Real-Time "RDFization"

Leveraging Linked Data Fragments for enhanced data publication: the Share-VDE case study

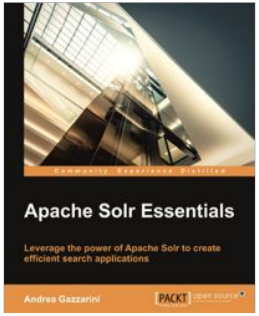
SWIB 2024, November 27th 2024

Andrea Gazzarini, Share-VDE Lead Architect

www.svde.org
info@svde.org

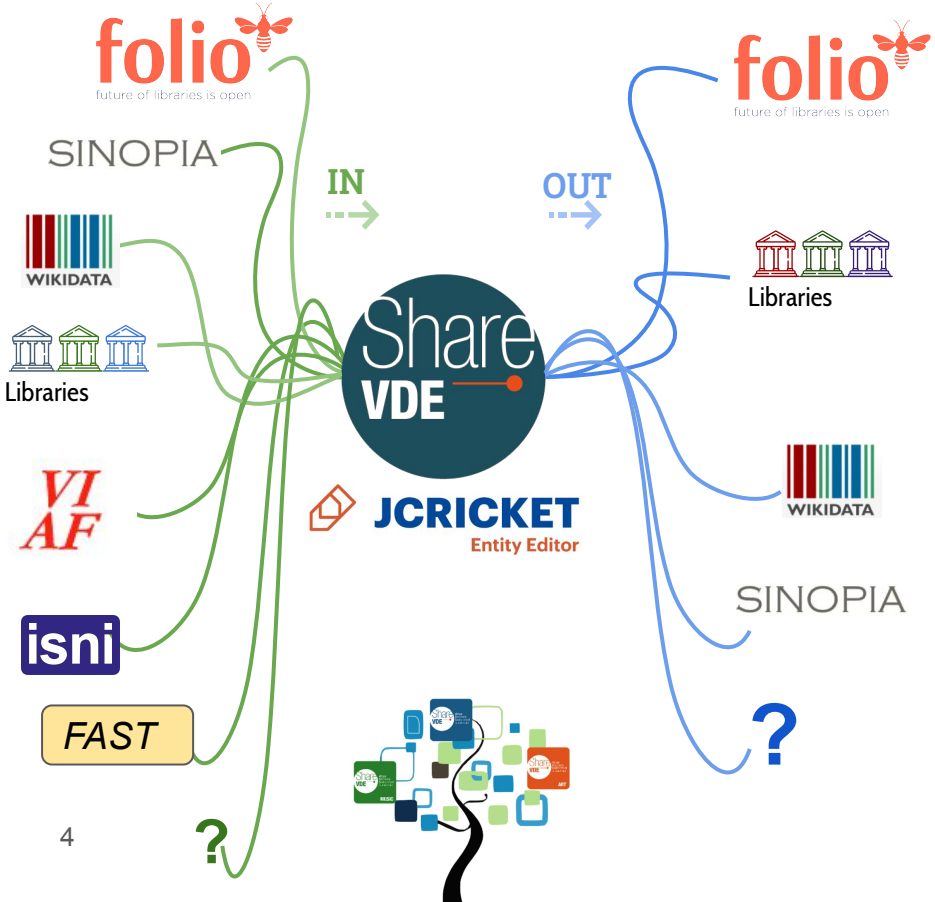
I, Andrea Gazzarini

- Software Engineer (1999-)
- “Hermit” Software Engineer (2010-)
- Programming Passionate
- Information Retrieval Passionate
- Author of “Apache Solr Essentials”
- Apache Qpid (past) Committer
- Founder of SpazioCodice
- Share-VDE Lead Architect
- Husband & Father
- Bass Player
- Chapman Stick (aspiring) Player



