

GraphDB Workshop

DMZ, 1st March 2023



ORACLE

verizon✓

Gartner®

Experfy Training

Big Data Analyst

\$150.00

Enroll Now

Add to Cart

Add to Wish List

LIVE SUPPORT

Certification

industry recognized certification enables you to add this credential to your resume

Duration: 10h 41m

ABOUT COURSE

PREREQUISITES

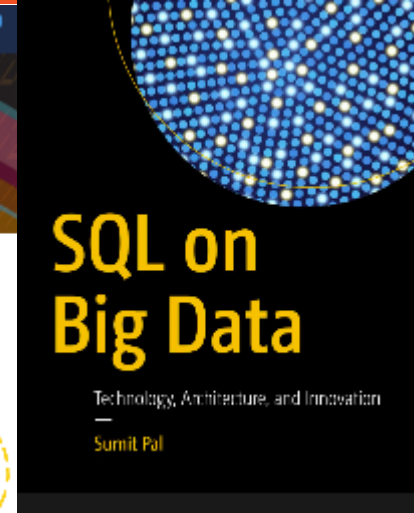
COURSE CURRICULUM

REVIEWS

The media playback was aborted due to a corruption problem or because the media used features your browser did not support.

Master the skills necessary to build a career in Big Data.

- Get up to speed with big data technologies and start doing analyst work on massive data sets.
- Instructor: Microsoft SQL Server team (1996-1997), Oracle development team (1997-2004) and Big Data team at Verizon Labs (2013-2015).
- This big data online training prepares you for Cloudera's Business Analyst Certification.



Graph General

DMZ, 1st March 2023

NETFLIX

UNLIMITED TV SHOWS & MOVIES

JOIN NOW

SIGN IN

N SERIES

connected

THE HIDDEN SCIENCE OF EVERYTHING

Connected

2020 | U/A 13+ | 1 Season | Science & Nature Docs

Science journalist Latif Nasser investigates the surprising and intricate ways in which we are connected to each other, the world and the universe.

Starring: Latif Nasser



ontotext

INVESTIGATIONS ▾ LATEST DATA ▾ JOURNALISTS LEAK TO ICIJ ABOUT ▾



INTERNATIONAL CONSORTIUM
OF INVESTIGATIVE JOURNALISTS

DONATE



An ICIJ Investigation

THE PANAMA PAPERS: EXPOSING THE ROGUE OFFSHORE FINANCE INDUSTRY

A giant leak of more than 11.5 million financial and legal records exposes a system that enables crime, corruption and wrongdoing, hidden by secretive offshore companies.



- Need to understand **latent relationships**
- Rapidly **traverse relationships** for insights, patterns
- Data is variable & **does not fit 2D model** of rows and columns
- Integrate **disparate heterogeneous data sources**
- Visualize **clustering** of data based on **connections** or patterns

Ontotext - GraphDB

DMZ, 1st March 2023

Company Highlights



OVERVIEW

- Ontotext - global leader in RDF graph databases, knowledge management and content analytics
- Established in 2000 as an R&D lab
- Growth funding round in 2022
- Employees: ~110 / Presence in Europe, USA and APAC
- Verticals include Financial Services, Life Sciences, Health Care, Manufacturing, Publishing

MANAGEMENT TEAM



Atanas Kiryakov (CEO)
Founded Ontotext in 2000



Veska Davidova (COO)
Joined Ontotext in 2013,
ex. Atos, Siemens, HP



Vassil Momtchev (CTO)
Joined Ontotext in 2005



Deron Ryan (CCO)
Joined Ontotext in 2022, ex.
ScienceLogic, Microsoft



Doug Kimball (CMO)
Joined Ontotext in 2022,
ex. Stibo Systems, Nielsen

Global Leaders Choose Ontotext



IT systems monitoring in retail banking



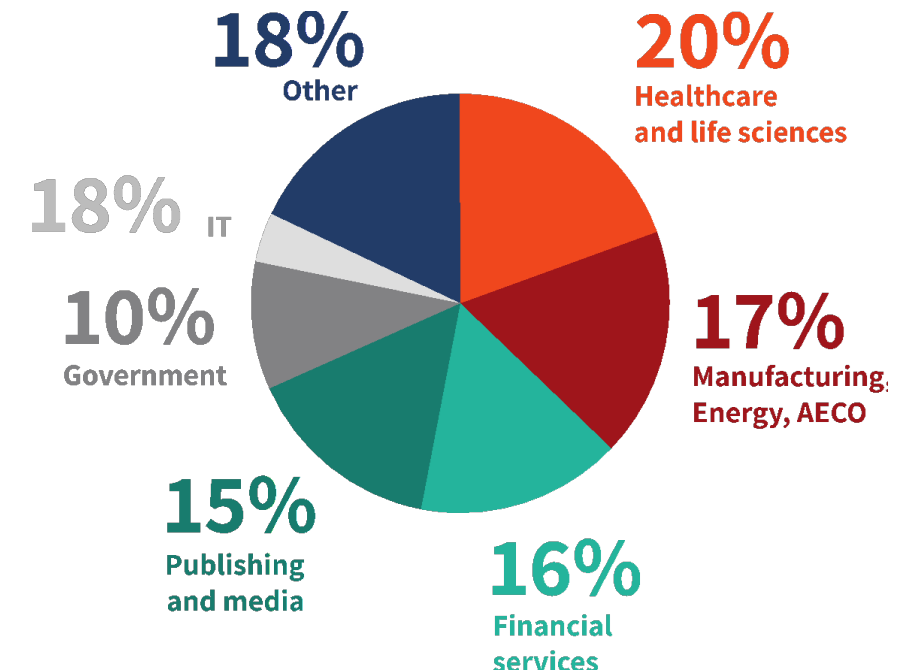
Building management systems



Oil & Gas Market Analysis Tool



Preclinical Discovery & Roche Scientific Interoperability Hub



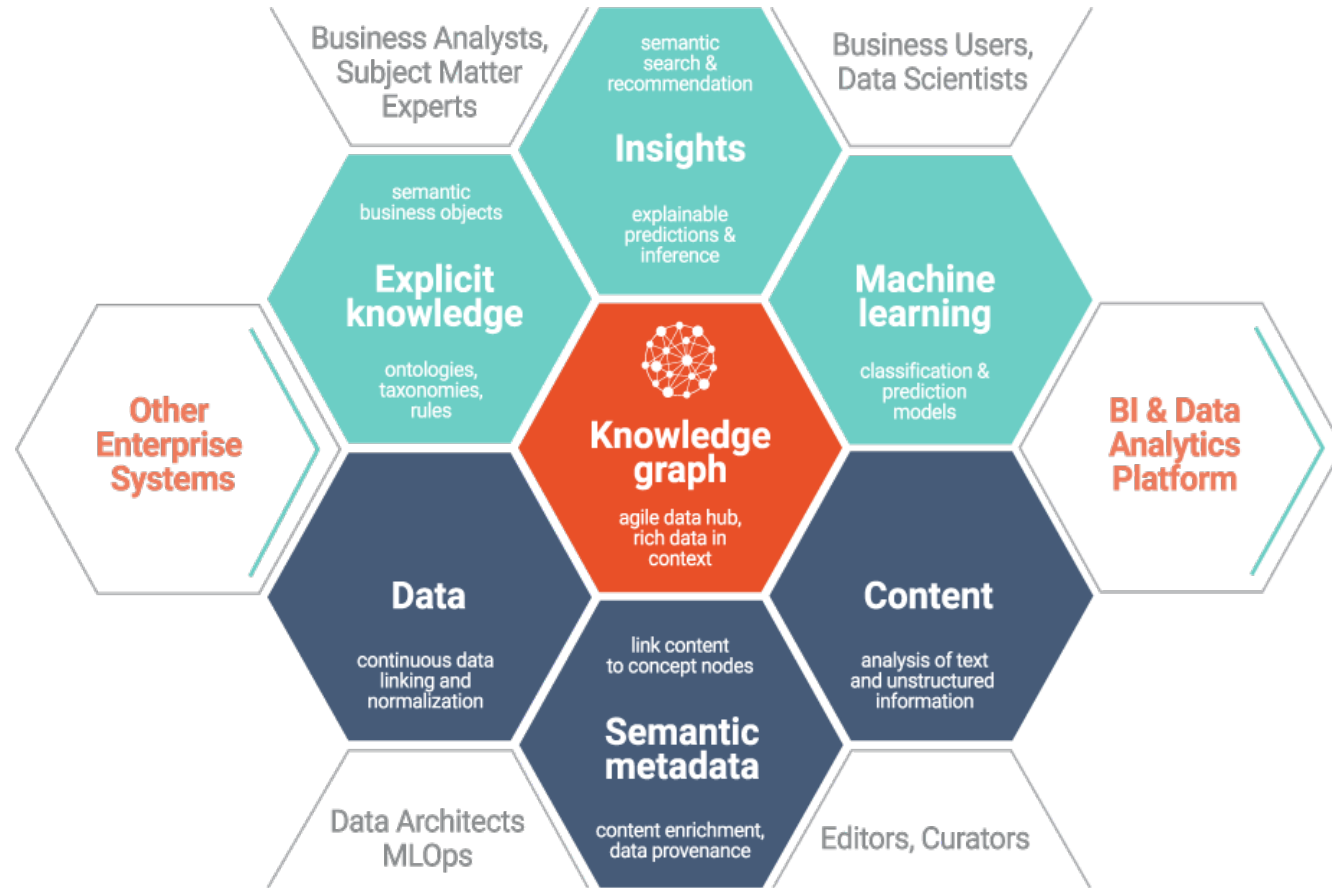
What Do We Do?



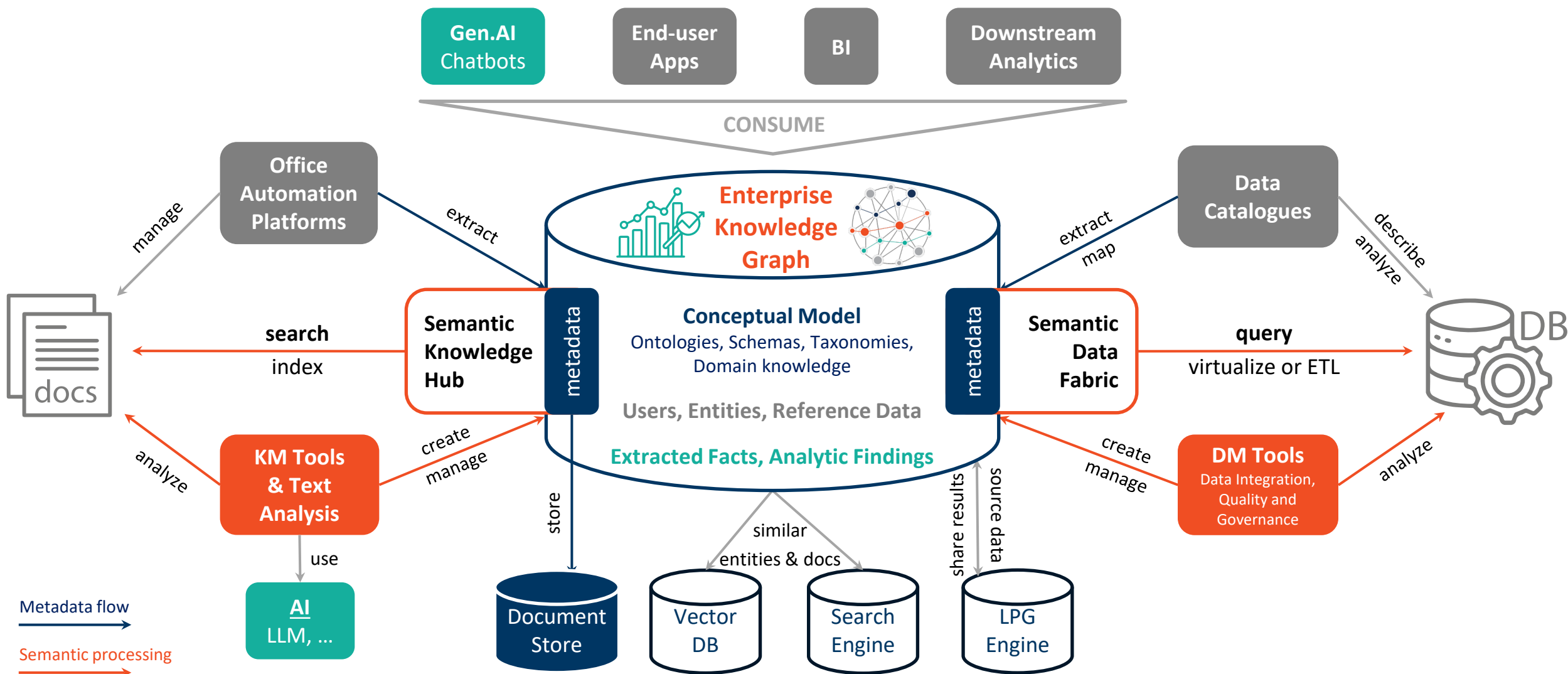
Connect data into reusable graphs to power deep analytics

We help enterprises get profound insights by linking:

- Disparate databases and systems
- Proprietary & global data
- Structured data and documents



The Big Picture of the Graph Platform



- **Scalable and Dependable RDF 1.1 engine**

- Predictable performance across wide range of workloads
- ACID compliant transactions



- **Platform Independent (Java)**

- **W3C Standards Compliant**

- Comprehensive support for RDF, SPARQL, OWL 2, and SHACL



- **Reasoning and Validation**

- **High-Availability Cluster and Enterprise-Grade Security**

- **Connectors for Upstream and Downstream Integration**

- **Free Community Support & Dedicated Commercial support and maintenance**



GraphDB Editions



FREE

Delivers a fully functional database optimized for desktop use and small commercial prototypes:

- No constraints on data scale
- Limited to one concurrent read
- Single inference thread
- Lucene connector for indexing

ENTERPRISE

Offers cluster support for enterprise resilience and high-availability:

- Eliminates having single point of failure
- Multi-data center support
- Unlimited scalability of read operations
- Elasticsearch, OpenSearch, Solr and Kafka

Multiple Operation and Deployment Options



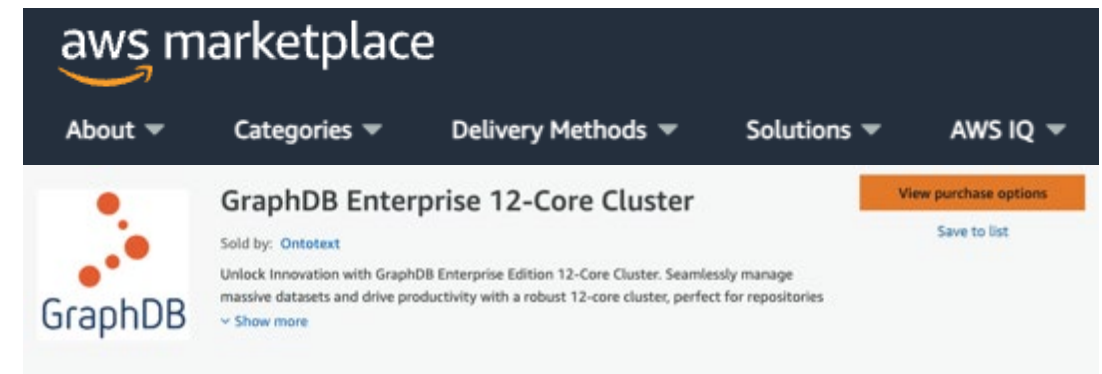
○ Operation mode

- Self-service
 - On-prem/Cloud
 - Technical support with SLA
- Managed service
 - AWS / Azure marketplaces
 - Service availability with SLA

○ Deployment options

- Native installation packages
(zip, msi, dmg, deb, rpm)
- Docker/Helm charts
- AMI & Azure VM

	GDB Self-hosted	GDB Self-hosted via Marketplace	GDB Managed Service
Channel	Ontotext Direct	AWS / Azure	AWS / Azure
SLA/Dev support	Recommended	Recommended	Required
Hosting	Client	Client	Ontotext
Operation	Client	Client	Ontotext



Architecture



○ GraphDB Workbench

The screenshot shows the GraphDB Workbench interface. On the left, there's a sidebar with icons for home, queries, graphs, settings, and help. The main area is titled 'SPARQL Query & Update'. It contains a text editor with a SPARQL query, a 'Run' button, and a 'test' dropdown. Below the editor, there are tabs for 'Table', 'Raw Response', 'Pivot Table', 'Google Chart', and 'Graph(beta)'. The 'Table' tab is active, showing a table of results. A 'Filter query results' bar is above the table, indicating 'Showing results from 1 to 1,000 of 2,082. Query took 0.1s.' The table has 10 rows of data, including values like 'vin:adjacentRegion', 'vin:locatedIn', and 'http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine'.

```
1 SELECT ?yr ?name ?document
2 WHERE {
3   ?class rdfs:subClassOf foaf:Document .
4   ?document rdf:type ?class .
5   ?document dcterms:issued ?yr .
6   ?document dc:creator ?author .
7   ?author foaf:name ?name
8   OPTIONAL {
9     ?class2 rdfs:subClassOf foaf:Document .
10    ?document2 rdf:type ?class2 .
11    ?document2 dcterms:issued ?yr2 .
12    ?document2 dc:creator ?author2
13    FILTER (?author=?author2 && ?yr2<?yr)
14  } FILTER (!bound(?author2))
15 }
```

	s
1	vin:adjacentRegion
2	vin:locatedIn
3	http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine
4	http://www.w3.org/TR/2003/CR-owl-guide-20030818/wine
5	vin:Wine
6	vin:Winery
7	vin:Region
8	vin:Vintage
9	vin:WineGrape
10	vin:WhiteWine

○ GraphDB Engine

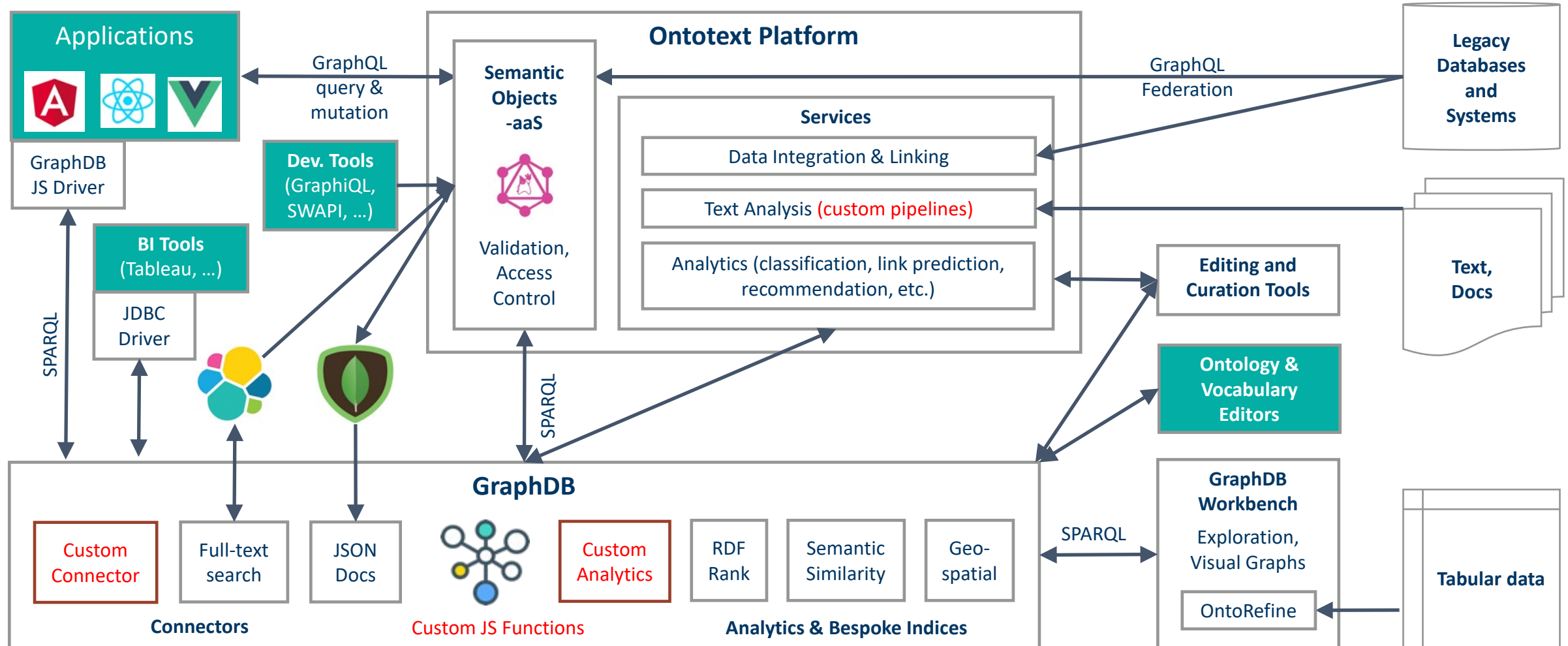
- storage and retrieval of triples through semantic queries
- REST API for database access

○ Plugins

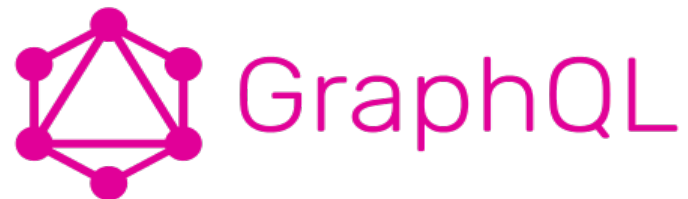
- Various extensions & connectors enable tailoring for specific needs and use cases.

repositories : Repository management		Show/Hide	List Operations	Expand Operations
sparql : SPARQL		Show/Hide	List Operations	Expand Operations
DELETE	/repositories/{repositoryID}/statements	Deletes statements from the repository.		
GET	/repositories/{repositoryID}/statements	Fetches statements from the repository.		
POST	/repositories/{repositoryID}/statements	Performs updates on the data in the repository		
PUT	/repositories/{repositoryID}/statements	Updates data in the repository, replacing any existing data with the supplied data		
contexts : Contexts management		Show/Hide	List Operations	Expand Operations
namespaces : Namespaces management		Show/Hide	List Operations	Expand Operations

Access, Customization and Integration Points



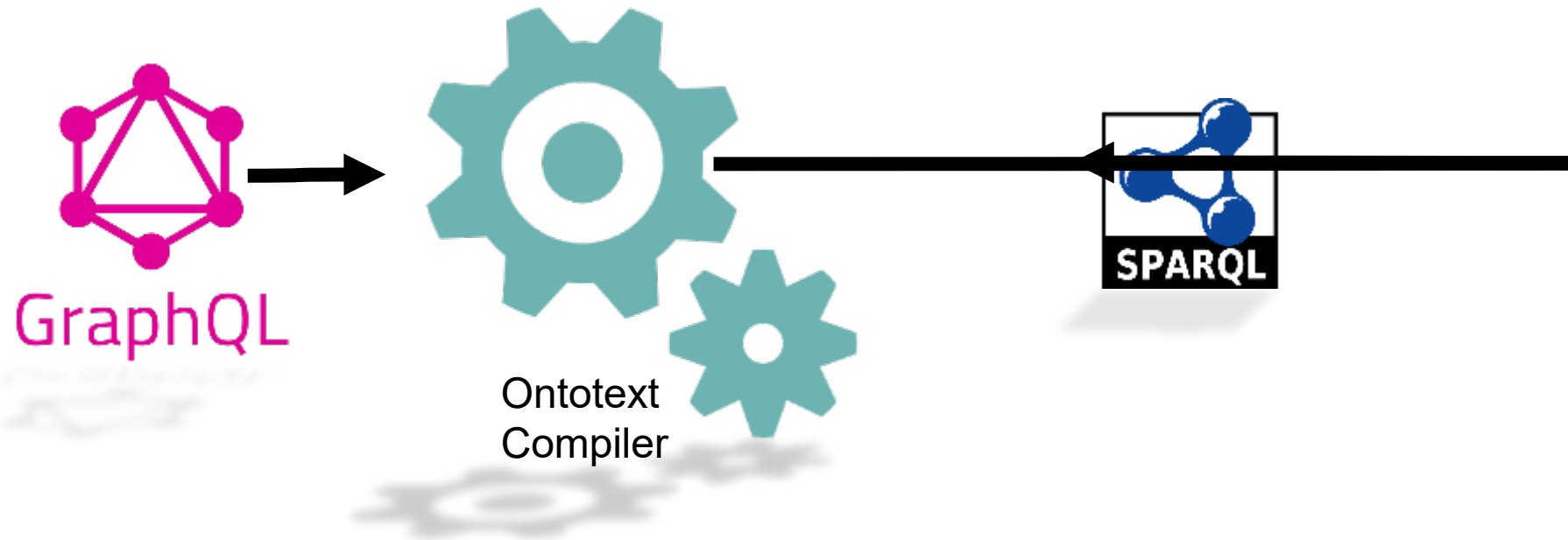
- **SPARQL 1.1 with SPARQL-star extensions**
- **Many SPARQL extensions that provide additional functionality**
- **GraphQL**
- **Beyond vanilla SPARQL**
 - Faceting and full-text search
 - Graph path search
 - Geographic query extensions
 - Ranking of RDF nodes
 - Similarity search



GraphQL Access via Semantic Objects



- Knowledge Graph access and updates via GraphQL
- Data validation via RDF Shapes
- Semantic Business Objects definitions done by business analysts
 - GraphQL Schema and shapes generated from Semantic Objects



Knowledge Graphs

DMZ, 1st March 2023

Fortune 100 Companies leverage Knowledge Graphs

Google Knowledge Graph

Article Talk

Read Edit View history Tools

From Wikipedia, the free encyclopedia

"Knowledge Graph" redirects here. For the general concept in information science, see *Knowledge graph*. For other uses, see *Knowledge Graph* (disambiguation).

