Linked data in a shared catalogued form

Share Family tenants and orchestration of data flow

https://wiki.svde.org
info@svde.org

Share Family tenant infrastructure

The Share Family core principles

- Redundancy of data is complex to manage
- Linking entities is easier than duplicate data
- Cooperate and maintain autonomy at the same time
- Homogeneity of datasets and possible services to be shared
- Centralize core data through a lightweight method
- Distribute the technologic load to achieve long-term sustainability
- Profile levels of cooperation among systems and initiatives

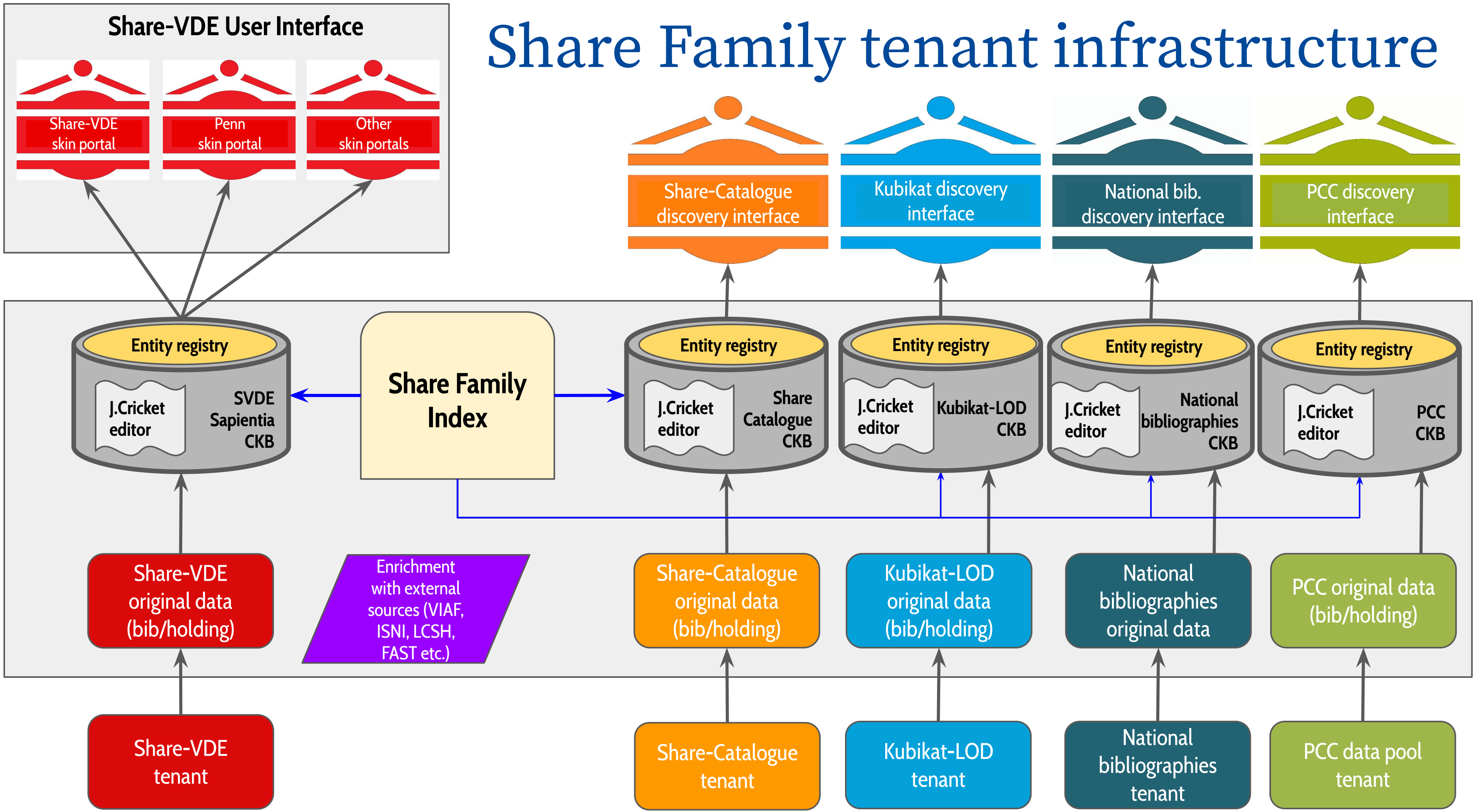
Solution in the Share Family architecture

Creation of more branches in the Share Family, named tenants

Consistent groups of institutions gathered by similar scope or from the same domain:

- Share-VDE
- Share-Catalogue
- PCC data pool
- National Bibliographies Group
- Kubikat-LOD
- Parsifal project (network of ecclesiastical university libraries in Rome)

Share Family tenant infrastructure



Benefits

- More efficient data management
- Technological sustainability
- Dedicated applications and services tailored to the institutions members of the various branches
- From the users perspective this enables richer and specialized sets of resources to be used.

What we have done so far

Live tenants and skin portals

- **SVDE tenant -** www.svde.org => with LC's authority data and the bibliographic data of Stanford, UPenn and Smithsonian Institution. Further catalogues of participant libraries will follow in July.

  In addition the specific skins for the following institutions containing their respective data are:
  https://penn.svde.org (integrations via APIs will follow)
  https://stanford.svde.org

- **PCC tenant -** https://pcc-lod.org => with the PCC datapool.

- **National Bibliographies tenant -** https://natbib-lod.org
  with the skin for the British National Bibliography https://bl.natbib-lod.org (*)

- **Kubikat tenant -** https://kubikat-lod.org
  Kubikat art history libraries group

  (*) *Note: the skin for the British National Bibliography is a preview of a beta side.*

Maximise efforts - Promote autonomy

The main purpose of this centralized architecture is to ensure **long-term sustainability** while favoring the **autonomy** of each tenant.

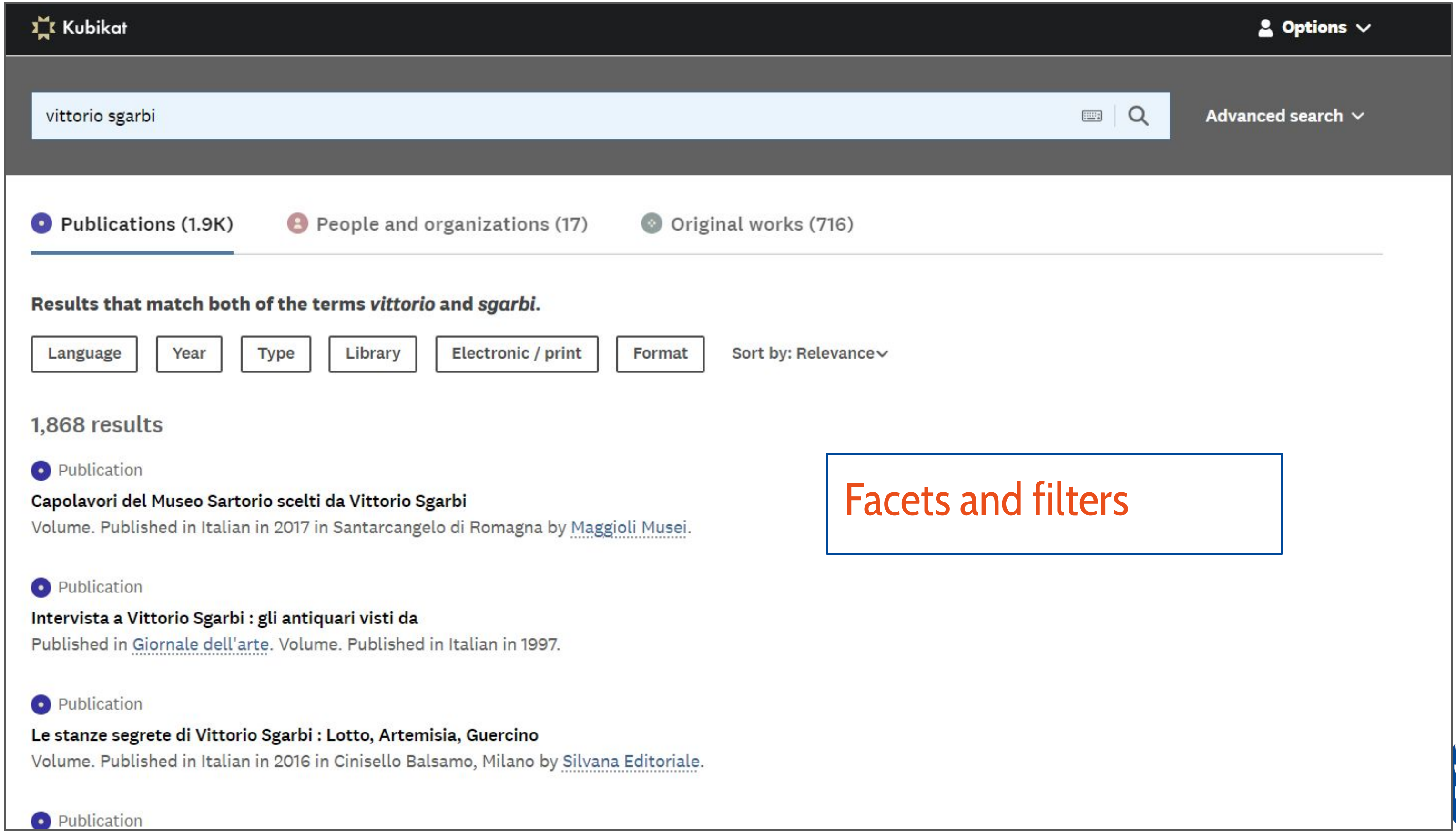
To foster this vision, it is essential to avoid ad hoc developments while ensuring the ability of local customizations. This flexibility is achieved through mechanisms that allow each tenant to selectively enable functions according to the purpose:

- on/off mechanism

- optional default configurations

- local features/services

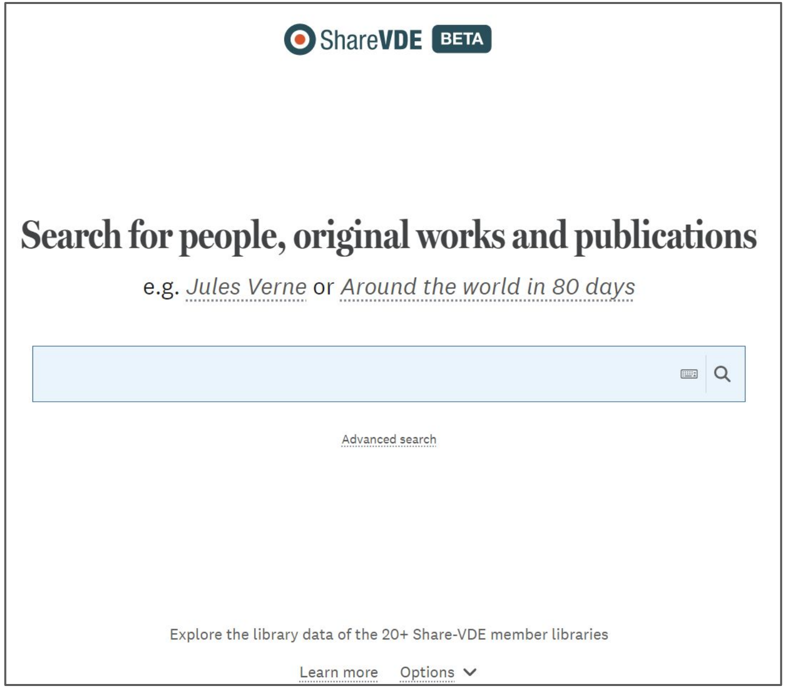
On/Off mechanism example



Facets and filters

Default configuration: SVDE and PCC data pool

Simple search default configuration



Default simple search configuration: the BNB

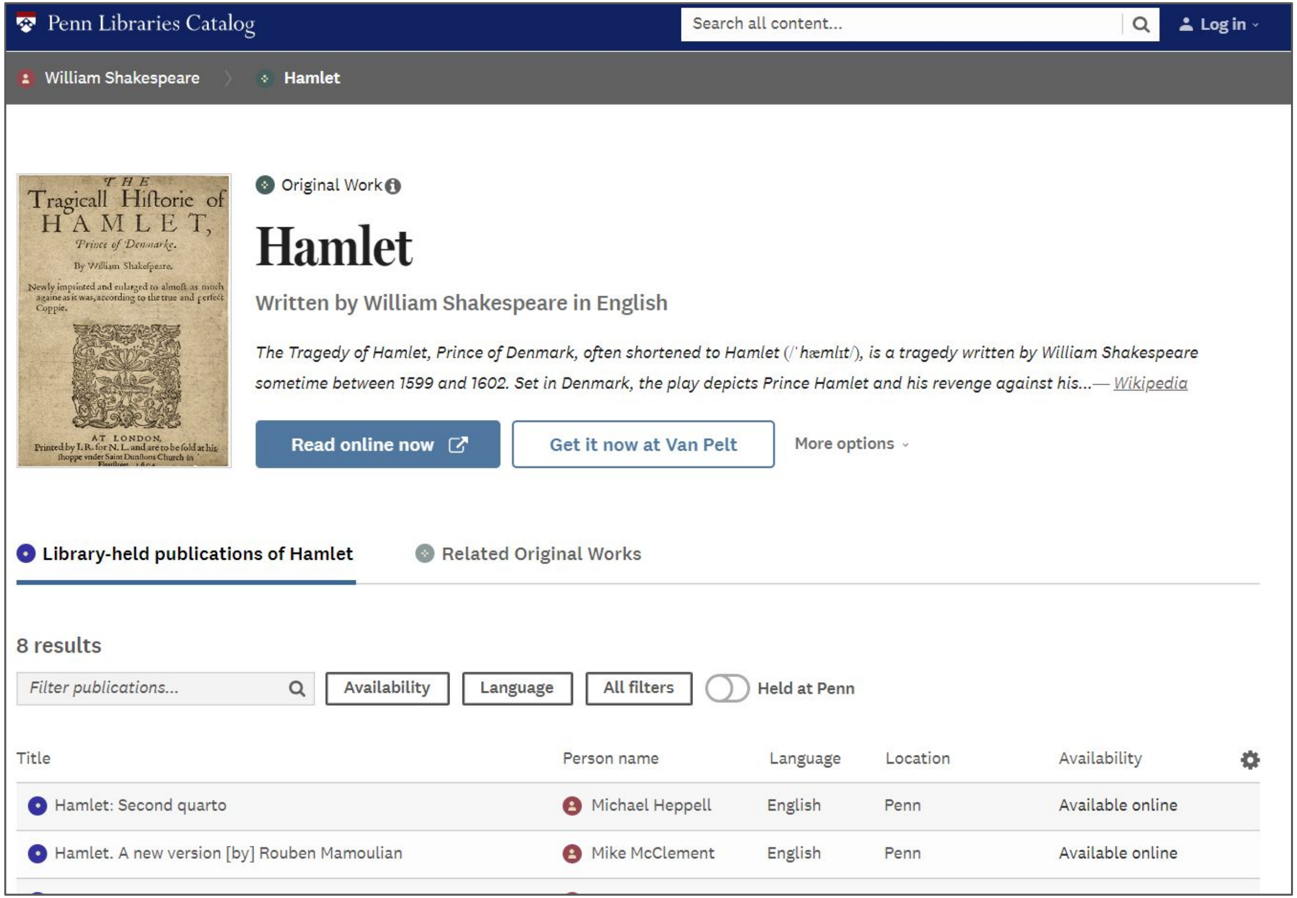
Simple search default configuration on Natbib tenant and the BNB - British National Bibliography skin* is set to Publications search, instead of the SVDE default.

This was done to comply with a different requirement whereby for the data stored in this tenant (ie. national bibliographies) it's meaningful to direct users to publications.