The Share-VDE Initiative

Share-VDE: Share Virtual Discovery Environment



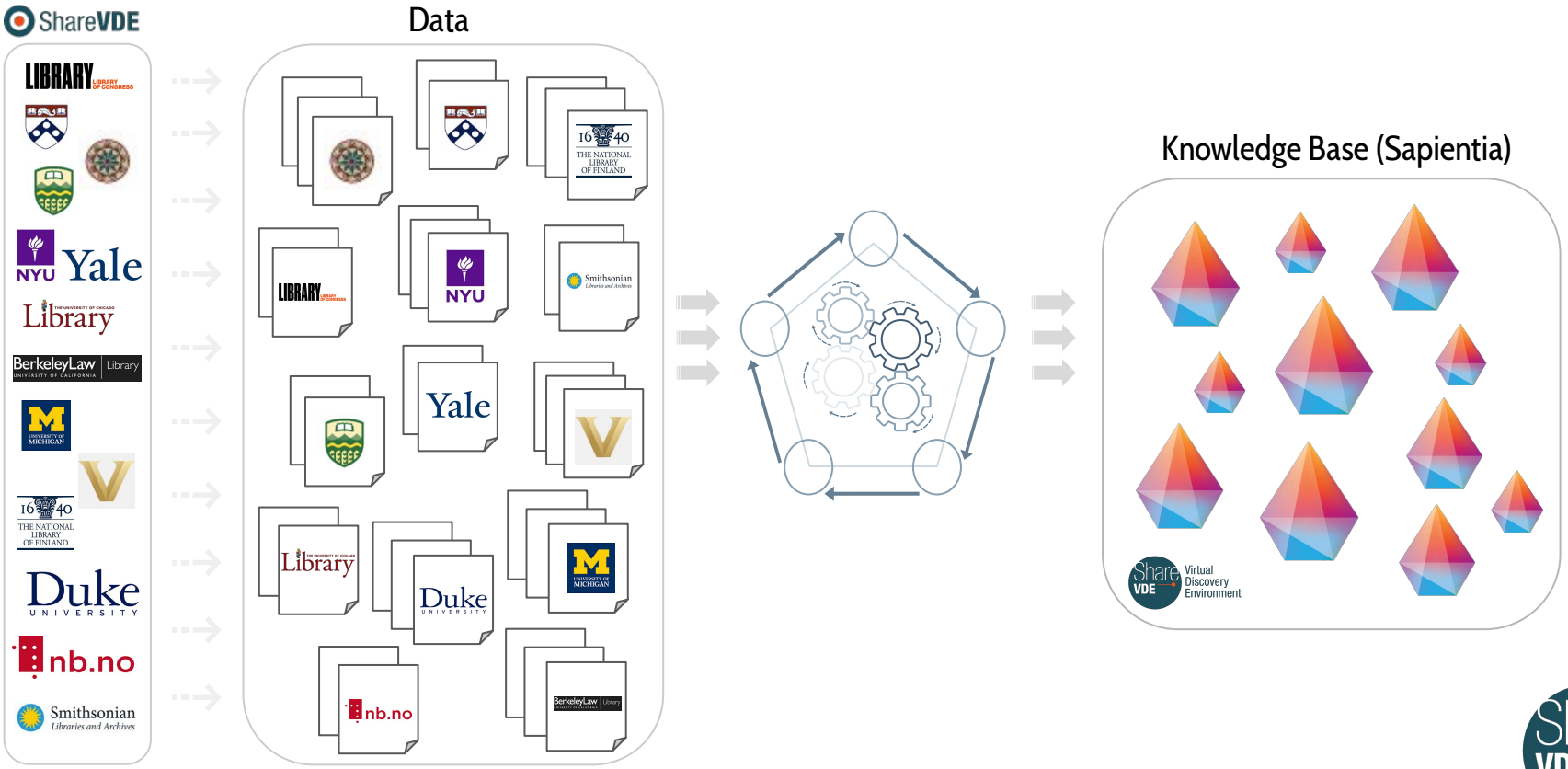
In a Nutshell

Share-Virtual Discovery Environment is a **library-driven** initiative which brings together, in a shared discovery environment, the **bibliographic catalogues** and **authority files** of a growing number of leading **academic** and **national libraries** from across **North America** and **Europe**.

