schema from Spreadsheets - 3

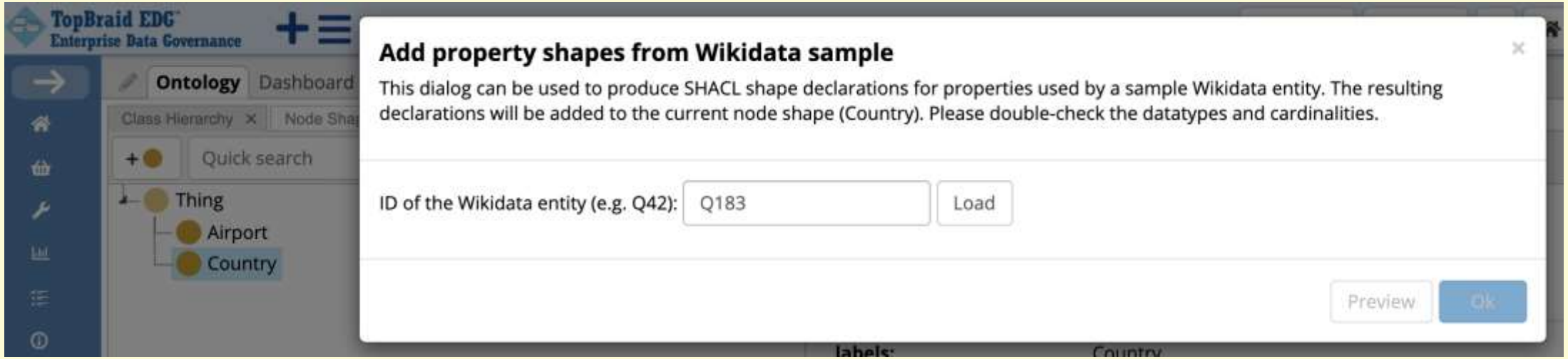
3

1. Select a spreadsheet
2. Select, modify, map as needed
3. Airport's properties are added



TopBraid EDG Derives Schema from External Knowledge Graphs - 1

1. Map a type (e.g., Country class) to an external Knowledge Graph (e.g., Wikidata)

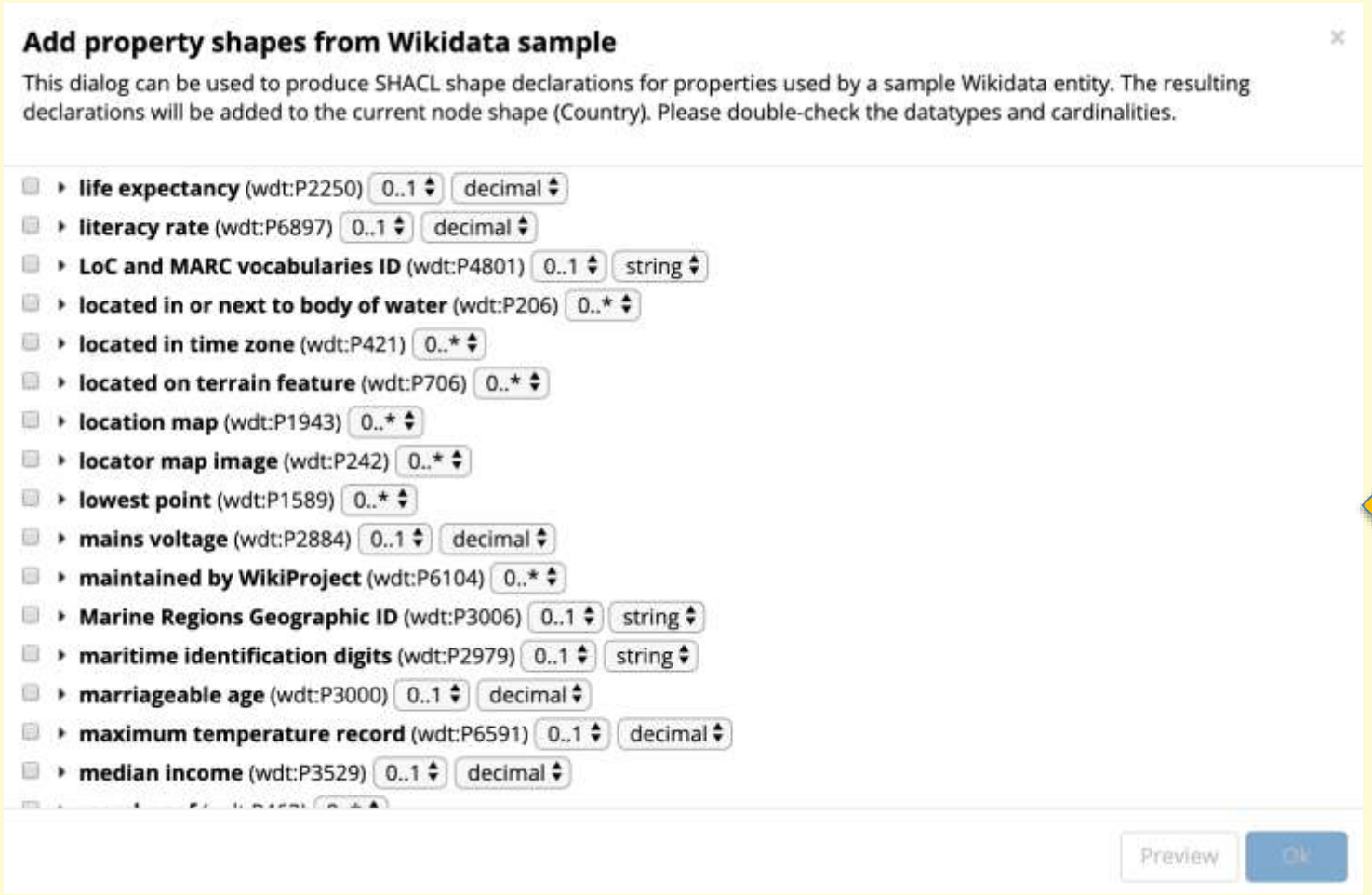


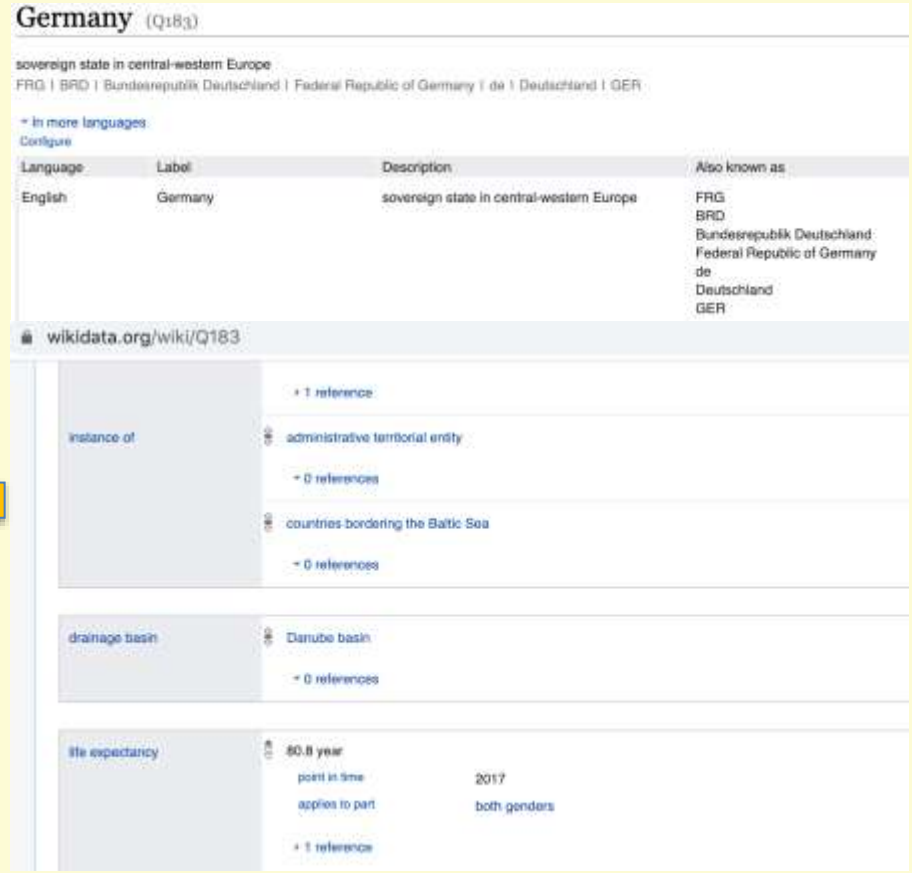
2. Select properties to be derived from the external graph (e.g., population)
3. Country's properties are added

TopBraid EDG Derives Schema from External Knowledge Graphs - 2

2

1. Map a type (e.g., Country class) to an external Knowledge Graph (e.g., Wikidata)
2. Select propertiesto be derived from the external graph (e.g., image, population, etc)