Different communities or types of institutions might need customised features

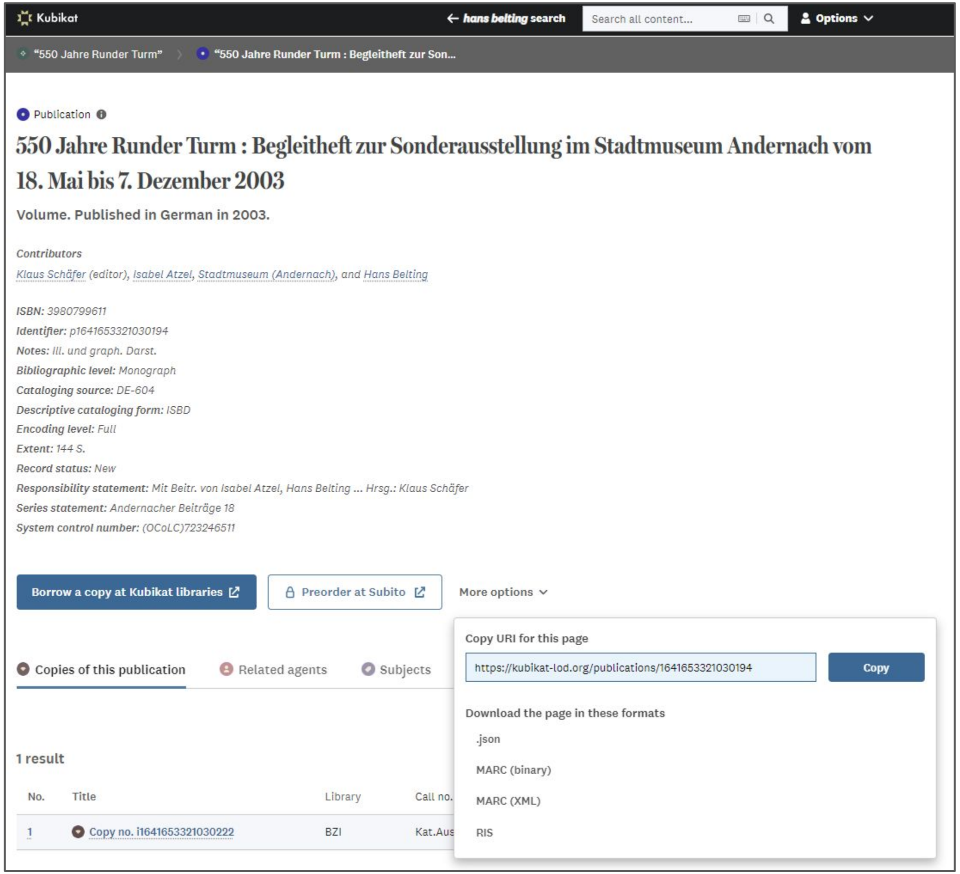
(*) *Note: the skin for the British National Bibliography is a preview of a beta site*

Local services: University of Pennsylvania



[Penn circulation services]

Local services: Kubikat-LOD pilot project



[Kubikat-LOD lending and Subito ordering service integration]()

What we will do

Share Family Index and Share Family Identifier

Share Family Index

SFI - Share Family Index

- index that centralizes core entity data from each CKB (e.g. entities URIs and very few data only for search and redirection purposes)
- central index able to point to all the URIs in all the CKBs of the different tenants within the Share Family

SFId - Share Family Identifier

- each entity has a unique URI within the tenant's CKB namespaces
- all the URIs for the same entity in different namespaces are grouped under a unique Share Family Identifier in the Share Family Index
- the SFId is a unique identifier linking to URIs that identify the entities in each CKB
- the SFId carries the minimum amount of data needed to identify the entities

Share Family Index and Share Family Identifier

E.g. Ernest Hemingway URIs in different CKB namespaces and the corresponding Share Family Identifier in the Index (the following URIs are for simulation purposes):

```
http://sfi/agents/456789 [Share Family Identifier]
    sameAs
https://svde.org/agents/101631288986955
    sameAs
https://svde.org/pcc/agents/7890123
    sameAs
http://kubikat-lod.org/agents/456789
```

Central orchestration: the SFI - Share Family Index

The Share Family Index is:

- a registry which is in charge to create and assign a unique **Share Family Identifier (SFId)**
- a central index that aggregates URIs for entities that are stored in different CKBs
- a metasearch engine able to run queries in all the Share Family tenants
- an orchestrator of queries and messages between tenants according to service user agreements/profiles, functioning similarly to the ESB - Enterprise Service Bus
- a pointer to entities URIs in the individual CKBs of the different tenants

The Share Family Index is not:

- a database/storage
- a CKB
- a Search Engine

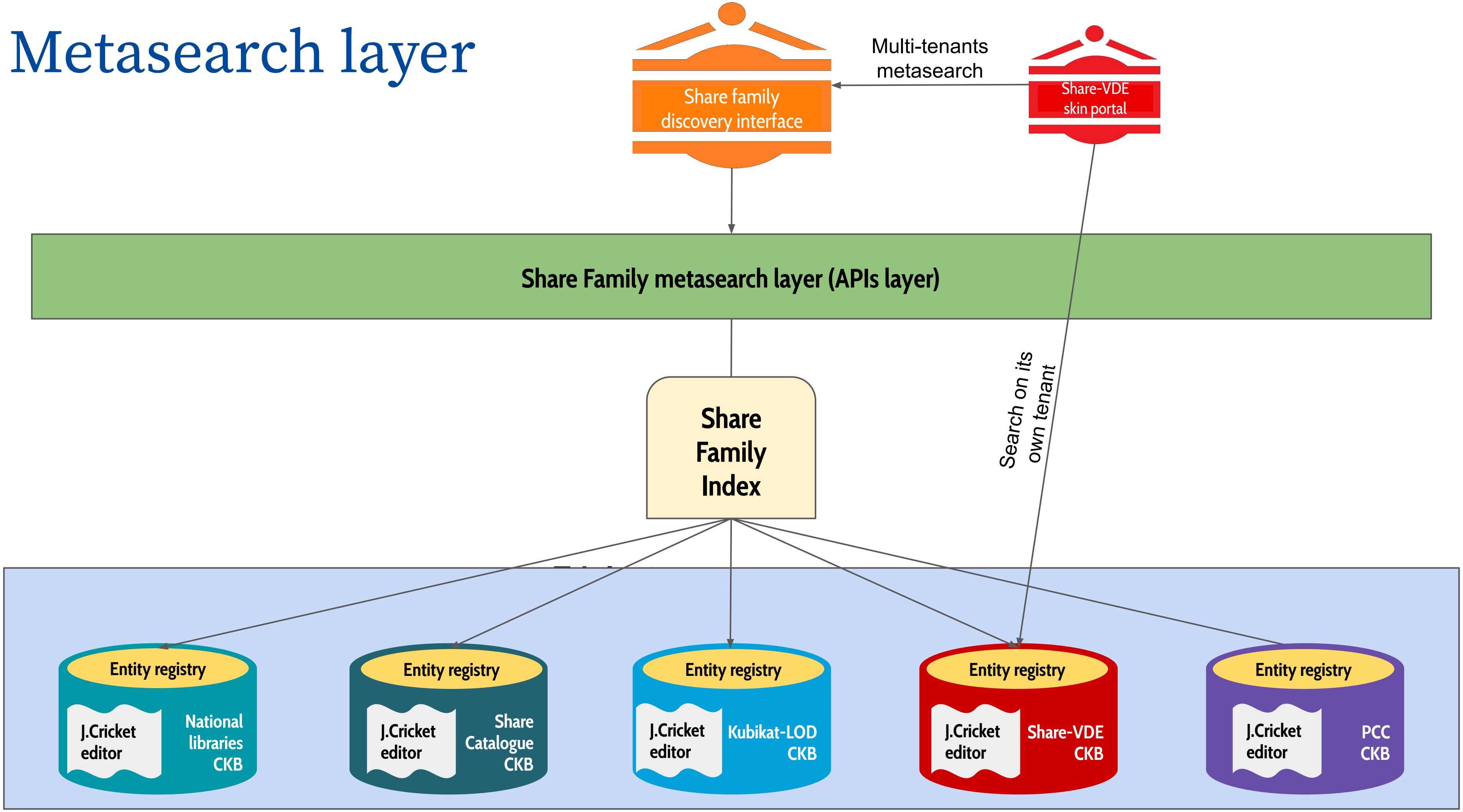
Examples of interaction use cases

1. Change alert only: two (or more) tenants hold the same entity; ***tenant 1*** receives a notification that some data of the entity has been changed in ***tenant 2***.