https://svde.org

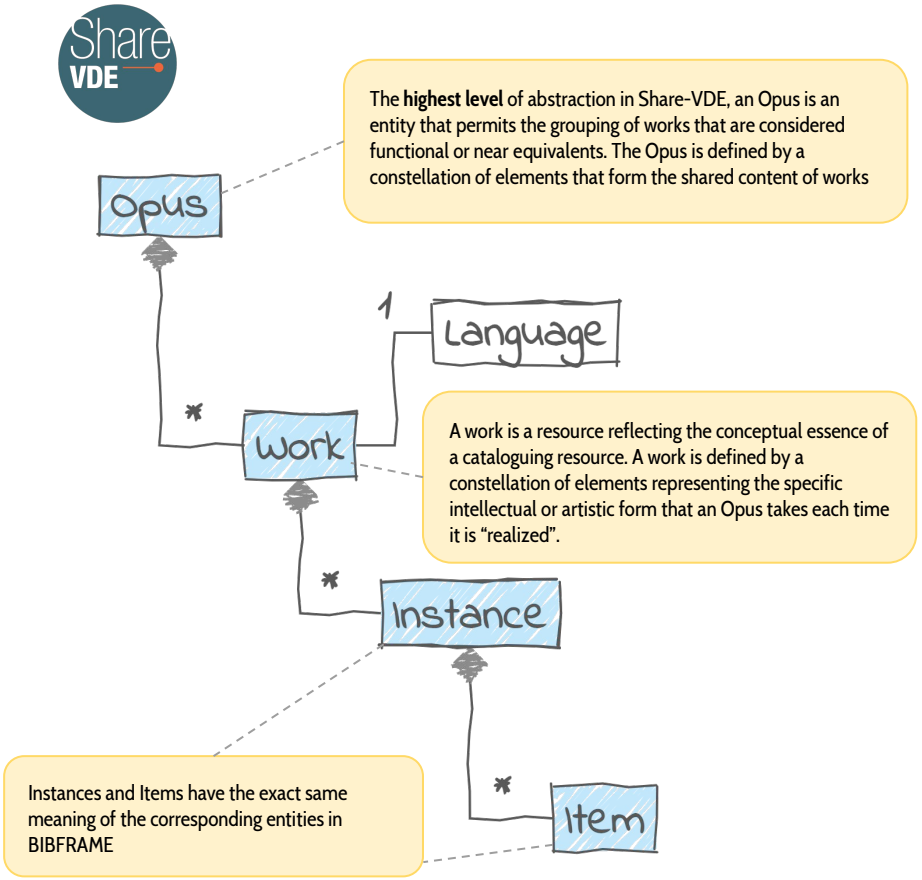
Sapientia: The Share-VDE Knowledge Base

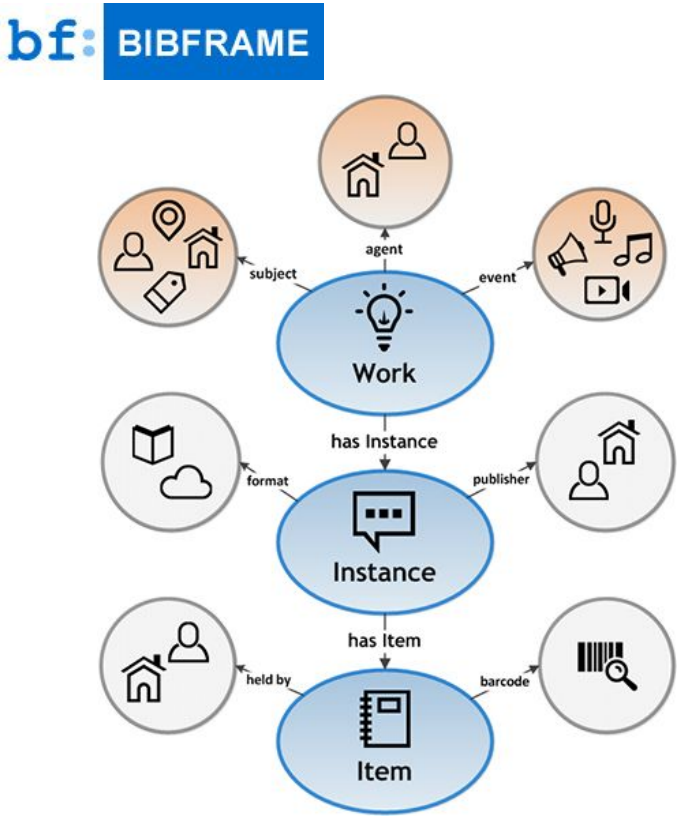
Sapientia: Genesis



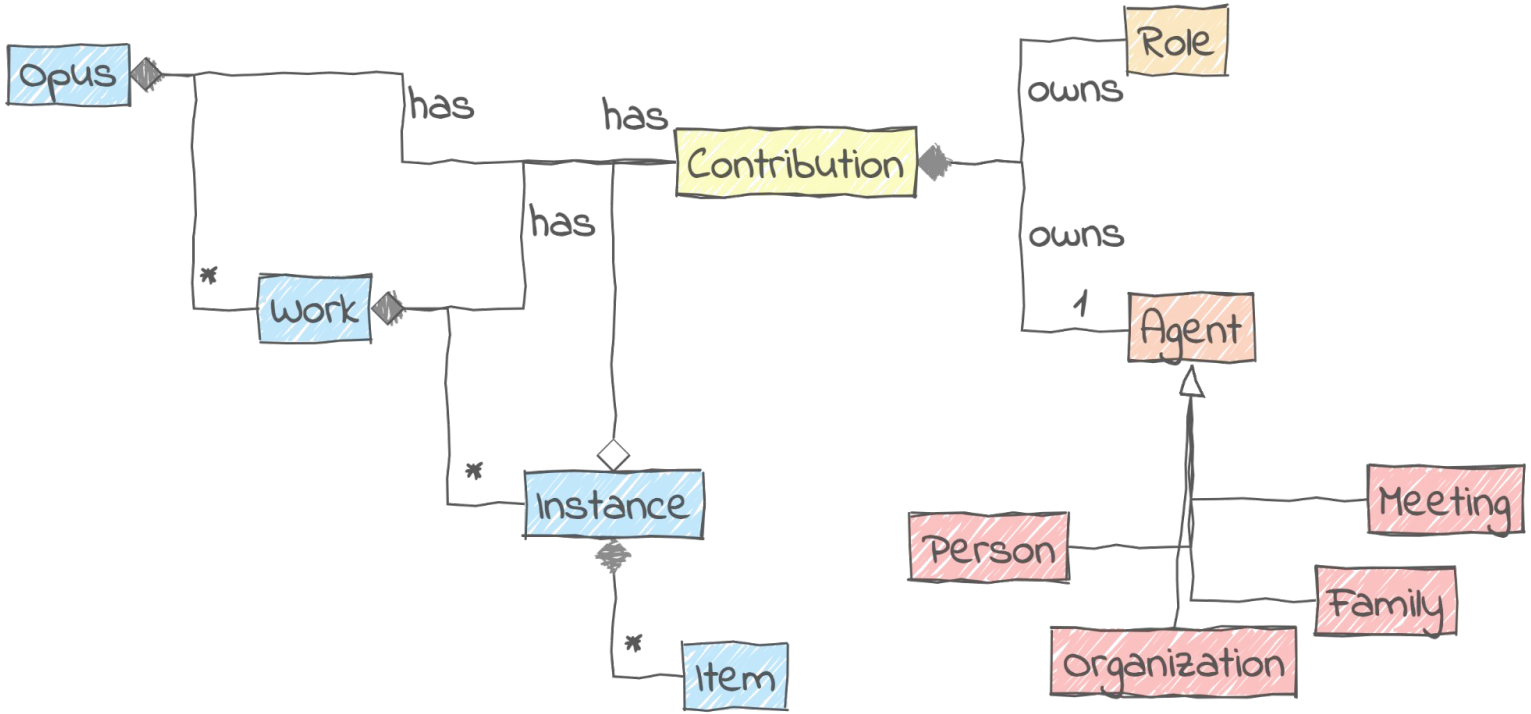
The Domain Model

Core entities

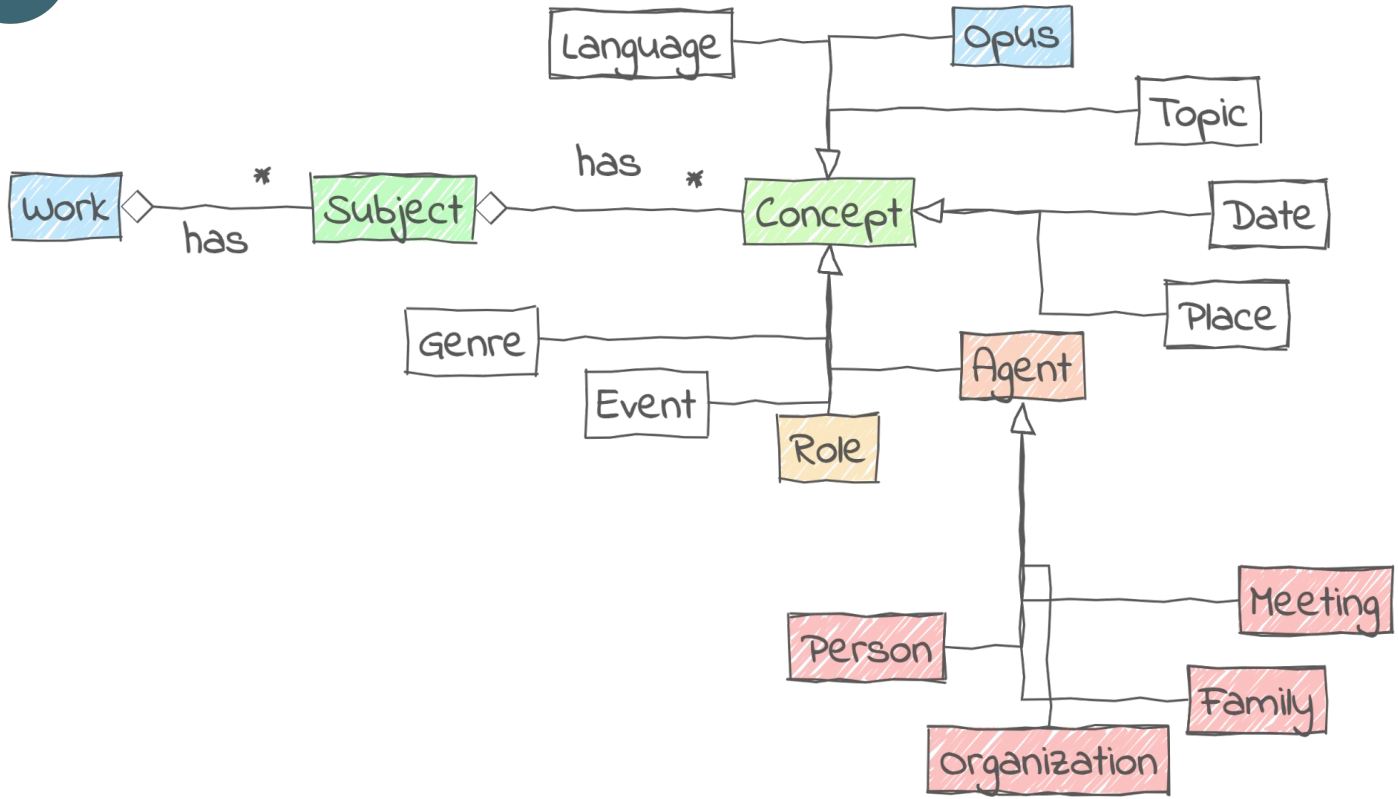




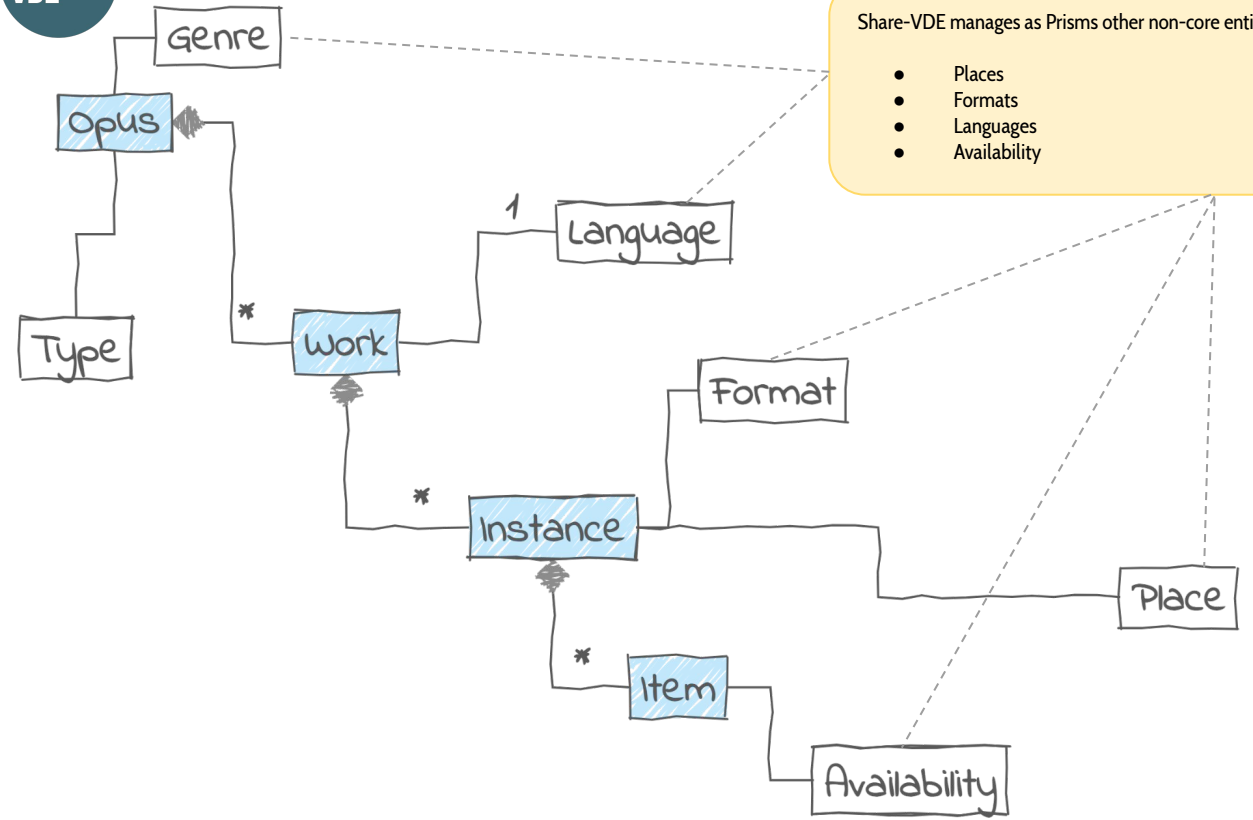
Agents, Contributions



Subjects


Language
Opus
Topic
Work
*
has
Subject
has
*
Concept
Date
Place
Genre