The **Google Knowledge Graph** is a *knowledge base* from which Google serves relevant information in an infobox beside its search results. This allows the user to see the answer in a glance, as an *instant answer*. The data is generated automatically from a variety of sources, covering places, people, businesses, and more.^{[1][2]}

The information covered by Google's Knowledge Graph grew quickly after launch, tripling its data size within seven months (covering 570 million entities and 18 billion facts^[3]). By mid-2016, Google reported that it held 70 billion facts^[4] and answered "roughly one-third" of the 100 billion monthly searches they handled. By March 2023, this had grown to 800 billion facts on 8 billion entities.^[5]

There is no official documentation of how the Google Knowledge Graph is implemented.^[6] According to Google, its information is retrieved from many sources, including the *CIA World Factbook* and Wikipedia.^[7] It is used to answer direct spoken questions in Google Assistant^{[8][9]} and Google Home voice queries.^[10] It has been criticized for providing answers with neither source attribution nor



Building The LinkedIn Knowledge Graph

Qi He October 6, 2016

Share Tweet Share



Public Knowledge Graphs

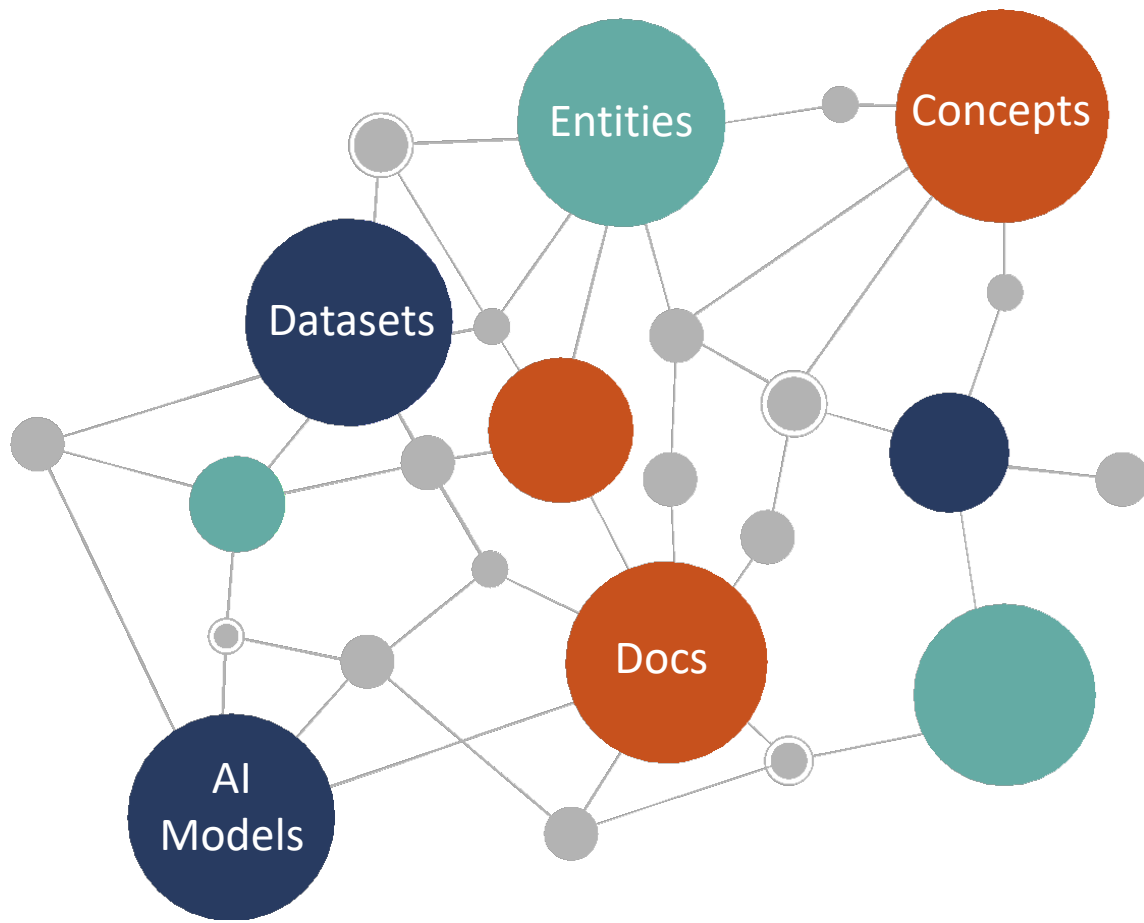


Examples of Knowledge Graphs



- **Microsoft Satori (powers Bing and Cortana)**
 - **Google KG (powers Google Search and Google Assistant)**
 - **Amazon Ivi Information Graph (powers Amazon Alexa)**
 - **ESCO – European Labor Market KG about Occupations, Skills and Qualifications**
 - **ERNIE: A knowledge graph-enhanced language model**
 - **LYNX PROJECT manage compliance, based on a legal knowledge graph (LKG)**
- **DiffBot**
 - **ShopBot**
 - **Wikidata / DBPedia**
 - **YAGO**
 - **ConceptNet**
 - **Metaweb**
 - **Freebase**
 - **OpenIE**
 - **Gdelt**
 - **GeoNames**
 - **Cyc**
 - **WIKI Media**

What is a Knowledge Graph?



Interlinked descriptions of concepts and entities

- Concepts describe each other
- Connections provide context
- Context helps comprehension

Can be used as a:

- **Database:** can be queried
- **Graph:** can be analyzed as a network
- **Knowledge base:** new facts can be inferred

Data analytics & AI/LLM initiatives →

Challenged in **unified view across diverse data** sources.

KGs are a Game Changer because they:

- Establish a machine-readable **contract about the meaning** of the data
- Use **semantic metadata** to interconnect and **harmonize siloed data**
- **Enrich data** with external **domain knowledge**

Our platform enables enterprises to build a **solid data foundation** for:

- **Expedited data discovery** and integration
- **Continuous unification** and layering of **usage-centric models** (e.g. ontologies)
- **Standardized data exchange** and publishing
- **Accurate interpretation** of data governed in decentralized manner (data mesh)

Why Knowledge Graphs?



Put data in context via semantic metadata and linking

Help enterprises gain **deeper insights** via linking, analysis and exploration of:

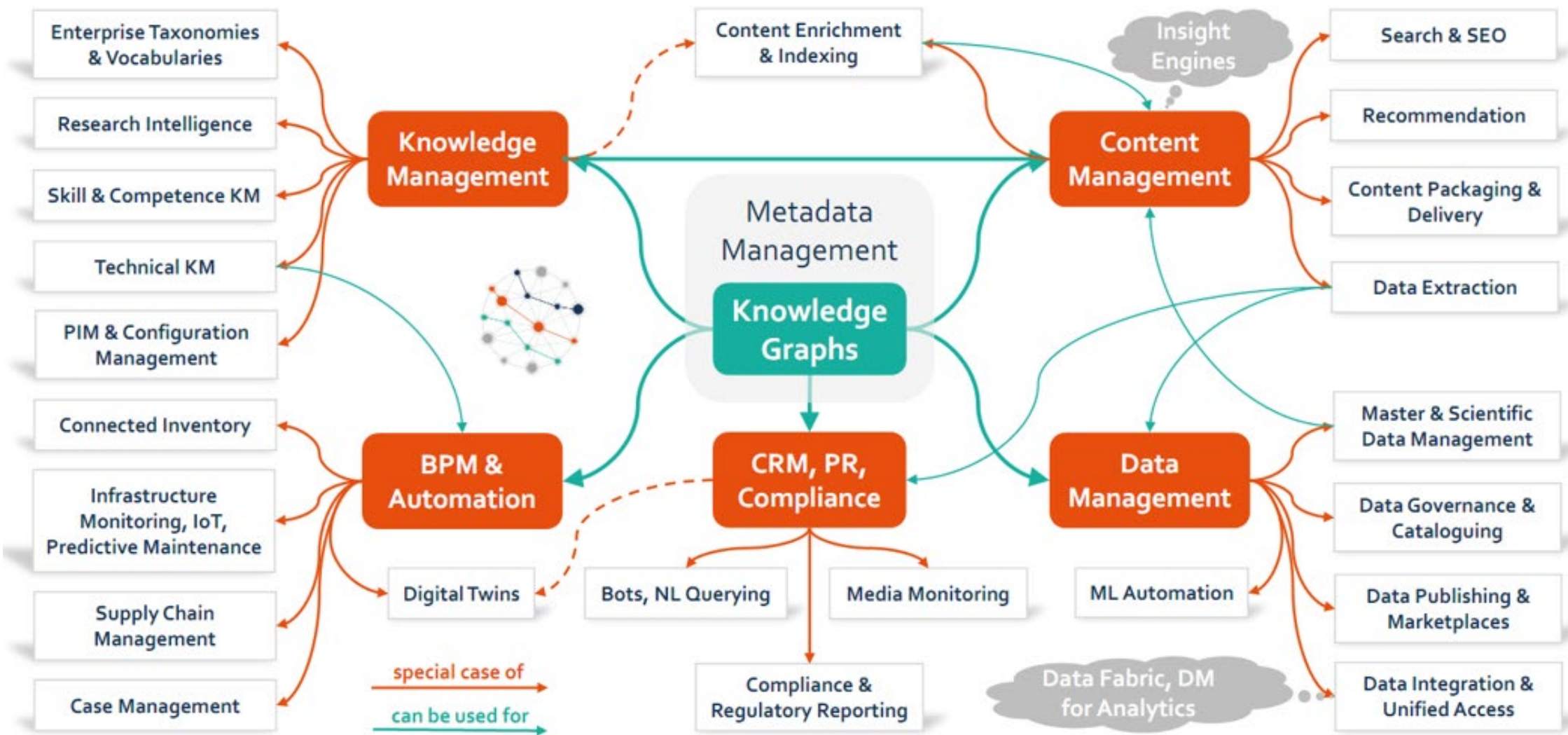
- Diverse databases - unconstrained
- Text documents and other content
- Proprietary & global data

The sweet spot

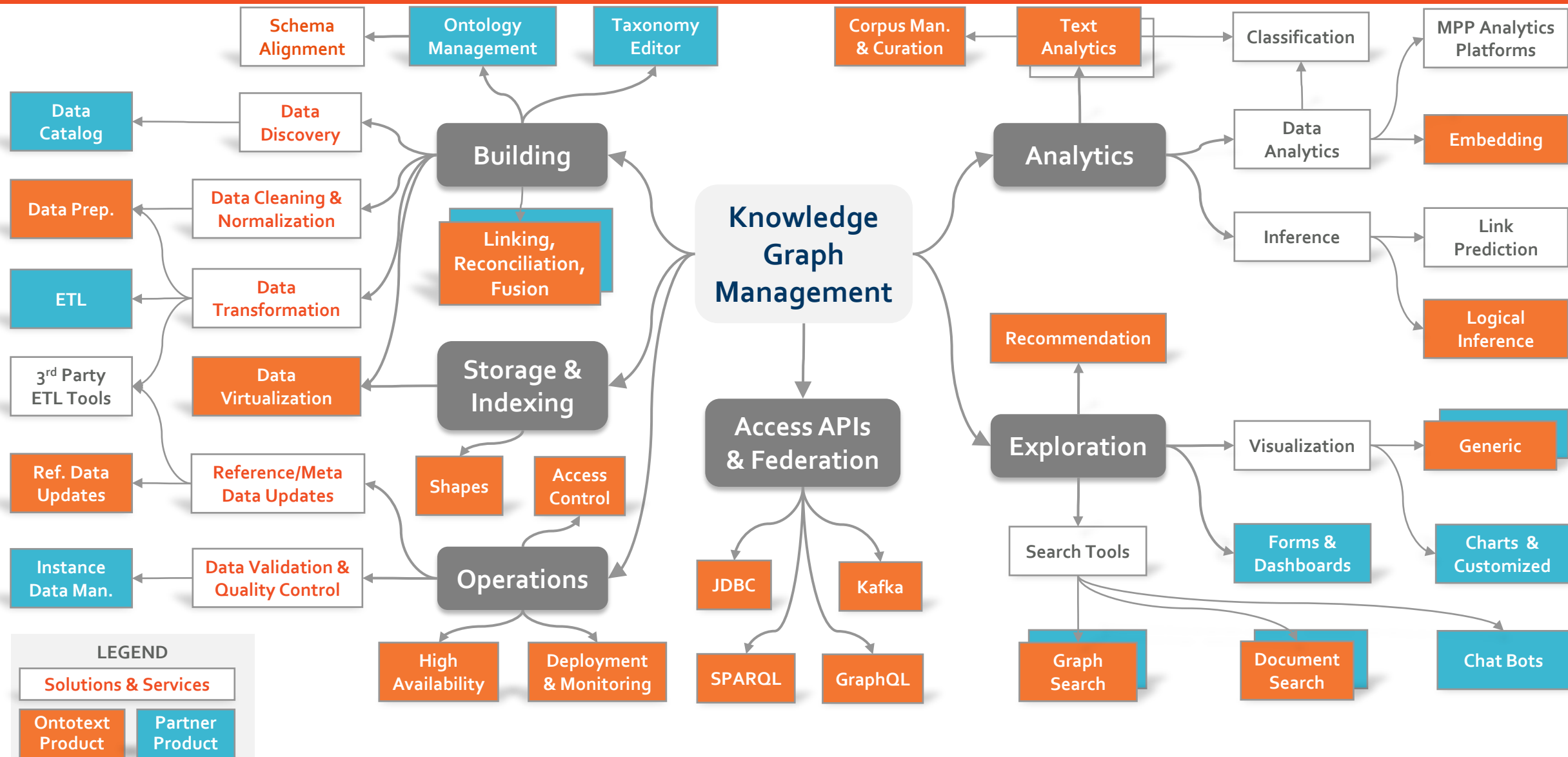
Knowledge intensive domains and applications, which require:

- Large highly interconnected reference data sets
- Very diverse data, distributed architecture need
- Complex relationships, simplified user engagement

Knowledge Graph Application Mindmap



Knowledge Graph Management Capabilities

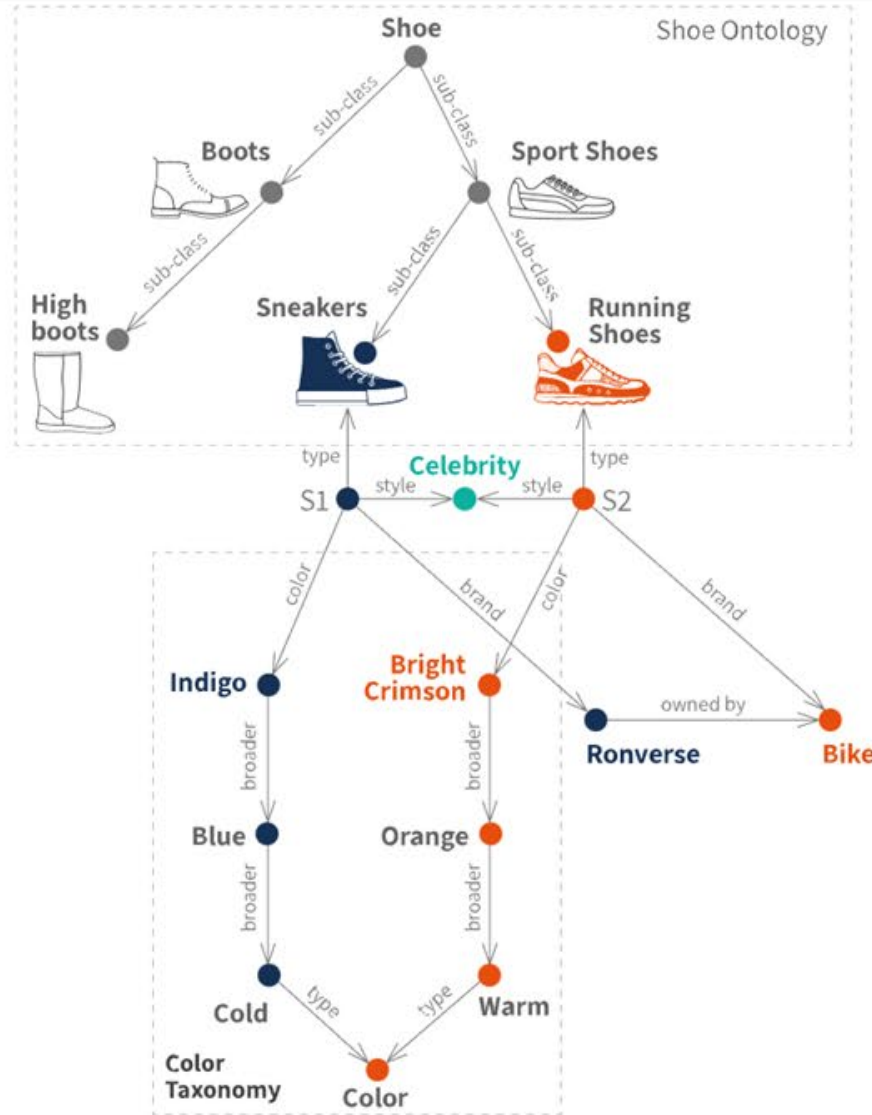


Graphs to Knowledge Graphs

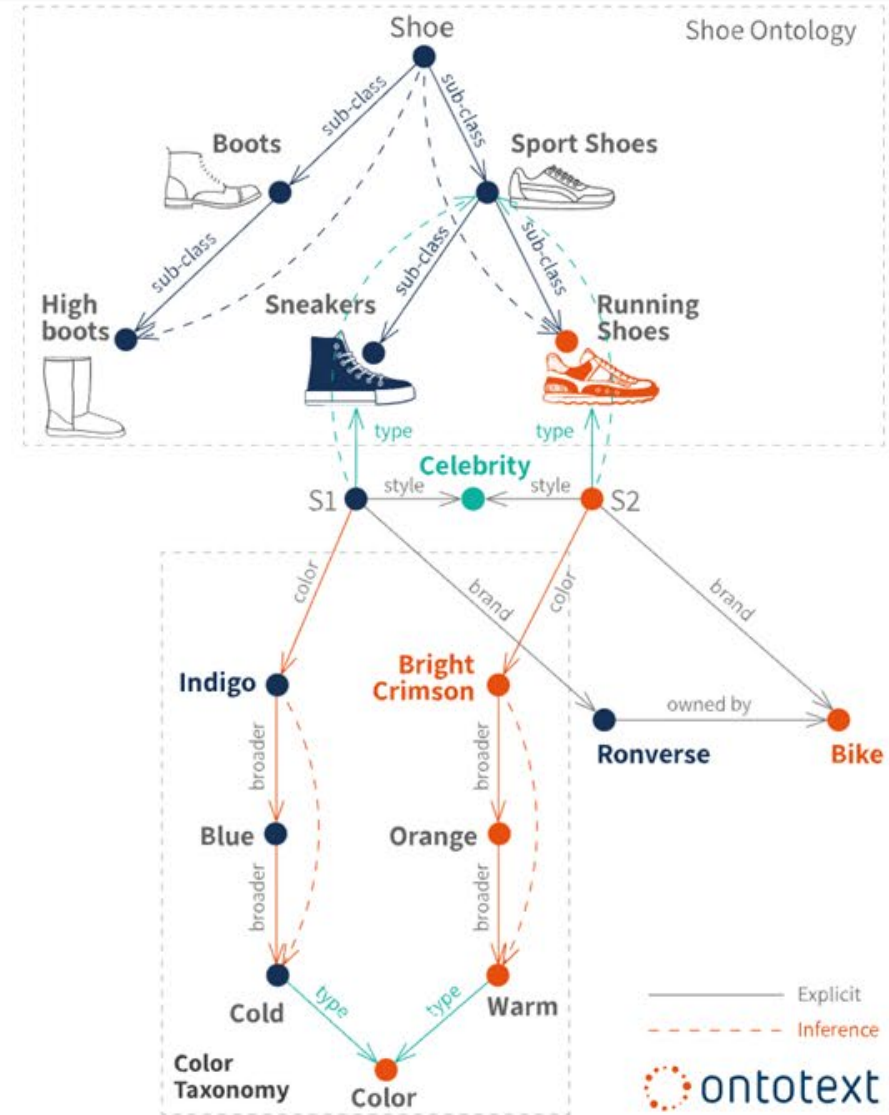
Plain Graph



Knowledge Graph



Knowledge Graph with Inference



- Best way to represent lineage is via a [knowledge graph](#)
- 2 ways to represent lineage right now
 - [OpenLineage standard](#) &
 - Semantic standard [PROV-O](#) (an open standard for describing Provenance in RDF)
- Easy to folding provenance into a KG or other graph-based representations
- Collect information from Systems, Information may be Incomplete or Contradictory.
- Inferencing and Graph Analytics [KGs](#) can sort contradictory information in reliably
- Graphs great for representing evolution of lineage and how it changes over time.

Not Every Graph is a KG

- Not every graph is a KG as the Property Graph vendor advocates allude.
- Graph of a LAN network is not a KG if there is no schema and semantics.
- **KGs are not about searching paths between nodes**
- KGs are about putting data in context via linking and semantic metadata.
- KGs are structured around collection of interlinked descriptions of entities – real-world objects and events, or abstract concepts.
- Descriptions must have formal semantics to allow people & machines to process unambiguously.

What Is a Knowledge Graph



- **Graph Database with a Knowledge Toolkit**
- **Knowledge of a Domain as Graph — Network of Entities & Relationships**
- **Represents Entities, Facts & Models of a Domain**
- **Includes Rich Rules with Inferencing, Reasoning from Relationships**
- **Entity Resolution and Extraction to deal with messy data**

1. **Identity Resolution** (IRI): Unique Address to Concept - Internationalized Resource Identifier
2. **Meaning Resolution** (Ontologies): Data modeling for shared understanding
3. **Triple Expression** (RDF, OWL): Columnar → Semantic Structure – Subjects, Objects linked by predicates. Ensures Concepts are Defined and Understood at granular level.
4. **Business Rules** (SHACL): Conditional Expressions For Criteria. Rules linked to ontologies ensure meaning is shared not obscured.

