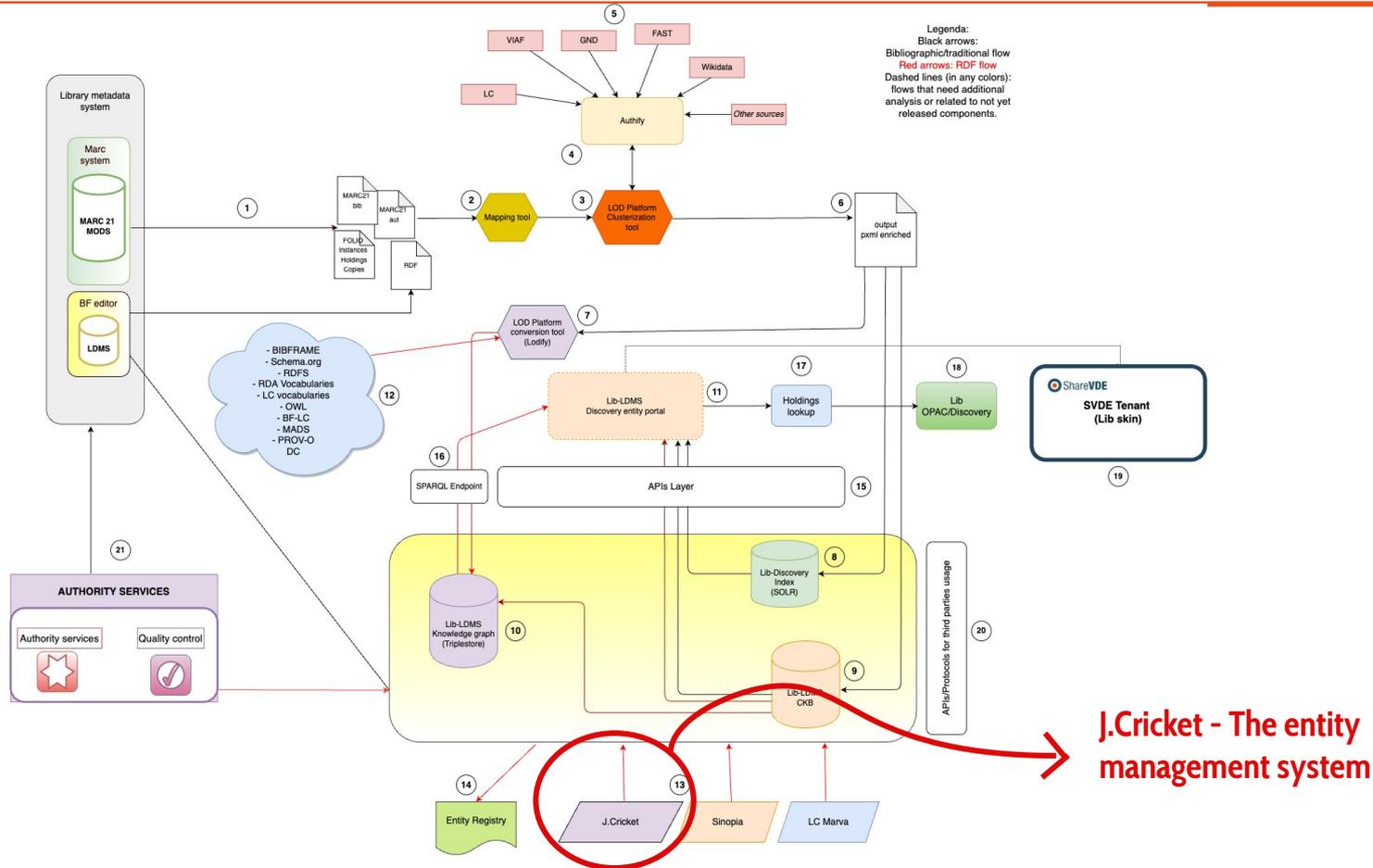


# J.Cricket entity editor

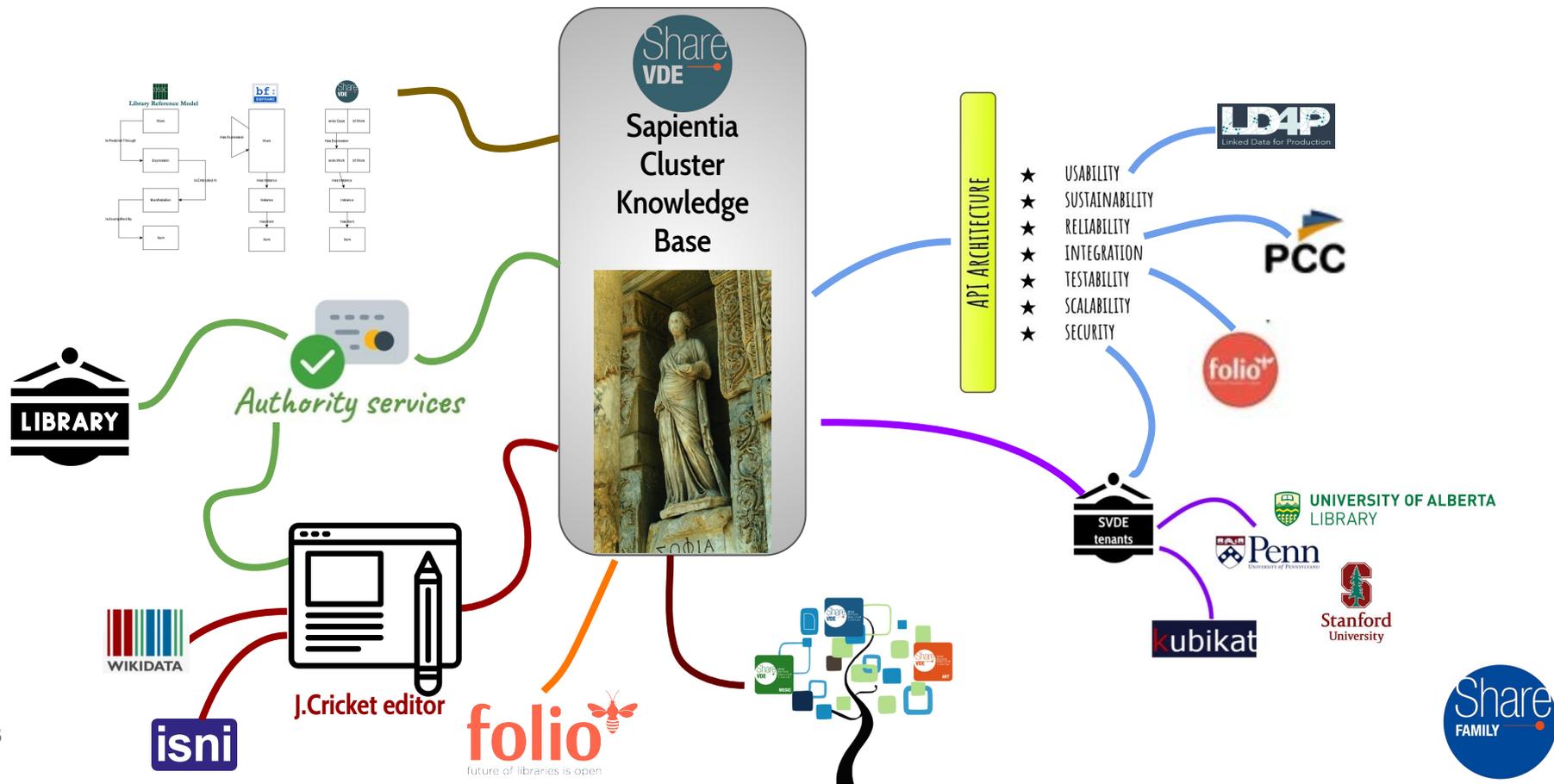
---



# Share-VDE data flow



# Towards the Share-VDE Sapientia CKB ecosystem



# From linked data publication to linked data editing

SHARE-VDE Search all content...