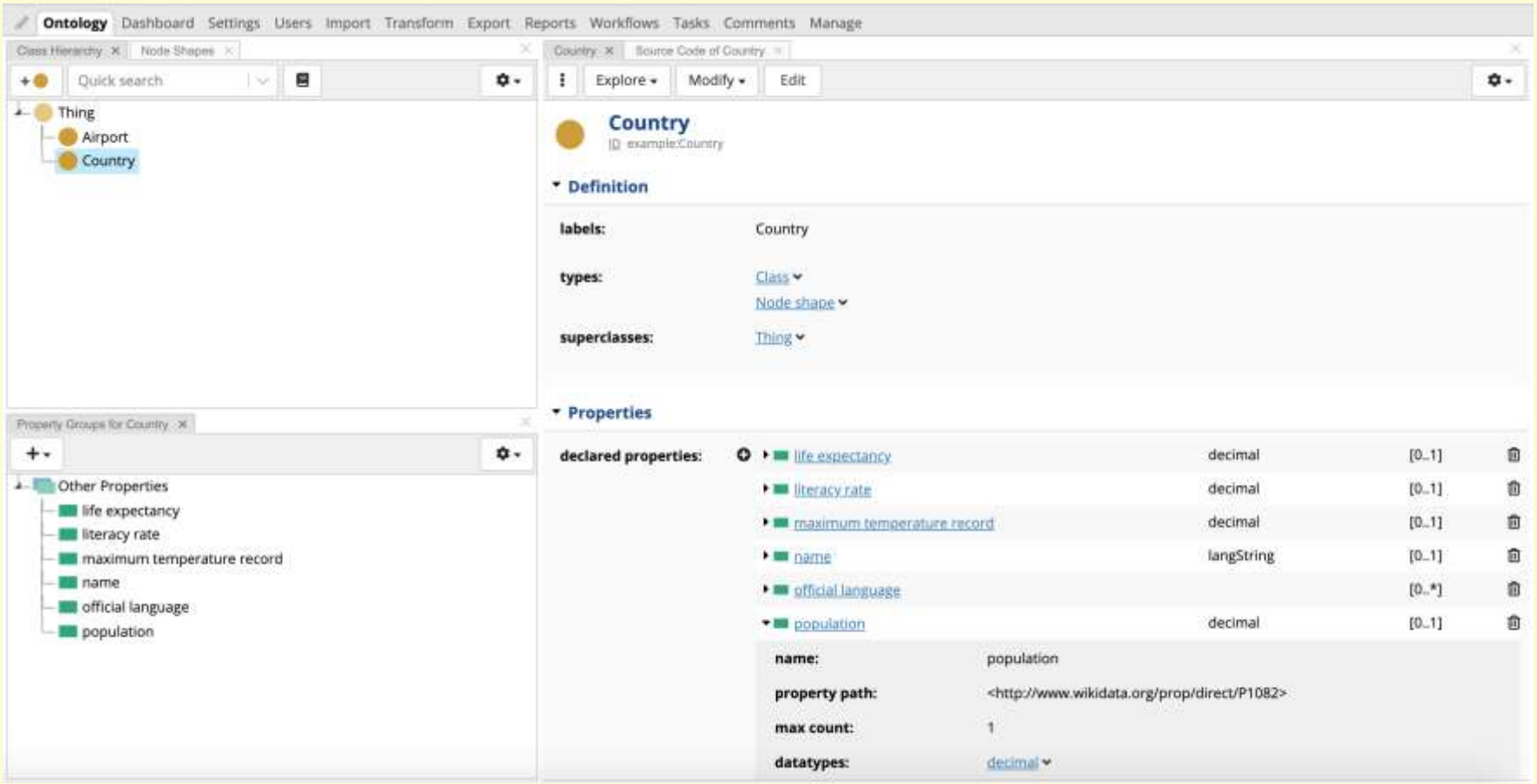




TopBraid EDG Derive Schema from External Knowledge Graphs - 3

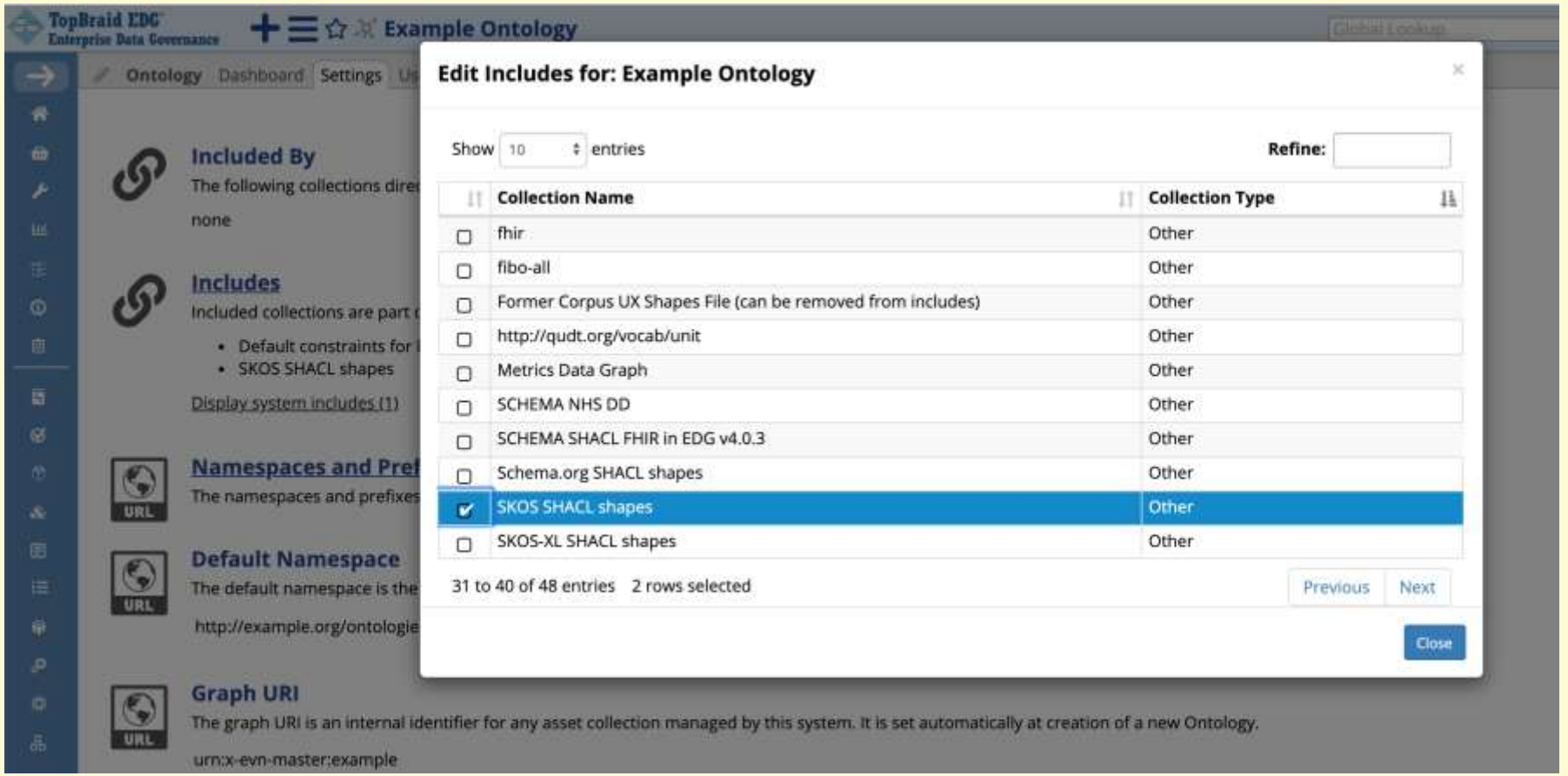
3

1. Map a type (e.g., Country class) to an external Knowledge Graph (e.g., Wikidata)
2. Select properties to be derived from the external graph (e.g., life expectancy, population, etc)
3. Country's properties are added



Model Re-Use - 1

1. Include an external ontology

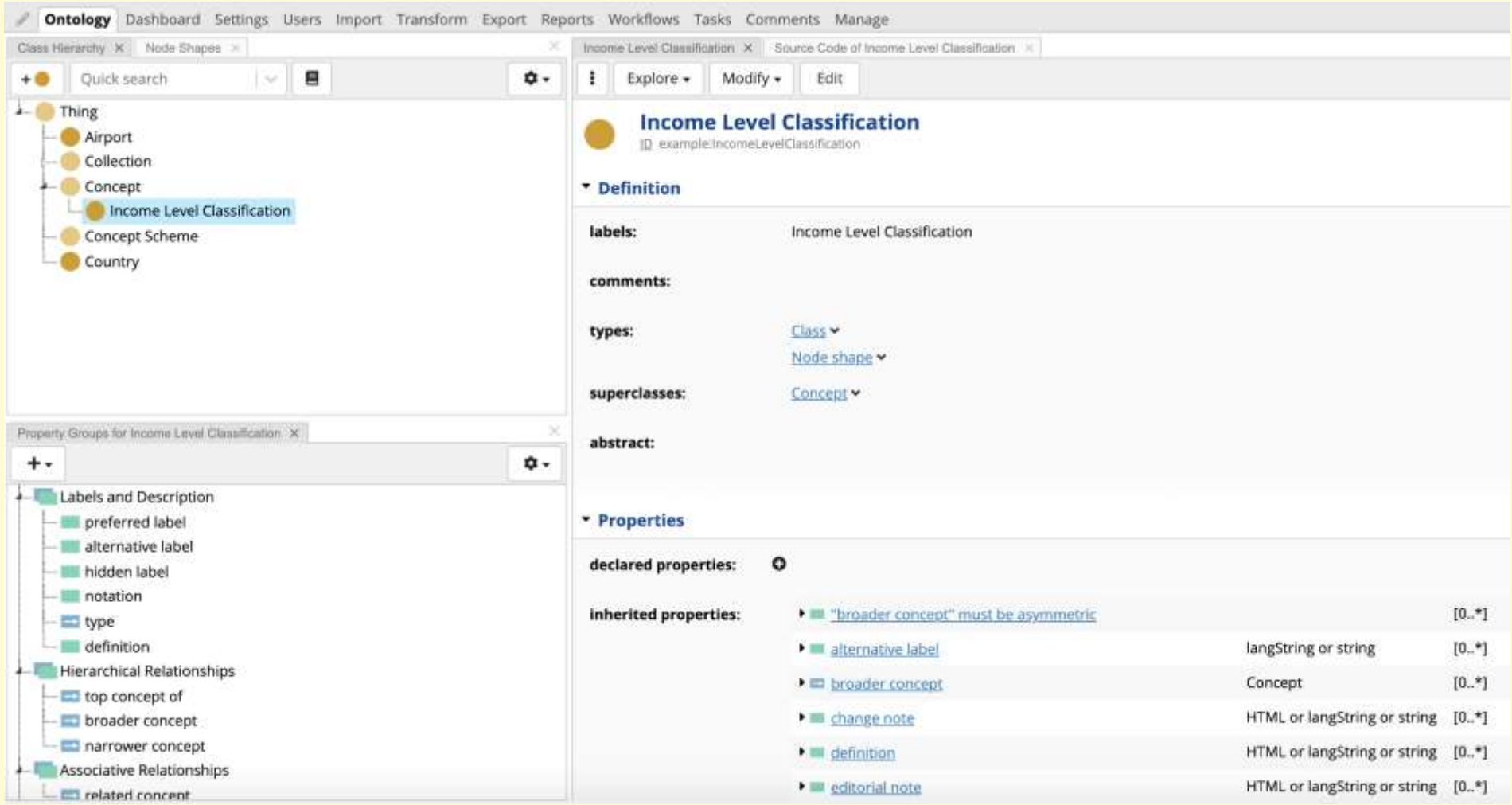


2. Integrate it e.g., add your own subclasses, use classes in the ontology as targets of relationships, etc.
3. For classes that you will use, deactivate properties you will not use

Model Re-Use - 2

2

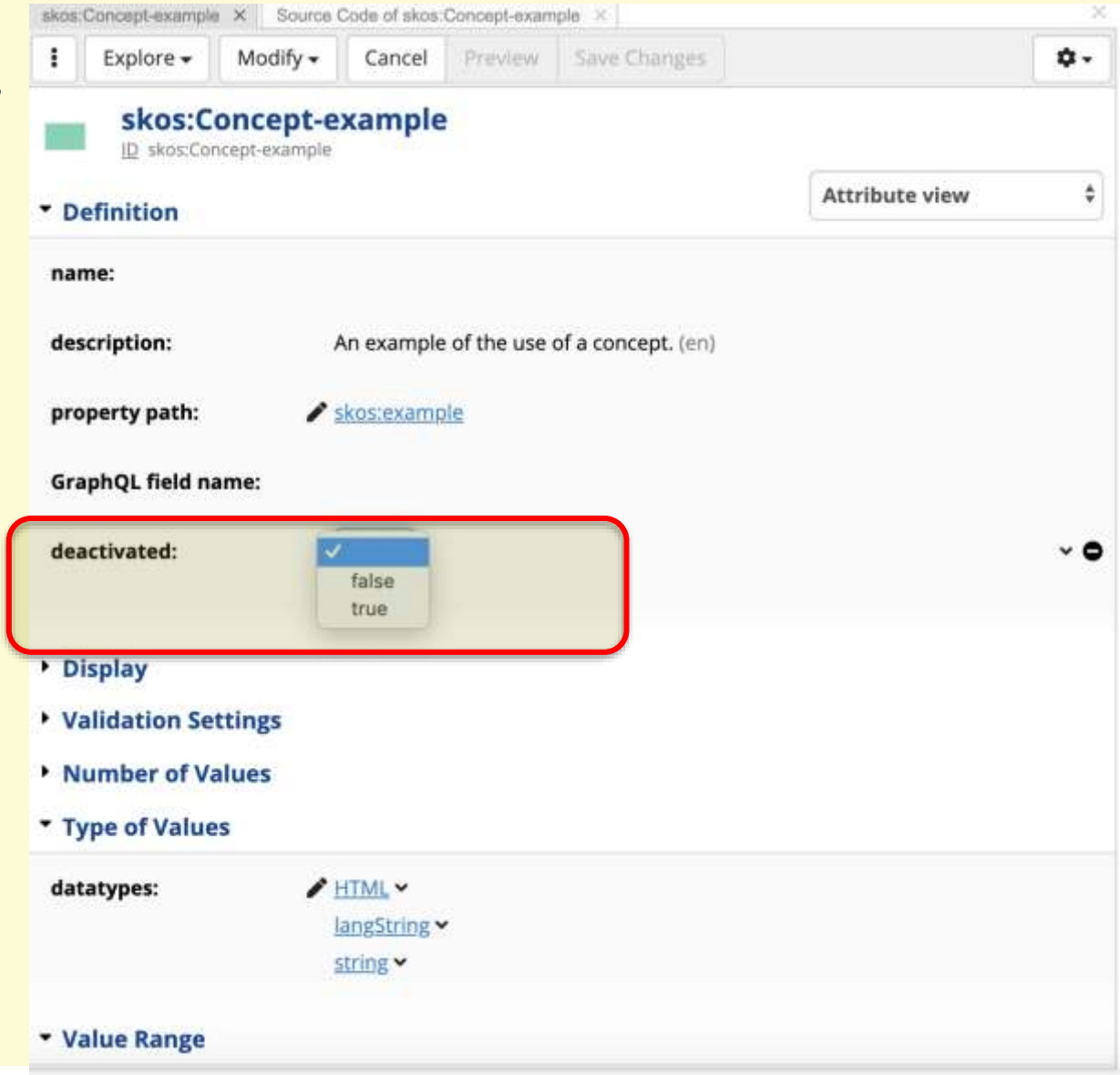
1. Include an external ontology
2. Integrate it e.g., add your own subclasses, use classes in the ontology as targets of relationships, etc.