Semantic Foundational Capabilities



Quality : Linking with ontology ensures Precision of Meaning - Concepts, Systems, People & processes

Concept Reuse: Using standards Eliminates Assumptions. Enables reuse of concepts

Context: Semantic standards allow Separation of Business Logic from Code

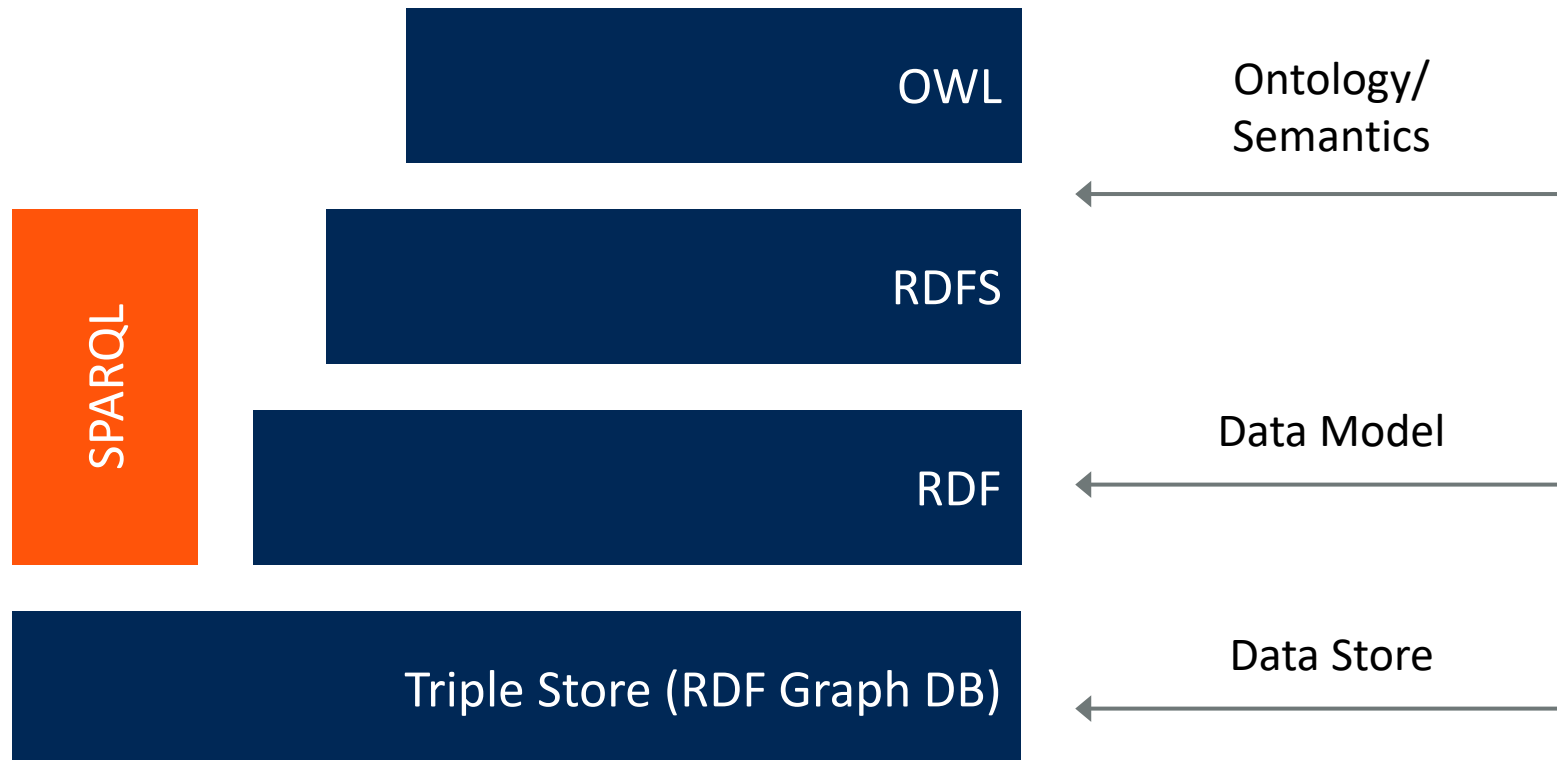
Lineage Traceability: Data is linked to Identifier - Enabling Tracing, Automates Lineage & Provenance

Governance: Standards & Identity govern meaning, Structural Validation

Machine-Readable: Standards written in language for humans & machines. Automatic validation & DQ

Automated Continuous Testing: With standards Change is Linked to a testing process

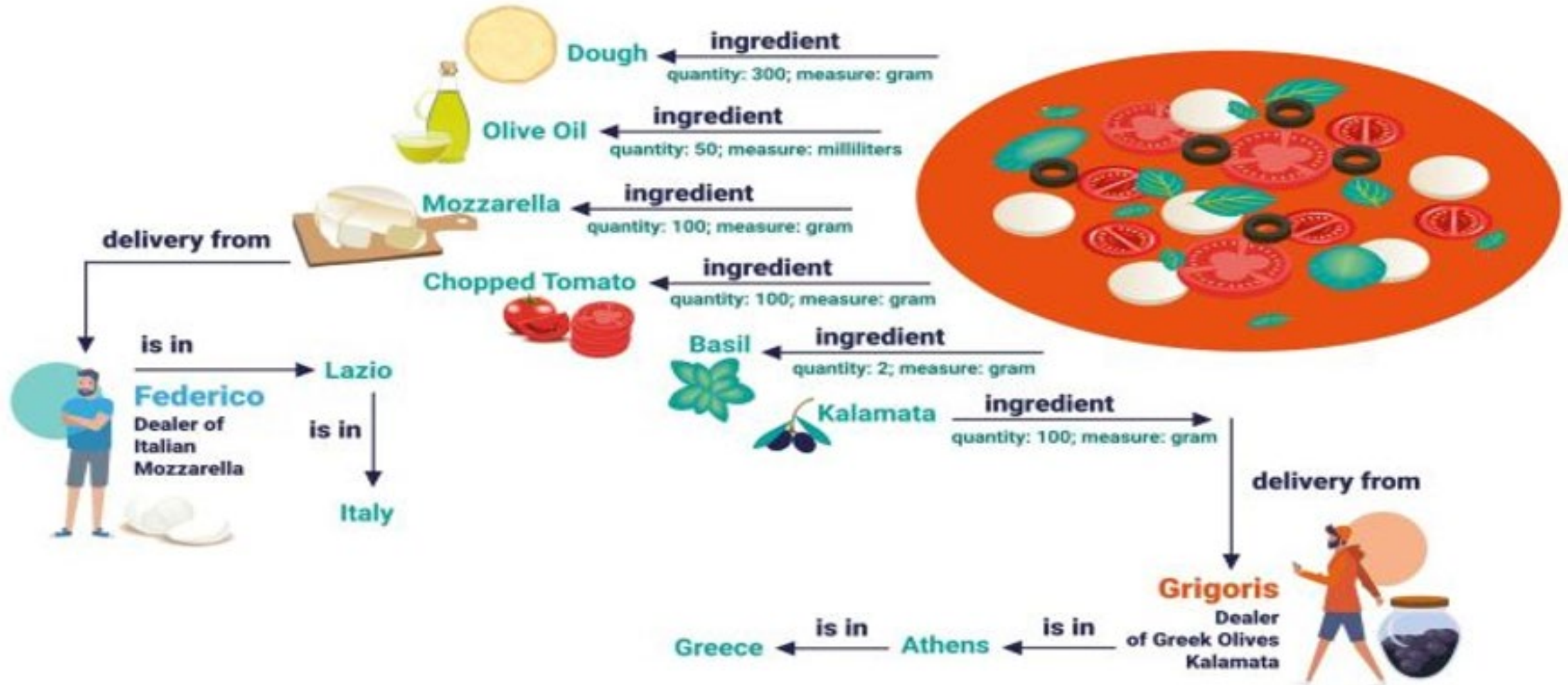
- **Machine-interpretable - Standards allows Shared Understanding, Use/ Reuse**
- **Relations as First-class citizens**
- **Follows Open Standards**



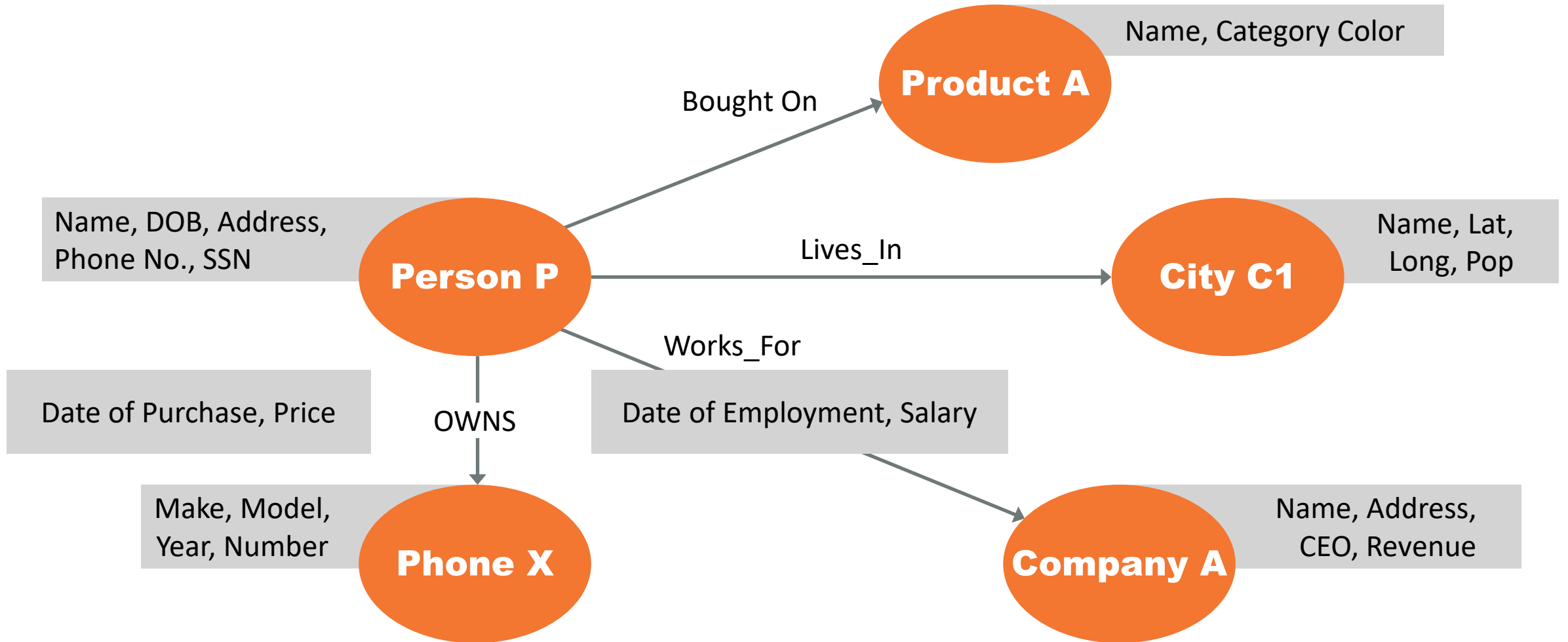
Graph Models – LPG, RDF

DMZ, 1st March 2023

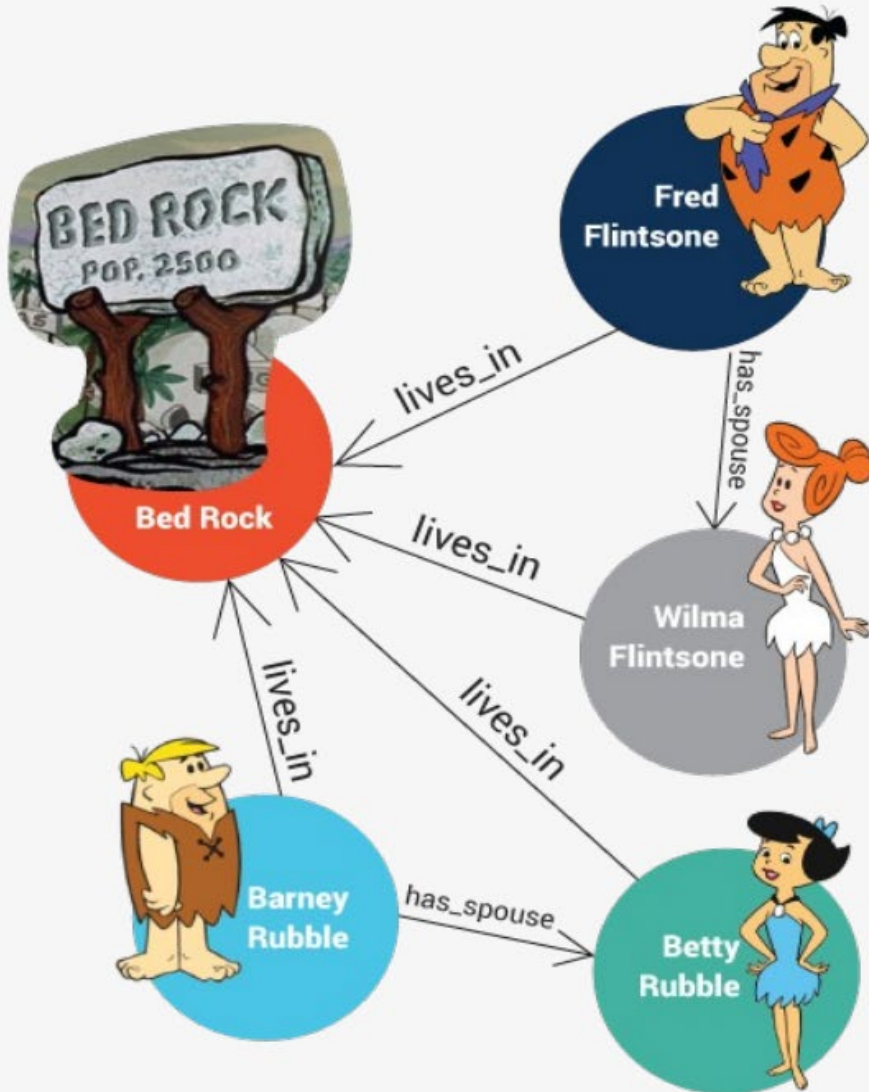
LPG Example



RDF Example



GraphDB: RDF Graphs



Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications, originally designed as a metadata data model

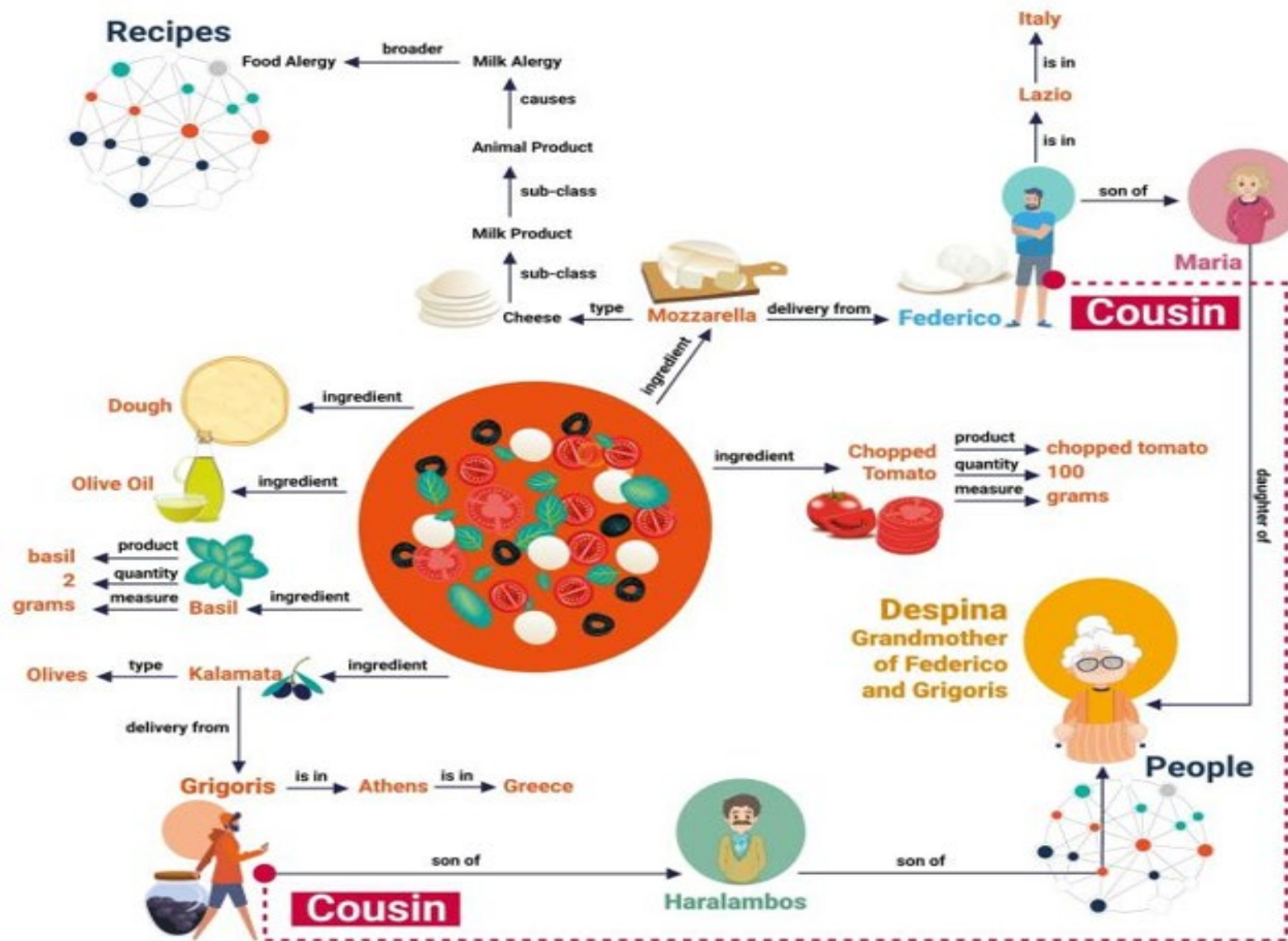
For example, just like there's lingua franca for representing documents on the Web and that is the Hypertext Markup Language (HTML), RDF is a common format for data to be represented and shared.

Subject

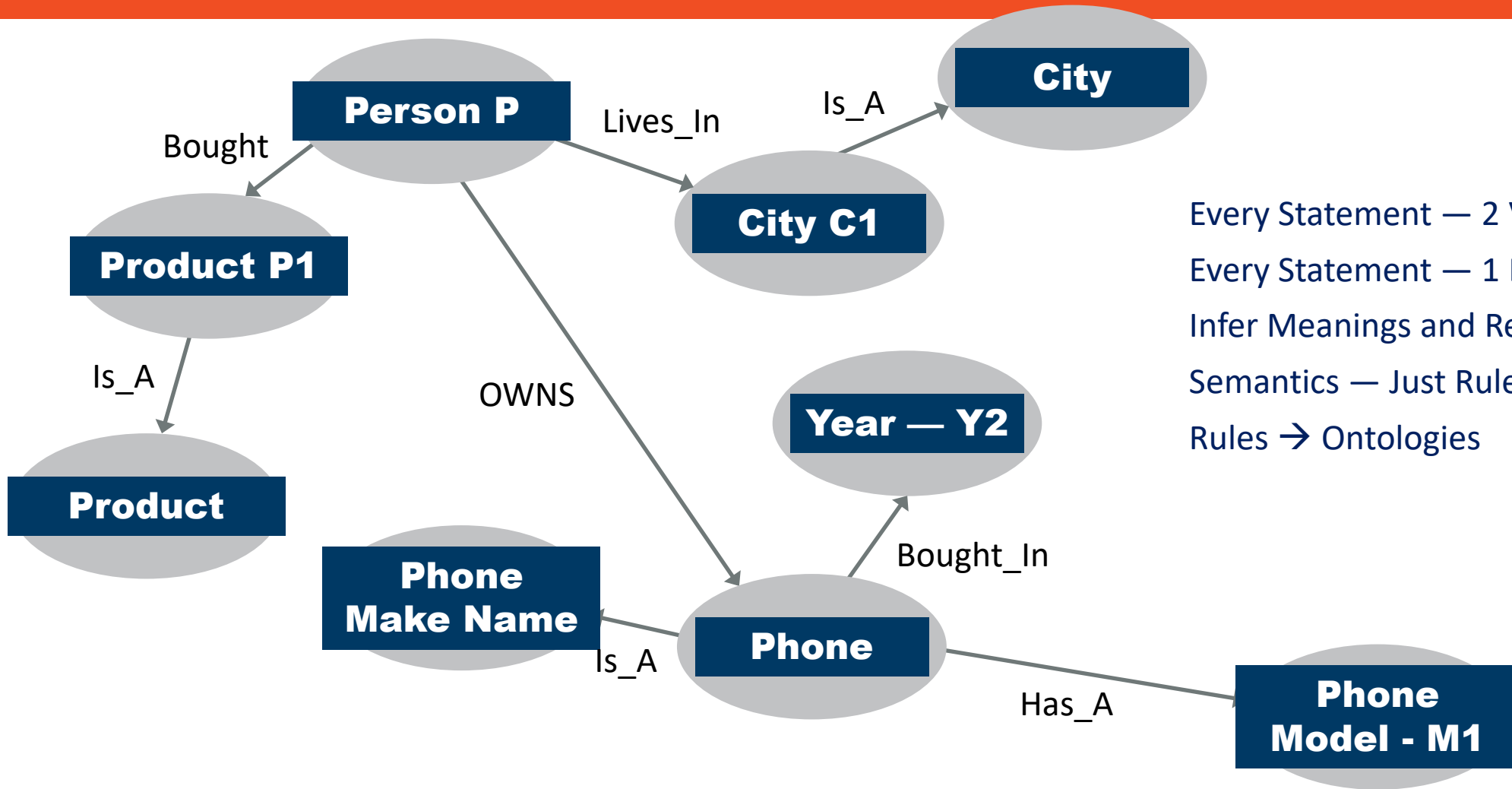
{Predicate}

Object

RDF Example



RDF Example



Every Statement — 2 Vertices

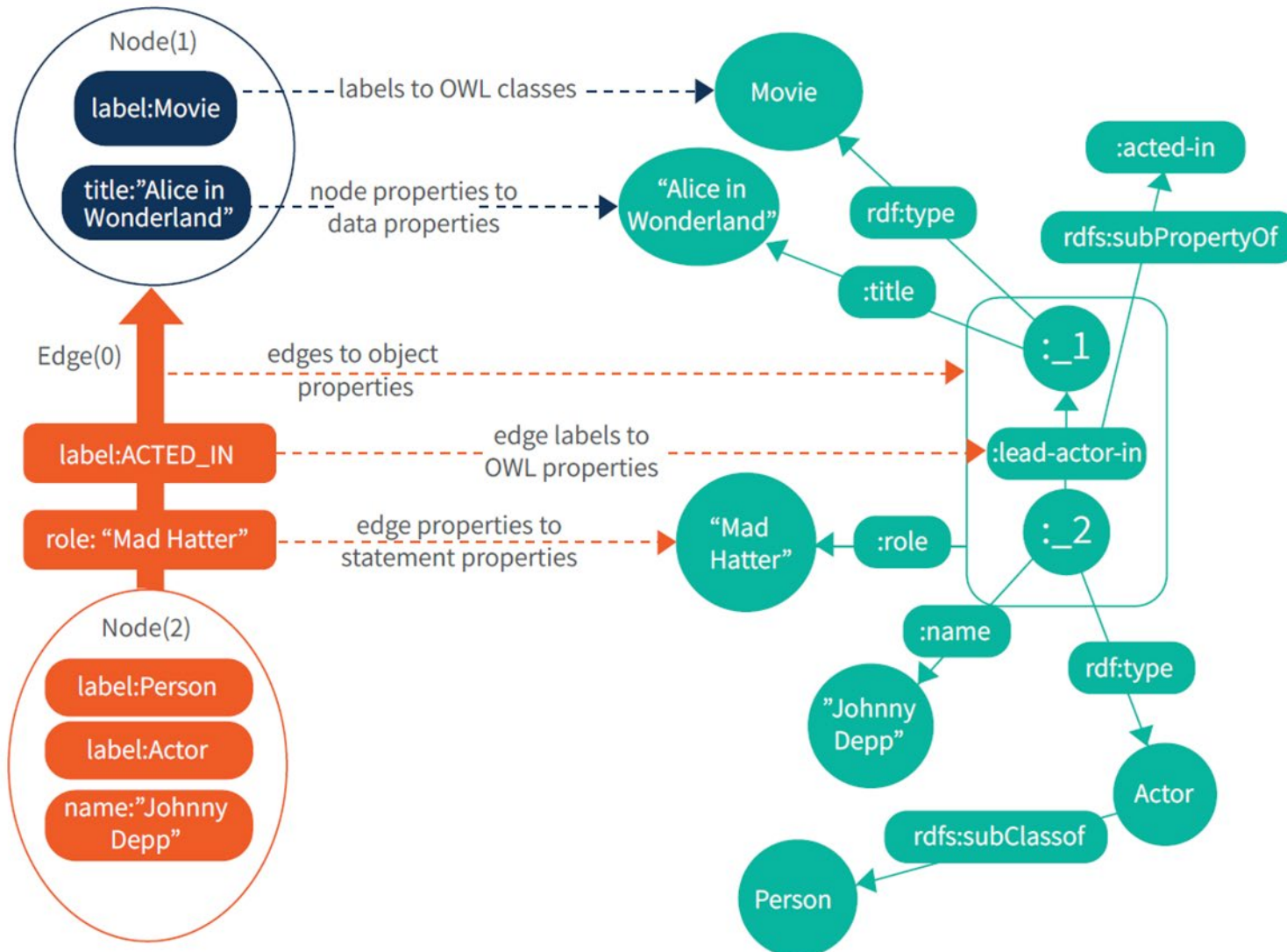
Every Statement — 1 Edge

Infer Meanings and Relationships

Semantics — Just Rules

Rules → Ontologies

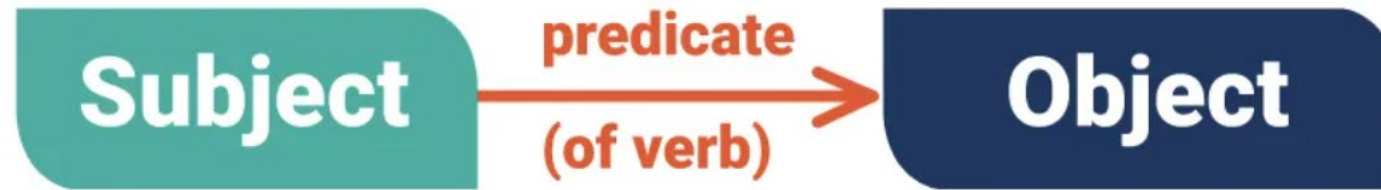
RDF and LPG: Graphs of Different Flavor



LPGs are easy to transform into RDF

The advantages of RDF's finer grained model:

- Schema and data can be queried together
- Property values are nodes and can be described
- Full control over the structure of the metadata

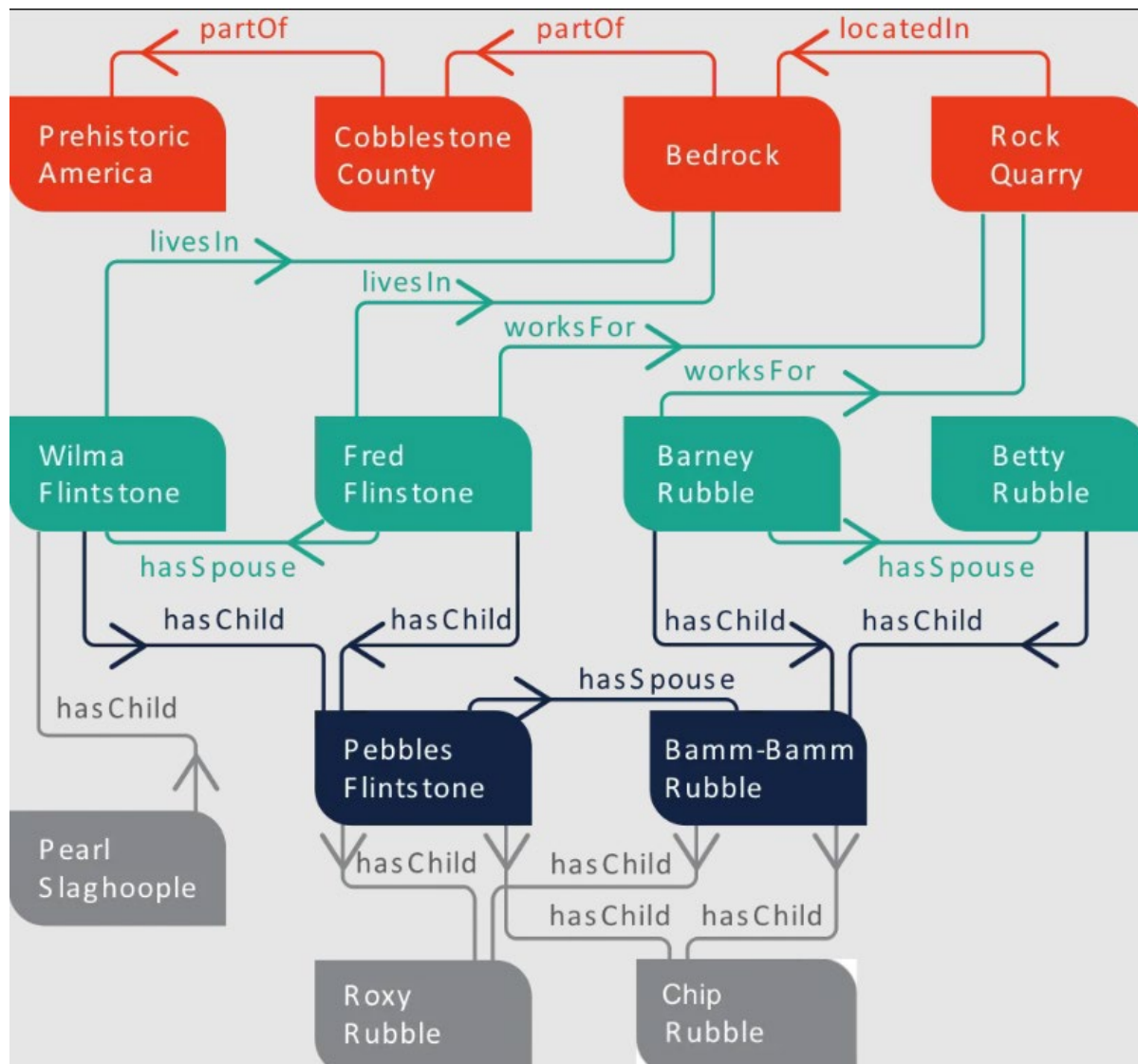


Three-part structures called triples.

Each triple component must have a unique identifier known as the Uniform Resource Identifier (URI), which looks like a web page address. (Except Object)

subject	predicate	object
:Wilma	:hasSpouse	:Fred
:Wilma	:hasAge	24

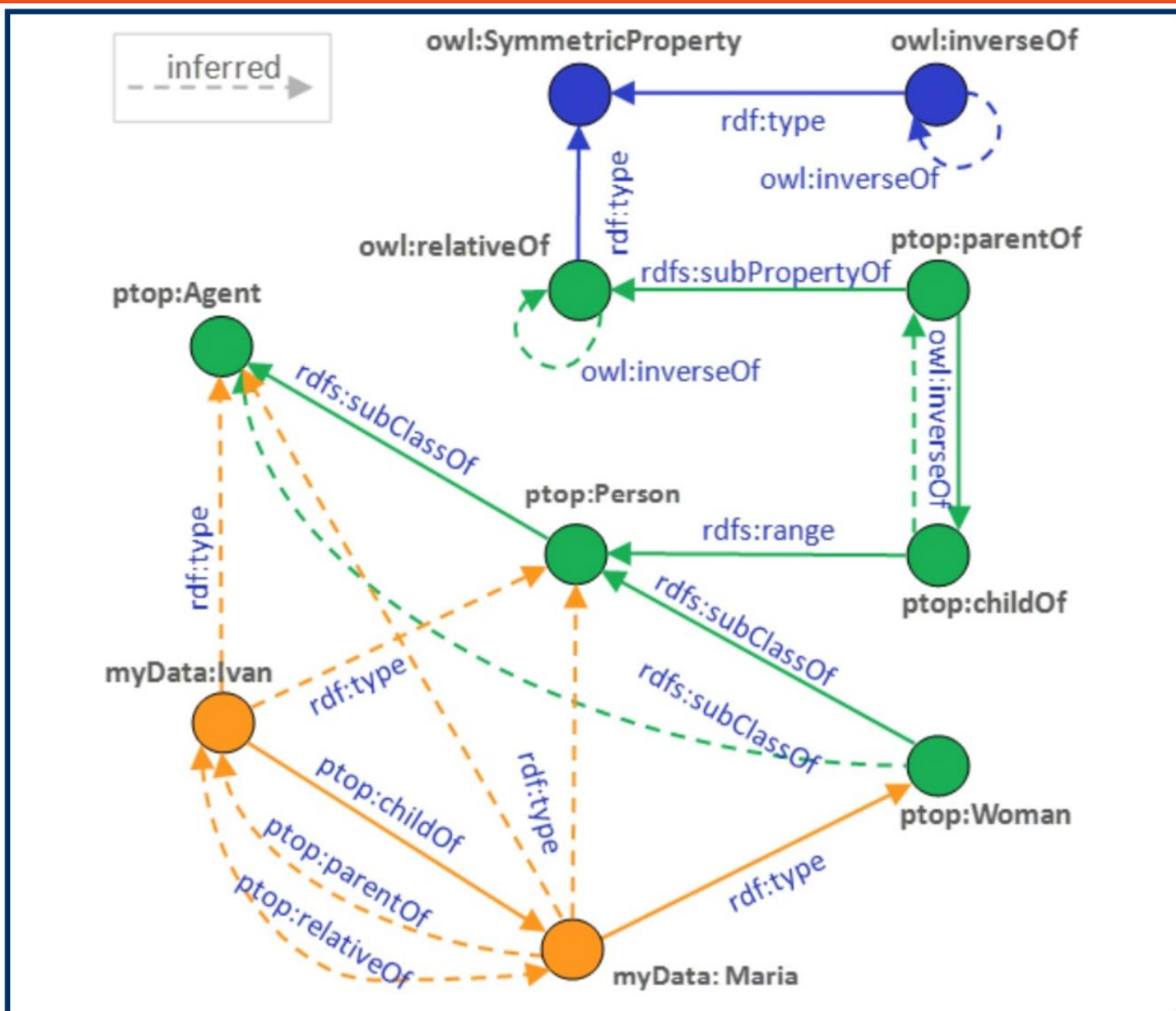
RDF Example



RDF Example

S t a t e m e n t		
Subject	Predicate	Object
myo:Person	rdf:type	rdfs:Class
myo:gender	rdf:type	rdf:Property
myo:parent	rdfs:range	myo:Person
myo:spouse	rdfs:range	myo:Person
myd:Maria	rdf:type	myo:Person
myd:Maria	rdfs:label	"Maria P."
myd:Maria	myo:gender	"F"
myd:Ivan	rdfs:label	"Ivan Jr."
myd:Ivan	myo:gender	"M"
myd:Maria	myo:parent	myd:Ivan
myd:Maria	myo:spouse	myd:John
...		

RDF Example



LPGs Vs RDFs

RDF

(Resource Description Framework, Triple Store)

Optimized for:

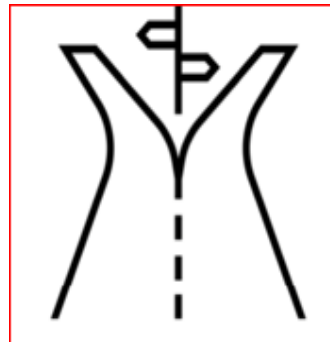
- ✓ Semantic Knowledge Graphs
- ✓ Complex Ontologies & Taxonomies
- ✓ Inferencing / Reasoning
- ✓ Standards Based
- ✓ Linked Open Data
- ✓ Data Re-usability
- ✓ Machine Readable
- ✓ Semantic Web Standards

LPG

(Labeled Property Graph)

Optimized for:

- ✓ Index-Free Adjacency
- ✓ Very Deep Traversals
- ✓ Graph Data Science
- ✓ ML Integration
- ✓ Single Purpose Graphs
- ✓ High Concurrency
- ✓ Ease of Deployment



Graph Technologies

LPGs Vs RDFs



	RDF and Knowledge Graphs	Property Graph
Designed for	Semantic Web / Linked Open Data: Publishing and interlinking data with formal semantics and no central control	Graph analytics
Most appropriate applications	Metadata Management Knowledge Management Data Linking & Unification Master/reference data management Data Fabric	Graph analytics ...
Champions	Enterprise Data Architects Knowledge Managers / Taxonomists	Data engineers Data scientists

LPGs Vs RDFs



	RDF	Property Graph
Data Management Strengths	Data interoperability via: <ul style="list-style-type: none">• global identifiers• standard schema language Data validation	Index-free adjacency Compact serialization, shorter learning curve, Functional graph traversal language (Gremlin)
Formal semantics	Yes, standard schema and model semantics foster data reuse and inference	No formal model representation
Standardization	Driven by W3C standardization processes	OpenCypher
Query languages	SPARQL specifications: Query Language, Updates, Federation, Protocol (end-point)...	Cypher, PGQL, GCore, GQL
Serialization format	Multiple standard serialization formats	No serialization format
Schema language	RDFS, OWL, Shapes	None

Advantages of LPG Over RDF are gone



Historical advantages of LPG:

- Attaching properties to the edges of a graph
- Efficient graph traversal

RDF completely leveled those over the last 3+ years:

- **RDF-star** – simple mechanism to attach metadata to the edges
- SPARQL extensions for **exploration of multi-hop relationships in graphs**
 - ✓ In 2022 GraphDB passed LDBC Social Network Benchmark – neo4j designed it to demonstrate its advantage in graph traversal and analytics. but never published results from it.

- **RDF(S), OWL, SPARQL and SHACL is a stack of W3C semantic standards**
 - ✓ Global identity => Interoperability
 - ✓ Formal semantics => Explicit Common Meaning
 - ✓ Standardization => Unification
 - ✓ Validation => Quality
- **Property Graphs are Designed for Graph Analytics**
 - ✓ GREMLIN and Cypher are good for searching path in a graph, but
 - ✓ PGs lack of standardization, formal semantics and even schema language

Fundamentals – RDF Modeling

DMZ, 1st March 2023

Standard data model for graph data structures

Describing resources on the web or elsewhere

Resource → any entity that we want to describe

Store in “Serializations” such as Turtle, JSON-LD, N-Triples, and RDF/XML.

Any tool conforms to RDF standards can read all formats

Standard maintained by W3C

RDF-related standards include Turtle, JSON-LD, N-Triples, and RDF/XML

Three-part statements known as **triples**.

RDF data is a set of triples.

Triple states certain resource, for a certain property, has a certain value.

Call these three parts the **subject**, **predicate** and **object**.

For example: *cust239* name "Adam Lee"

This is a vague

Term "name" mean different things in different contexts, and many different companies may have a customer 239.

RDF requires subject and predicate of a triple have URIs to unambiguously identify what they're referencing.

URI – Uniform Resource Identifier



Look like URLs, but they're not required to be locators

Describe *where* something is; they may just serve as identifiers.

IRI, or Internationalized Resource Identifier - URIs allow a wider choice of characters than the URI

URIs come from known, standardized vocabularies, or your organization make up your own

RDF Schema - Define vocabularies from Scratch OR Extension of Existing ones

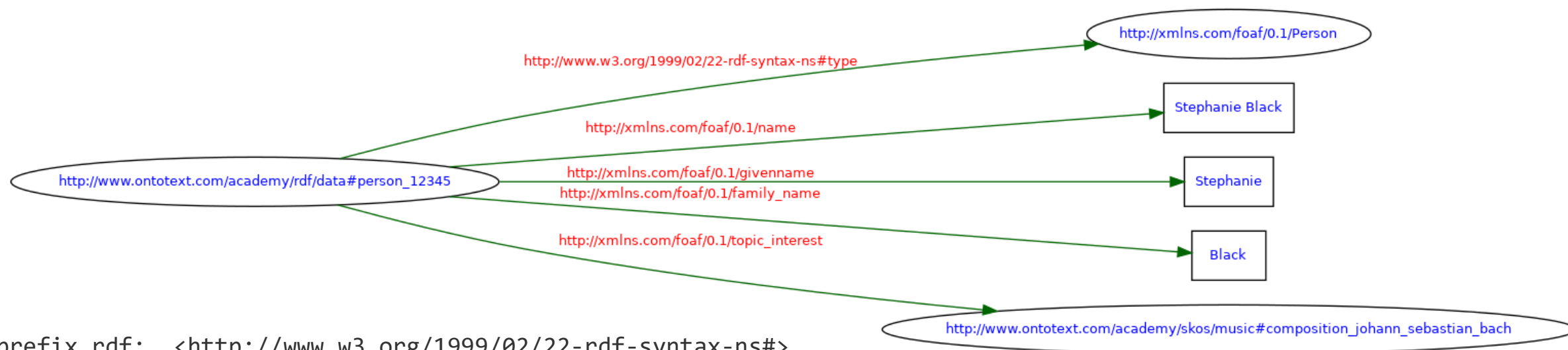
Organizations use combination of standardized vocabularies and custom ones

Example : Predicate uses the mbox property from the standard FOAF vocabulary to show email address.

```
<http://www.altostrat.com/ns/cust239> <http://xmlns.com/foaf/0.1/mbox>  
"adam.lee@cymbalgroupp.com" .
```

Advantage of URIs data can more easily interoperate and integrate with data from other sources

Using URIs to represent things makes easier to connect Our data with Other data



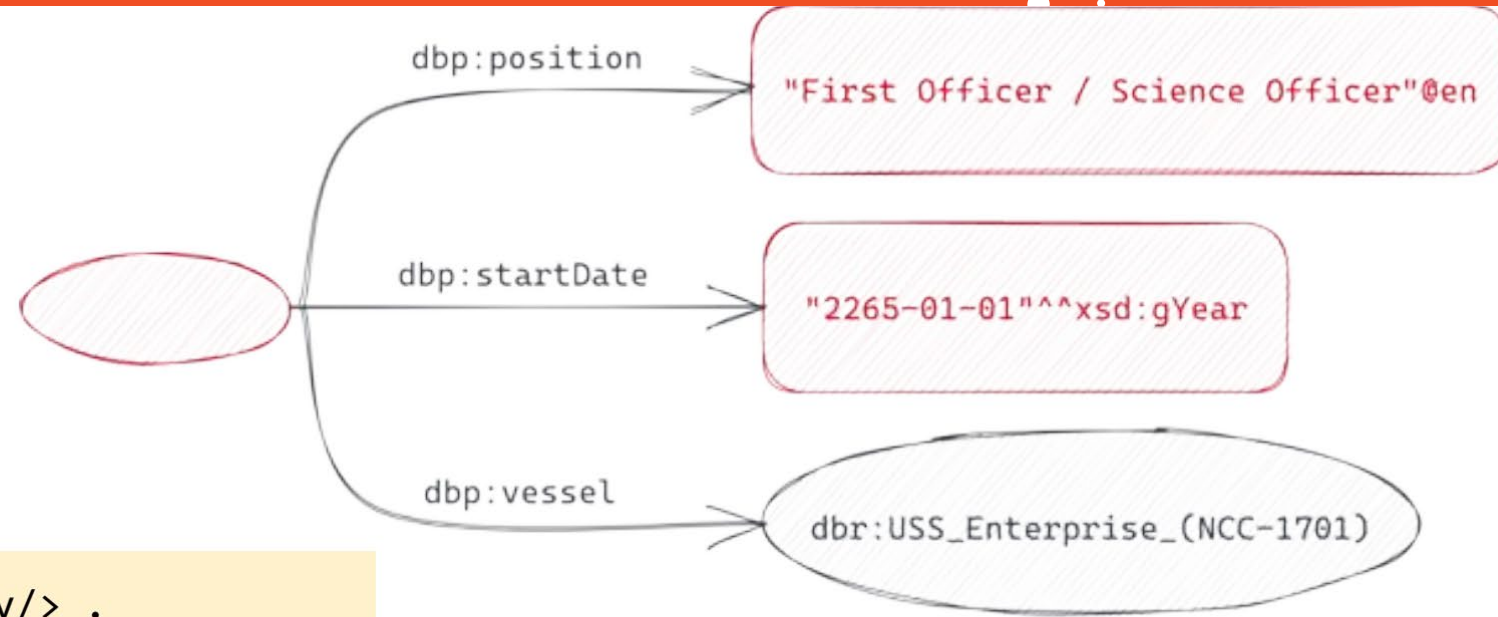
```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix d: <http://www.ontotext.com/academy/rdf/data#> .
```

```
<d:person_12345>
```

```
  rdf:type      foaf:Person ;
  foaf:name     "Stephanie Black" ;
  foaf:givenname "Stephanie" ;
  foaf:family_name "Black" ;
  foaf:topic_interest
```

```
<http://www.ontotext.com/academy/skos/music#composition_johann_sebastian_bach> .
```

Blank Node



```
@prefix dbp: <http://dbpedia.org/ontology/> .  
@prefix dbr: <http://dbpedia.org/resource/> .  
@prefix xsd: <http://www.w3c.org/2001/XMLSchema#> .
```

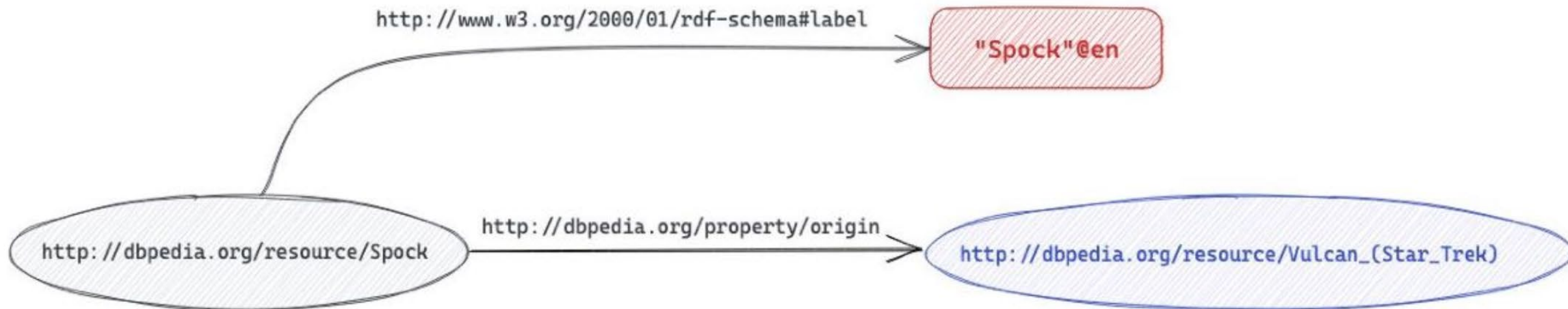
```
[ ] dbp:position "First Officer/ Science Officer"@en ;  
    dbp:startDate "2265-01-01"^^xsd:gYear ;  
    dbp:vessel    dbr:USS_Enterprise_(NCC-1701) .
```

anonymous blank node as subject

Semicolon in a Triple

```
@prefix dbp: <http://dbpedia.org/property/> .  
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
@base <http://dbpedia.org/resource/> .  
  
<Spock> rdfs:label "Spock"@en ;  
        dbp:origin <Vulcan_(Star_Trek)> .
```

semicolon indicates that subsequent
triples have the same subject
(predicate list)

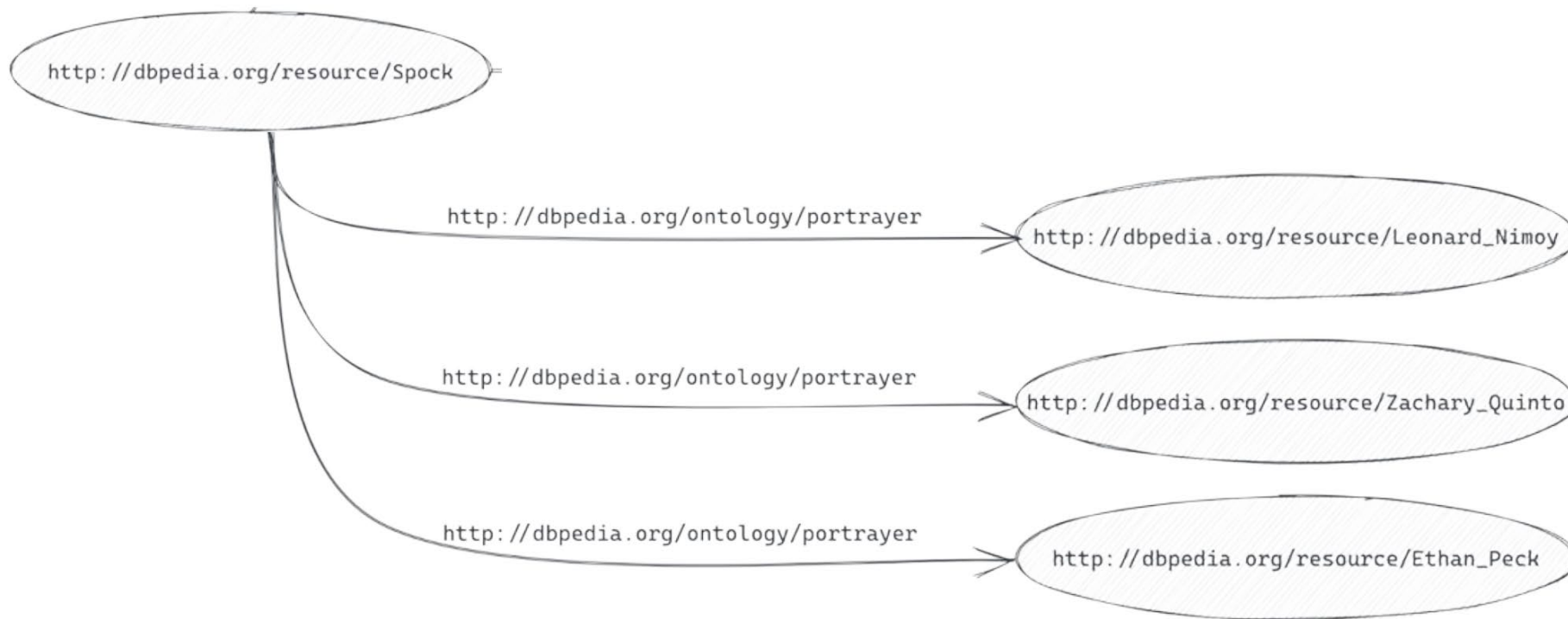


Comma in a Triple

```
@prefix dbo: <http://dbpedia.org/ontology/> .  
@base <http://dbpedia.org/resource/> .
```

```
<Spock> dbo:portrayer <Zachary_Quinto> ,  
                <Leonard_Nimoy> ,  
                <Ethan_Peck> .
```

comma indicates that subsequent
triples have same subject and
property (**object list**)