Event
Role
Agent
Meeting
Person
Family
Organization

Non-core Entities



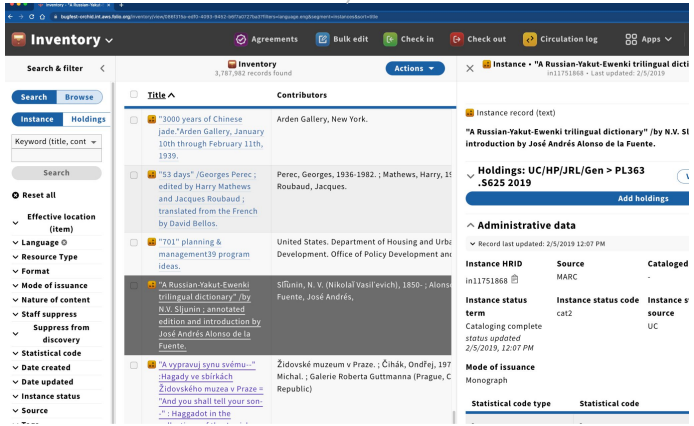
Share-VDE manages as Prisms other non-core entities, too. Some example

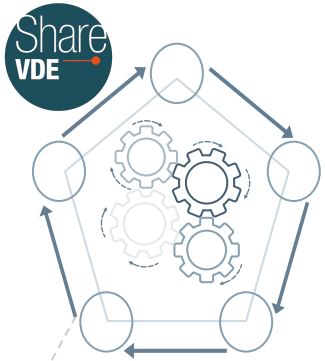
- Places
- Formats
- Languages
- Availability

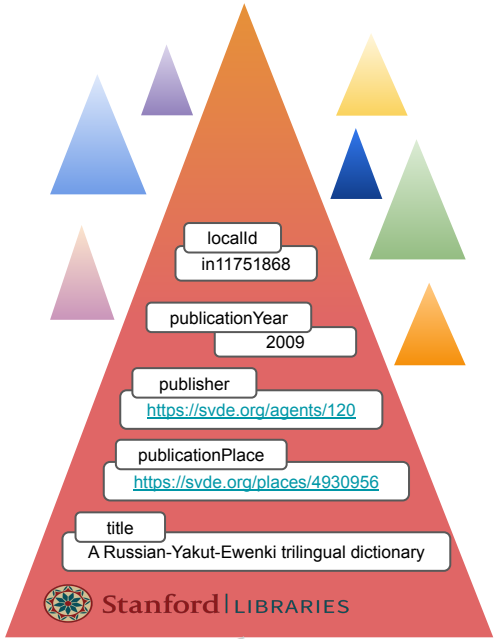
The Entity as a “Prism”

From Library Data to Sapientia

A Share-VDE member uses an ILS for managing its data.