Model Re-Use - 3

3

1. Include an external ontology
2. Integrate it e.g., add your own subclasses, use classes in the ontology as targets of relationships, etc.
3. For classes that you will use, deactivate properties you will not use

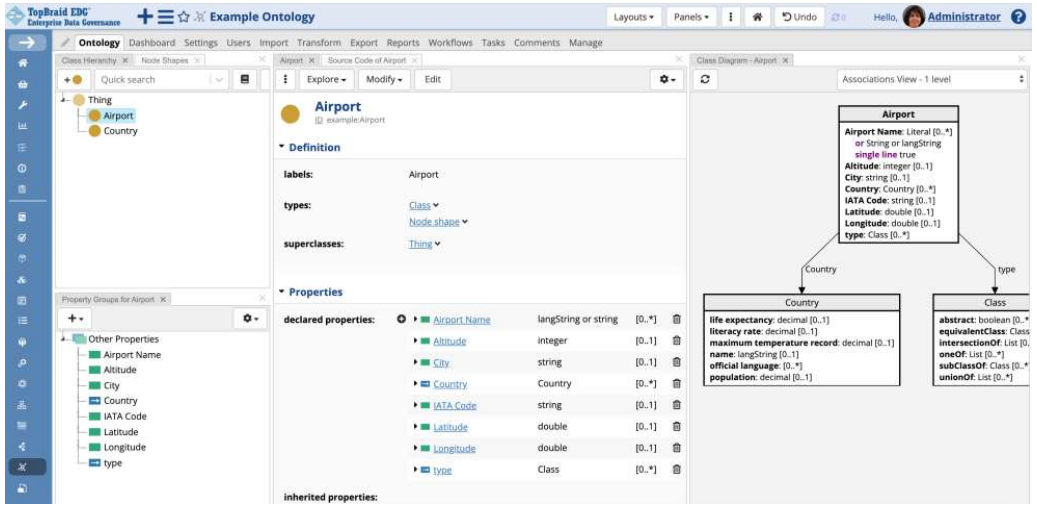


Challenge 3: Ensuring that Developers Like to Work with RDF

Use of introspection to query the semantic models

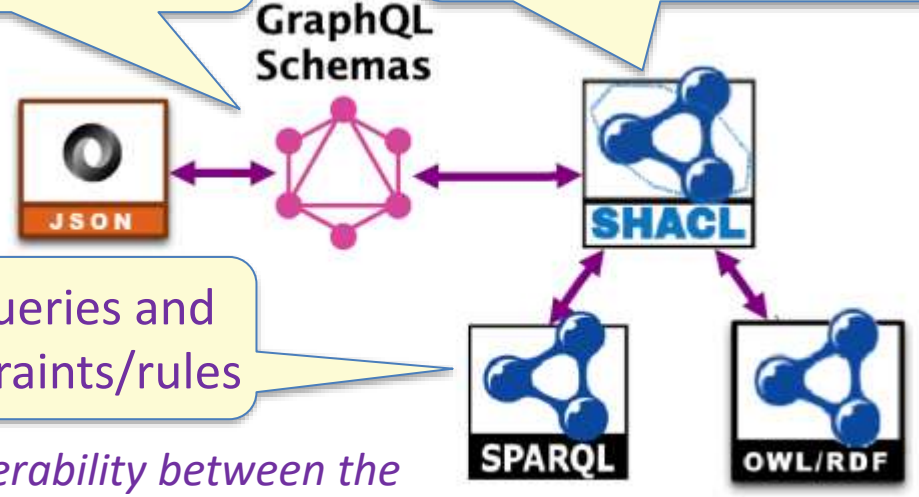
Translates GraphQL to and from SHACL and RDF

User Interface ↔ ReactJS Components ↔ Servlets ↔ GraphQL Engine ↔ Knowledge Graphs



Bridging the Knowledge Graph and JSON worlds

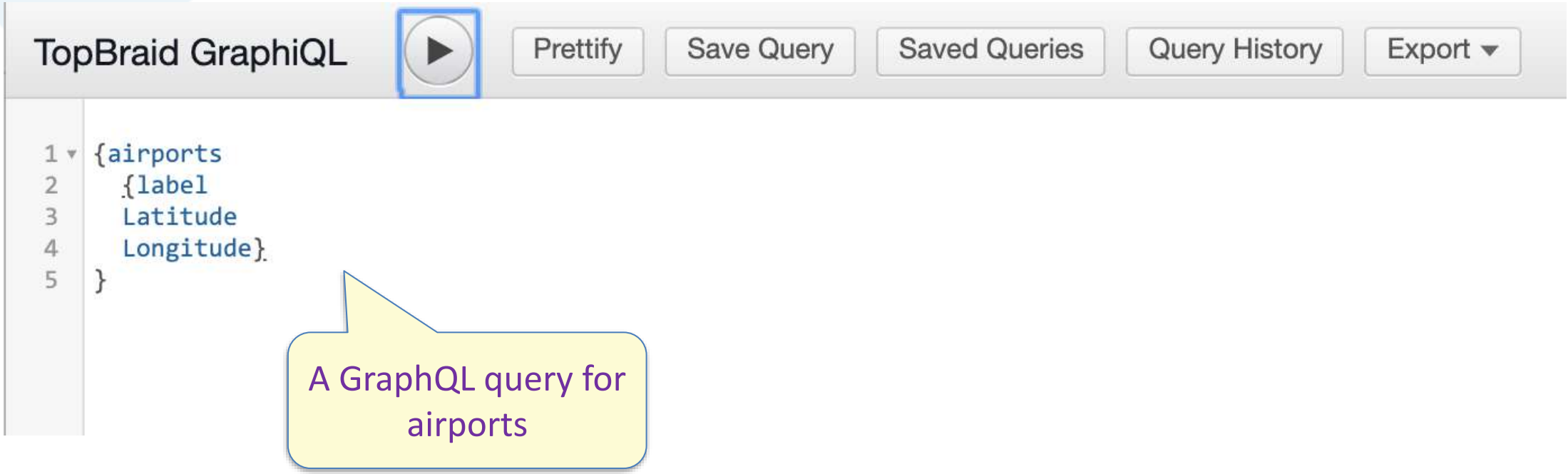
SHACL Schema and Data Shapes Models



SPARQL for queries and complex constraints/rules

GraphQL provides an interface technology for interoperability between the Javascript world of User Interfaces and SHACL-based Data Models and Rules

Example of Querying Knowledge Graph Data with GraphQL



For more see video at: https://www.topquadrant.com/project/graphql_json_rdf/

Example of Querying Knowledge Graph Data

TopBraid GraphiQL ▶ Prettify | Save Query | Saved Queries | Query History | Export ▾ | Docs

```
{airports
  {label
  Latitude
  Longitude}
}
```

JSON Results


```
{
  "data": {
    "airports": [
      {
        "label": "Yenbo",
        "Latitude": 24.144244,
        "Longitude": 38.06335
      },
      {
        "label": "Namorik Atoll Airport",
        "Latitude": 5.63167,
        "Longitude": 168.125
      },
      {
        "label": "Bantry Aerodrome",
        "Latitude": 51.668598,
        "Longitude": -9.48417
      },
      {
        "label": "Labe",
        "Latitude": 11.326058,
        "Longitude": -12.28685
      },
      {
        "label": "Chennai Intl",
        "Latitude": 12.994414,
        "Longitude": 80.180517
      },
      {
        "label": "John F Kennedy Intl",
        "Latitude": 40.639751,
        "Longitude": -73.778925
      },
      {
        "label": "Guemar Airport",
        "Latitude": 33.5114,
        "Longitude": 6.77679
      },
      {
        "label": "Chimbu Airport",
        "Latitude": -6.02429,
        "Longitude": 144.971
      },
```


For more see video at: https://www.topquadrant.com/project/graphql_json_rdf/

Example of Querying Knowledge Graph Data with GraphQL

TopQuadrant™

TopBraid GraphiQL ▶ Prettify | Save Query | Saved Queries | Query History | Export

```
{airports {
  label
  Latitude
  Longitude
}}
```

QUERY VARIABLES


```
{
  "data": {
    "airports": [
      {
        "label": "Yenbo",
        "Latitude": 24.144244,
        "Longitude": 38.06335
      },
      {
        "label": "Namorik Atoll Airport",
        "Latitude": 5.63167,
        "Longitude": 168.125
      },
      {
        "label": "Bantry Aerodrome",
        "Latitude": 51.668598,
        "Longitude": -9.48417
      },
      {
        "label": "Labe",
        "Latitude": 11.326058,
        "Longitude": -12.28685
      },
      {
        "label": "Chennai Intl",
        "Latitude": 12.994414,
        "Longitude": 80.180517
      },
      {
        "label": "John F Kennedy Intl",
        "Latitude": 40.639751,
        "Longitude": -73.778925
      },
      {
        "label": "Guemar Airport",
        "Latitude": 33.5114,
        "Longitude": 6.77679
      },
      {
        "label": "Chimbu Airport",
        "Latitude": -6.02429,
        "Longitude": 144.971
      },
      {
        "label": "Mosul International Airport",
        "Latitude": 36.305833,
        "Longitude": 43.1475
      },
```


The GraphQL schema automatically transformed from a SHACL model

Schema **RootRDFQuery**

Search RootRDFQuery...

No Description

FIELDS

```
airports(
  queryText: String
  filter: String
  uri: ID
  orderBy: Airport_FieldsEnum
  orderByExpr: String
  orderByDesc: Boolean
  orderAll: Boolean
  Altitude: Int
  City: String
  Country: ID
  IATACode: String
  Latitude: Float
  Longitude: Float
  rdf_type: ID
  where: Airport_where
  first: Int
  skip: Int
): [Airport]
```

graphReport(excludeIds: [String], resourceType: String): _GraphReport

The results report of the data graph, validating all resources.

_rootTypeShapes: [_TypeShape]

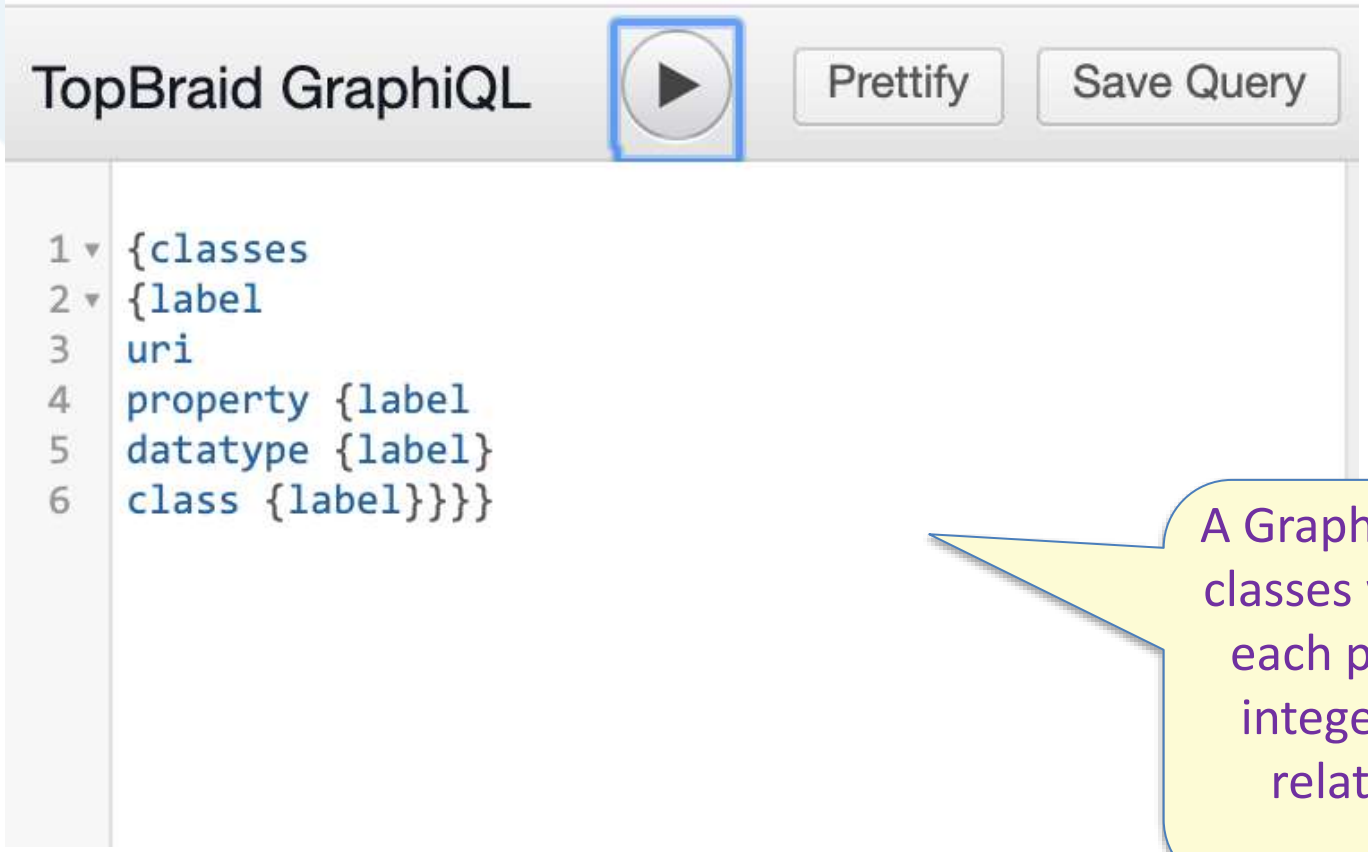
Requests the shape information for all root types.

_typeShapeByName(name: String!): _TypeShape

Requests the shape information of a type with a given name.