Link between Nodes



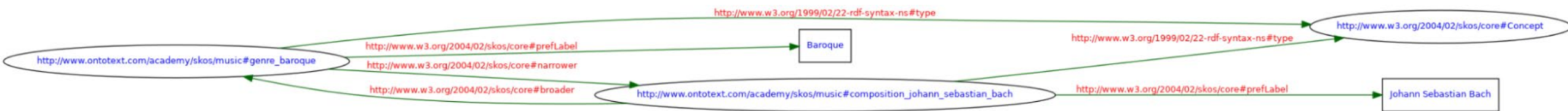
Property representing topic of interest points to the external node `composition_johann_sebastian_bach`.
This node described with the following URI identifiers and additional properties as name-value pairs:

URI	<code>http://www.ontotext.com/academy/skos/music#composition_johann_sebastian_bach</code>
Preferred label	"Johann Sebastian Bach"
Broader concept	<code>http://www.ontotext.com/academy/skos/music#genre_baroque</code>

broader concept is pointing to resource "genre_baroque" which could be described as such

URI	<code>http://www.ontotext.com/academy/skos/music#genre_baroque</code>
Preferred label	"Baroque"
Narrower concept	<code>http://www.ontotext.com/academy/skos/music#composition_johann_sebastian_bach</code>

Link between Nodes



```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
```

```
<http://www.ontotext.com/academy/skos/music#genre_baroque>
```

```
  rdf:type      skos:Concept ;
```

```
  skos:prefLabel "Baroque" ;
```

```
  skos:narrower  <http://www.ontotext.com/academy/skos/music#composition_johann_sebastian_bach> .
```

```
<http://www.ontotext.com/academy/skos/music#composition_johann_sebastian_bach>
```

```
  rdf:type      skos:Concept ;
```

```
  skos:prefLabel "Johann Sebastian Bach" ;
```

```
  skos:broader   <http://www.ontotext.com/academy/skos/music#genre_baroque> .
```

Link between Nodes



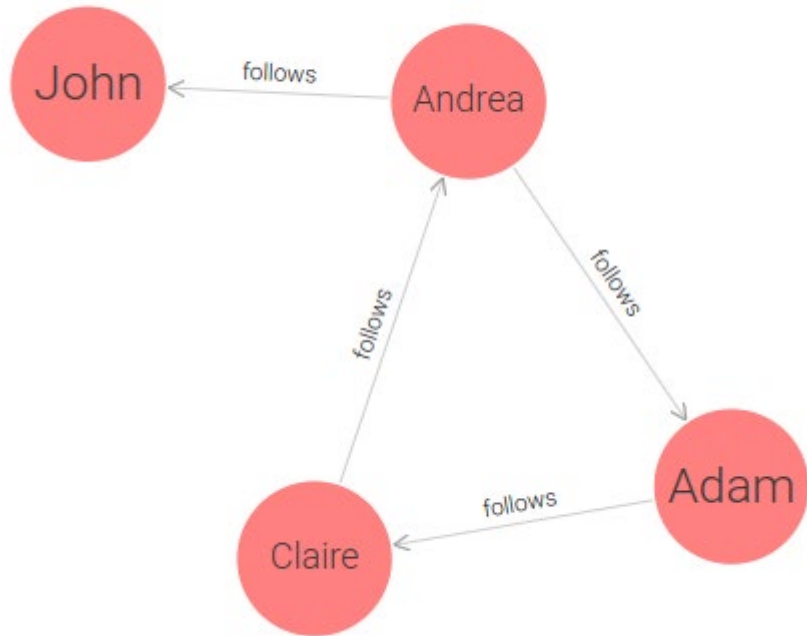
Combining different RDF datasets is easy.

Combine examples 2 and 3, we can say Stephanie Black (person_12345) has an interest (foaf:topic_interest) in Johann Sebastian Bach (composition_johann_sebastian_bach) Which is related to the Baroque genre (genre_baroque).

Navigate graph reverse way by locating genre_baroque and navigate the graph in the reverse direction to find all people interested in this style of music.



Example



Turtle uses the **# character to comment** out the rest of a given line.

```
@prefix as: <http://www.altostrat.com/ns/> .
```

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
@prefix schema: <https://schema.org/> .
```

```
as:cust239 foaf:mbox "adam.lee@cymbalgroupp.com" ;  
foaf:givenname "Adam" ;  
foaf:family_name "Lee" .
```

```
as:cust725 foaf:givenname "Claire" ;  
foaf:family_name "Graham" ;  
schema:follows as:cust016 .
```

```
as:cust016 foaf:givenname "Andrea" ;  
foaf:family_name "Gray" ;  
schema:follows as:cust239 ;  
schema:follows as:cust644 .
```

```
as:cust644 foaf:givenname "John" ;  
foaf:family_name "Bell" .
```

```
as:cust239 schema:follows as:cust725 .
```

Give more context about the data

- List the Classes of Entities
- Relationships between Classes
- Properties

Describe semantics about the classes, properties, and their relationships

W3C RDF Schema standard is sometimes called RDFS

First triple uses `rdf` and `rdfs` vocabularies to declare a class of resources called *as:Person*

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

```
as:Person rdf:type rdfs:Class .
```

```
as:Person a rdfs:Class .
```

```
as:Customer rdfs:subClassOf as:Person .
```

```
as:cust644 a as:Customer .
```

Nothing to prevent *as:cust644* from being an instance of other related or unrelated classes such as *foaf:Person* or *as:Teacher*

Properties



`rdfs:domain` means it is a member of the class named in that triple's object.

Means any resource that has `as:familyName` or `as:givenName` property is an instance of the `as:Person` class

Doesn't say it's required for an `as:Person` instance to have an `as:familyName` value

Properties exist independently of classes

Unintuitive to people accustomed to OOAD

Adds flexibility when building model with external data whose original model may not be documented a common scenario in data integration projects.

Subproperties of properties

```
as:givenName rdfs:subPropertyOf rdf:label .  
as:familyName rdfs:subPropertyOf rdf:label .
```

`rdfs:subClassOf` and *`rdfs:subPropertyOf`* → data's structure its relationship to other known schemas

```
as:familyName a rdf:Property .  
as:givenName a rdf:Property .  
as:memberSince a rdf:Property .
```

Schema properties go with certain classes:

```
as:familyName rdfs:domain as:Person .  
as:givenName rdfs:domain as:Person .  
as:memberSince rdfs:domain rdf:Customer .
```


OWL Ontologies



W3C standard

Adds information about Classes & Properties to build a Schema

Richer model of a domain than Class and Property

Describe Simple or Complex Conditions for Class Membership

Life sciences uses this with inference engines to compute which treatments may be candidates to address a new medical condition.

Inference engine that can implement the entire OWL Full

@prefix owl: <http://www.w3.org/2002/07/owl#> .

as:cust239 as:spouse as:cust016 .
as:spouse a owl:SymmetricProperty .

SPARQL Query

```
PREFIX as: <http://www.altostrat.com/ns/>
SELECT ?cust ?custSpouse
WHERE {
    ?cust as:spouse ?custSpouse
}
```

cust	custSpouse
http://www.altostrat.com/ns/cust239	http://www.altostrat.com/ns/cust016
http://www.altostrat.com/ns/cust016	http://www.altostrat.com/ns/cust239

Which Schemas / Ontologies to Use



Use an existing schema or ontology which makes your data interoperable

Creating own from Scratch gives Customizability **but** Reduces Interoperability

Look for an existing schema or ontology that describes the data

Find one that almost but not quite completely meets your needs.

Customizability of RDFS schemas and OWL ontologies becomes very useful.

vCard Ontology describes most of what you need about people - base class of `vcard:Individual`

Way to describe customers

use *vCard* ontology & add own customer class as a subclass of its existing `vcard:Individual` class:

```
as:Customer rdfs:subClassOf vcard:Individual .
```

Declare custom *as:memberSince* & *as:goldStarCustomer* properties use with *as:Customer* instances

Use *vcard:given-name* and *vcard:last-name* Properties with those instances

`rdfs:domain` indicates that `as:familyName` property goes with the `as:Person` class

Doesn't mean it's required for an `as:Person` instance to have an `as:familyName` value

Define constraints and check `as:Person` resources meet this condition

Define constraints by creating "shapes", sets of triples that use properties and classes from the SHACL vocabulary

SHACL – Shape Constraints Language - DQ



Customer data make as:memberSince property a required value.
Value no earlier than January 1, 2020

```
@prefix as: <http://www.altostrat.com/ns/> .  
@prefix sh: <http://www.w3.org/ns/shacl#> .  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

as:CustomerShape

```
  a sh:NodeShape ;  
  sh:targetClass as:Customer ;  
  sh:property as:memberSinceShape .
```

as:memberSinceShape

```
  a sh:PropertyShape ;  
  sh:path as:memberSince ;  
  sh:datatype xsd:date ;  
  sh:minInclusive "2020-01-01"^^xsd:date ;  
  sh:minCount 1 ;  
  sh:maxCount 1 .
```

SHACL - Example



```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
@prefix as: <http://www.altostrat.com/ns/> .
```

```
as:cust644 a as:Customer ;  
    as:memberSince "2023-10-08"^^xsd:date .
```

```
as:cust755 a as:Customer ;  
    as:memberSince "2019-03-14"^^xsd:date .
```

```
as:cust947 a as:Customer .
```

Tool Validate dataset against set of SHACL constraints

Resulting report in RDF

Easier to add validation process to an RDF-based pipeline of data processing

SHACL - Results

```
_ :8f6e49eb7f54453fb89a2d771c8e958346467 a sh:ValidationResult;  
sh:focusNode <http://www.altostrat.com/ns/cust947>;  
rsx:shapesGraph rdf4j:SHACLShapeGraph;  
sh:resultPath <http://www.altostrat.com/ns/memberSince>;  
sh:sourceConstraintComponent sh:MinCountConstraintComponent;  
sh:resultSeverity sh:Violation;  
sh:sourceShape <http://www.altostrat.com/ns/memberSinceShape> .
```

```
_ :8f6e49eb7f54453fb89a2d771c8e958346469 a sh:ValidationResult;  
sh:focusNode <http://www.altostrat.com/ns/cust755>;  
rsx:shapesGraph rdf4j:SHACLShapeGraph;  
sh:value "2019-03-14"^^xsd:date;  
sh:resultPath <http://www.altostrat.com/ns/memberSince>;  
sh:sourceConstraintComponent sh:MinInclusiveConstraintComponent;  
sh:resultSeverity sh:Violation;  
sh:sourceShape <http://www.altostrat.com/ns/memberSinceShape> .
```

SHACL - Results



`http://www.altostrat.com/ns/cust947` node violated `sh:MinCountConstraintComponent` for `memberSince` property.

Had no values for this property and was required to have a minimum of one.

`cust755 as:memberSince` value of `"2019-03-14"` violated the `sh:MinInclusiveConstraintComponent`.

Value was lower than one specified to be the minimum allowable one

Show

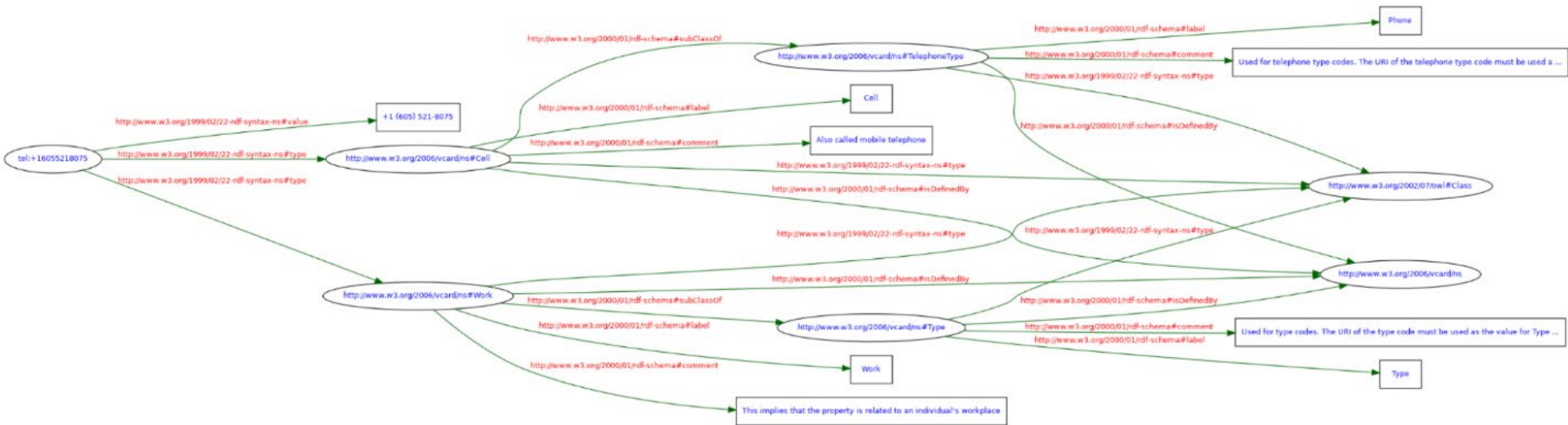
<https://www.w3.org/2006/vcard/ns#>.

Explicit schema reduces ambiguity of data while integrating or sharing

When publishing your data, other systems may recognize schema and do the same mapping on their side.

The data is all linked to a specific schema, it is possible to visualize definition by extending the schema in the graph.

Example



Example – FOAF – Friend of Friend - Ontology



Describes Persons, Activities and their Relations to other People and Objects.

Classes & Properties in the FOAF ontology.

(Classes begin with UPPERCASE letters and Properties with lowercase)

- Basics :**

- Agent, Person, name, nick, title, homepage, mbox, mbox_sha1sum, img, depiction, surname, family_name, givenname, firstName

- Personal Info:**

- weblog, knows, interest, currentProject, pastProject, plan, based_near, workplaceHomepage, workInfoHomepage, schoolHomepage, topic_interest, publications, geekcode, myersBriggs, dnaChecksum

- Projects and Groups:** Project, Organization, Group, member, membershipClass, fundedBy, theme

- Documents and Images:**

- Document, Image, PersonalProfileDocument, topic, primaryTopic, tipjar, sha1, made, thumbnail, logo

- OnlineAccounts :** OnlineAccount, OnlineChatAccount, OnlineEcommerceAccount, OnlineGamingAccount, holdsAccount, accountServiceHomepage, accountName, icqChatID, msnChatID, aimChatID, jabberID, yahooChatID

FOAF - Example



@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<http://www.ontotext.com/academy/rdf/data#person_12345>

a foaf:Person ;

foaf:name "Stephanie Black" ;

foaf:givenname "Stephanie" ;

foaf:family_name "Black" ;

foaf:homepage <http://www.stephanie_black.com> ;

foaf:knows <http://www.ontotext.com/academy/rdf/data#person_54321>;

foaf:topic_interest <http://www.ontotext.com/academy/skos/music#composition_johann_sebastian_bach>,
<http://www.ontotext.com/academy/skos/music#composition_aaron_copland> ;

foaf:currentProject <http://www.ontotext.com/academy/rdf/data#project_12345> .

<http://www.ontotext.com/academy/rdf/data#project_12345>

a foaf:Project ;

foaf:name "Building a classical music catalog" ;

foaf:homepage <https://www.music.com/catalog/> .

<http://www.ontotext.com/academy/rdf/data#group_00001>

a foaf:Group ;

foaf:name "The Classical music lovers" ;

foaf:member <http://www.ontotext.com/academy/rdf/data#person_12345> ,

<http://www.ontotext.com/academy/rdf/data#person_54321> .

Vocabulary describing Classes / Properties -- building blocks to create a schema

Define Classes, Properties, Relationships

Classes:

rdfs:Resource, rdfs:Class, rdfs:Literal, rdfs:Datatype, rdf:langString, rdf:HTML, rdf:XMLLiteral, rdf:JSON, rdf:Property

Properties :

rdfs:range, rdfs:domain, rdf:type, rdfs:subClassOf, rdfs:subPropertyOf, rdfs:label, rdfs:comment

Declaring property uses triple that names something as an instance of the **rdfs:Property** Class

```
@prefix acad: <http://www.ontotext.com/academy/rdf/data#> .
```

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

```
acad:serviceProvider a rdfs:Property .
```

Describes properties separately, with optional information about the resource class to which they apply.

RDFS also lets you define a property as a **subproperty** of another one.

W3C Recommendation

Representation of Thesauri, Classification Schemes, Taxonomies, Subject-heading systems

Express a Parent-Child relationship between resources

Concepts: Concept, ConceptScheme, inScheme, hasTopConcept, topConceptOf
Labels & Notation: prefLabel, altLabel, hiddenLabel, notation

Semantic Relations: broader, narrower, related, broaderTransitive, narrowerTransitive, semanticRelation

Mapping Properties: broadMatch, narrowMatch, relatedMatch, closeMatch

Collections: Collection, orderedCollection, member, memberList

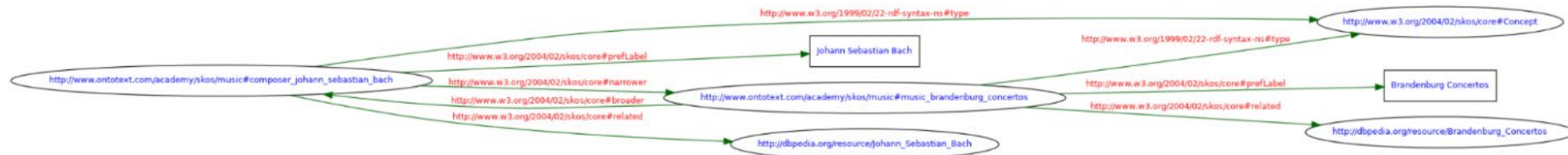
Documentation: note, changeNote, definition, editorialNote, example, historyNote, scopeNote

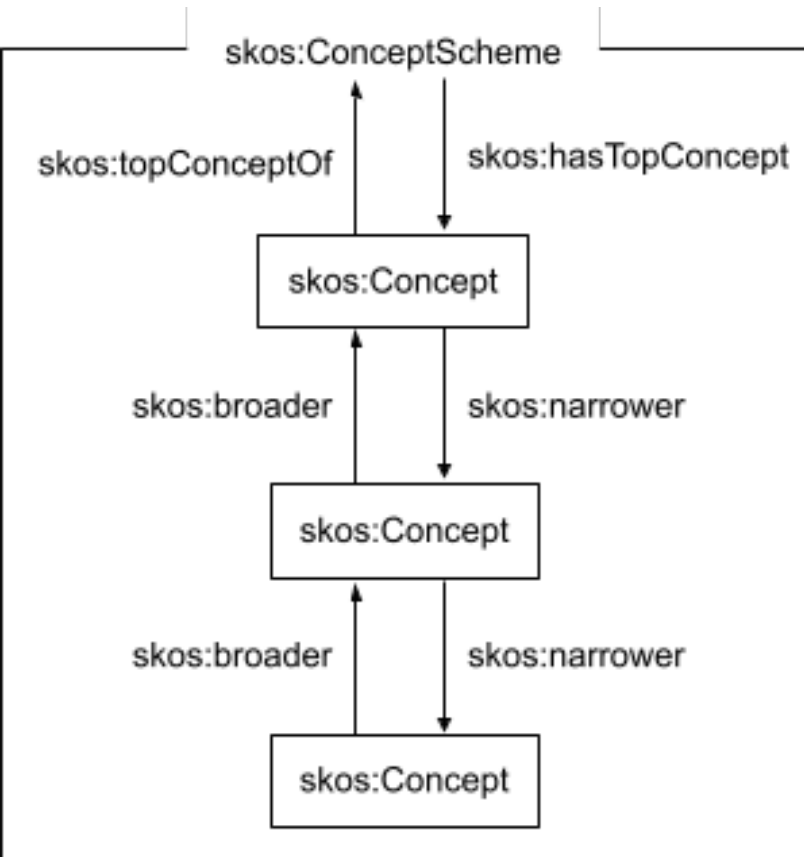
State "Brandenburg Concertos" is a **narrower concept** of "Johann Sebastian Bach" compositions.

Opposite relation by saying that "Johann Sebastian Bach" composition is a **broader concept** of "Brandenburg Concertos".

Property *skos:related* - Relationship with another Concept where no hierarchy or generality relation is implied.

Describing relationships between concepts in one part of a hierarchy with concepts in another part, or even in a different hierarchy.





@prefix skos: <<http://www.w3.org/2004/02/skos/core#>> .

```

<http://www.ontotext.com/academy/skos/music#composition\_johann\_sebastian\_bach>
  a skos:Concept ;
  skos:prefLabel "Johann Sebastian Bach" ;
  skos:narrower
    <http://www.ontotext.com/academy/skos/music#music\_brandenburg\_concertos> ;
  skos:related <http://dbpedia.org/resource/Johann\_Sebastian\_Bach> .
  
```

```

<http://www.ontotext.com/academy/skos/music#music\_brandenburg\_concertos>
  a skos:Concept ;
  skos:prefLabel "Brandenburg Concertos" ;
  skos:broader
    <http://www.ontotext.com/academy/skos/music#composition\_johann\_sebastian\_bach> ;
  skos:related <http://dbpedia.org/resource/Brandenburg\_Concertos> .
  