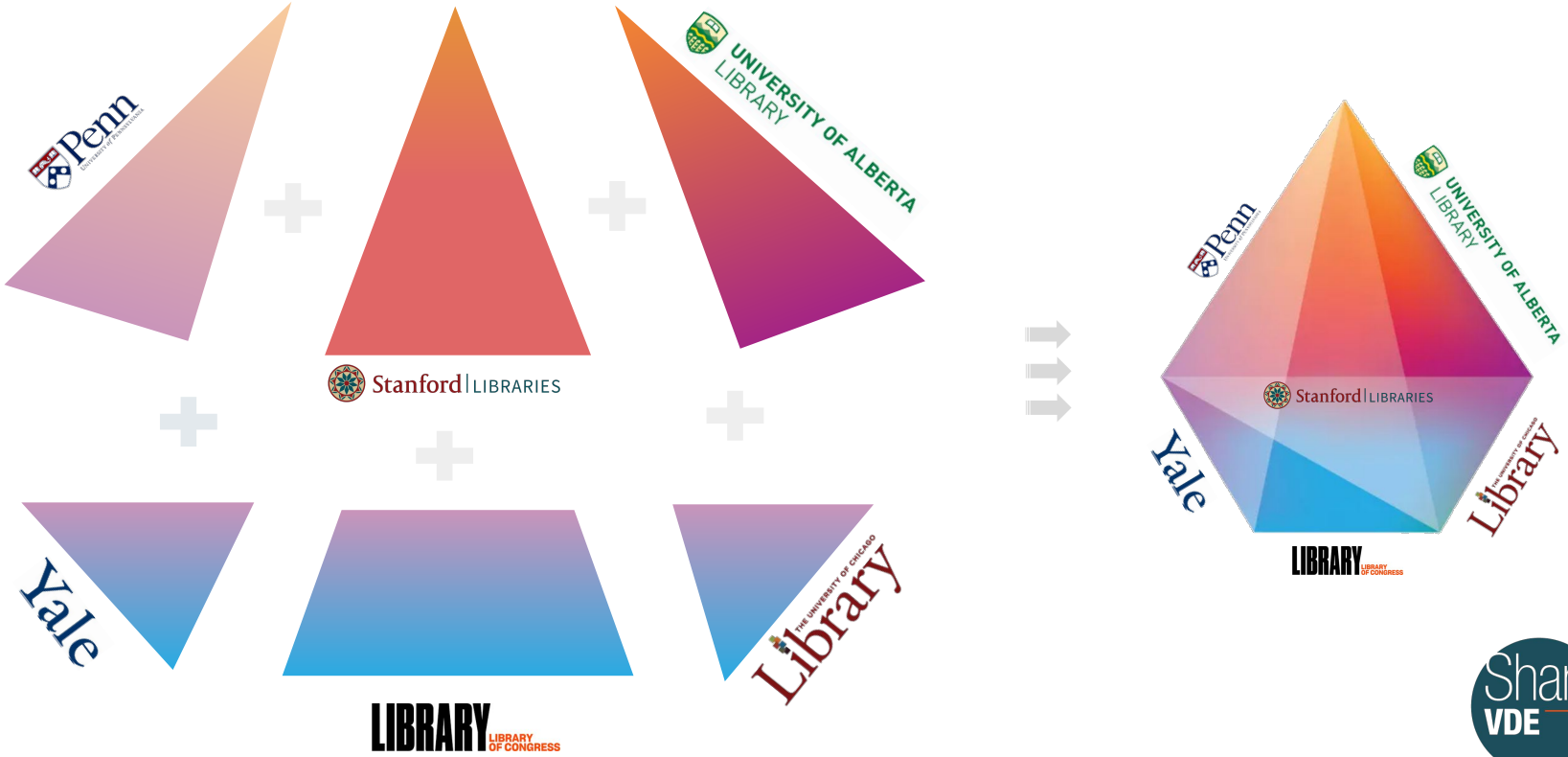




Library data is sent to Share-VDE, through API or offline batches.

Source data is split across the entities that form the Share-VDE domain model. In this example we focus on the properties that are assigned to a Share-VDE instance (red triangle above)

Prism, faces: the Share-VDE Entity



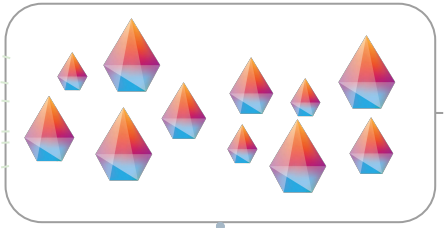
Linked Data Fragments

Share-VDE: The Big Picture

Data flows into Share-VDE from libraries, institutions and third-party sources (e.g. VIAF, ISNI, FAST)

The Share-VDE knowledge base (**Sapientia**) contains the integrated/clustered/enriched entities.

Data is mainly edited through JCricket, the Share-VDE entity editor.



JCRICKET
Entity Editor

GraphQL/REST

API

JDBC (Private)

RDBMS

HTTP (Private)

Search Engine

SPARQL

SPARQL/HTTP (Public)

RDF Store

Changes are asynchronously propagated towards the three data storages

Share VDE

Let's analyze a (simple) SPARQL Query

```
PREFIX opuses: <https://svde.org/opuses/>
PREFIX works: <https://svde.org/works/>
PREFIX instances: <https://svde.org/instances/>
PREFIX items: <https://svde.org/items/>
PREFIX bf: <http://id.loc.gov/ontologies/bibframe/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?barcode
WHERE {

    opuses:401 bf:hasExpression ?work .

    ?work bf:hasInstance ?instance .

    ?instance bf:hasItem ?item .

    ?item bf:isIdentifiedBy ?uri .

    ?uri rdf:value ?barcode
}
```

Prefixes: useful for associating a (long) URI to a short mnemonic code in the query.