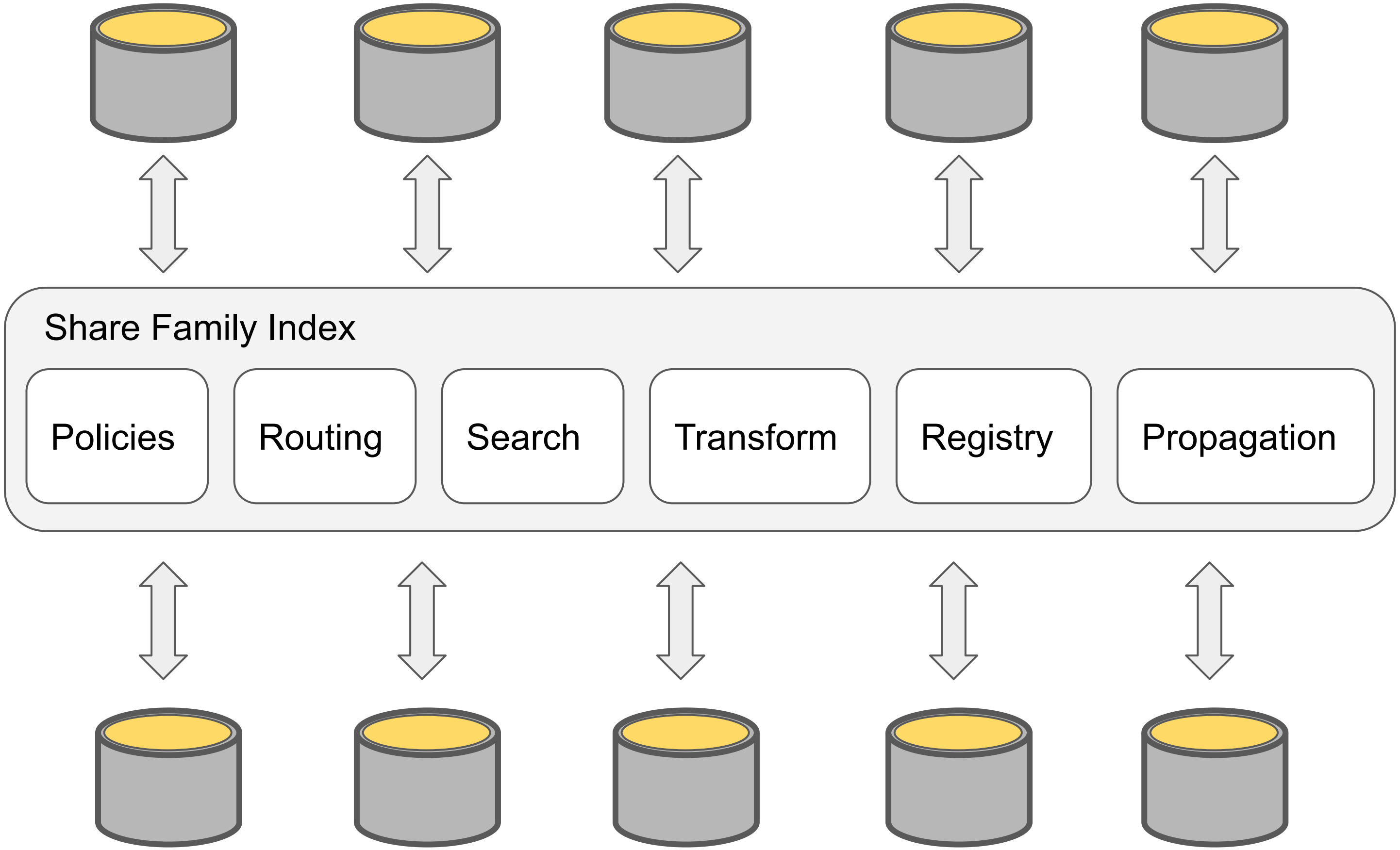
2. Propagate changes done to entity data: two (or more) tenants hold the same entity; ***tenant 1*** changes some data of the entity and propagates the changes to ***tenant 2***.

- Example: A library using Sinopia creates an Instance that already exists in the PCC data pool of SVDE
  - Scenario 1: every time an entity undergoes changes, the SFI receives push notifications from Sinopia and vice versa
  - Scenario 2: periodic triggers unsolicited to collect and register changes

Metasearch layer

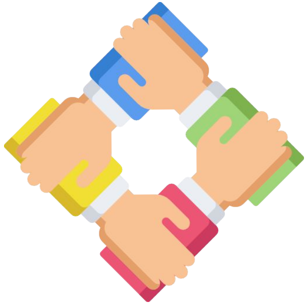


Share Family Index: an ESB++ for the tenants



Levels of cooperation

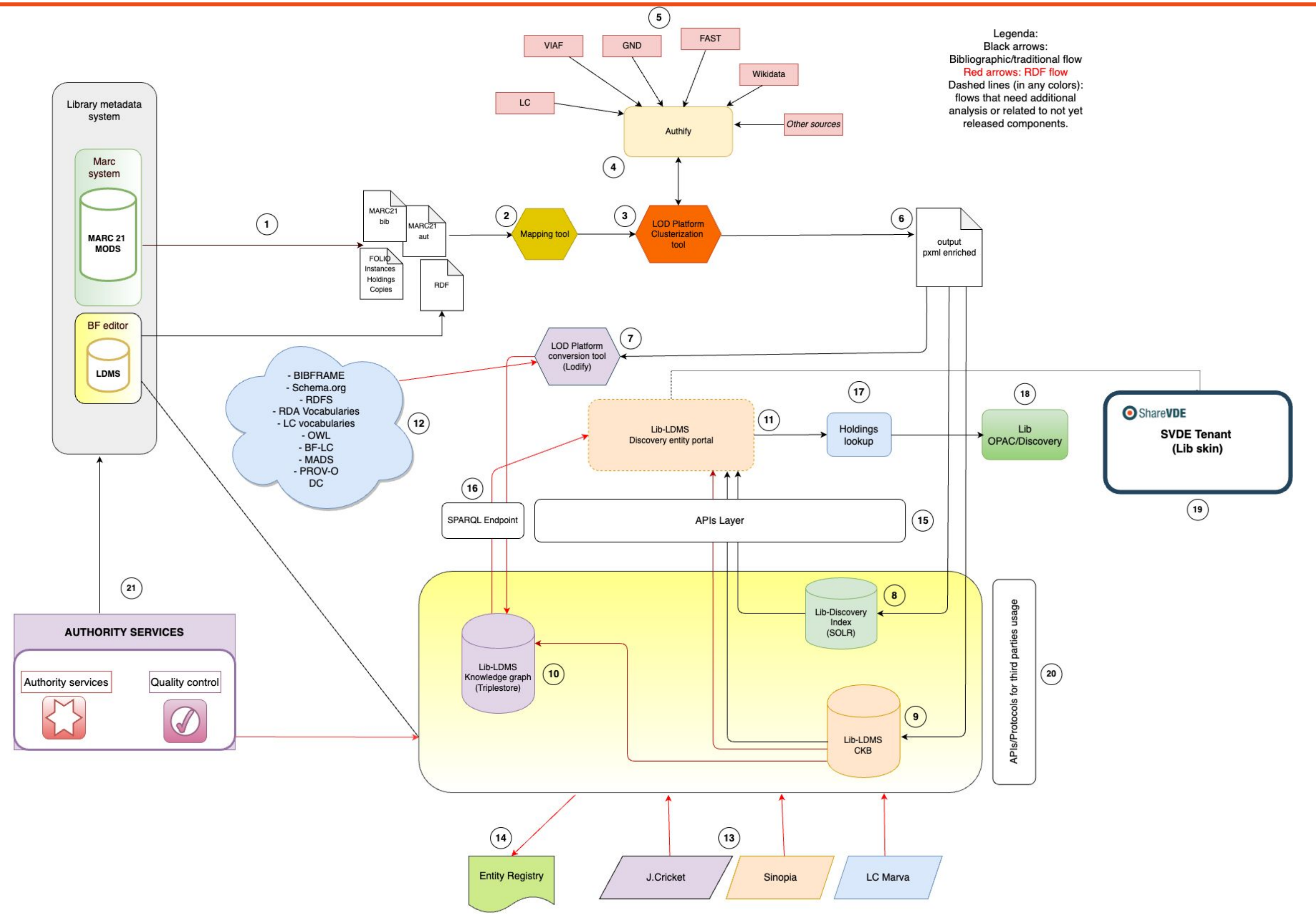
Consensus from the groups of Institutions involved