```

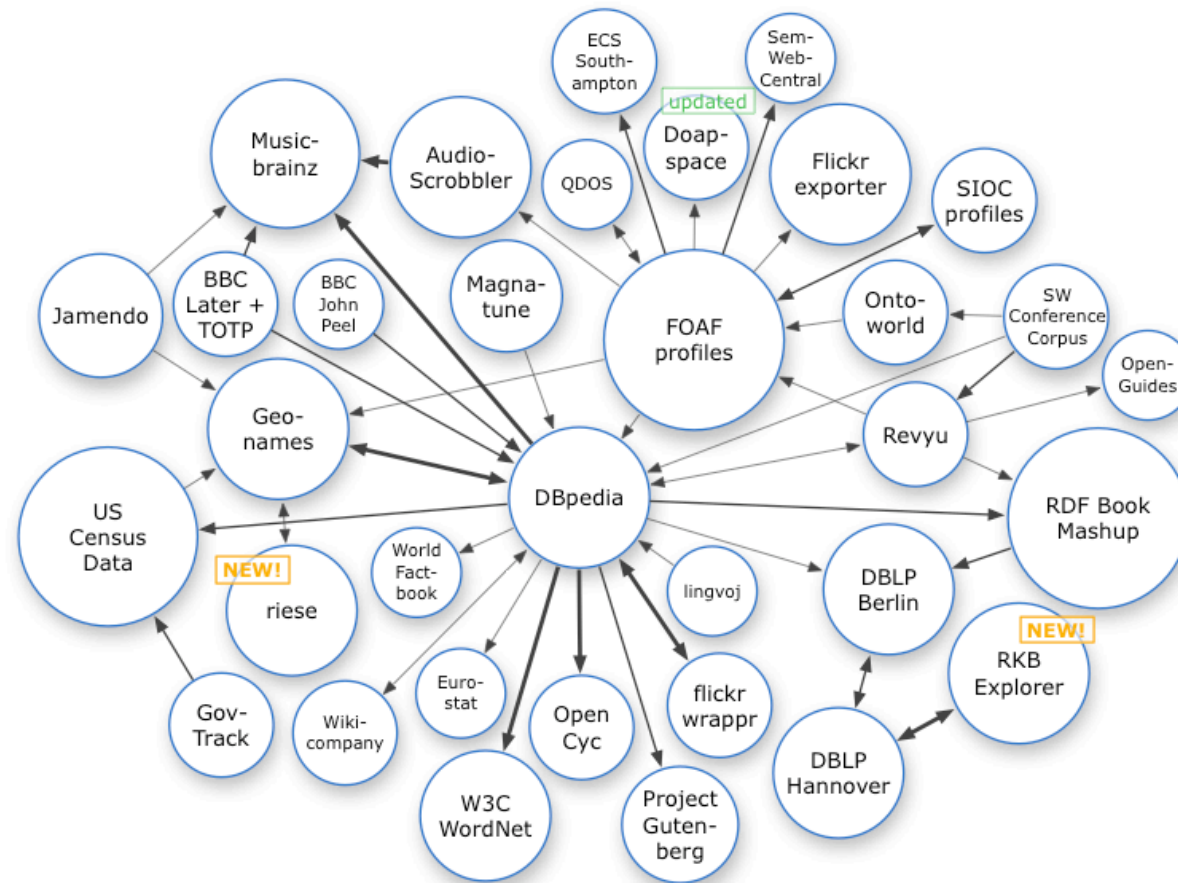

Reuse data already available from [Linked Open Data Cloud](#) (LOD)

Interlinked datasets in RDF

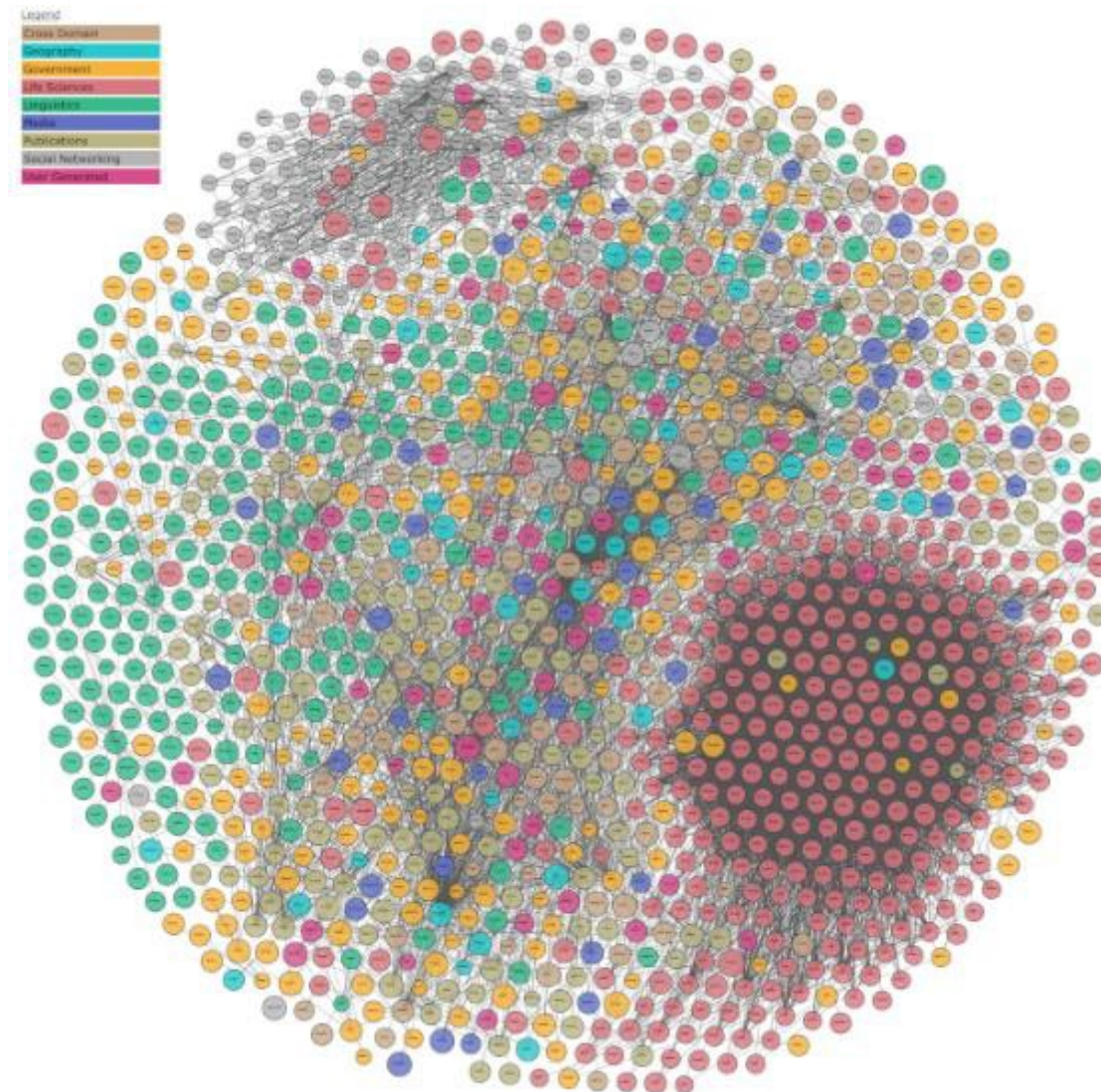
31 billion RDF triples, interlinked by around 504 million RDF links

Key nodes of the cloud include DBpedia (an RDF extract of Wikipedia) and Wikidata

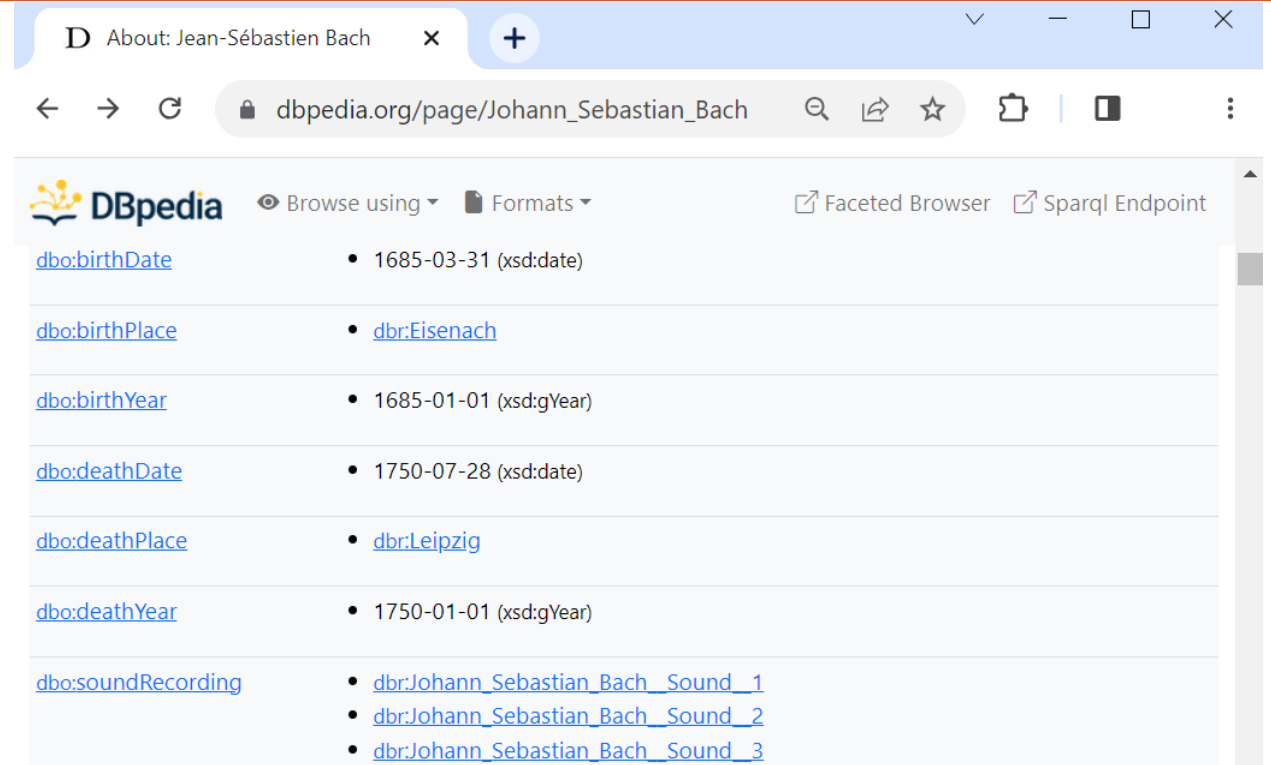
LOD Example - 2008



LOD Example - 2023



Linked Open Data - 2023



Access content of LOD both as a human or as a machine (reading RDF).

As human use search interface to locate an entry such as "Johann Sebastian Bach".
The corresponding web page will show you all its attributes in a tabular format.

Linked Open Data - 2023



Machine access this information in RDF format; Excerpt of DBpedia's data on JS Bach:

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

@prefix dbo: <http://dbpedia.org/ontology/> .

@prefix dbp: <http://dbpedia.org/property/> .

<http://dbpedia.org/resource/Johann_Sebastian_Bach>

a foaf:Person ;

rdfs:label "Johann Sebastian Bach"@en, "Jean-Sébastien Bach"@fr ;

dbo:birthDate "1685-03-31"^^xsd:date ;

dbp:birthPlace <http://dbpedia.org/resource/Eisenach> ;

dbo:birthYear "1685"^^xsd:Year ;

dbo:deathYear "1750"^^xsd:Year ;

dbo:deathDate "1750-07-28"^^xsd:date ;

dbp:deathPlace <http://dbpedia.org/resource/Leipzig> ;

owl:sameAs <http://www.wikidata.org/entity/Q1339> ;

dbo:soundRecording <http://dbpedia.org/resource/Johann_Sebastian_Bach_Sound_1> ,

<http://dbpedia.org/resource/Johann_Sebastian_Bach_Sound_2> ,

<http://dbpedia.org/resource/Johann_Sebastian_Bach_Sound_3> .

Call comes from an application
Server sends information in RDF.

Call from browser redirected to web page.
(Technically, the resource/ part of the URL will be replaced by page/.)

Redirection done automatically depending on the value of the HTTP User-Agent request header coming from the client.

- http://dbpedia.org/resource/Johann_Sebastian_Bach
- http://dbpedia.org/page/Johann_Sebastian_Bach

Use LOD to complement information of graph.

Using ***skos:related*** to link current resource to entry in DBpedia

@prefix skos: <http://www.w3.org/2004/02/skos/core#> .

```
<http://www.ontotext.com/academy/skos/music#composition_johann_sebastian_bach>  
  a skos:Concept ;  
  skos:prefLabel "Johann Sebastian Bach" ;  
  skos:narrower <http://www.ontotext.com/academy/skos/music#music_brandenburg_concertos> ;  
  skos:related <http://dbpedia.org/resource/Johann_Sebastian_Bach> .
```

```
<http://www.ontotext.com/academy/skos/music#music_brandenburg_concertos>  
  a skos:Concept ;  
  skos:prefLabel "Brandenburg Concertos" ;  
  skos:broader <http://www.ontotext.com/academy/skos/music#composition_johann_sebastian_bach> ;  
  skos:related <http://dbpedia.org/resource/Brandenburg_Concertos> .
```

SHACL – Shape Constraint Language



Number of values that a property may have,

- Type of values,
- Numeric Ranges,
- String Matching,
- Logical combinations of constraints.

Alert inappropriate values in datasets improve the quality of your data.

Properties that enable this checking of conditions include:

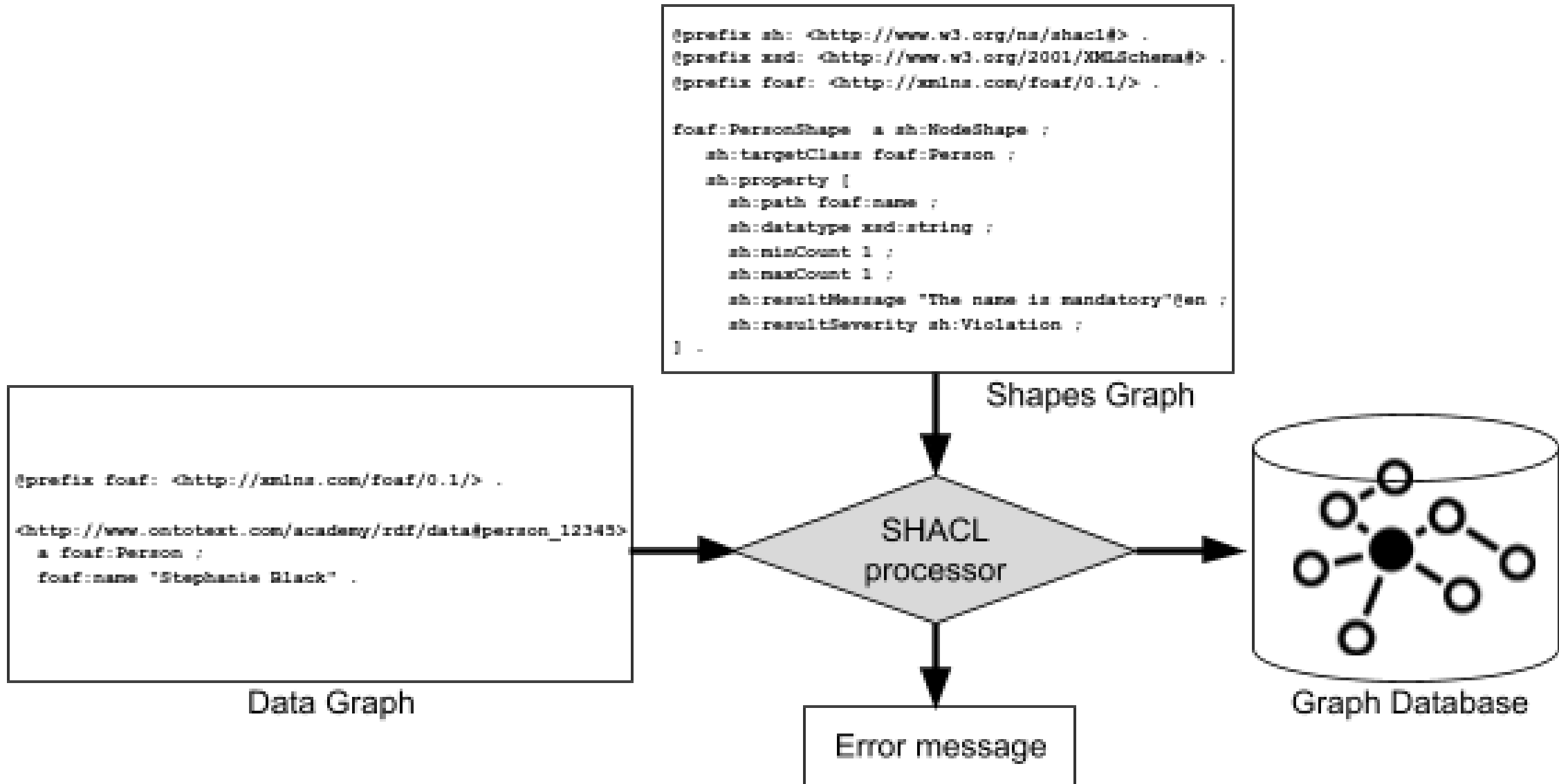
- **Cardinality:** sh:minCount, sh:maxCount
- **Types of values:** sh:class, sh:datatype, sh:nodeKind
- **Values:** sh:hasValue, sh:in, dash:hasValueIn
- **Ranges:** sh:minInclusive, sh:maxInclusive, sh:minExclusive, sh:maxExclusive
- **String:** sh:minLength, sh:maxLength, sh:pattern (and sh:flags), sh:languageIn, sh:uniqueLang
- **Logical constraints:** sh:and sh:not, sh:or
- **Condition:** sh:maxCount, sh:minCount, sh:nodeKind, sh:path, sh:inversePath, sh:property, sh:target, sh:targetClass, sh:targetNode, sh:targetObjectsOf, rx:targetShape, sh:targetSubjectsOf
- **Administration:** sh:deactivated, sh:message, sh:severity, sh:shapesGraph

Conditions expressed in RDF and are built around "shapes"
Describe specific constraint of a target.

Each target can be specified in several ways:

- `sh:targetClass`– All instances of a class
- `sh:targetNode`– Specific nodes
- `sh:targetObjectsOf`– All object of a specific property
- `sh:targetSubjectsOf`– All subjects of a specific property

SHACL Architecture



- SPARQL Protocol and RDF Query Language query data source in RDF
- SPARQL query Contain Triple Patterns
- Each of Subject, Predicate and Object may be a Variable.
- Not case-sensitive

SPARQL uses the # to comment out a line

```
SELECT * WHERE {  
  ?s ?p ?o .  
}
```

List all resources that have a type of *foaf:Person*

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT * WHERE {  
  ?s rdf:type foaf:Person .  
}
```



```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT * WHERE {  
  ?s a foaf:Person .  
  ?s foaf:name ?n  
}
```

- Compact syntax that does not repeat the ?s variable in different statements.
- Using a semicolon ";" at the end of a triple pattern

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT * WHERE {  
  ?s a foaf:Person ;  
    foaf:name ?n .  
}
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

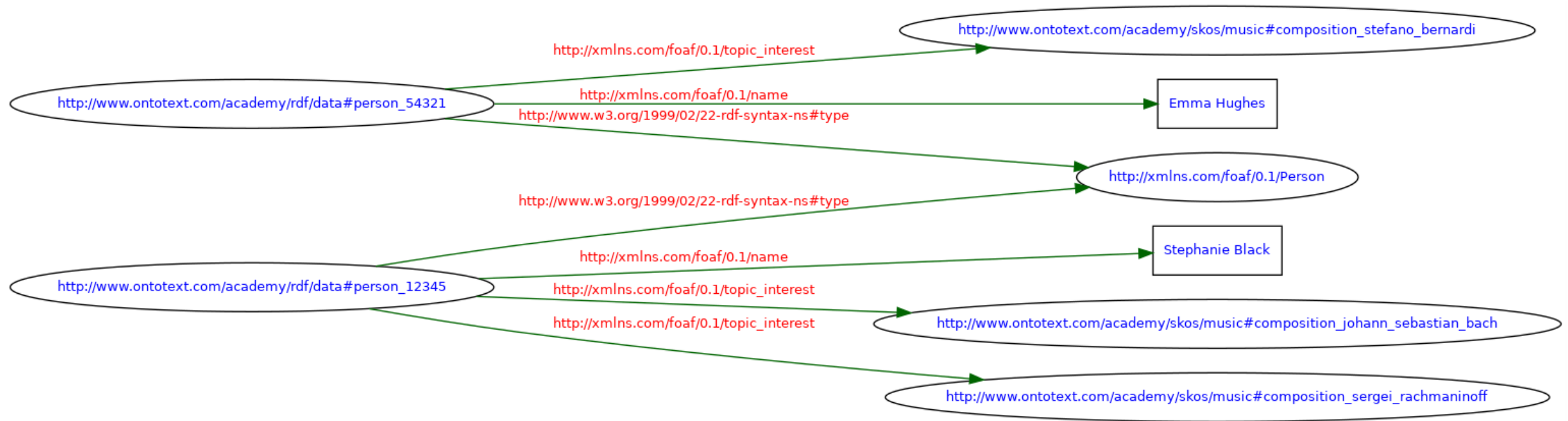
```
SELECT * WHERE {  
  ?s a foaf:Person ;  
    foaf:name "Stephanie Black" .  
}
```

```
PREFIX data: <http://www.ontotext.com/academy/rdf/data#>  
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT * WHERE {  
  data:person_12345 a foaf:Person ;  
    foaf:name ?n .  
}
```

SPARQL Examples

Extract - all people who have only a specific interest in two types of music composition:
Johann Sebastian Bach and Sergei Rachmaninoff



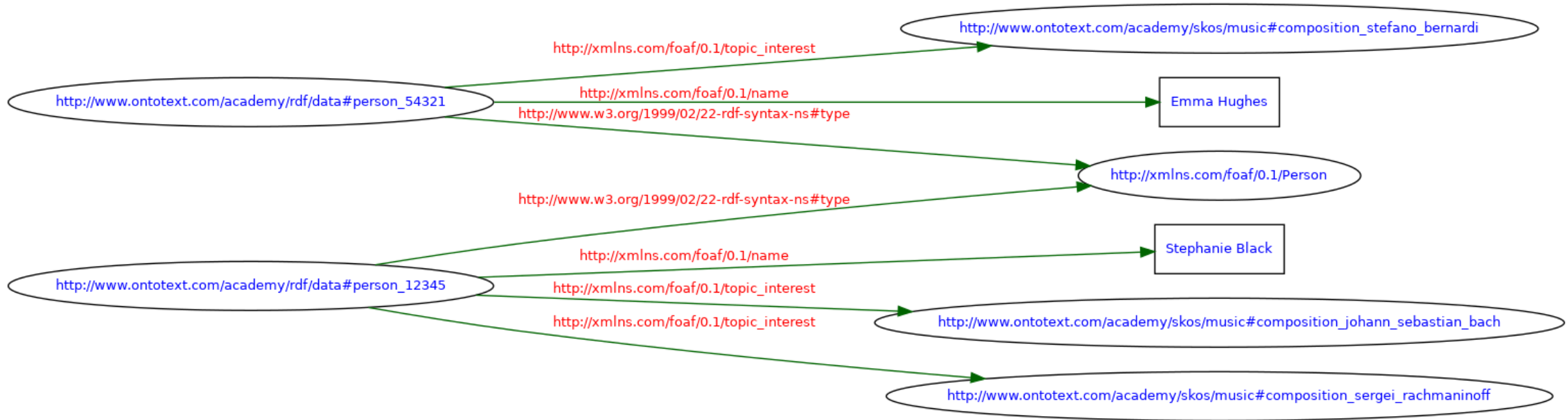
Extract - all people who have only a specific interest in two types of music composition:
Johann Sebastian Bach and Sergei Rachmaninoff

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX music: <http://www.ontotext.com/academy/skos/music#>
```

```
SELECT * WHERE {  
    ?s a foaf:Person ;  
    foaf:name ?n ;  
    foaf:topic_interest music:composition_johann_sebastian_bach ;  
    foaf:topic_interest music:composition_sergei_rachmaninoff .  
}
```

SPARQL Examples



Know Stephanie's musical interests but want to list them

Know Stephanie's musical interests but want to list them

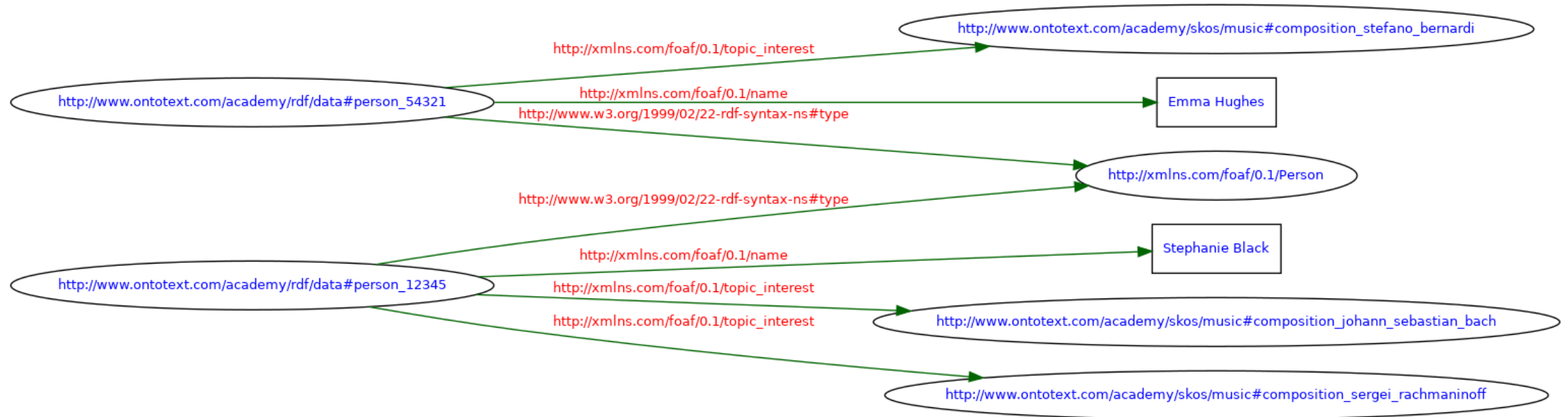
```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT ?interestTopic WHERE {  
  ?s a foaf:Person ;  
    foaf:name "Stephanie Black" ;  
    foaf:topic_interest ?interestTopic .  
}
```

Result URI identifiers of interest nodes

Nodes have properties

Want to see them ? Nodes have a skos:prefLabel property that shows a name for the interest topic




```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

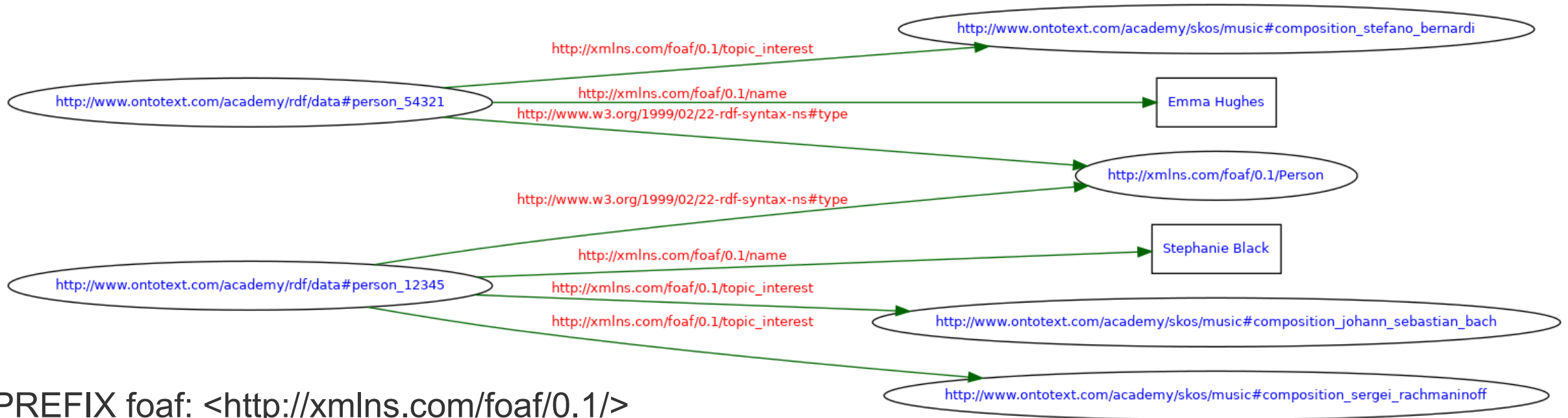
SELECT ?topicName WHERE {
  ?s a foaf:Person ;
    foaf:name "Stephanie Black" ;
    foaf:topic_interest ?interestTopic .

  ?interestTopic skos:prefLabel ?topicName .
}
```

SPARQL Examples – Optional Clause

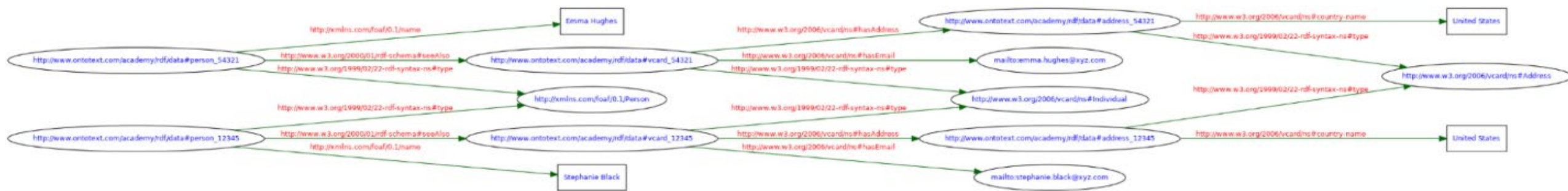


Names of the foaf:Person instances and their nicknames if they have



PREFIX foaf: <http://xmlns.com/foaf/0.1/>

```
SELECT ?name ?nickname WHERE {  
  ?p a foaf:Person ;  
    foaf:name ?name .  
  OPTIONAL { ?p foaf:nick ?nickname . }  
}
```



Identify resources distant from each other on the graph.