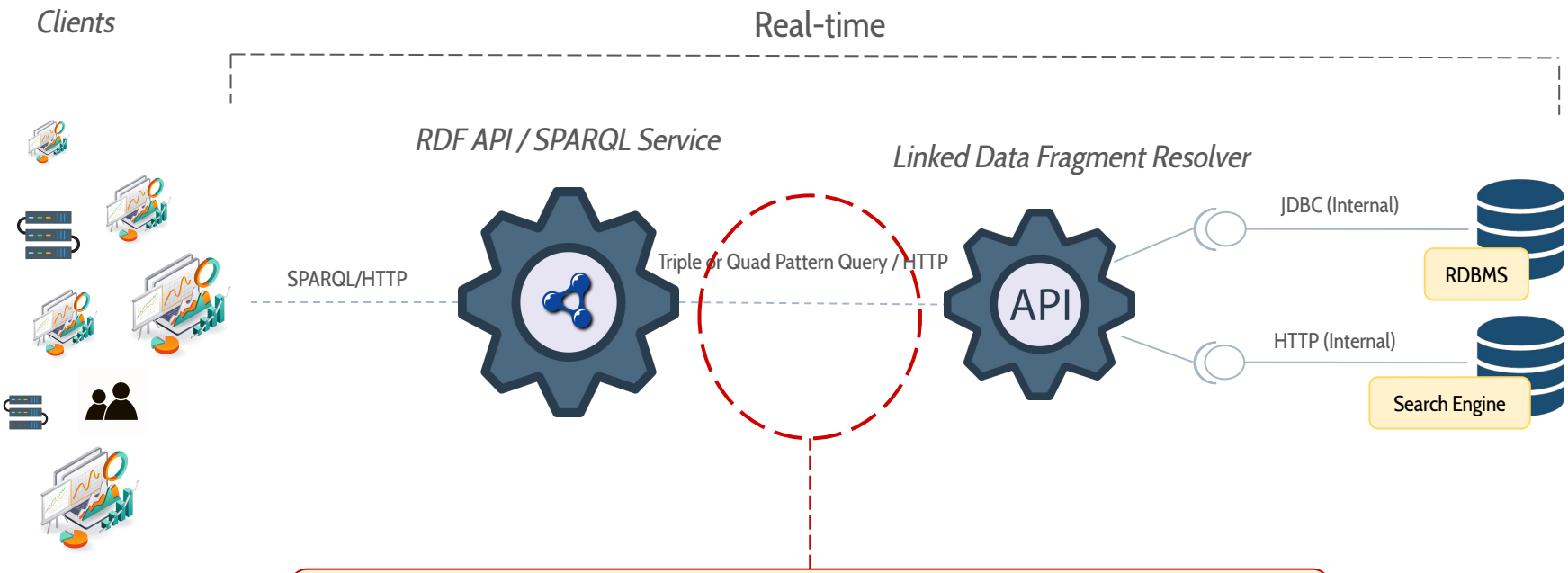
A variable called ***?barcode*** referenced in the query below, whose value(s) will compose the output results

Query statements, composed by a subject, a predicate and an object, ending with a dot.
The three parts can be an explicit value (e.g. bf:hasExpression) or a variable, eventually bound with another statement (see the ?work variable). For that reason they are also referred as **Triple Patterns**

Simplifying, we could say a SPARQL query is a set of multiple triple patterns, potentially independent and executable as an atomic computation units.

Their execution offers a partial view of the whole SPARQL result, a **Fragment**, a **Linked Data Fragment**

Linked Data Fragments: Participants



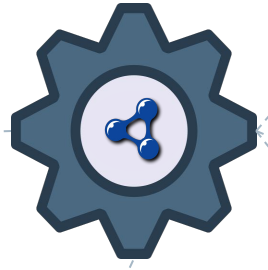
The **RDF representation** of the requested data **is created on the fly**, according to **one or more ontologies** that can be indicated in the request, as well.

Linked Data Fragments In Action

SPARQL/HTTP

Triple Pattern #1
Triple Pattern #2
Triple Pattern #3

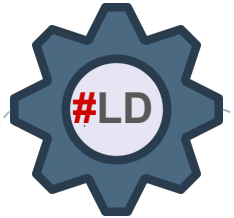
Linked Data Fragment Client
OR
SPARQL Endpoint

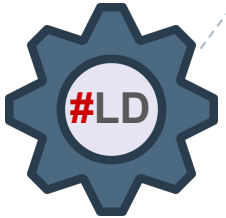


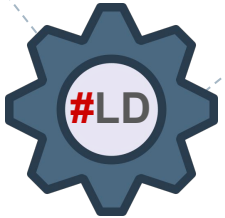
Triple Pattern #1

Triple Pattern #2

Triple Pattern #3







A Triple Pattern Server (or Linked Data Fragment Server) exposes a HTTP endpoint which is able to compute and resolve (i.e. produce the corresponding triples) a given pattern.

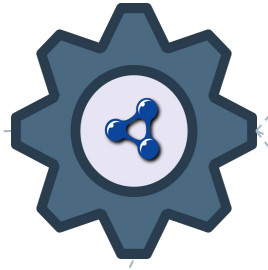
- Provides a SPARQL Endpoint to clients
- Receives the SPARQL query
- Destructures the SPARQL query in triple patterns
- Optimizes / rewrites / reorders the patterns
- Calls the Triple Pattern Server(s) for each pattern
- Merge the results
- Returns the response to clients

Scaling up Linked Data Fragment Resolvers...