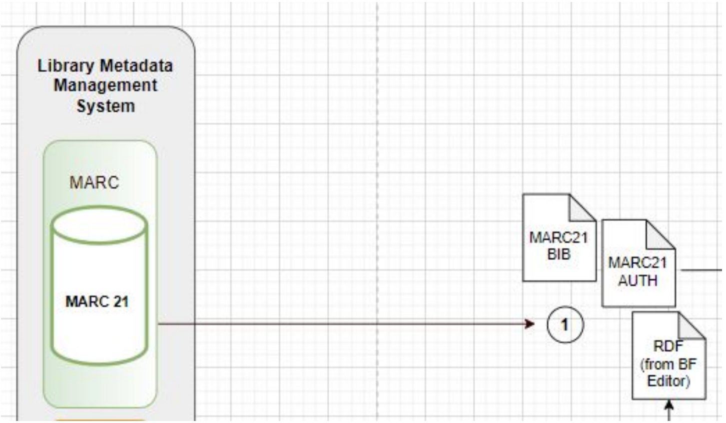
Ad hoc agreements to set up among the communities/groups of institutions to establish Service User Profiles

Orchestration of data flow

Share data flow



Share data flow - Step 1



1) Main event: the Share-VDE member library exports the records to SVDE system in one of the supported formats

Extended description

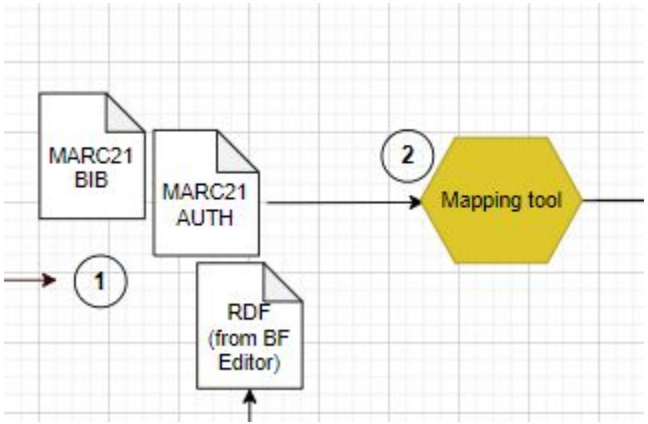
Original source data, produced in different formats in each library (through its own Library Metadata Management System) e.g.:

MARC 21 bibliographic records;

MARC 21 authority records;

RDF from LoC editor or from Sinopia (or other source): in case an RDF editor will be included in the workflow, such as Sinopia - see item 13 - the RDF format will be managed as original format, for data created as linked data from scratch.

Share data flow - Step 2



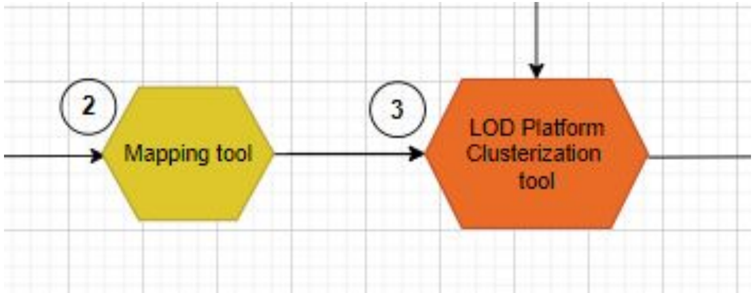
2) Main event: SVDE analyses the data to comply with internal processing (clustering of entities and conversion)

Extended description

Mapping tool: the system that allows to map the source data, in different formats, to identify the elements (content and format) useful for subsequent clustering and conversion processes into LOD. The system is sufficiently flexible to allow the extension of the source formats over time, allowing it to adapt the clustering and conversion processes in an agile way. This mapping phase, supported, in the case of new formats, by the analysis of domain experts, allows for the adjustment of the clustering and conversion logic in order to accept a wide and rich range of formats.

Share data flow - Step 3

3) Main event: SVDE clustering tool called LOD Platform creates clusters of entities (Agents, Work, Instance etc.)



Extended description

LOD Platform - Clusterization tool: the tool includes the clustering logics of the data coming from different sources often non-homogeneous in order to create the entity as Real-World-Object (RWO) and assign a unique identifier. By clustering we mean the mechanism of identification of the entities with Large Scale Fuzzy Name Matching Techniques, through different text analysis methods such as: Common key, List, Edit distance, Statistical similarity, Word embedding

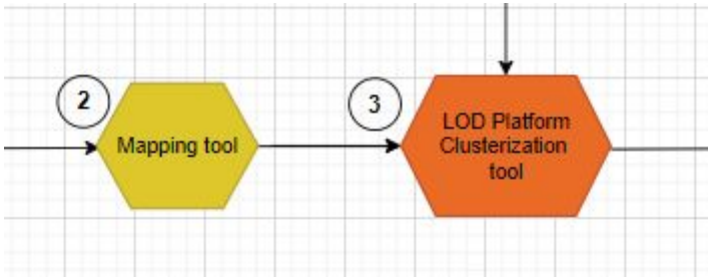
These methods tackle issues about data identification, among them: similar names, split database fields, phonetic similarity, spelling differences, truncated components, titles and honorifics, initials and nicknames, etc.

Other logics support the creation of a cluster / entity as well as the definition of the preferred form, among the many possible variants, to be assigned during the data presentation phase, including access point size/weight, usage count, identifiers presence, etc.

(continues) Share data flow - Step 3

The process produces a cluster of data in which the many possible variant forms of a name (of whatever entity it is) are reconciled and collected in a cluster, which is assigned a unique identifier (proprietary URI). The entities already managed with clustering processes are the following:

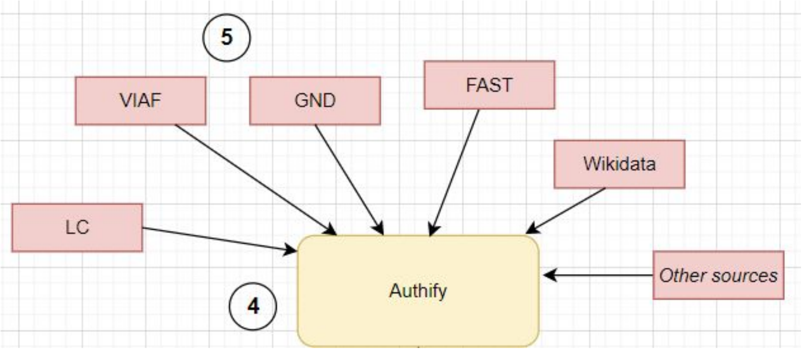
- Agent (Person, Organization, Family, Meeting)
- Work (with also HUB type)
- Instance
- Item (work-in-progress)
- Place
- Subject (Agents Person, right now)
- Topics: we currently have algorithms and processes to enrich a topic with external URIs from external sources (such as FAST or LCSH) using mostly string matching logics. Intense work is being carried on to enrich terms from subject strings with Wikidata source; this allows to expand the number of identifiers, thus partially overcoming the issue of different alphabets and reducing the risk of creating clusters that seem to identify the same entity but actually don't (entity recognition process).