_typeShapeByURI(uri: String!): _TypeShape

Example of Querying Model with GraphQL

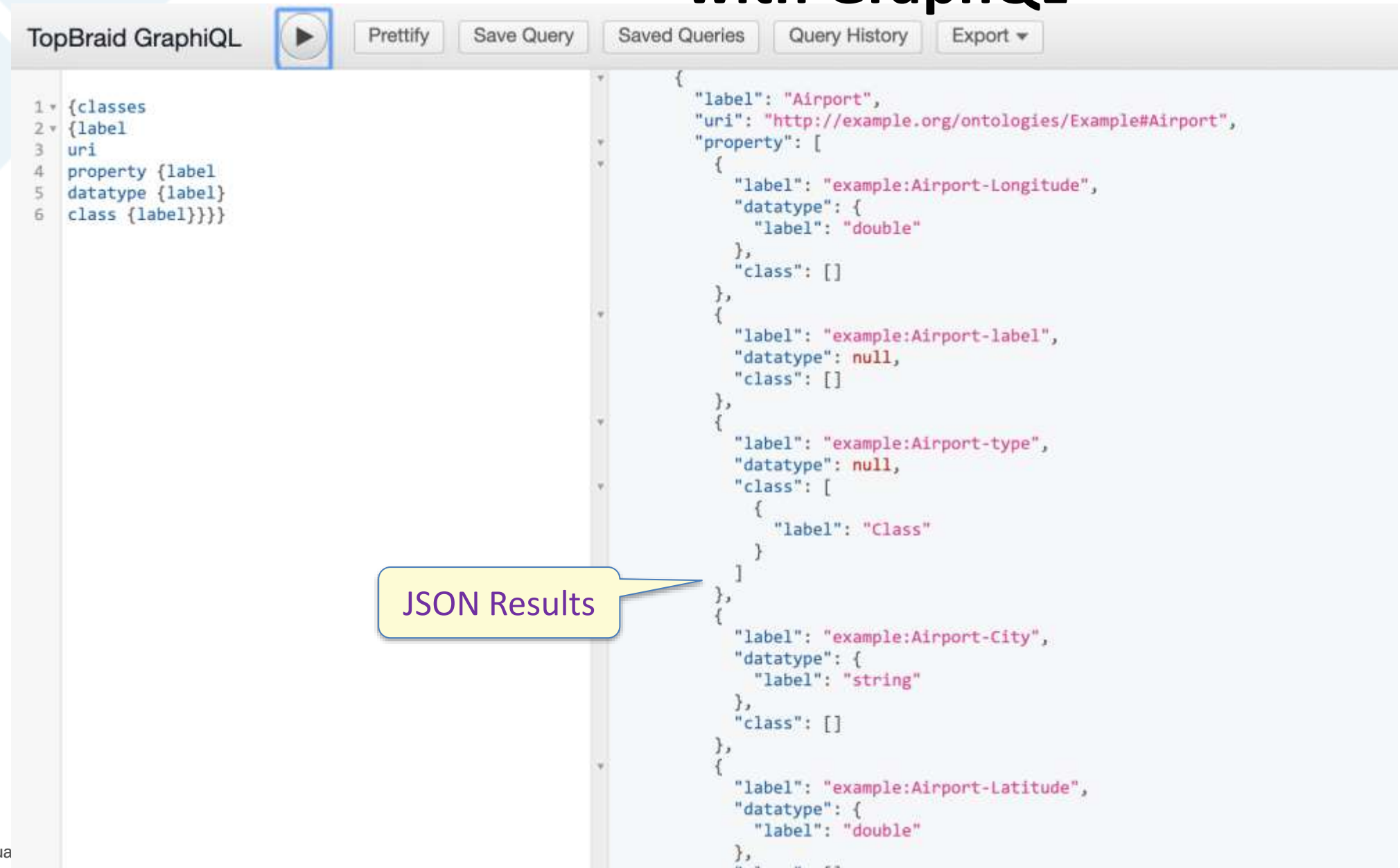


A GraphQL query over a SHACL model – all classes with their properties, returning for each property a datatype (e.g., string or integer) and class (this is applicable for relationships such as a link between airports and countries)

For more see video at: https://www.topquadrant.com/project/graphql_json_rdf/

Example of Querying SHACL/RDF with GraphQL

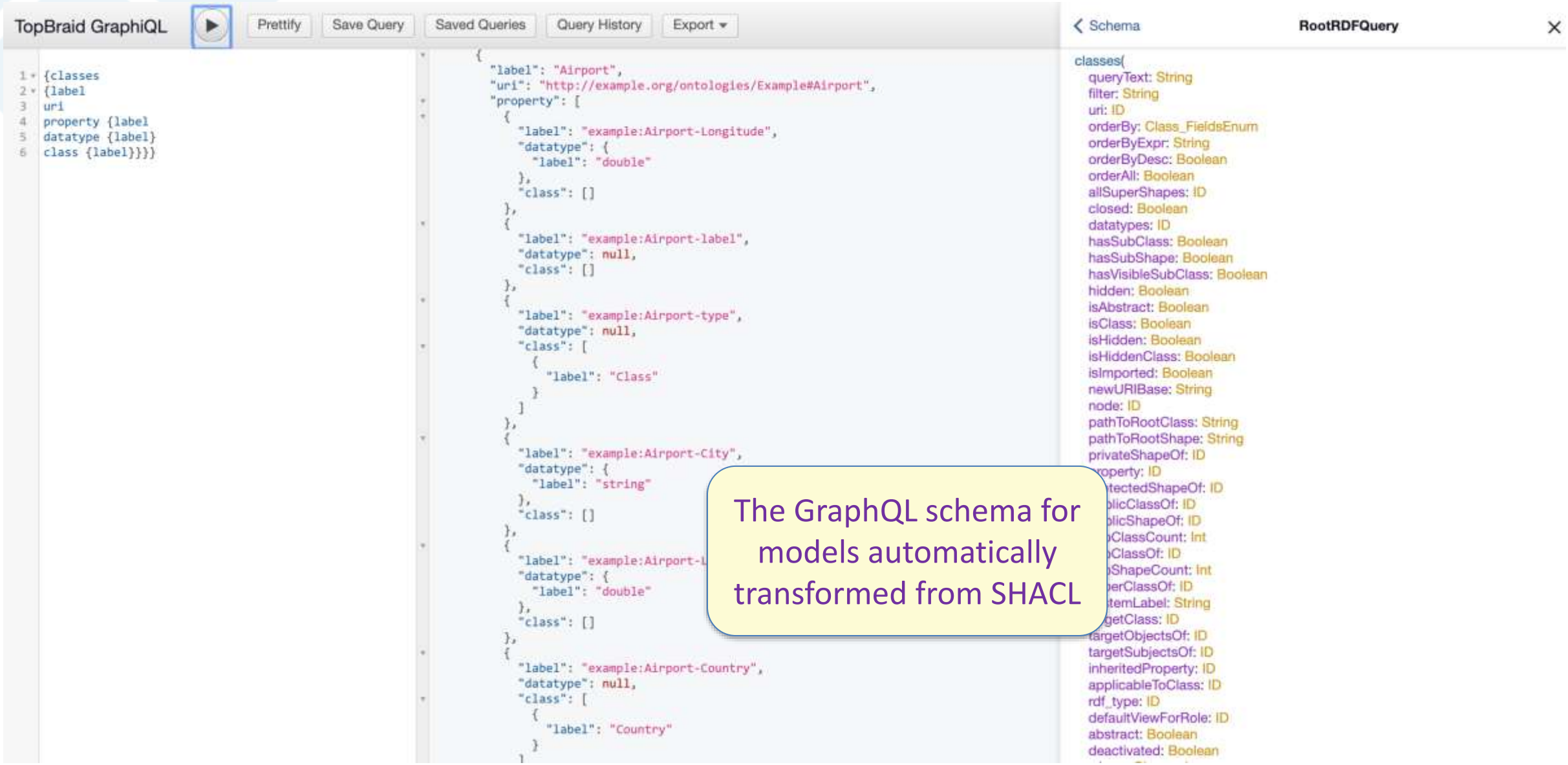
TopQuadrant™



JSON Results

Example of Querying SHACL/RDF with GraphQL

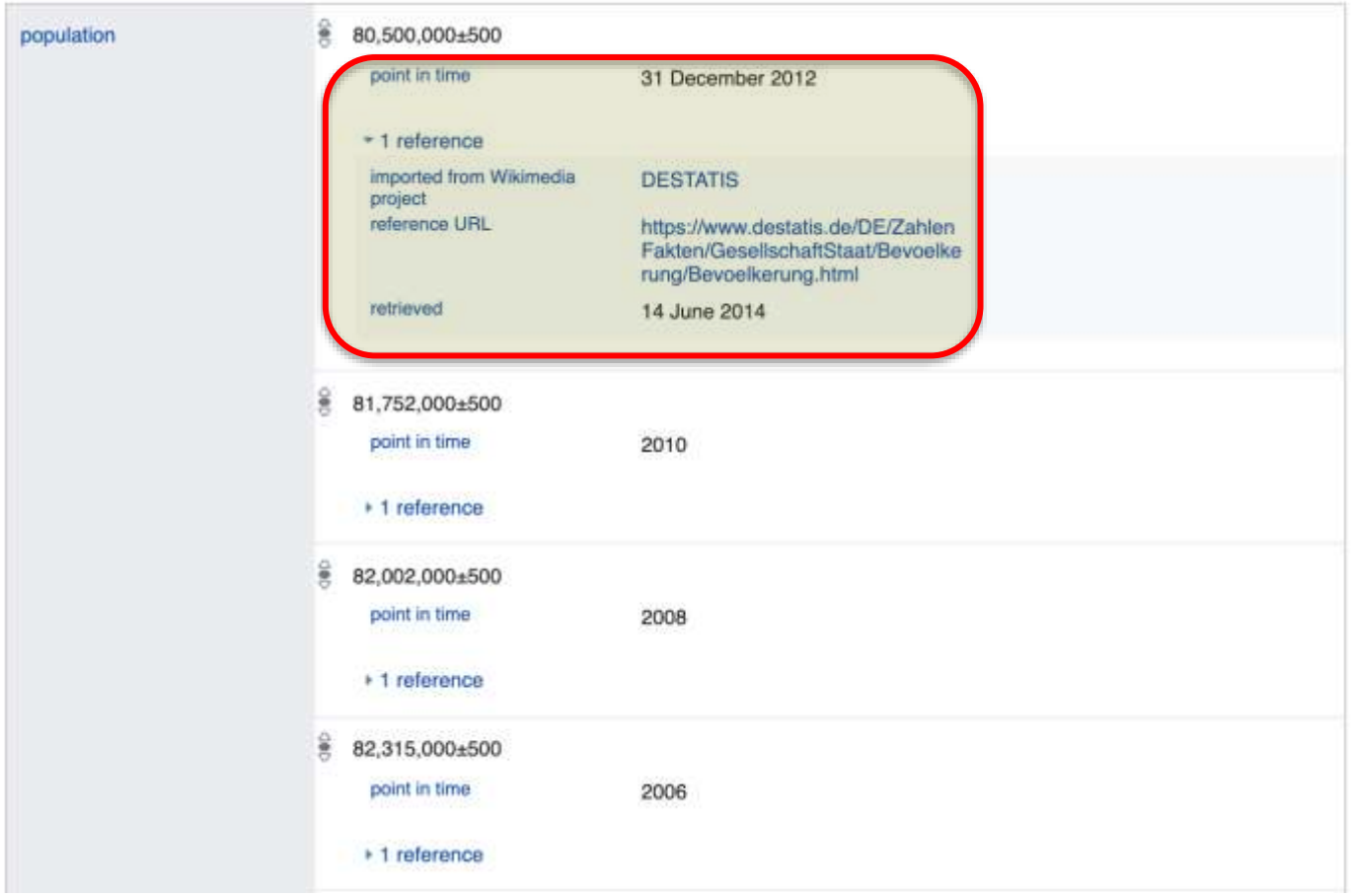
TopQuadrant™



The GraphQL schema for models automatically transformed from SHACL

Challenge 4: Capturing Statements About Statements

Some facts have additional information that you may want to capture



This is typically called “reification” - of a property value

Two approaches

1. Turn an attribute into a relationship

   Use Population as a class with instances

   For example:

   ex:Germany ex:population ex:P123.

   ex:P123 a ex:Population;

   ex:value 80500000;

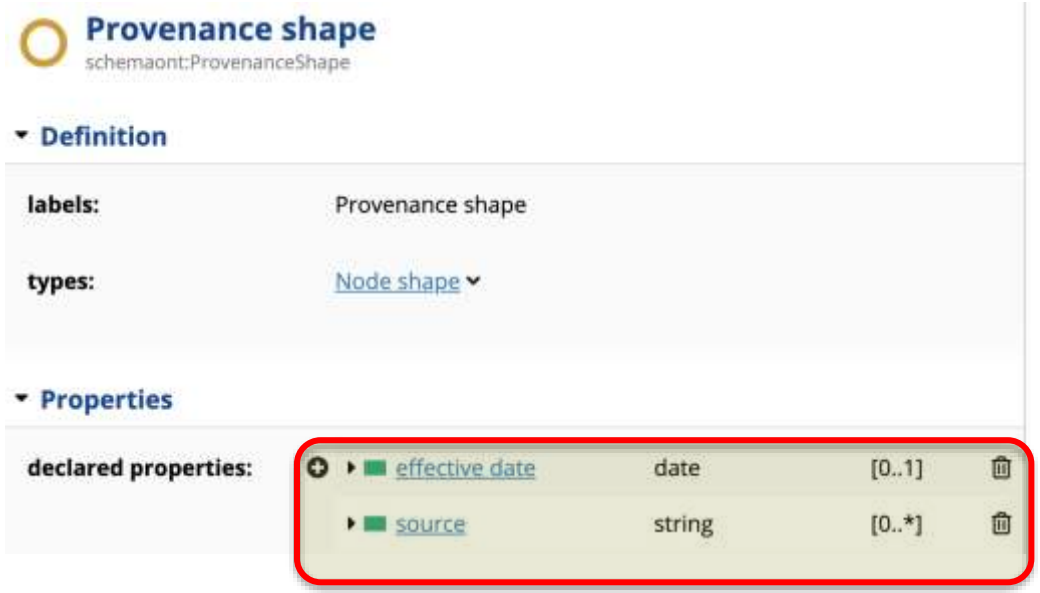
   ex:date 2012.