SPARQL/HTTP

Triple Pattern #1
Triple Pattern #2
Triple Pattern #3

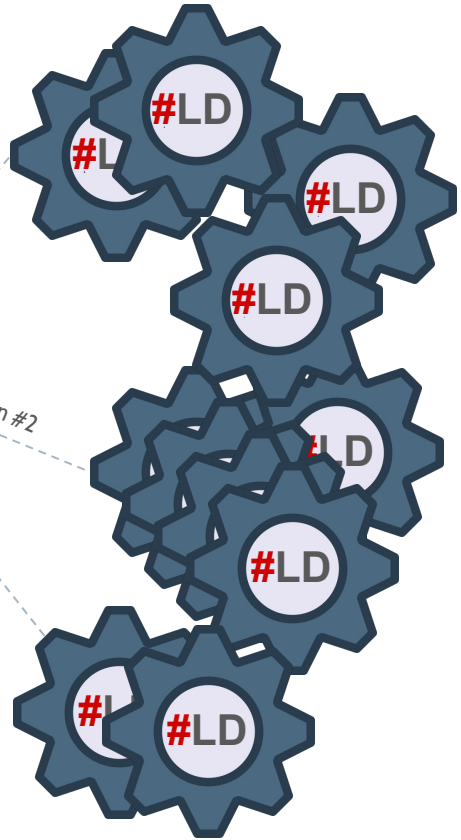
Linked Data Fragment Client
OR
SPARQL Endpoint



Triple Pattern #1

Triple Pattern #2

Triple Pattern #3



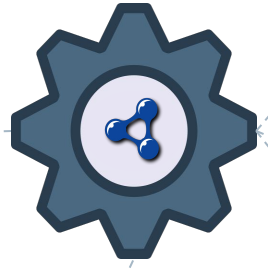
- Provides a SPARQL Endpoint to clients
- Receives the SPARQL query
- Destructures the SPARQL query in triple patterns
- Optimizes / rewrites / reorders the patterns
- Calls the Triple Pattern Server(s) for each pattern
- Merge the results
- Returns the response to clients

...and the Datasource layer behind

SPARQL/HTTP

Triple Pattern #1
Triple Pattern #2
Triple Pattern #3

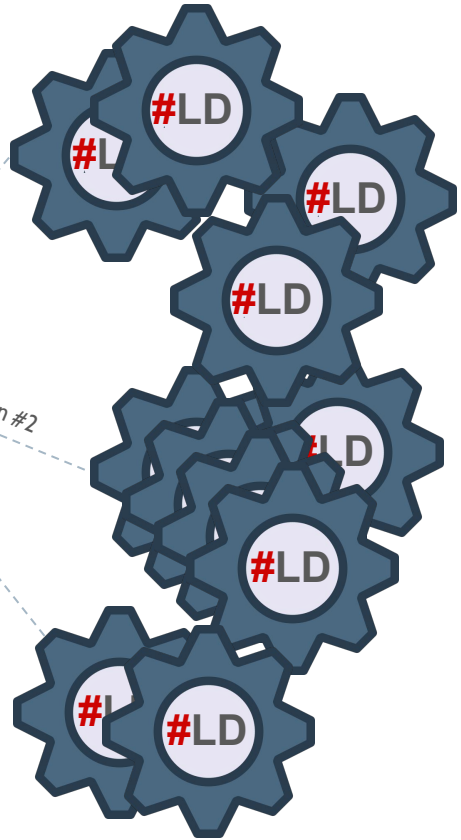
Linked Data Fragment Client
OR
SPARQL Endpoint



Triple Pattern #1

Triple Pattern #2

Triple Pattern #3



- Provides a SPARQL Endpoint to clients
- Receives the SPARQL query
- Destructures the SPARQL query in triple patterns
- Optimizes / rewrites / reorders the patterns
- Calls the Triple Pattern Server(s) for each pattern
- Merge the results
- Returns the response to clients

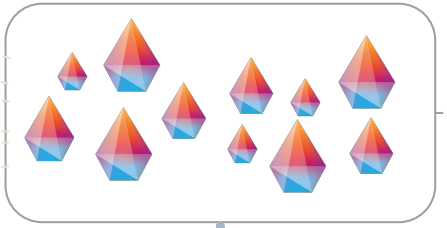
(Let’s Simplify The) Architecture

Share-VDE: The Big Picture

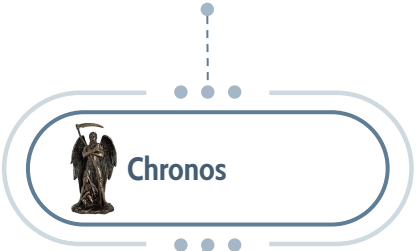
Data flows into Share-VDE from libraries, institutions and third-party sources (e.g. VIAF, ISNI, FAST)

The Share-VDE knowledge base (**Sapientia**) contains the integrated/clustered/enriched entities.

Data is mainly edited through JCricket, the Share-VDE entity editor.



JCRICKET
Entity Editor



GraphQL/REST

API

JDBC (Private)

RDBMS

HTTP (Private)

Search Engine

SPARQL

SPARQL/HTTP (Public)

RDF Store

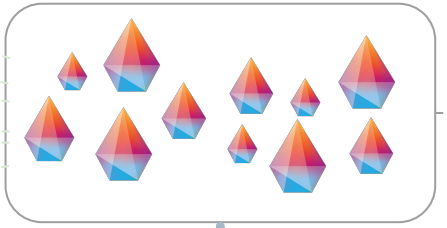
Changes are asynchronously propagated towards the three data storages

Share-VDE: The Big Picture

Data flows into Share-VDE from libraries, institutions and third-party sources (e.g. VIAF, ISNI, FAST)

The Share-VDE knowledge base (**Sapientia**) contains the integrated/clustered/enriched entities.

Data is mainly edited through JCricket, the Share-VDE entity editor.



JCRICKET
Entity Editor

GraphQL/REST

API

JDBC (Private)

NoSQL

HTTP (Private)

Search Engine

SPARQL

Linked Data Fragments in Share-VDE: benefits

No RDF Storage

- RDF Data is translated/generated on demand.