In addition we manage some "domain" entities, such as places; occupations; roles; languages; other data coming from controlled vocabularies.

Share data flow - Steps 4-5

4-5) Main event: the data imported from libraries are enriched with data from external sources (e.g. VIAF, Wikidata, FAST etc.)



Extended description

Authify: the clustering process is completed with the enrichment of data from external sources through the proprietary tool Authify, within the LOD Platform. Authify manages the data sources through different processes depending on the query methods available from the external sources (APIs/WS, Dump, protocols etc.).

Authify is a RESTFul module that offers several search and detection services. At the very beginning, the LOD Platform project aimed at overcoming some limitations of the public VIAF Web API with the goal to obtain more comprehensive and precise URI retrieval results. VIAF, being a public project, doesn't allow a massive invocation of its API: for those use cases where such a requirement is needed, the project provides a download of the whole dataset.

That was the main reason why Authify was implemented: indexing and storing the VIAF clusters dataset and providing, on top of that, powerful full-text and bibliographic search services. Other sources will be added progressively, to answer different libraries' needs.

(continues) Share data flow - Steps 4-5

Extended description

The Authify Cluster Search Services provide, as the name suggests, a full-text search service among names and works clusters. The search Web API uses, behind the scenes, an "invisible queries" approach in order to try and find a match, as precise as possible, within the managed clusters.

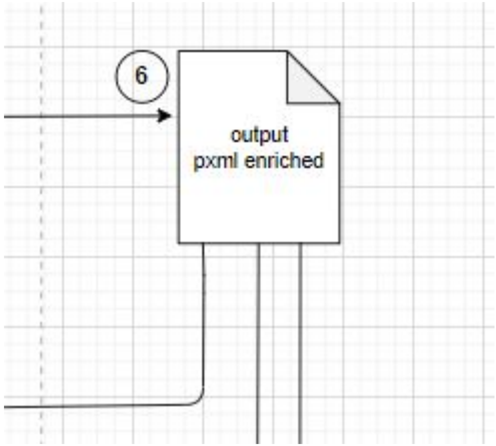
Thanks to the invisible queries approach, everything is more transparent to the caller: on top of a single search request, the system executes a chain of different search strategies with different priorities, and the first match that produces a result populates the response that is returned. For debugging purposes, the response also includes the matching strategy that produced the results.

The system has been built in order to be flexible and extensible, so the chain mentioned above is fully configurable; for instance, here's a brief description of the current configuration when searching names clusters:

- Subfields matching: the query language allows the caller to specify the source tag/subfields that compose the heading (which is the actual input query string). If those subfields are found within the query string, the system tries to find an exact match with their values;
- Input heading exact match: the system tries to find an exact match with the provided query string;
- FullText search: if exact match is not possible, then a regular full text search is executed, with things like proximity search for names (e.g. Bertrand Meyer = Meyer Bertrand), special detection for some entities (e.g. birth and death dates). Differently from the previous strategies, here precision is lower and it's possible for this strategy to return more than one result: in this case the response will contain the total number of matched items and a ranking assigned to each cluster;
- As a last attempt, the system executes a search by "initials", in order to find a valid match in those cases when the input string (or the indexed heading) contains the name in its short form. Same as the previous point, this could lead to a response with minor precision.

In Authify, a complete hyperlinking process is created and the final result is a much richer and well-identified entity than the original one.

Share data flow - Step 6



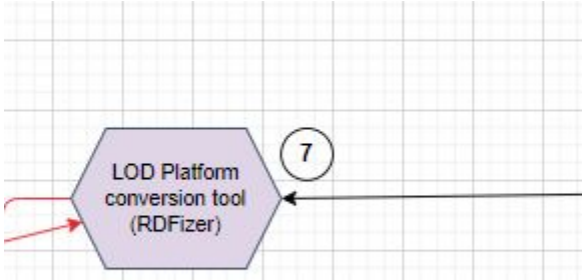
6) Main event: SVDE creates an intermediate file in a proprietary format to feed different internal pipelines

Extended description

Pxml file - the clusterization/reconciliation/enrichment processes have as output a pxml file, a proprietary file format defined to express the richness of data in a standard way. This file is used to feed two distinct processes, the LOD conversion and the text indexing into SOLR.

Share data flow - Step 7

7) Main event: SVDE converts to linked data the data imported from libraries and processed through steps 1-6



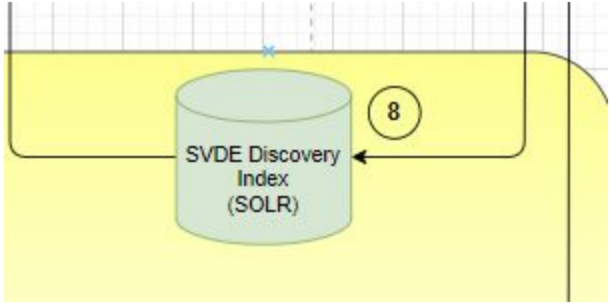
Extended description

RDFizer - The LOD Platform RDF conversion tool. RDFizer (an evolution of the previous Lodify module) is a RESTFul module that automates the entire process of converting and publishing data in RDF according to the BIBFRAME 2.0 ontology in a linear and scalable way. It is flexible and adaptable to multiple situations: it allows, therefore, to manage the classes and properties not only of BIBFRAME but also of other ontologies as needed. Lodify works strictly in conjunction with other LOD Platform tools and components such as Authify, the database of relationships and the Cluster Knowledge Base. The platform represents an enhanced and expanded version of the ALIADA framework (mentioned in the previous section) within an infrastructure that is better adapted to handle large amounts of data. The enriched pxml file cited in the item 6) acts as the input for RDFizer, which translates it into triples and uploads them to the selected triplestore. Upon completion, the RDF data can be extracted as a Turtle file by using the APIs provided by the triplestore.

RDFizer manages two conversion procedures:
- Cluster Conversion: converts data obtained from the cluster enrichment process;
- Record Conversion: converts bibliographical data obtained from an enriched MARC file into a triple.

Share data flow - Step 8

8) Main event: the data processed are indexed by the Solr search engine



Extended description

The Discovery Index (SOLR) - The same pxml file is used for indexing in the inverted index of SOLR: this search engine, used in combination with the triplestore for the presentation of data in the search portal, allows to enormously extend the entities search and retrieval. Combined with what is made available by the triplestore, it allows the end user to access the data by having the entity as the subject of the research, and no longer the bibliographic or authority records. A complex and extended knowledge panel will be proposed for each entity addressed in the system, to show its attributes and the rich network of relationships with other entities, in a way that tries to combine the richness of data with the user-friendly and intuitive discovery. A long list of search and retrieve logics, offered by the SOLR system, can be applied to extend the search capabilities of the system.

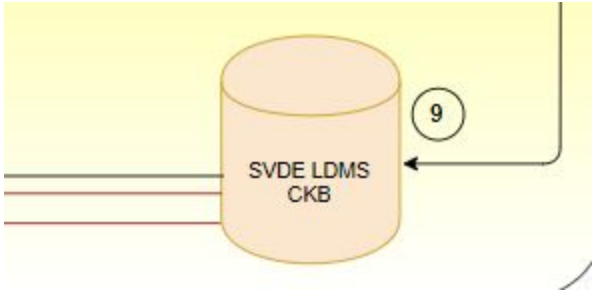
Share data flow - Step 9

9) Main event: the data converted feed the Cluster Knowledge Base, ie the database of SVDE entities

Extended description

The Cluster Knowledge Base - The Cluster Knowledge Base on PostgreSQL database and the corresponding RDF version are the result of the data processing and enrichment procedures with external data sources for each entity; currently, the clusters mentioned in point 3) typically: clusters of Agents (authorized and variant forms of the names of Persons, Institutions, Families), clusters of Works (authorized access points and variant forms for the titles of the Works and Instances) and clusters of Instances, as entities produced combining data coming from different library records. The CKB is populated with clusters of all the linked data entities that are created within the specific project that uses the LOD Platform. Such clusters derive from the reconciliation and clustering of the bibliographic and authority records (both records internal to the library system and from external sources) to form groups of resources that are converted to linked data to represent a real world object.