2. Keep an attribute and its value as-is

   Add a statement with the date “on top” of the population value

Enabling of a “less verbose” Way to Capture Statements About Statements

- Create a node shape that will define facts a user may need to add to values



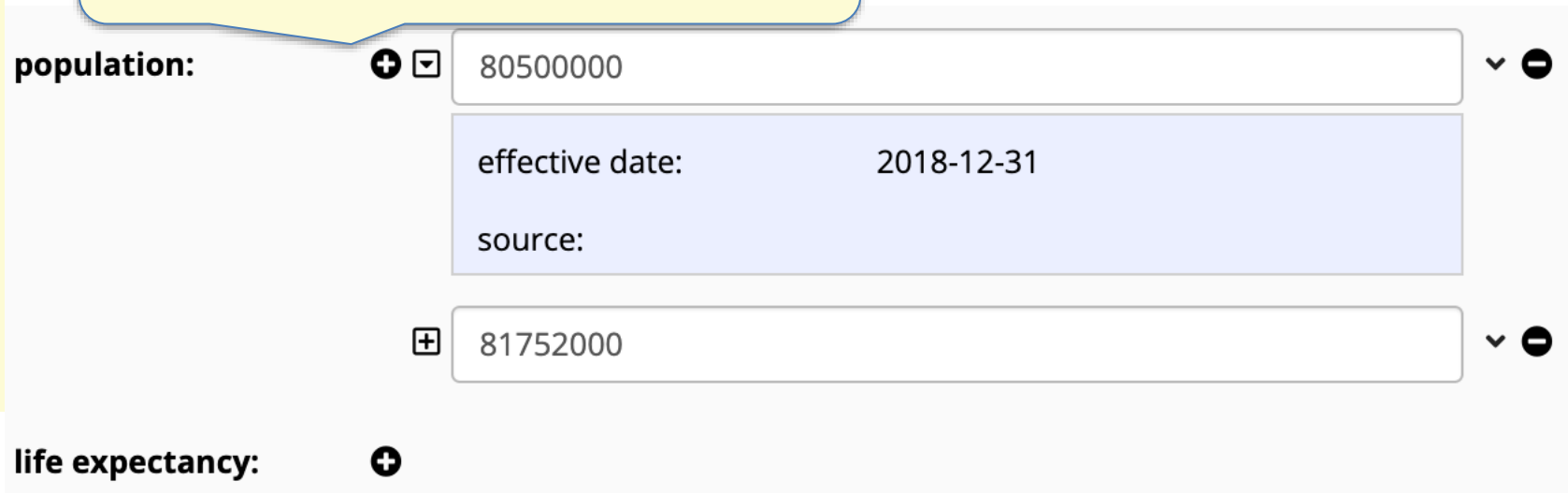
- Identify a property which values should be annotated with additional facts and connect it to the shape

see: http://datashapes.org/reification.html

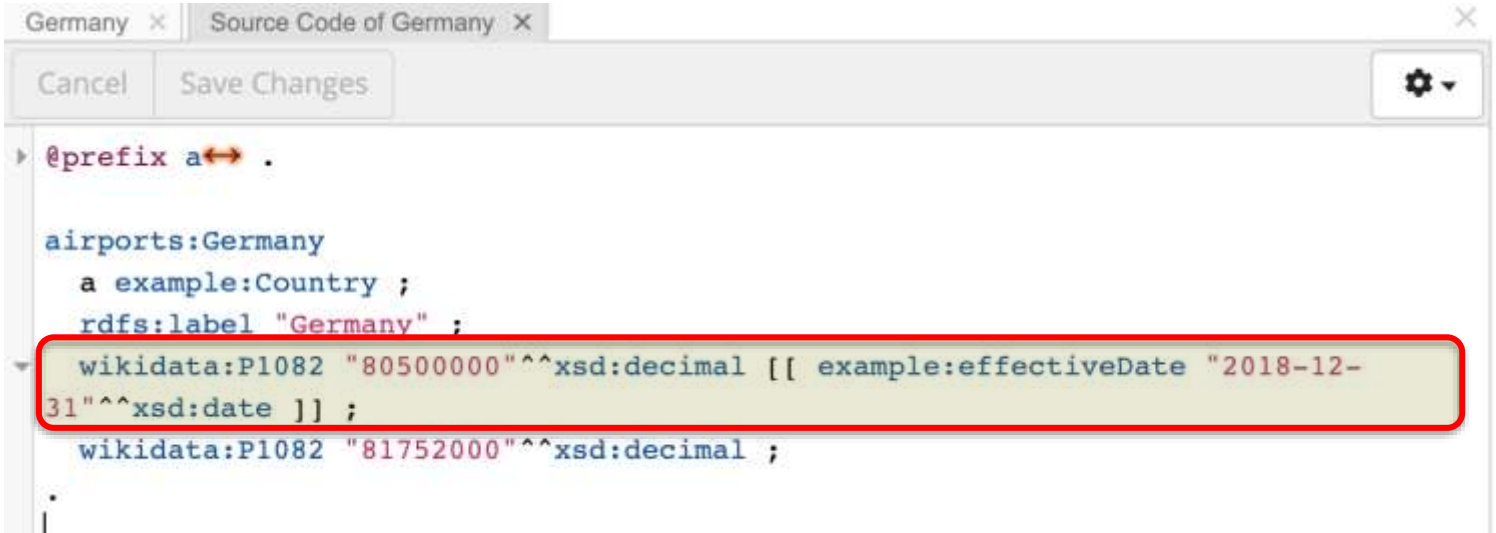
Example of Reification: Adding Annotations

In this example, users working with data can view and add a source and an effective date to statements capturing country's population

Reification Button opens the sub-form of reification properties

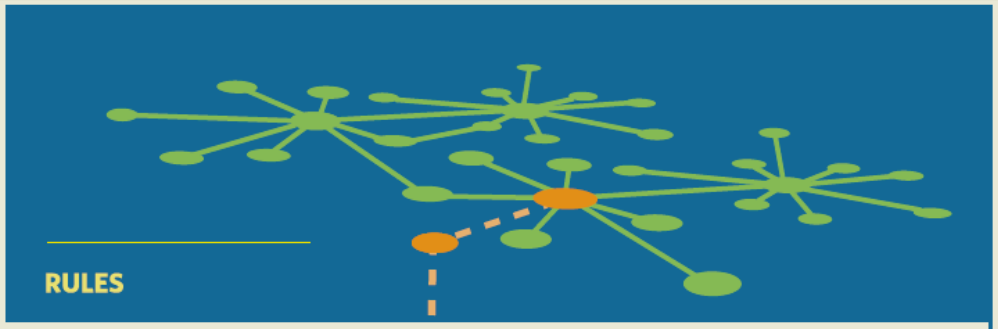


Stored as a single extra triple fact

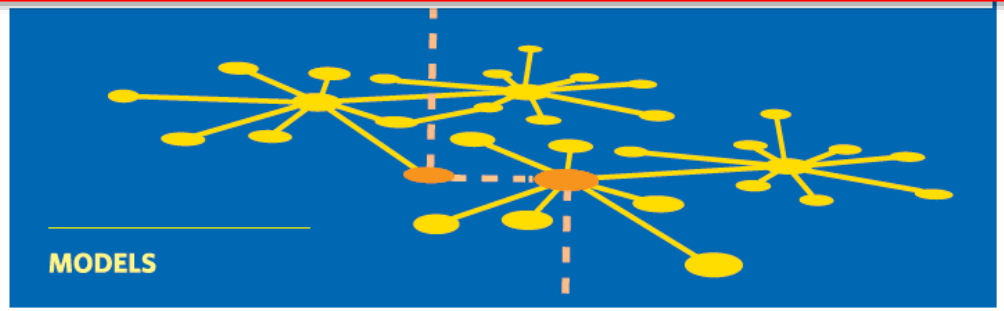


Challenge 5: Effective Ways to do Reasoning

- One of the advantages of the Knowledge Graphs is ability to do reasoning
  - Infer additional facts based on the facts that were provided



RULES: If both of a person's parents have blue eyes, they will also have blue eyes

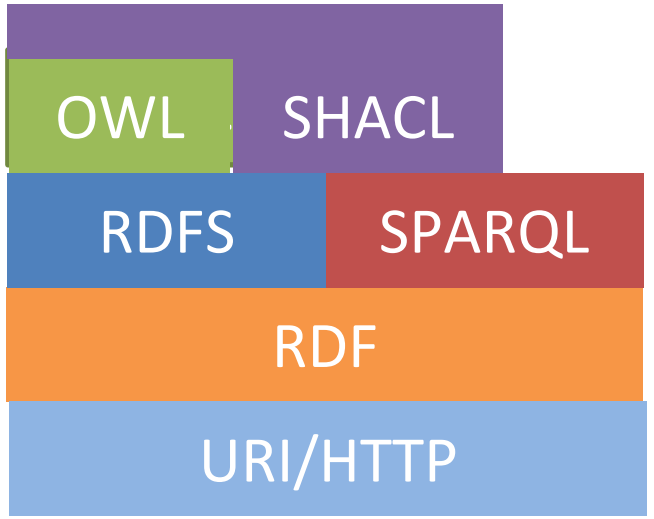


MODELS : A person has eye color. A person has two parents.
A person's father is also a person and he is male.

FACTS: James has blue eyes. James' father is Andrew. James is a person.

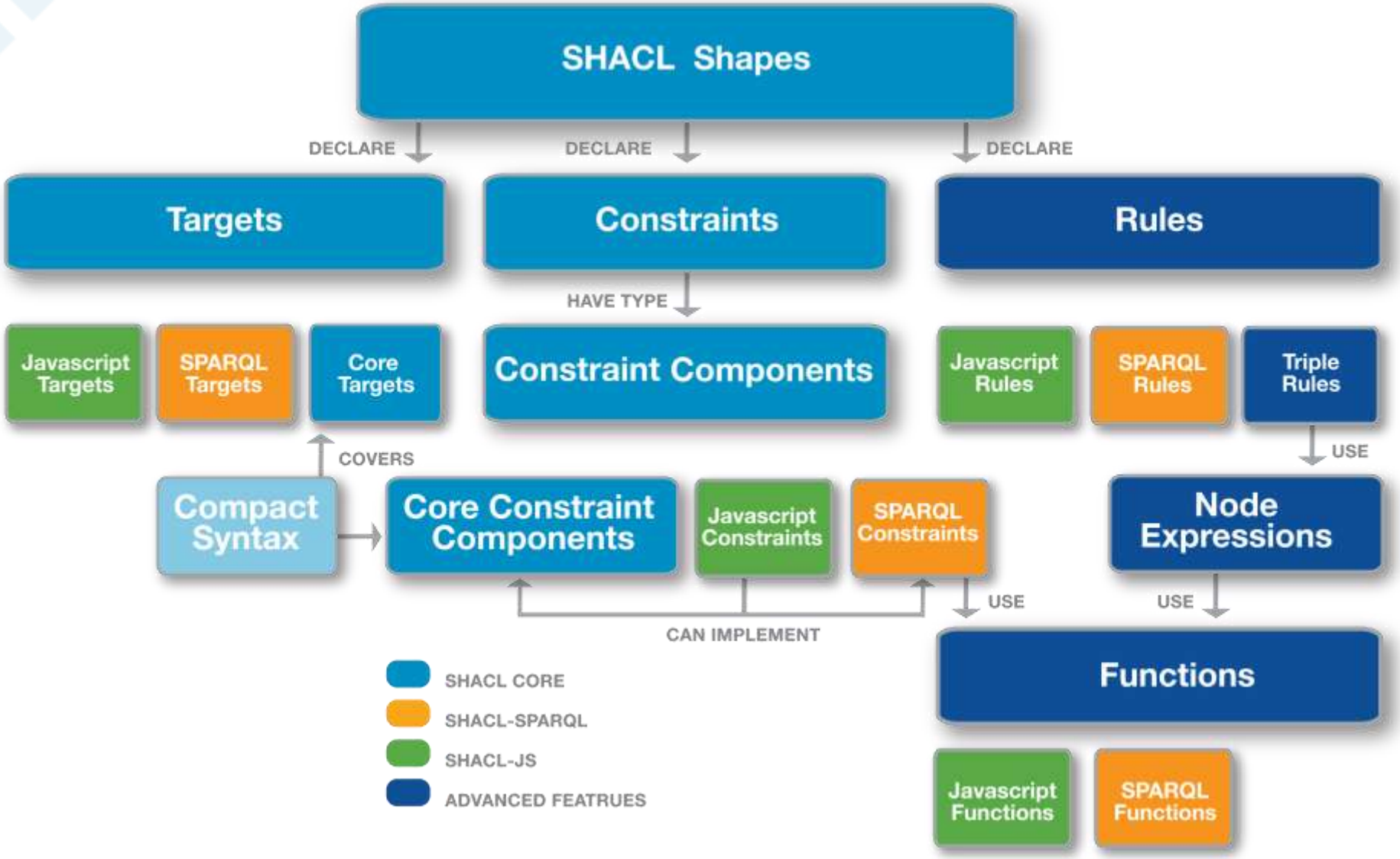
Reasoning or inferencing is a complex topic, often misunderstood