Looking for all instances of foaf:Person who have a foaf:name property.

Go further for other resources that are linked to the resulting nodes by a rdfs:seeAlso relationship

Target of rdfs:seeAlso a vcard:Individual should have a vcard:hasEmail property.

Looking to find all persons who have a business card (vcard) with an email address attached to it.

SPARQL Example - Traversing

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
```

```
SELECT ?s ?n ?email WHERE {
    ?s a foaf:Person ;
        foaf:name ?n ;
        rdfs:seeAlso ?vcard .

    ?vcard a vcard:Individual ;
        vcard:hasEmail ?email
}
```

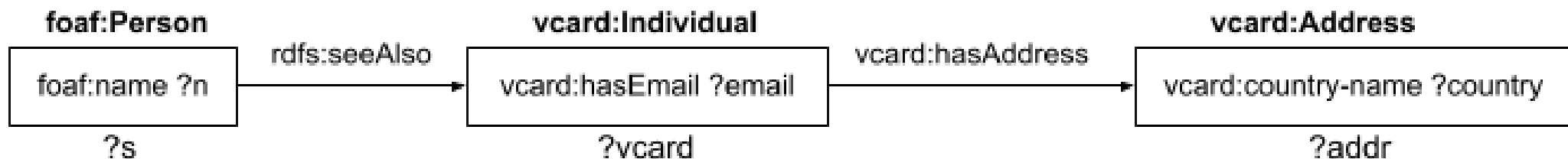
SPARQL Example - Traversing

Further navigate adding more statements to the query.

Use vCard instance ?vcard that and follow its vcard:hasAddress relationship to a resource of type vcard:Address

that has a vcard:country-name.

Find address attached to the business card (vcard) and locate the country name of this address.



SPARQL Example - Traversing

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
```

```
SELECT ?s ?n ?email ?country WHERE {
```

```
  ?s a foaf:Person ;
```

```
    foaf:name ?n ;
```

```
    rdfs:seeAlso ?vcard .
```

```
  ?vcard a vcard:Individual ;
```

```
    vcard:hasEmail ?email ;
```

```
    vcard:hasAddress ?addr .
```

```
  ?addr a vcard:Address ;
```

```
    vcard:country-name ?country
```

```
}
```

SPARQL – Path Traversal



Express the graph route

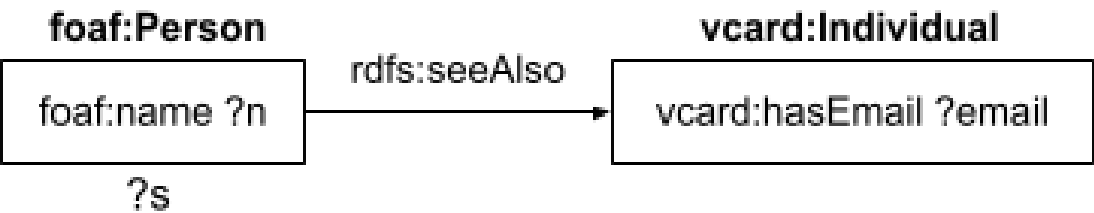
/ to denote moving to a subsequent arc
(edge on the graph corresponding to a property path step).

^ indicates the opposite navigation direction.

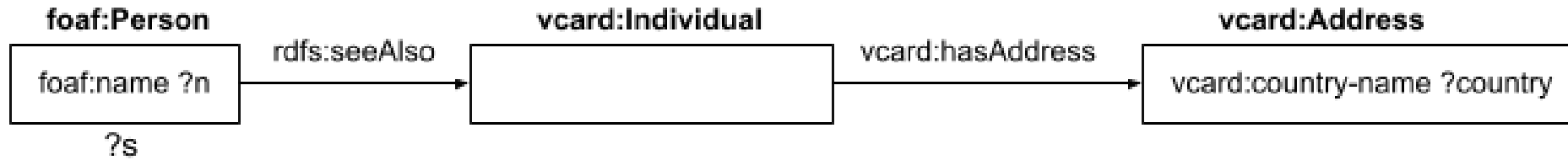
```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
```

```
SELECT ?s ?n ?email WHERE {
  ?s a foaf:Person ;
    foaf:name ?n ;
    rdfs:seeAlso / vcard:hasEmail ?email
}
```

/	Sequence path
^	Reverse path
	Alternative path
*	Path of zero or more occurrences
+	Path of one or more occurrences
?	Path of zero or one



SPARQL – Path Traversal



Reuse sequence path / to traverse further in the relationship

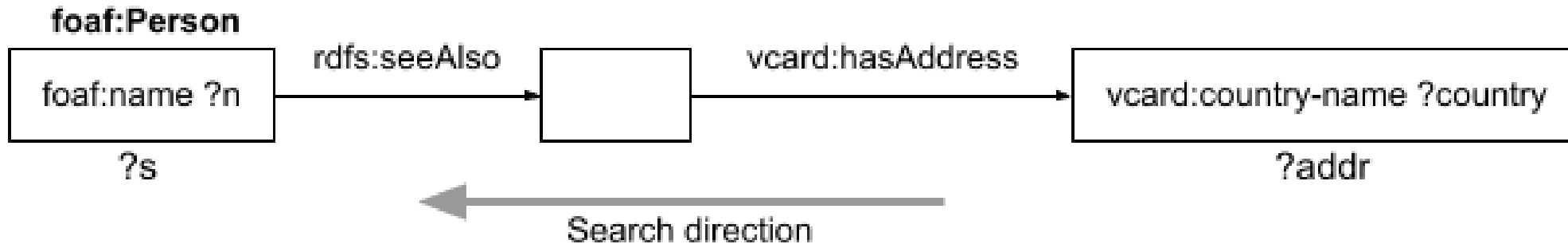
```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
```

```
SELECT ?s ?n ?country WHERE {  
  ?s a foaf:Person ;  
    foaf:name ?n ;  
    rdfs:seeAlso / vcard:hasAddress / vcard:country-name ?country  
}
```

SPARQL – Path Traversal



Reverse path ^ works the same way but navigating the arc in the reverse direction.

Starts by finding all resources that have a vcard:country-name property.

If found, looks for the incoming arc vcard:hasAddress.

Repeats the same operation with the incoming rdfs:seeAlso.

Finally, it does a descendant search to find the value of the foaf:name property.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT * WHERE {
```

```
  ?addr vcard:country-name ?country;
```

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

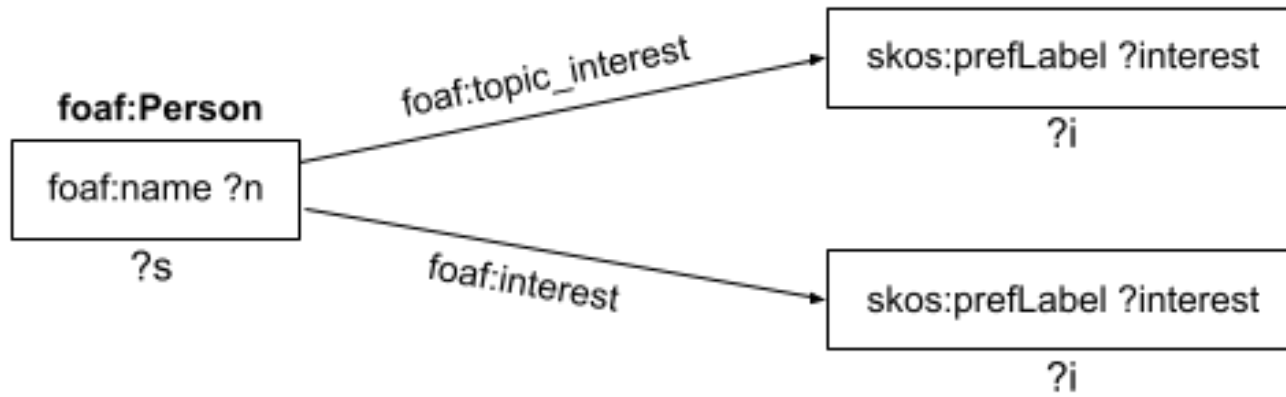
```
SELECT * WHERE {
```

```
    ?addr vcard:country-name ?country;
```

```
        ^vcard:hasAddress / ^rdfs:seeAlso / foaf:name ?n .
```

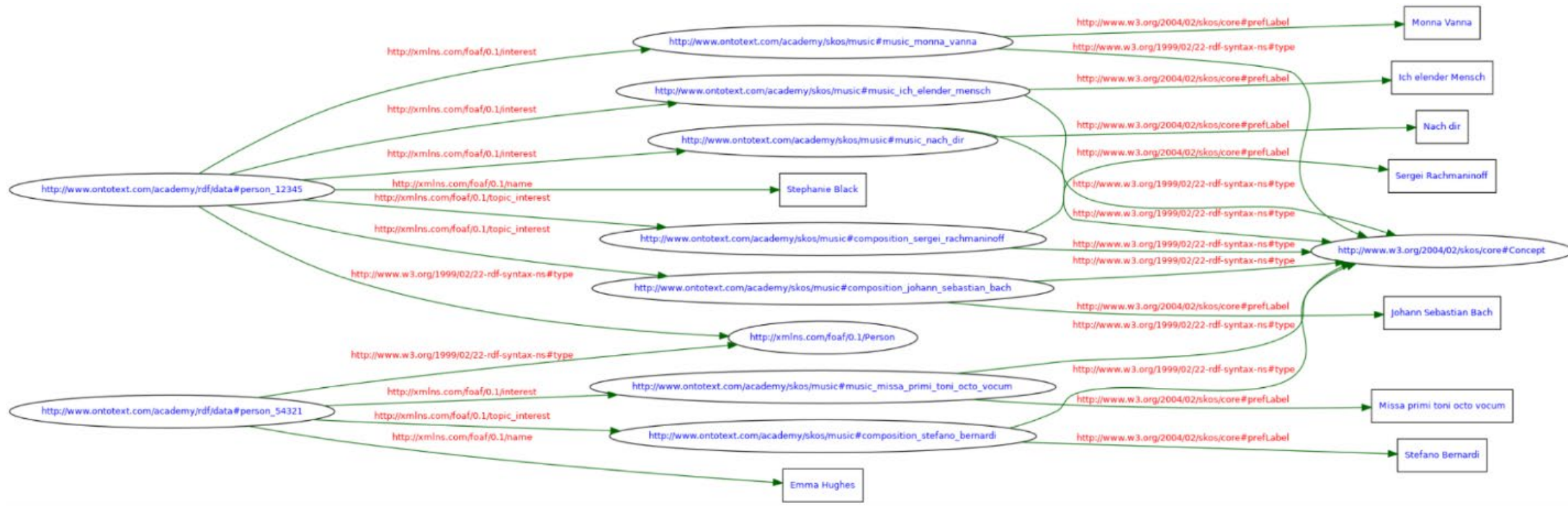
```
}
```

SPARQL – Path Traversal



Branch to different relations
Selects All persons have at least one relation
foaf:topic_interest OR ***foaf:interest***.

If none exist, selection of node invalidated
If at least one relation exists, Query succeed,
and corresponding resources will be selected.



SPARQL – Alternative Path Traversal



```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
```

```
SELECT ?s ?interest WHERE {
  ?s a foaf:Person ;
    foaf:name ?n ;
    foaf:topic_interest | foaf:interest ?i .

  ?i skos:prefLabel ?interest
}
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
```

```
SELECT ?s ?interest WHERE {
  ?s a foaf:Person ;
    foaf:name ?n ;
    (foaf:topic_interest | foaf:interest) / skos:prefLabel ?interest
}
```


`FILTER()` expression limits the search based on conditions.
Expression specifies boolean condition, and then data for which that condition is true will be returned.

Can combine the `FILTER()` expression with different operators such as:

String: `STRLEN`, `SUBSTR`, `UCASE`, `LCASE`, `STRSTARTS`, `STRENDS`, `CONTAINS`, `STRBEFORE`, `STRAFTER`, `ENCODE_FOR_URI`

Logical: `!`, `&&`, `||`

Comparison: `=`, `!=`, `>`, `<`, `IN`, `NOT IN`

Math: `abs`, `round`, `ceil`, `floor`, `RAND`, `+`, `-`, `*`, `/`, ```

Match: `langMatches`, `REGEX`, `REPLACE`

Conditional: `IF`, `COALESCE`, `EXISTS`, `NOT EXISTS`

Date/Time: `NOW`, `YEAR`, `MONTH`, `DAY`, `HOURS`, `MINUTES`, `SECONDS`, `TIMEZONE`, `TZ`

Tests: `isIRI`, `isURI`, `isBlank`, `isLiteral`, `isNumeric`, `bound`

Hashing: `MD5`, `SHA1`, `SHA256`, `SHA512`

Logical and Comparison operators

B

Regular Expressions

SET Operations

Sorting and Limiting and OFFSETting Operations

Functions – Concat, Coalesce

Aggregation

Not Exists - you retrieve data that is missing a specified kind of accompanying data.

Example - Lists people with a family name of "Brown" that have no friends (or rather, for whom we have no foaf:knows data

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT ?person WHERE {  
    ?person a foaf:Person ;  
        foaf:family_name "Brown" .  
    FILTER NOT EXISTS { ?person foaf:knows ?friend }  
}
```

Finding the number of people in each country

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT ?country (COUNT(?p) as ?total) WHERE {  
    ?p a foaf:Person ;  
        rdfs:seeAlso / vcard:hasAddress / vcard:country-name ?country  
}  
GROUP BY ?country
```

When not familiar with a dataset's structure

Discover properties and relationship of a resource.

Returns triples

W3C SPARQL specification is vague about exactly what a query processor is required to return

GraphDB returns all triples that have that URI as a subject or object.

```
DESCRIBE <http://www.ontotext.com/academy/rdf/data#person\_9249006>
```

Whether a query pattern matches or not

Result Boolean true or false.

Example asks if anyone has the salary shown.

It will return a boolean true.

PREFIX dbo: <<http://dbpedia.org/ontology/>>

```
ASK { ?person dbo:salary "3.800000e+1"^^xsd:double }
```

SPARQL – Query Federation



SERVICE query processor to pass a graph pattern or a SELECT query to a service

Retrieve resource `Johann_Sebastian_Bach` in DBpedia

Using SPARQL endpoint that makes structured data from Wikipedia available as RDF triples.

Try to connect yourself to the DBpedia entry as a human

http://dbpedia.org/page/Johann_Sebastian_Bach

SPARQL – Query Federation



Machine Readable RDFs

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
PREFIX dbr: <http://dbpedia.org/resource/>
```

```
PREFIX dbo: <http://dbpedia.org/ontology/>
```

```
SELECT *
```

```
WHERE {
```

```
    SERVICE <https://dbpedia.org/sparql> {
```

```
        dbr:Johann_Sebastian_Bach rdfs:label ?label ;
```

```
        dbo:abstract ?abstract .
```

```
        FILTER (LANG(?label) = "en")
```

```
        FILTER (LANG(?abstract) = "en")
```

```
    }
```

```
}
```


SPARQL – Query Federation



```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
CONSTRUCT {
  <http://www.ontotext.com/academy/rdf/maurice-ravel> ?p ?o
}
WHERE {
  {
    SERVICE <https://dbpedia.org/sparql> {
      dbr:Maurice_Ravel ?p ?o .
      FILTER (isLiteral(?o) && LANG(?o) = "en")
    }
  }
  UNION {
    SERVICE <https://query.wikidata.org/sparql> {
      <http://www.wikidata.org/entity/Q1178> ?p ?o .
      FILTER (isLiteral(?o) && LANG(?o) = "en")
    }
  }
}
```

Taxonomies + Ontology + Instance data = Knowledge graph

Ontology DNA

Taxonomy RNA

Vocabulary / Glossary

Graph Databases

Semantic Mapping Tools / Languages

Data Mapping Framework

ETL / ELT Tools

What Is a Taxonomy



- Information model → Describe, Structural Hierarchy
- Effective for Organizing Data
- Capture Context, Meaning → Easy to Find & Understand
- Form of Classification Scheme
- Knowledge Map

Foundation for Smart Search/Discovery Applications

Finding Appropriate Term / Provide Metadata and Tagging

Models how Terms / Entities organized Hierarchically

Categorize data to be found / retrieved

Supports Better Findability, Precision/ Recall metrics

- Schema, Structure, and Rules for Graph Data
- Blueprint / Common Data Model
- Identifies/Distinguishes Concepts & Relationships
- Formal Specifications of Terms in Domain
- Defines : Entities, Relationships & Properties
- Semantic Types, Properties and Relations

Avoid / Minimizes Ambiguity

Used for – Interoperability (Shared Vocabularies) + Inference

Shared Vocabulary → Describe Semantics

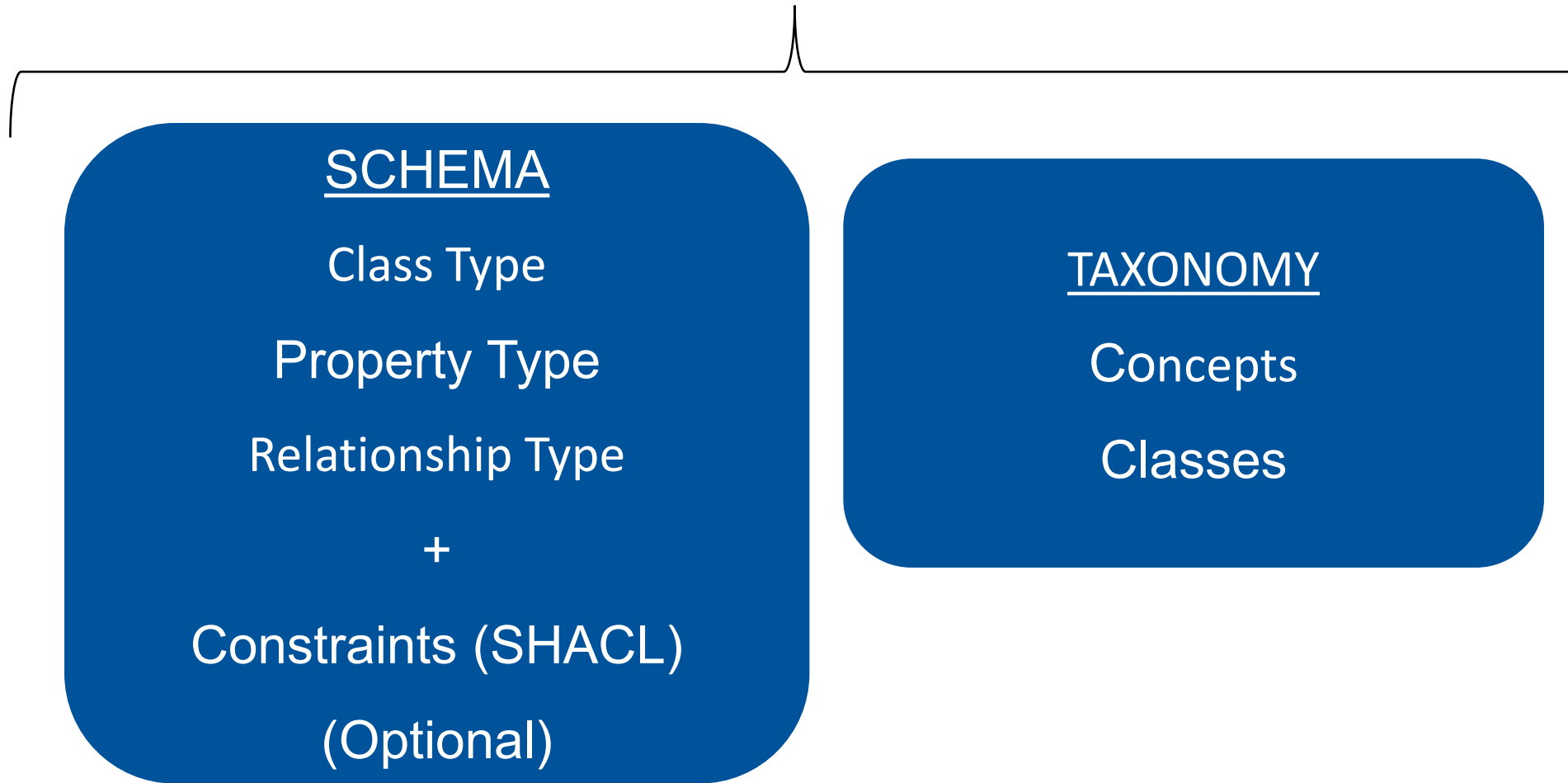
Core components of an ontology

- **1. Classes**
 - May include **subclasses** (and superclasses)
- **2. Properties**
 - May include **subproperties** (and superproperties)
 - Called “slots” in older terminology
- **3. Instances**
 - Also called “individuals”
 - Specific members of a class

Ontology statement examples

- **Class definition statements:**
 - Parent isA Class
 - Mother isA Class
 - Mother subClassOf Parent
 - Child isA Class
- **Property definition statements:**
 - isMotherOf isA Property
 - isMotherOf domain Mother
 - isMotherOf range Child
- **Individual/instance statements:**
 - MariaTaylor isA Mother
 - AdamJTaylor isA Child
 - MariaTaylor isMotherOf AdamJTaylor