The CKB is the pool where new entities are collected, as the clustering processes go along. Currently, the CKB version dedicated to the Share-VDE initiative hosts the clusters mentioned in point 3). The CKB is the authoritative source of the system and it's available both on the relational database PostGres (mostly for internal maintenance purposes, reports etc.), as well as in RDF in order to be used for the Entity Discovery Interface and public exposure. The CKB is updated both through automated procedures as well as through manual actions via the entity editing module CKB editor. Each change performed on the entities of the CKB (both manual and automatic) is reported in the Entity Registry, which has the key role of keeping track of every variation of the resource URI, in order to guarantee the effective and broad sharing of resources.



Share data flow - Step 10

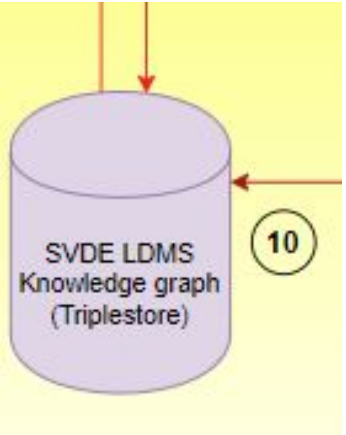
10) Main event: the data converted to RDF are sent to the triple store

Extended description

The Knowledge Graph (triplestore) - The data converted to RDF according to the agreed entity model (BIBFRAME 2.0 as core ontology, and other classes and properties derived from the ontologies indicated in point 12) are indexed in the triplestore. Also the data stored in the triplestore can vary (according to the update cycles defined by the target library), both through manual and automatic procedures, via the CKB editor module.

The Cluster Knowledge Base (CKB) is the result of the processing of the data. It contains groups of entities (clusters) created from original data, that are identified by URIs. The entities here included are cited in the item 3). The CKB is available in RDF, and relations and connections among resources can be inferred and queried via SPARQL endpoint. Advanced API layers are currently under development.

The value of the CKB is that the library can benefit from enriched information produced by the integration of their data with the data from other libraries' catalogues.



Share data flow - Step 11

11) Main event: The data are published to the Entity Discovery Interface on the web portal https://svde.org

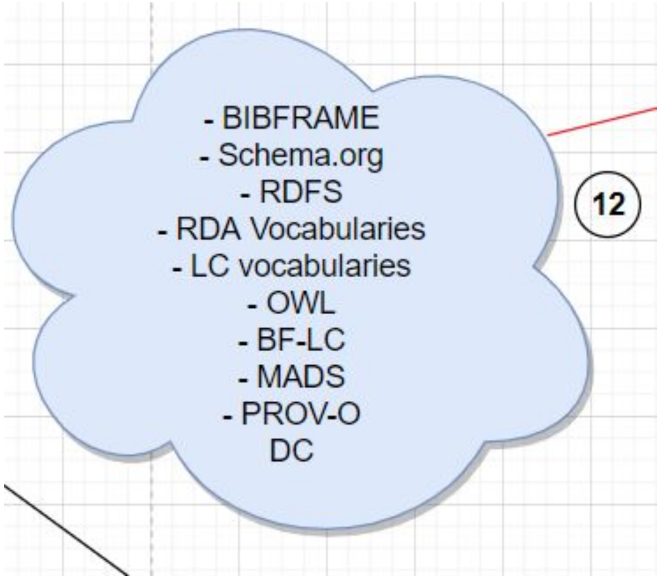
Extended description

The Discovery interface - The Entity Discovery Interface will harness the potential of linked data to offer an easy and intuitive user experience and deliver ever more wide-ranging and detailed search results to library patrons and library staff, based on an adaptation of the BIBFRAME data model.

The design focus is to provide intuitive access to complex data and make BIBFRAME easy to understand and benefit from. In order to achieve that goal, within the international Share-VDE community where this work stemmed from a two way process where stakeholders and UX designers continuously give feedback to each other in iterative phases has been put in place. Moreover, input from real users has been gathered in order to shape design around users' needs (i.e. library patrons, mostly university students), at the same time taking into account the requirements from librarians that play a key role in the wider linked data community (Share-VDE, LD4 etc.).

Share data flow - Step 12

12) Main event: Entity modeling is a key process in SVDE. The system creates BIBFRAME data but can integrate different ontologies



Extended description

Ontologies and controlled vocabularies used in the project - The data modeling in a LOD project is one of the most delicate and crucial aspects of the whole data management process. The conversion tool, Lodify, is built in a way that allows to extend it, following an approach as open as possible to receive and manage changes/extensions in existing ontologies and the inclusion of new ontologies and controlled vocabularies. Currently, the conversion tool uses the following ontologies: Bibframe, in its version 2.0; BF-LC; RDFS; OWL; MADS; PROV-O; RDA Vocabularies; LC Vocabularies

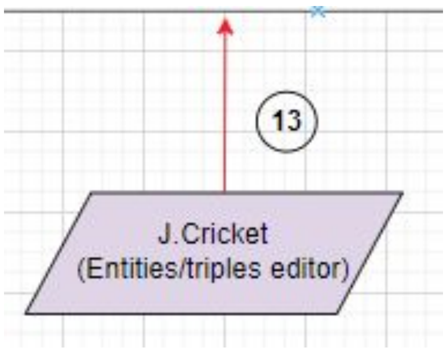
The peculiarity of the LOD Platform is the ability to handle different ontologies and vocabularies. An example of such flexibility is the data model of Share-VDE that combines the BIBFRAME oriented approach with the IFLA-LRM oriented approach. This has been done in order to foster interoperability among the community of Share-VDE libraries that comprises different entity modelling practices, and between other projects that apply the pure BIBFRAME model.

Share data flow - Step 13

13) Main event: The data produced by SVDE can be edited by J.Cricket linked data entity editor

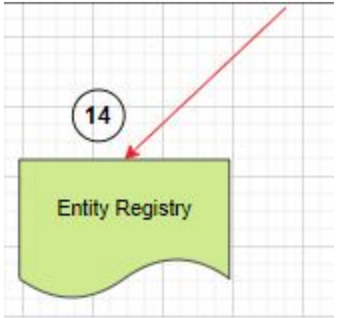
Extended description

J.Cricket entities editor - All data produced by conversion processes can be modified manually, to better address the issue of data quality. Within the Share-VDE initiative, an entity editor was designed for cooperative entity editing and it's currently under development. As already mentioned, all changes in the clusters, also through manual actions, are reported in the Entity Registry (see item 14). The J.Cricket development plan is far-reaching and spans over the current and the next year. In addition to J.Cricket, APIs are being developed for a direct connection to the open-source, RDF editor Sinopia (part of the LD4P initiative).



Share data flow - Step 14

14) Main event: Changes done to entities are tracked in the Entity Registry



Extended description

Entity Registry - As mentioned before, the management and tracking of changes to the URIs associated with each CKB cluster is entrusted to the Entity Registry. It is a special tool in which the association between clusters and the URIs that identify such clusters and changes done to the entities are registered, and where all the changes affecting this association are reported. An interesting example is the Redirect, that is the registration of the redirect from a cluster no longer valid to a valid one: this guarantees the recovery of the entities and their persistent identification even in the presence of heavy cluster modifications, such as the merge/matching process.

The status of the cluster is also reported in the Entity Registry, and it can be:

- Active: the cluster is active when it is fed by bibliographic records and, consequently, it is visible in the user interface of the Entity Discovery Interface portal.
- Inactive: the cluster is inactive when it is fed only by authority records or when it is not fed by any record. Inactive clusters, although they cannot be displayed in the user interface portal, are involved in the same way as active clusters in the clustering processes and can change their status in the case of association with bibliographic records.