- OWL is difficult
  - for classification not for specification (open world)
  - Instance-level (A-Box) constraints and rules are hard if not impossible
- Emerging from practice, SHACL (previously SPIN) has proven success:
  - Developed from experiences of real-world problems across several application domains
  - Shapes as models of constraints
    - Highly expressive
      - Node Shapes
      - Property Shapes
    - Highly effective for data (and model) validation
    - Supporting rules and computed values



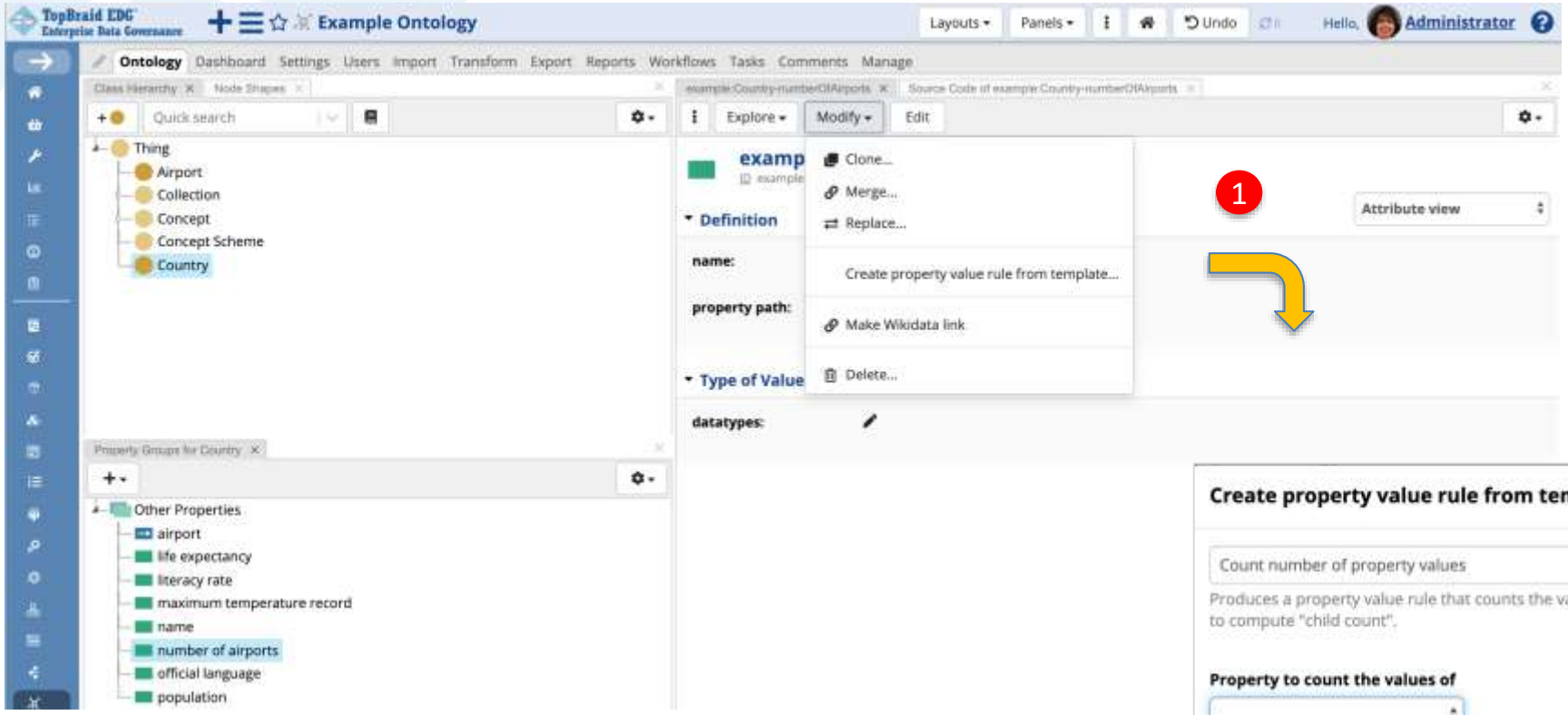
TopQuadrant™

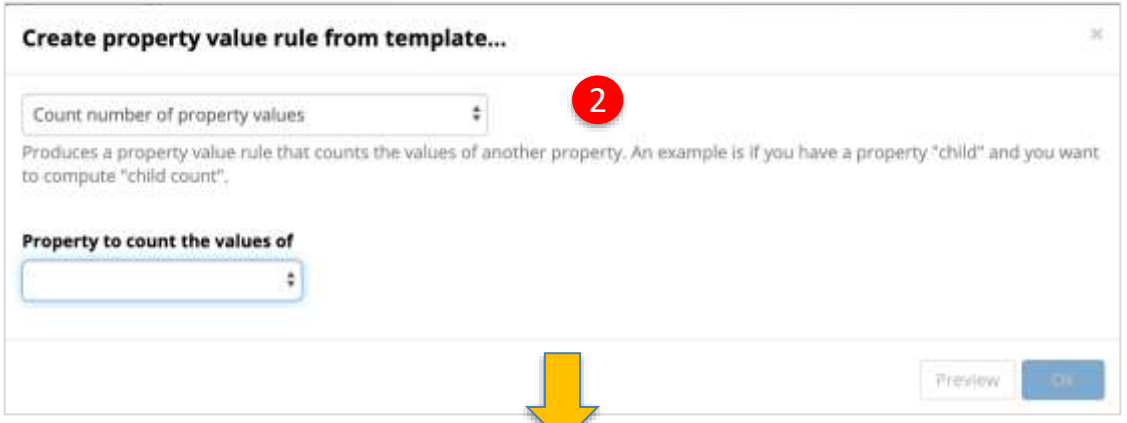
SHACL Components



Example of a Computed Value

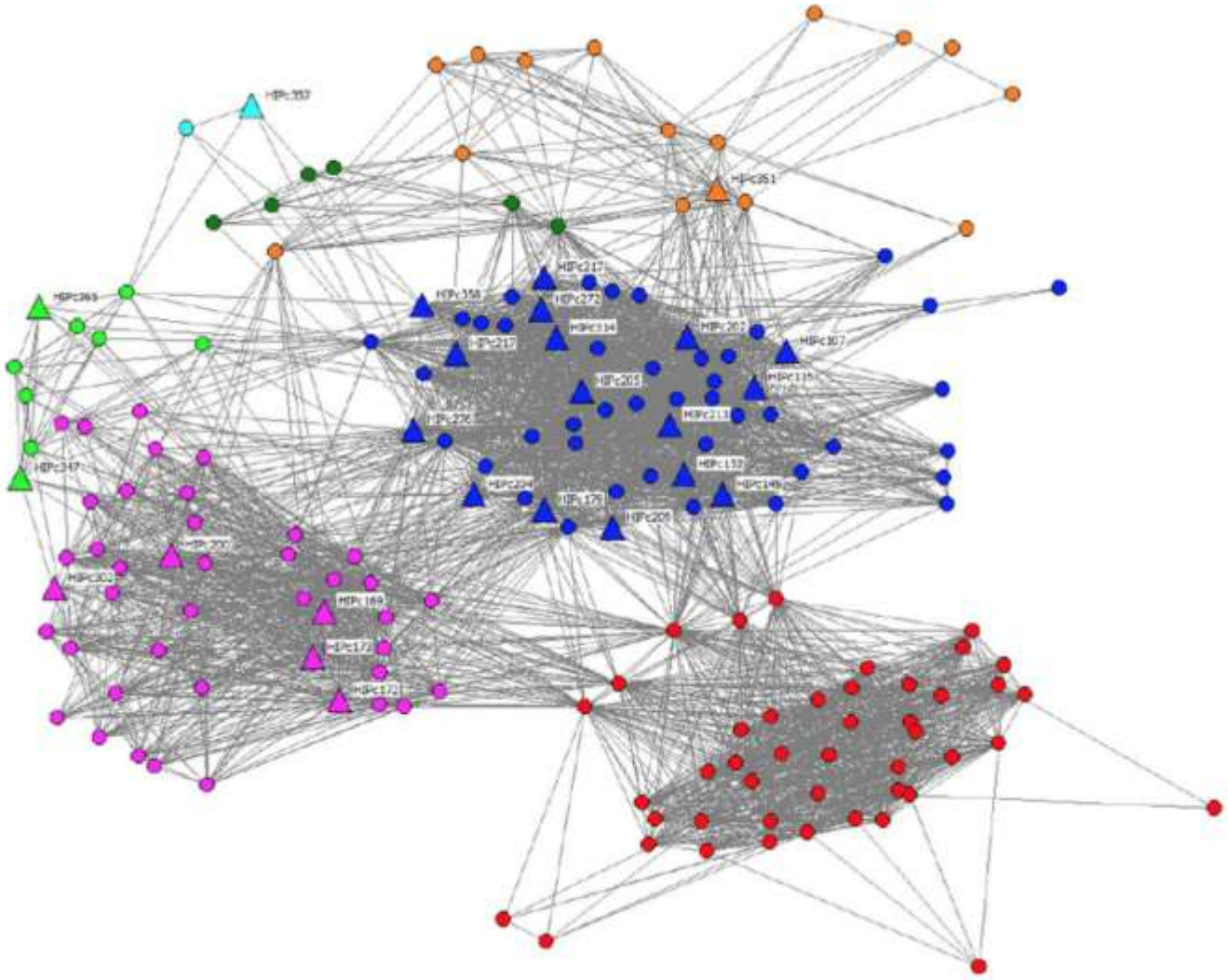
Several templates are available to simplify rule creation. More can be added.





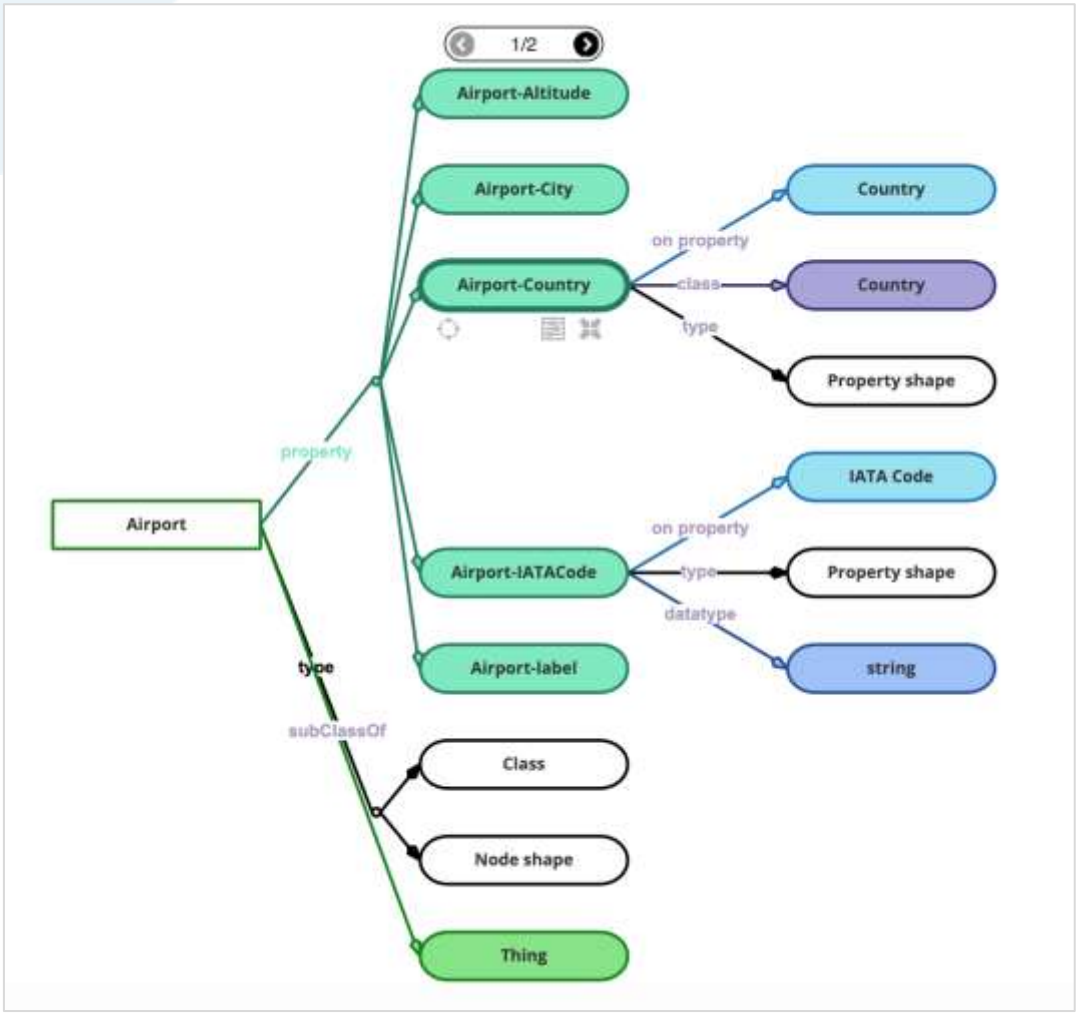
Challenge 6: Task Appropriate Visualizations