Ontology + Taxonomy → Semantics



FIBO RDF Modeling

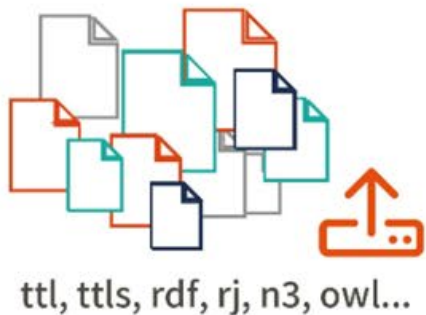
DMZ, 1st March 2023

Maximize your use of The Financial Industry Business Ontology (FIBO) with GraphDB



Register now
for free one-on-one
training on using FIBO
with
GraphDB


Upload Data



Reconcile



Map to FIBO

total statements
390,249 

86,860 explicit
303,389 inferred
4.49 expansion ratio

Explore

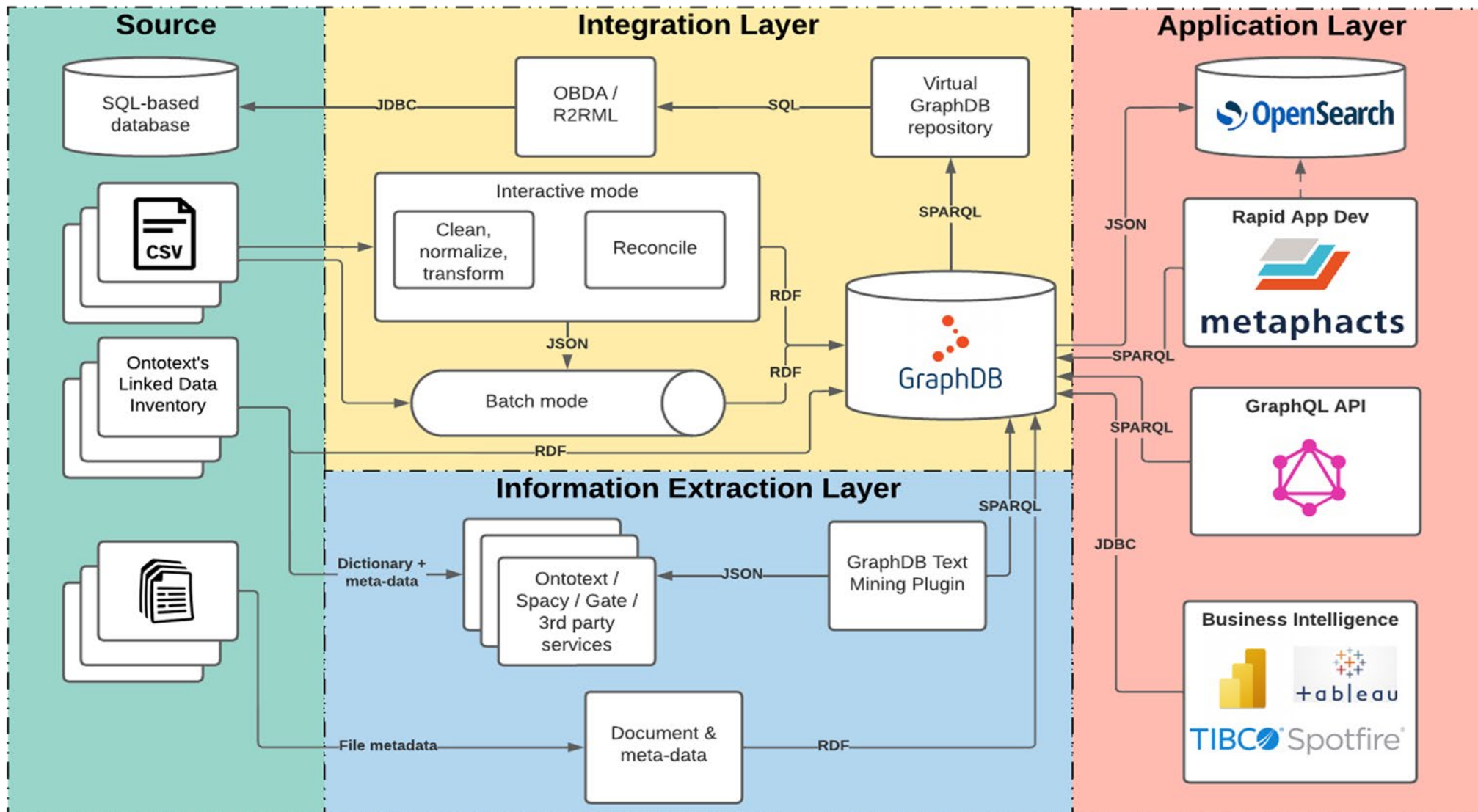


Visualize



- Conceptual Model developed by the [Enterprise Data Management Council](#) (EDMC)
- Supports an [open process](#) for maintenance and development of FIBO
- Goal of FIBO provide precise meaning to the data artifacts that describe the business of finance
- Contains entities / associations describe information needed to build, extend & integrate financial business applications
- Specified using RDF(S) and OWL
- Analysis using SPARQL and OWL inference

KG High-level Application Architecture



Key capabilities from FIBO powered KG



- Knowledge Representation - Express Domain Knowledge
- Data Interoperability - Common Vocabulary to Describe and Exchange data;
- Reusability and Extensibility – Use / Reuse / Refine
- Inference and Reasoning - Derive New Insights and Facts
- Abstractions Alignment - Normalization and Harmonization
- Improved Discoverability
- Content & Knowledge Management

Autocomplete










Autocomplete index

Autocomplete for repository FIBO is **ON** with status **Ready**

Build Now

Index IRIs is **ON** **+ Add label**

Label IRI	Languages
http://xmlns.com/foaf/0.1/name	any language  
https://spec.edmcouncil.org/fibo/ontology/FND/Utilities/AnnotationVocabulary/preferredDesignation	any language  
https://spec.edmcouncil.org/fibo/ontology/FND/Utilities/AnnotationVocabulary/synonym	any language  
https://spec.edmcouncil.org/fibo/ontology/FND/Utilities/AnnotationVocabulary/commonDesignation	any language  
http://www.w3.org/2000/01/rdf-schema#label	any language  
https://spec.edmcouncil.org/fibo/ontology/FND/Utilities/AnnotationVocabulary/explanatoryNote	any language  
http://www.w3.org/2004/02/skos/core#prefLabel	any language  

[The Autocomplete index](#) is one of the most important utilities of the GraphDB Workbench.

It offers suggestions for the IRIs' local names in the SPARQL editor, the View Resource page, Search RDF resources and in the OntoRefine Mapping editor.

[RDF Rank](#) is an algorithm that identifies the more important or more popular entities in the repository by examining their interconnectedness. The popularity of entities can then be used to order the query results in a similar way to the internet search engines, the way Google orders search results using [PageRank](#). This can be used with the Reconciliation server or for your own API.

```
PREFIX rank: <http://www.ontotext.com/owlim/RDFRank#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT * WHERE {
    ?entity a <https://www.omg.org/spec/LCC/Countries/CountryRepresentation/Country>;
    rdfs:label ?name;
    rank:hasRDFRank5 ?rank .
}
ORDER BY DESC(?rank) LIMIT 100
```

The [Prominence functionality](#), the prominence for a resource is defined as the sum of the number of outgoing connections (where the resource is the subject of a triple) and the number of incoming connections (where the resource is the object of a triple). The numbers are automatically maintained by GraphDB.

Here are the most prominent Public companies in FIBO

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?node ?name ?prominence {
  ?node <http://www.ontotext.com/owlim/entity#hasProminence> ?prominence;
    rdfs:label ?name;
    a <https://spec.edmcouncil.org/fibo/ontology/BE/Corporations/Corporations/PubliclyHeldCompany>.
#    a owl:NamedIndividual.
} order by desc(?prominence)
```

Advantages of Semantic Technology is various IR Search as:

- Full-Text Search (Lucene style), including integration with Elastic and SOLR
- Text Similarity (word2vec)
- Graph Similarity
- Semantic Properties (synonyms, hypernyms)
- Taxonomic Structure (skos:broader, skos:narrower)
- Visual of the graph relationships

Semantic Vectors: Text-based similarity



4 types of Semantic Vectors Similarity searches:

1. **Term to Term** - Closest Semantically Related Terms
2. **Term to Document** - Most Representative Documents for a specific searched Term
3. **Document to Term** - Most Representative Terms for a Specific Document
4. **Document to Document** - Closest Related Texts

Similarity Plugin integrates Semantic Vectors Library and Random Indexing algorithm. Algorithm uses tokenizer to translate documents to Sequences of Words (terms) to represent a Vector Space Model

Feature of the algorithm is the Dimensionality Reduction based on Random Projection
Initial Vector State is generated randomly.

Semantic Vectors: Term to Term Example

Search in TextSim



FIBO

admin

en

Search type: Term Result type: Term

CEO

Show

Hint: "abC" matches "abC*", "ab c*" and "ab-c*"

search options

Showing results for "CEO" [View SPARQL Query](#)

	documentID	score
1	"ceo"	"1.0"xsd:double
2	"chairperson"	"0.8789604040941613"xsd:double
3	"leader"	"0.8789604040941613"xsd:double
4	"firm's"	"0.7861661199104802"xsd:double
5	"she"	"0.7625867194235817"xsd:double
6	"success"	"0.6723673802898552"xsd:double
7	"officer"	"0.6367817223376875"xsd:double
8	"chief"	"0.5631670810658115"xsd:double
9	"cfo"	"0.5408987362959166"xsd:double
10	"implement"	"0.5269860561819623"xsd:double

Semantic Vectors: Doc to Doc Example



Search in TextSim



FIBO

admin

en

Search type: Document Result type: Document

https://spec.edmouncil.org/fibo/ontology/BE/OwnershipAndControl/Executives/ChiefExecutiveOfficer

Show


Hint: "abc" matches "abC*", "ab c*" and "ab-c*"

search options

Showing results for https://spec.edmouncil.org/fibo/ontology/BE/OwnershipAndControl/Executives/ChiefExecutiveOfficer [View SPARQL Query](#)

	documentID	score
1	https://spec.edmouncil.org/fibo/ontology/BE/OwnershipAndControl/Executives/ChiefExecutiveOfficer	"1.0"^^xsd:double
2	https://spec.edmouncil.org/fibo/ontology/BE/OwnershipAndControl/Executives/CorporateOfficer	"0.9438975551228896"^^xsd:double
3	https://spec.edmouncil.org/fibo/ontology/BE/OwnershipAndControl/Executives/CorporateOfficer	"0.9438975551228896"^^xsd:double
4	https://spec.edmouncil.org/fibo/ontology/BE/OwnershipAndControl/Executives/ChiefFinancialOfficer	"0.9172196816608629"^^xsd:double
5	https://spec.edmouncil.org/fibo/ontology/BE/OwnershipAndControl/Executives/isOfficerOf	"0.9138366642165311"^^xsd:double

Refine

Organisation  Project ID: 2317443267871 Aliases:

Open... Export Help

Facet / Filter Undo / Redo 1 / 1

469 rows

Show as: rows records Show: 5 10 25 50 100 500 1000 rows

« first < previous 1 next > last »

Using facets and filters
 Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

 Not sure how to get started?
[Watch these screencasts](#)

				▼ All	▼ Rank	▼ Title	▼ Website	▼ Employees	▼ Sector	▼ Industry	▼ Hqlocation	▼ Hqaddr	▼ Hqcity	▼	
☆	🗨	1.	1				Walmart	http://www.walmart.com	2300000	Retailing	General Merchandisers	Bentonville, AR	702 S.W. Eighth St.	Bentonville	AR
☆	🗨	2.	2				Berkshire Hathaway	http://www.berkshirehathaway.com	367700	Financials	Insurance: Property and Casualty (Stock)	Omaha, NE	3555 Farnam St.	Omaha	NE
☆	🗨	3.	3				Apple	http://www.apple.com	116000	Technology	Computers, Office Equipment	Cupertino, CA	1 Infinite Loop	Cupertino	CA
☆	🗨	4.	4				Exxon Mobil	http://www.exxonmobil.com	72700	Energy	Petroleum Refining	Irving, TX	5959 Las Colinas Blvd.	Irving	TX
☆	🗨	5.	5				McKesson	http://www.mckesson.com	68000	Wholesalers	Wholesalers: Health Care	San Francisco, CA	1 Post St.	San Francisco	CA
☆	🗨	6.	6				UnitedHealth Group	http://www.unitedhealthgroup.com	230000	Health Care	Health Care: Insurance and Managed Care	Minnetonka, MN	9900 Bren Rd. E.	Minnetonka	MN
☆	🗨	7.	7				CVS Health	http://www.cvshealth.com	204000	Health Care	Health Care: Pharmacy and Other Services	Woonsocket, RI	1 CVS Dr.	Woonsocket	RI
☆	🗨	8.	8				General Motors	http://www.gm.com	225000	Motor Vehicles & Parts	Motor Vehicles and Parts	Detroit, MI	300 Renaissance Center	Detroit	MI

CSV headers

Mapping

Rank	Title	Website	Employees	Sector	Industry	Hqlocation	Hqaddr	Hqcity	Hqstate	Hqzip	Hqtel	Ceo
Revchange	Profits	Prftchange	Assets	Totshequity	Ceo-title	Address	Ticker	Fullname	Revenues			

Base IRI
https://ontotext.com/rdf/resource/data/

Use the current repository prefixes or add new using the Turtle or SPARQL syntax, i.e PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

ex x fibo-be-corp-corp x fibo-be-lc-cb x fibo-fnd-org-fm x fibo-fnd-plc-adr x fibo-fnd-plc-loc x lcc-3166-1 x lcc-3166-2-us x rdfs x

ex:corporation_	@ Ticker	<IRI>	a	<IRI>	fibo-be-corp-corp: PubliclyHeldCompany	<IRI>
		rdfs: label			@ Title	"Literal"

Prefix	Column
ex:corporation_	: Ticker

Prefix	Constant
fibo-be-corp-corp	: PubliclyHeldCompany

ex:corporation_WMT a fibo-be-corp-corp:PubliclyHeldCompany

RDF

Prefix	Constant
rdfs	: label

Column
Title

rdfs:label "Walmart"

Data Integration from Relational Databases



Database

financialresult-# ;										
index	ticker	years_index	date	cost_revenue	cost_good	cost_admin	cost_rd	cost_other	sector	industry
0	AAL	Year 1	2012-12-31	\$24,855,000,000	\$10,499,000,000	\$12,977,000,000	\$-	\$845,000,000	Industrials	Airlines
1	AAL	Year 2	2013-12-31	\$26,743,000,000	\$11,019,000,000	\$12,913,000,000	\$-	\$853,000,000	Industrials	Airlines
2	AAL	Year 3	2014-12-31	\$42,650,000,000	\$15,620,000,000	\$20,686,000,000	\$-	\$1,295,000,000	Industrials	Airlines
3	AAL	Year 4	2015-12-31	\$40,990,000,000	\$11,096,000,000	\$21,275,000,000	\$-	\$1,364,000,000	Industrials	Airlines
4	AAP	Year 1	2012-12-29	\$6,205,003,000	\$3,106,967,000	\$2,440,721,000	\$-	\$-	Consumer Discretionary	Automotive Retail
5	AAP	Year 2	2013-12-28	\$6,493,814,000	\$3,241,668,000	\$2,591,828,000	\$-	\$-	Consumer Discretionary	Automotive Retail
6	AAP	Year 3	2015-01-03	\$9,843,861,000	\$5,390,248,000	\$3,601,903,000	\$-	\$-	Consumer Discretionary	Automotive Retail
7	AAP	Year 4	2016-01-02	\$9,737,018,000	\$5,314,246,000	\$3,596,992,000	\$-	\$-	Consumer Discretionary	Automotive Retail
8	AAPL	Year 1	2013-09-28	\$170,910,000,000	\$106,606,000,000	\$10,830,000,000	\$4,475,000,000	\$-	Information Technology	Computer Hardware
9	AAPL	Year 2	2014-09-27	\$182,795,000,000	\$112,258,000,000	\$11,993,000,000	\$6,041,000,000	\$-	Information Technology	Computer Hardware
10	AAPL	Year 3	2015-09-26	\$233,715,000,000	\$140,089,000,000	\$14,329,000,000	\$8,067,000,000	\$-	Information Technology	Computer Hardware
11	AAPL	Year 4	2016-09-24	\$215,639,000,000	\$131,376,000,000	\$14,194,000,000	\$10,045,000,000	\$-	Information Technology	Computer Hardware

Configuration

Create Ontop Virtual SPARQL repository

Repository ID*

Repository description

Connection Information

Database driver

Host name*

Port

Database name *

Username

Password

Driver class *

JDBC URL *

Ontop settings

OBDA or R2RML file *

```
[PrefixDeclaration]
ex: https://ontotext.com/rd/resource/data/
voc: https://ontotext.com/rd/resource/schema#
skos: http://www.w3.org/2004/02/skos/core#

[MappingDeclaration]
@collection [[
  mappingId data
  target ex:result_{ticker}_{date} a voc:FinancialResult ;
  skos:related ex:corporation_{ticker};
  voc:date {date}^^xsd:date ;
  voc:costRevenue {cost_revenue}^^xsd:string ;
  voc:costGood {cost_good}^^xsd:string ;
  voc:costAdmin {cost_admin}^^xsd:string ;
  voc:costRD {cost_rd}^^xsd:string ;
  voc:costOther {cost_other}^^xsd:string .
  source SELECT * FROM "data"
]]
```

Data Integration from Relational Databases



financialresult-# ;											
index	ticker	years_index	date	cost_revenue	cost_good	cost_admin	cost_rd	cost_other	sector	industry	
0	AAL	Year 1	2012-12-31	\$24,855,000,000	\$10,499,000,000	\$12,977,000,000	\$-	\$845,000,000	Industrials	Airlines	
1	AAL	Year 2	2013-12-31	\$26,743,000,000	\$11,019,000,000	\$12,913,000,000	\$-	\$853,000,000	Industrials	Airlines	
2	AAL	Year 3	2014-12-31	\$42,650,000,000	\$15,620,000,000	\$20,686,000,000	\$-	\$1,295,000,000	Industrials	Airlines	
3	AAL	Year 4	2015-12-31	\$40,990,000,000	\$11,096,000,000	\$21,275,000,000	\$-	\$1,364,000,000	Industrials	Airlines	
4	AAP	Year 1	2012-12-29	\$6,205,003,000	\$3,106,967,000	\$2,440,721,000	\$-	\$-	Consumer Discretionary	Automotive Retail	
5	AAP	Year 2	2013-12-28	\$6,493,814,000	\$3,241,668,000	\$2,591,828,000	\$-	\$-	Consumer Discretionary	Automotive Retail	
6	AAP	Year 3	2015-01-03	\$9,843,861,000	\$5,390,248,000	\$3,601,903,000	\$-	\$-	Consumer Discretionary	Automotive Retail	
7	AAP	Year 4	2016-01-02	\$9,737,018,000	\$5,314,246,000	\$3,596,992,000	\$-	\$-	Consumer Discretionary	Automotive Retail	
8	AAPL	Year 1	2013-09-28	\$170,910,000,000	\$106,606,000,000	\$10,830,000,000	\$4,475,000,000	\$-	Information Technology	Computer Hardware	
9	AAPL	Year 2	2014-09-27	\$182,795,000,000	\$112,258,000,000	\$11,993,000,000	\$6,041,000,000	\$-	Information Technology	Computer Hardware	
10	AAPL	Year 3	2015-09-26	\$233,715,000,000	\$140,089,000,000	\$14,329,000,000	\$8,067,000,000	\$-	Information Technology	Computer Hardware	
11	AAPL	Year 4	2016-09-24	\$215,639,000,000	\$131,376,000,000	\$14,194,000,000	\$10,045,000,000	\$-	Information Technology	Computer Hardware	

```

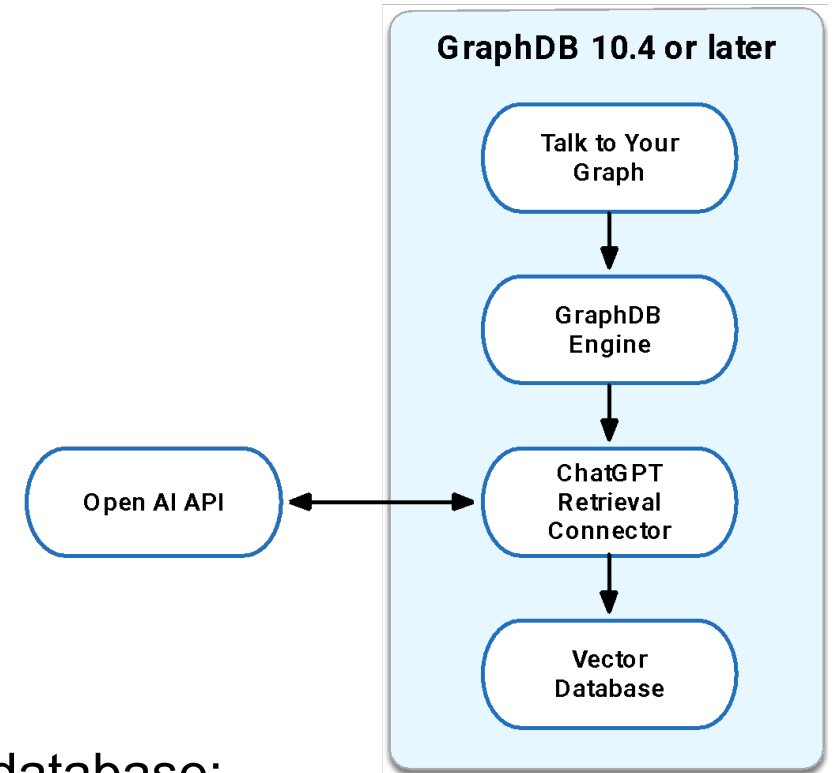
[PrefixDeclaration]
ex: https://ontotext.com/rdf/resource/data/
voc: https://ontotext.com/rdf/resource/schema#
skos: http://www.w3.org/2004/02/skos/core#

[MappingDeclaration]
@collection [
  mappingId data
  target ex:result_{ticker}_{date} a voc:FinancialResult ;
  skos:related ex:corporation_{ticker};
  voc:date {date}^^xsd:date ;
  voc:costRevenue {cost_revenue}^^xsd:string ;
  voc:costGood {cost_good}^^xsd:string ;
  voc:costAdmin {cost_admin}^^xsd:string ;
  voc:costRD {cost_rd}^^xsd:string ;
  voc:costOther {cost_other}^^xsd:string .
  source SELECT * FROM "data"
]
    
```

subject	predicate	object
https://ontotext.com/rdf/resource/data/result_AAL_2012-12-31	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	https://ontotext.com/rdf/resource/schema#FinancialResult
https://ontotext.com/rdf/resource/data/result_AAL_2012-12-31	http://www.w3.org/2004/02/skos/core#related	https://ontotext.com/rdf/resource/data/corporation_AAL
https://ontotext.com/rdf/resource/data/result_AAL_2012-12-31	https://ontotext.com/rdf/resource/schema#costAdmin	"\$12,977,000,000"
https://ontotext.com/rdf/resource/data/result_AAL_2012-12-31	https://ontotext.com/rdf/resource/schema#costGood	"\$10,499,000,000"
https://ontotext.com/rdf/resource/data/result_AAL_2012-12-31	https://ontotext.com/rdf/resource/schema#costOther	"\$845,000,000"
https://ontotext.com/rdf/resource/data/result_AAL_2012-12-31	https://ontotext.com/rdf/resource/schema#costRD	"\$-"
https://ontotext.com/rdf/resource/data/result_AAL_2012-12-31	https://ontotext.com/rdf/resource/schema#costRevenue	"\$24,855,000,000"
https://ontotext.com/rdf/resource/data/result_AAL_2012-12-31	https://ontotext.com/rdf/resource/schema#date	"2012-12-31"^^xsd:date

Talk To Your Graph

- SOTA for General Purpose Q&A
- Addresses key challenges of LLMs:
 - Lack Domain Knowledge;
 - Hard to Update Incrementally;
 - Black boxes prone to hallucination.
- How it works :
 - User selects parts of KGs to index;
 - Documents are indexed by OpenAI and stored in a vector database;
 - Full integration with SPARQL.





**THANK YOU FOR
YOUR TIME!**