(continues) Share data flow - Step 14

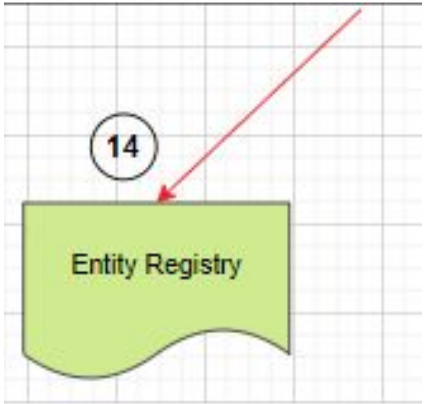
- Incorrect: the designation of a cluster as "incorrect" is entrusted to a special flag which, unlike the "active" and "inactive" status, can only be activated by manual modification via the CKB management module, defined as CKB editor. The use of the flag makes the cluster unusable (invalidation) and allows to activate the redirect mechanism from the wrong cluster to a valid cluster by creating an uri_alias. The invalidation action actually makes the cluster unusable and disables any form of modification both through the CKB editor, or through automatic delta updates processes.

Both active and inactive clusters must be visible in the CKB editor, while the attribution of "wrong" status implies that the cluster is not displayed in the user interface.

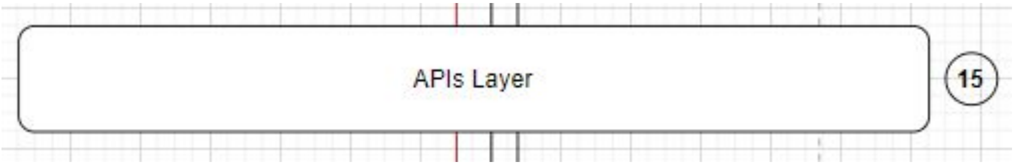
The definition of the statuses allows an easy management of the clusters, with distinction between clusters that may or may not be displayed in the portal, but most of all it guarantees the maintenance of the clusters that are no longer fed by any records, thus avoiding their cancellation and ensuring full correspondence between URIs and clusters.

There are two types of processes that can change clusters and their URIs:

- automatic: these are periodical processes starting from the LDMS according to the Libraries schedule, and are activated to manage the "delta" of the data;
- manual: through the use of the CKB Editor.



Share data flow - Step 15

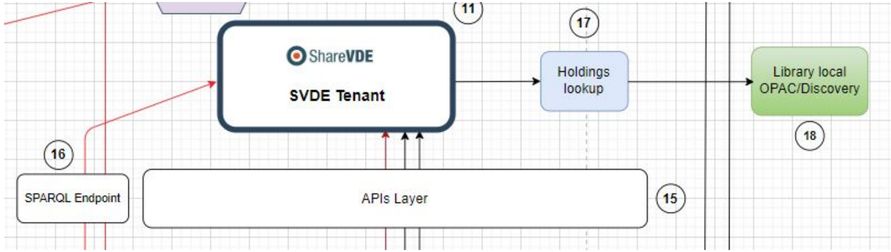


15) Main event: The API layer provides REST and a GraphQL APIs to query the data produced by SVDE

Extended description

APIs Layer - the data produced in the conversion or creation workflow are accessible through an API layer that will offer 2 interfaces to query the entities: a REST APIs layer and a GraphQL layer. In the REST API several endpoints serve different request types. In the GraphQL layer the same endpoint will perform the queries, using a language similar to SQL. GraphQL is a query language which allows to expose the underlying dataset as a graph. That allows a powerful mechanism for introspecting the entities and the relationships that form the domain model. The same entities are also exposed using a RESTFul paradigm, where a persistent URI is assigned to each entity and HTTP verbs are used for querying / manipulating the dataset.

Share data flow - Steps 16, 17, 18



16) Main event: SVDE makes available a SPARQL end point via the triple store

Extended description - SPARQL endpoint - Users can query triple data via SPARQL. A list of properties of datasets will make the query formulation easier for the user.

17) Main event: The Entity Discovery portal will also display holding data

Extended description - Holdings Lookup - Entity Discovery Interface will also display holding data, integrating APIs to retrieve holding information.

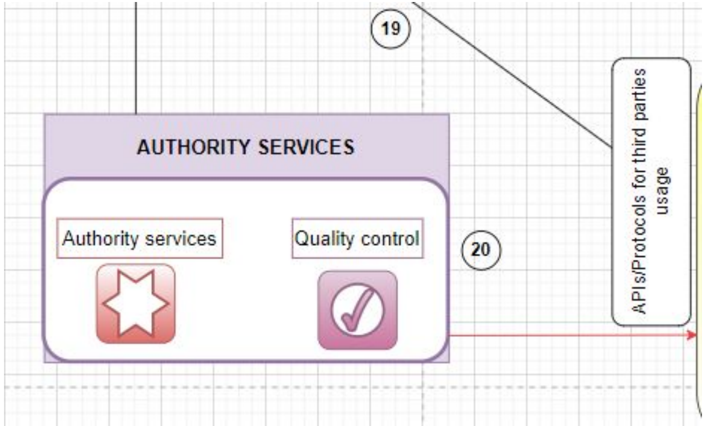
18) Main event: The Entity Discovery portal is connected with local library OPAC

Extended description - Library OPAC discovery: the SVDE entity discovery portal will be connected with the local OPAC of individual libraries.

Share data flow - Steps 19, 20

19) Main event: SVDE will make available further tools for data harvesting, including APIs, OAI-PMH, Atom feeds and Activity stream

Extended description
APIs/Protocols for third parties usage - to close the loop, the data elaborated in the SVDE workflow will be made available to libraries for their use, through protocols including APIs, OAI-PMH, Atom feeds and Activity stream



20) Main event: SVDE is connected with services for the authority control

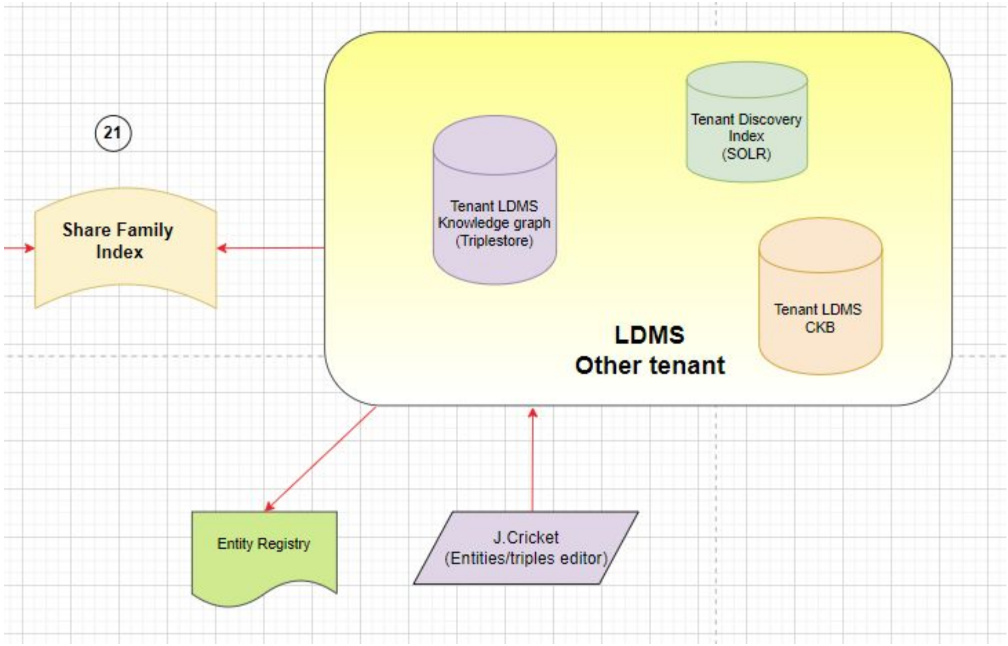
Extended description
Authority services - the services for the authority control are connected to the MARC and BIBFRAME data workflow

Share data flow - Step 21

21) Main event: SVDE will be able to connect with other tenants of the Share Family that apply the same data flow

Extended description

Example of connection with another Share Family tenant through the SFI - Share Family Index. The SFI is a registry which is in charge to create and assign a unique Share Family Identifier (SFId); it's a central index that aggregates URIs for entities that are stored in different CKBs



Thank you

https://wiki.svde.org/
https://svde.org
info@svde.org