- Commonly used, basic graph visualization shows all nodes and links in a graph
- Useful to identify clusters of data or to graphically see all links from/to a given node
- Does not necessarily fit all use cases and activities

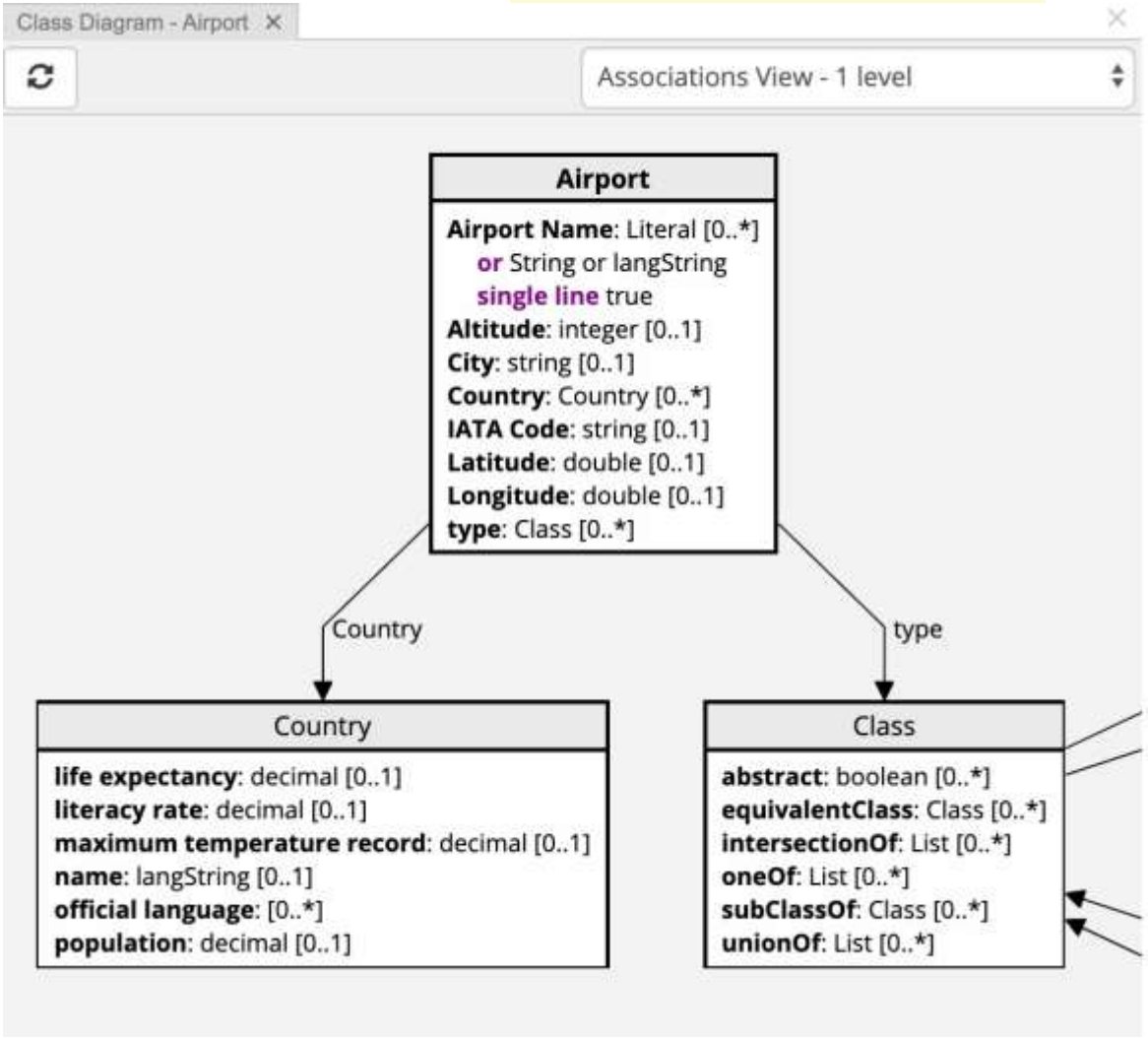


Example of Task Appropriate Visualization for an Ontology

Graph Diagram

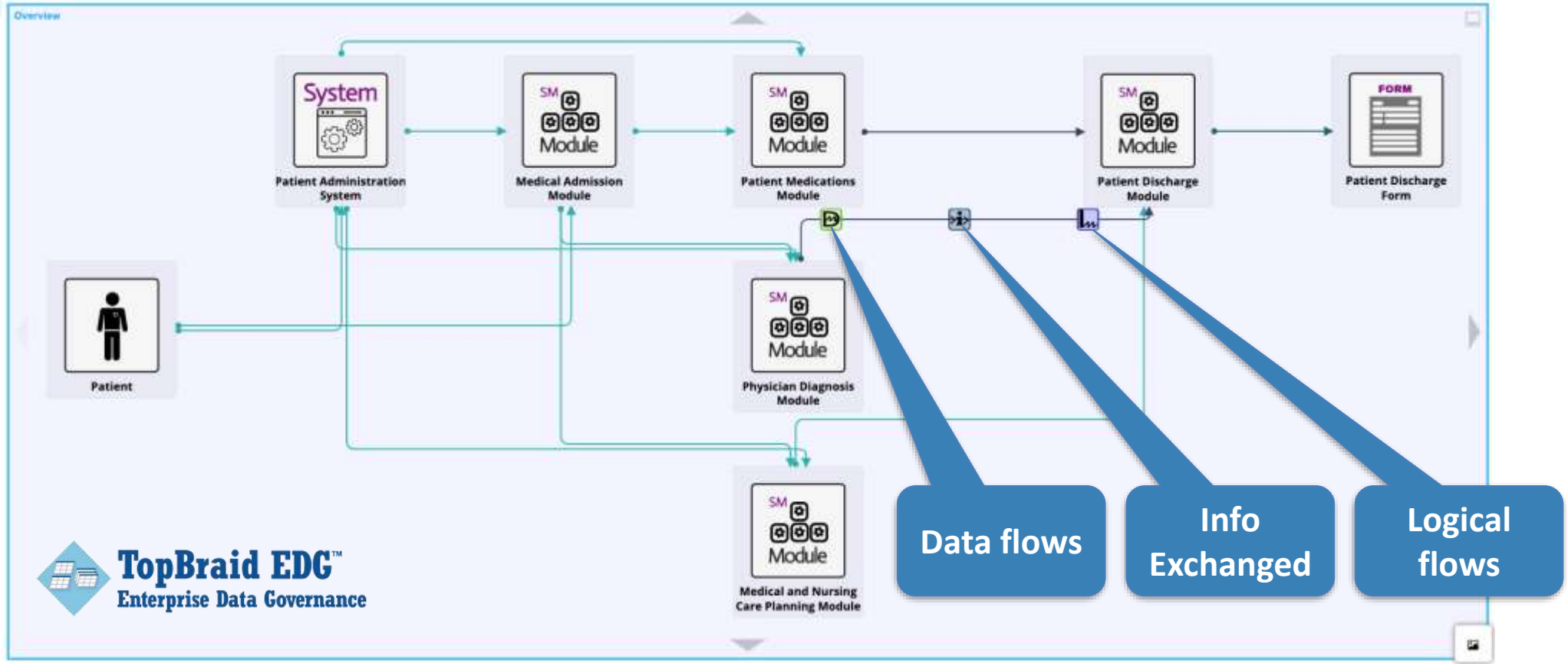


Class Diagram

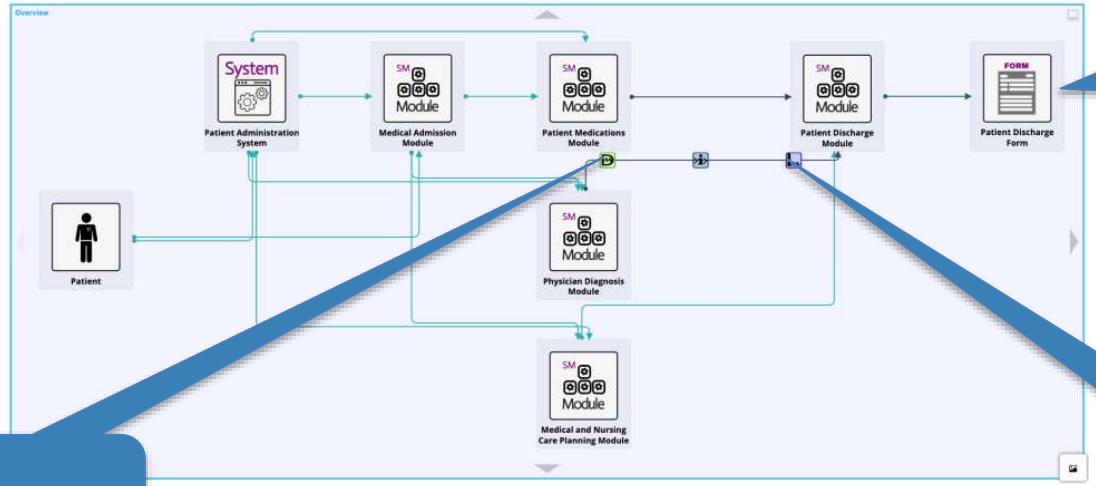


Example of Data Lineage Diagram

- A Viewpoint-specific visualization that uses reasoning. For example, to present links that are semantic inferences of dependencies.



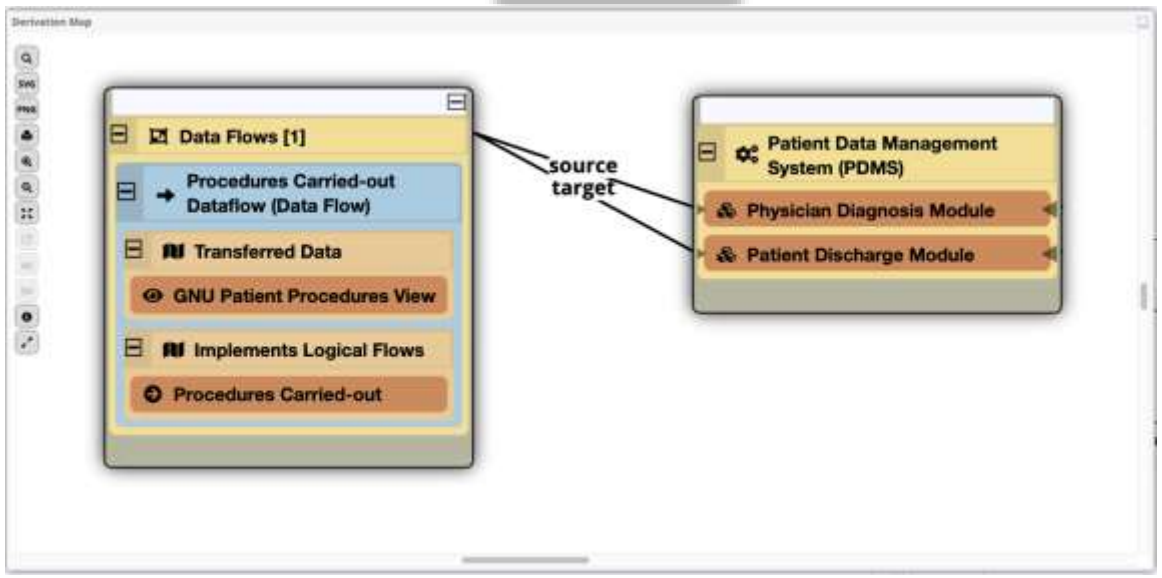
Digging Deeper from Logical (what) to Physical (how) Levels