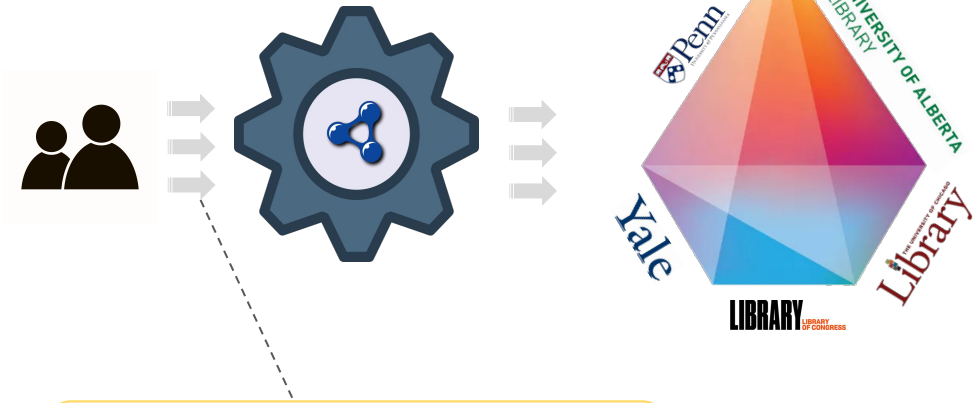
Distributed Computation

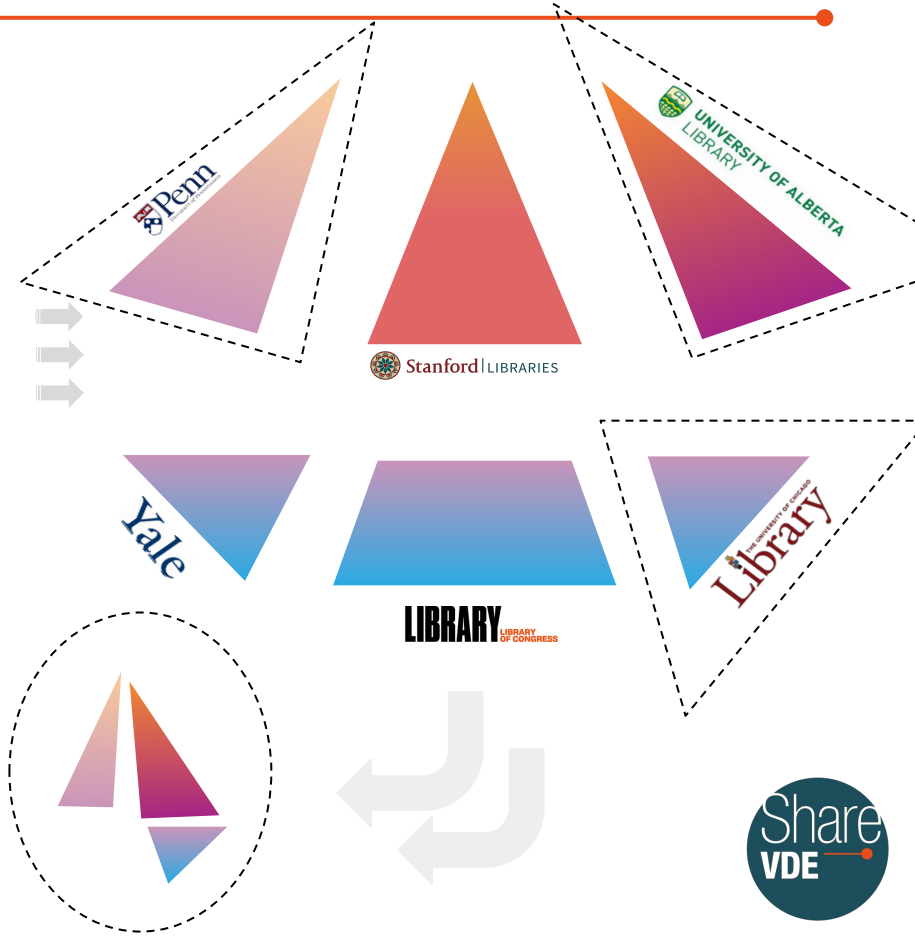
- **Computation is distributed** across the **Linked Data Client** (the SPARQL endpoint) and the **Triple/Quads Pattern Server**
  - The **destructuration**, the **optimization/rewriting** of the SPARQL query is done in the **Linked Data Client**
  - The **execution** of each single t**riple/quad pattern** is done at **Linked Data Fragment Server** level
- The CKB is required to answer to a **lot of small and simple requests**, instead of dealing with **one huge query**

Query Time

- **Request-driven approach benefits.**
  - (Example) No fixed mapping, **different queries** can request a **different mapping** in results
  - (Example) using the same query, requesters can **selectively ask for specific prism faces**
- **Federated search** is natively enabled

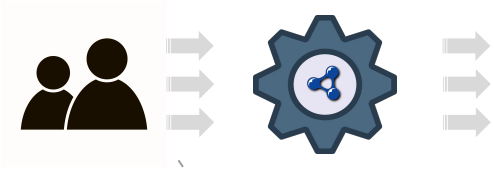
Query Time: Provenance-based de-structuration





GET http://svde.org/agents/201
...
x-svde-provenances: upenn, ualberta, uchicago

Query-Time Response “Shaping”



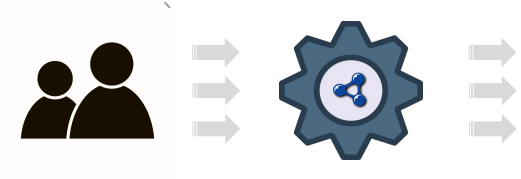
```
<bf:Agent rdf:about="http://svde.org/agents/201">
     <rdf:type rdf:resource="http://id.loc.gov/ontologies/bibframe/Person"/>
     <wkd:P569>27 January 1832</wkd:P569>
     <wkd:P570>14 January 1898</wkd:P570>
     <wkd:P735>Carroll</wkd:P735>
     <wkd:P734>Lewis</wkd:P734>
     ...
</bf:Agent>
```

GET http://svde.org/agents/201
...
x-svde-mapping: xbf

xbf = BIBFRAME + Wikidata

GET http://svde.org/agents/201
...
x-svde-mapping: xdc

xdc = DublinCore + schema.org



```
<rdf:Description rdf:about="http://svde.org/agents/201">
        <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
        <dc:name>Carroll, Lewis</dc:name>
        <schema:birthDate>1832-01-27</schema:birthDate>
        <schema:deathDate>1898-01-14</schema:deathDate>
        ...
</rdf:Description>
```

Thank you!

Thank you!

Real-Time “RDFization”

Leveraging Linked Data Fragments for enhanced data publication: the Share-VDE case study

SWIB 2024, November 27th 2024

Andrea Gazzarini, Share-VDE Lead Architect

www.svde.org
info@svde.org