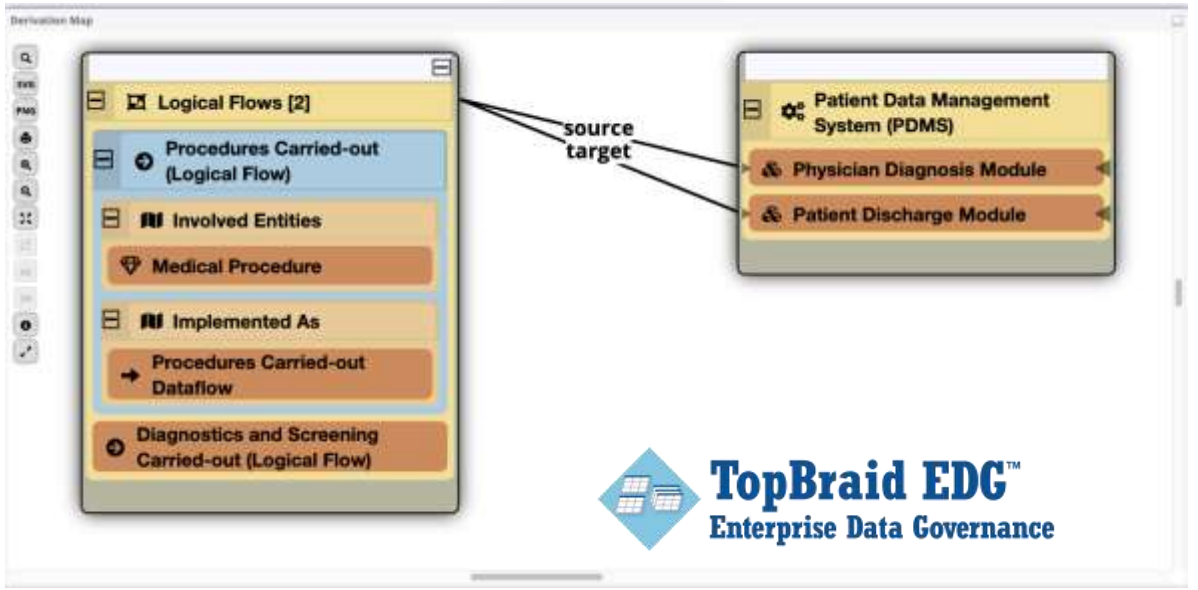


How a "Patient Discharge Form" gets created

Data flows

Logical flows





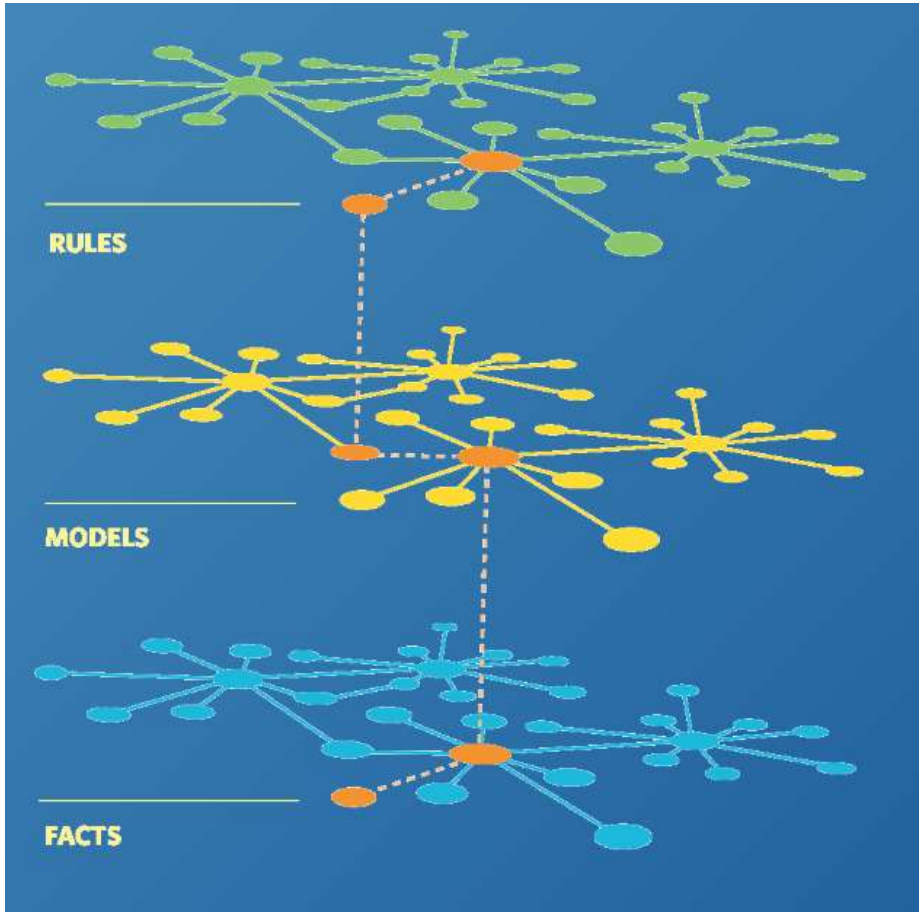
TopBraid EDG™
Enterprise Data Governance

Concluding Remarks

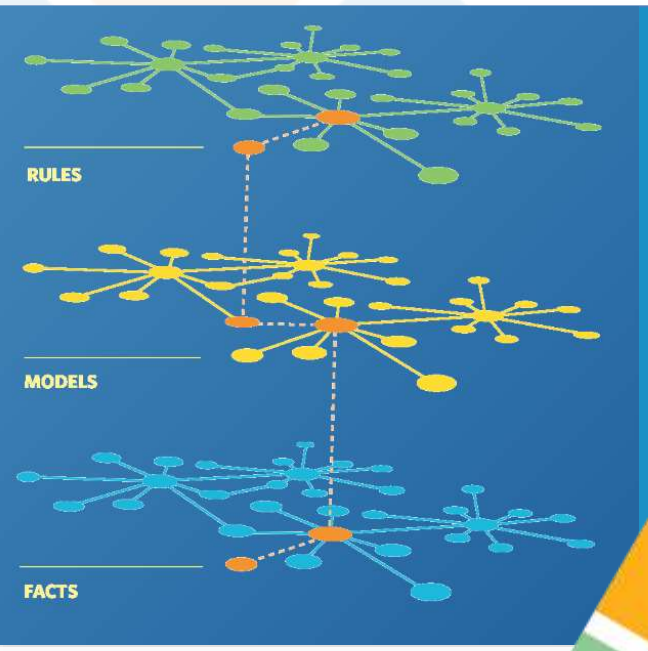
- Practices around RDF stack have matured
- Being too different (and too academic) hindered adoption
- Move to the mainstream means aligning with and enhancing broadly used development and data management approaches
- Products like TopBraid EDG help this happen by embedding and facilitating best practices

Benefits of a Knowledge Graph based Platform for Data Governance 2.0

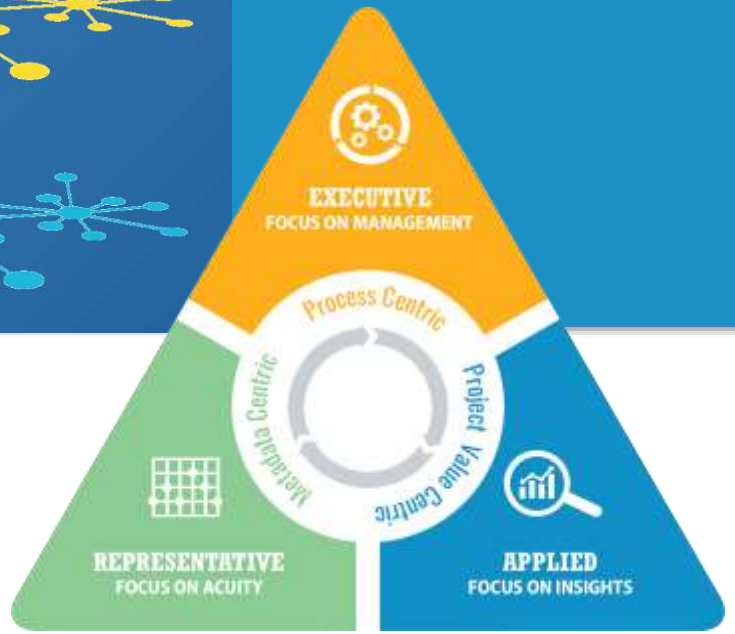
- Flexibility and extensibility based on standards
- Integration of reasoning and machine learning
- Enabling people (UI) and software (APIs/web services) to view, follow and query
- Bridging of data and metadata “silos” to provide seamless data governance
- Delivery of Knowledge-driven data governance



As an enterprise knowledge graph infrastructure, TopBraid EDG supports Data Governance 2.0 and applications of AI / ML



Thank You !



… Questions?

To Learn More about TopBraid EDG and Knowledge Graphs:

EDG Product Info:

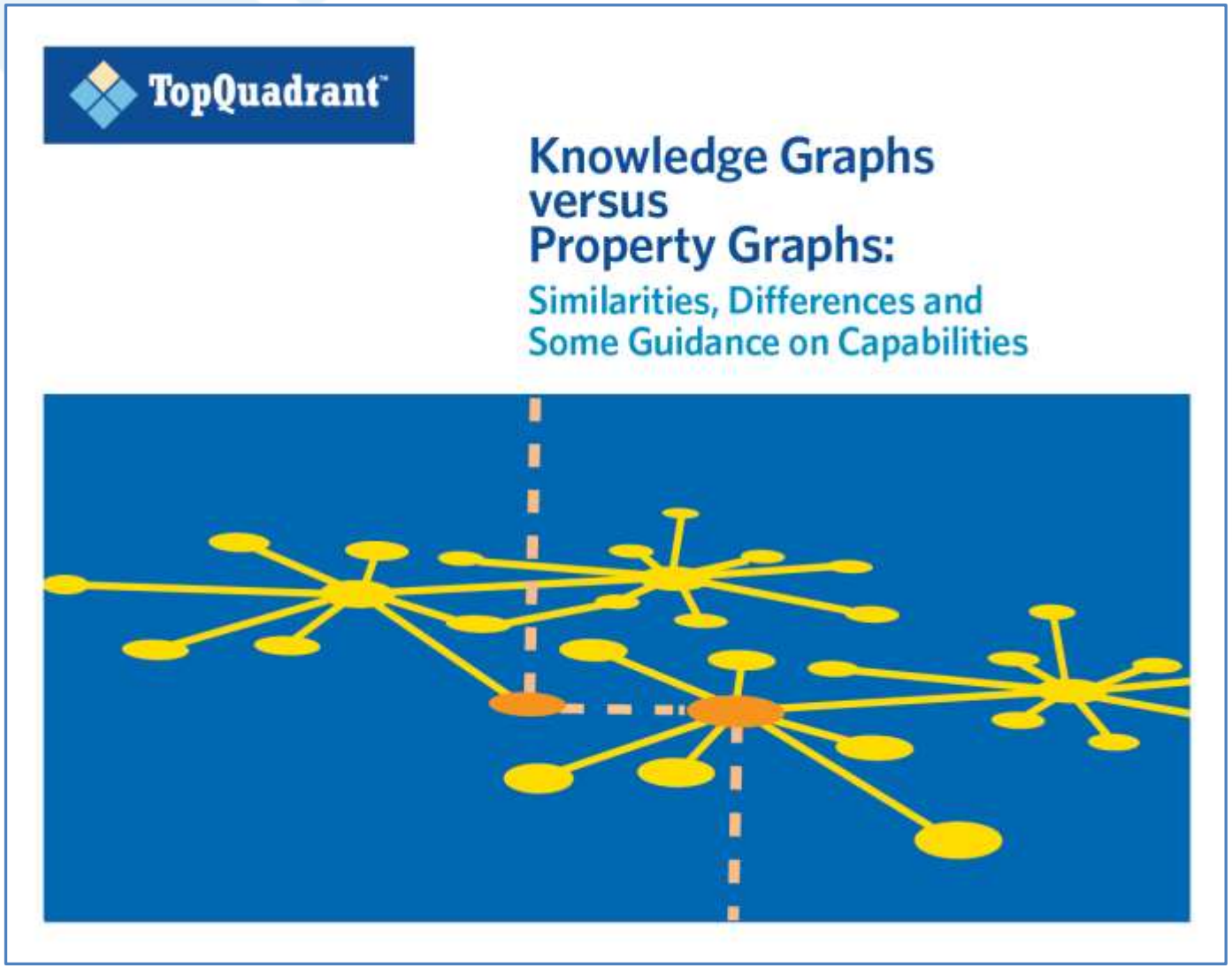
- TopBraid Enterprise Data Governance (TopBraid EDG)

(https://www.topquadrant.com/products/topbraid-enterprise-data-governance/)

Contact us: at info@topquadrant.com to:

- Discuss data governance and knowledge graphs
- Request a more targeted demo of TopBraid EDG
- Ask for a free EDG evaluation account

Latest Whitepaper



Download a copy on our website at:
topquadrant.com/knowledge-assets/whitepapers/

More Resources ...

More Webinar Recordings, Slides, Q&A:

- https://www.topquadrant.com/knowledge-assets/topquadrant-webinars/

Short Videos:

- TopBraid EDG “Quick Grok” Videos
  https://www.topquadrant.com/knowledge-assets/videos/

- TopBraid EDG Animated Video
  https://www.topquadrant.com/project/edg_agile_modular/

Blog:

- https://www.topquadrant.com/the-semantic-ecosystems-journal/

Data Governance White Papers

- https://www.topquadrant.com/knowledge-assets/whitepapers/

References - SHACL

- SHACL:
  - Constraints:
    - https://www.w3.org/TR/shacl/
  - Functions and Rules
    - https://www.w3.org/TR/shacl-af/
    - https://www.topquadrant.com/graphql/values.html
  - Reification with SHACL:
    - http://datashapes.org/reification.html

References - GraphQL

- GraphQL
  - An introduction to GraphQL in EDG:
    - https://www.topquadrant.com/technology/graphql/
  - Querying using GraphQL:
    - https://www.topquadrant.com/graphql/graphql-queries.html
  - TopQuadrant's Webinar on GraphQL:
    - https://www.youtube.com/watch?v=Muj0m8Qrdig