Person  
**William Goulding**  
British novelist, poet, playwright and Nobel Prize for Literature laureate

Sir William Gerald Goulding, CBE (19 September 1911 – 19 June 1993) was a British novelist, playwright, and poet. Best known for his debut novel *Lord of the Flies* (1954), he would go on to publish another eleven novels in his lifetime. In 1980, he was awarded the Booker Prize for *Rites of Passage*, the first novel in what became his sea trilogy, *To the Ends of ...* — [Wikipedia](#)

More options ▾

Original works by Original works about Related people

Filter... Location Language All filters

Publication title	Contributors	Language	Year
<input type="checkbox"/> Lord of the flies	William Goulding	English	1954
<input checked="" type="checkbox"/> Pincher Martin, 1956 novel.	William Goulding	English	1956
<input checked="" type="checkbox"/> Pincher Martin	William Goulding	English	1956

↕ End of list

Click to add to merge list

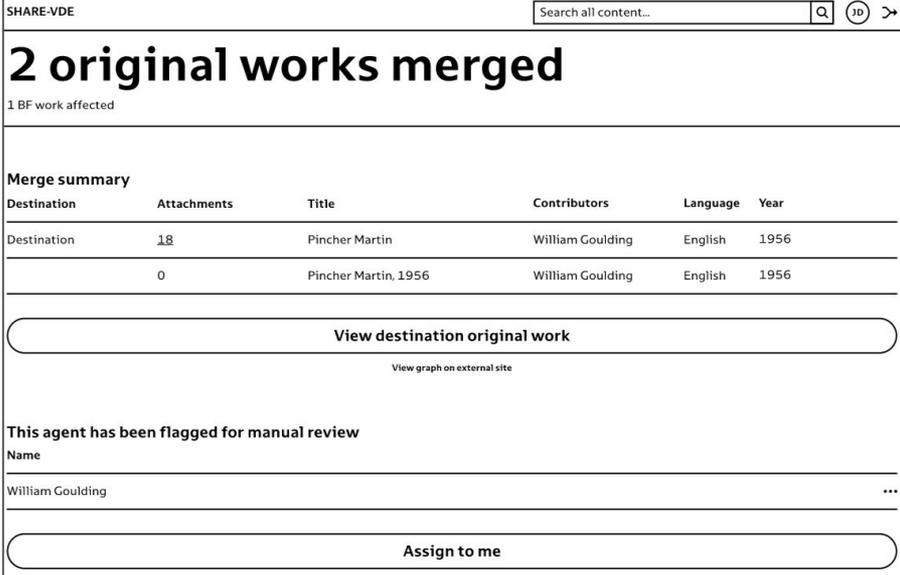
2 original works selected View selected ▾ Deselect all Add relationship Merge Add to my merge list

The Share family platform is evolving from a discovery environment that converts traditional MARC data of libraries in Linked Open Data to an interactive authoritative source providing real services for libraries. This transition is happening through the editor named J.Cricket, that is the new application dedicated to the editing of the clusters of data in a collaborative and integrated environment.

# From linked data publication to linked data editing

The editing tool J.Cricket will allow for editing the SVDE Cluster Knowledge Base, Sapientia, enabling several actions on the clusters (entities) saved in the SVDE database, including creation, modification, merge of clusters of works, of agents etc.

J.Cricket will extend authority capabilities through the integration with external data sources such as Wikidata and ISNI.



SHARE-VDE

## 2 original works merged

1 BF work affected

**Merge summary**

Destination	Attachments	Title	Contributors	Language	Year
Destination	18	Pincher Martin	William Goulding	English	1956
	0	Pincher Martin, 1956	William Goulding	English	1956

[View destination original work](#)

View graph on external site

**This agent has been flagged for manual review**

Name

William Goulding ...

[Assign to me](#)

# J.Cricket 1.1.0: Features Recap

---

- **AAA:** Authentication + Authorization + Auditing
- **Cluster Status API**
- **Edit Cluster**
  - real time notifications (through GraphQL subscriptions) about cluster property changes
- **Merge:** C1, C2, C3 => ~~C1~~, ~~C2~~, C3
  - Multiple phases: create the merge list, edit the merge list, edit clusters, request for review, approve (or deny the merge)
- **Split (Cluster):** C1 => C1, C2
  - C2 could even be a new cluster
  - Multiple phases: create the split-set, edit the split-set, edit clusters, request for review, approve (or deny the merge)
- **Dictionary API:** What are the available cluster types? Which attributes belong to a cluster type? Which relationships? Given an attribute, which is its cardinality? Is it mandatory or not?
- **Data changes synchronization across Share-VDE storages (e.g. RDF Store, Search Engine, RDBMS)**
- **Entity Event Log (aka cluster changes):** give me the history of changes of a given cluster
- **User notifications:** for managing the merge/split review lifecycle

# Edit - Overview

---

The **Edit Cluster** operation is available for **J.Cricket editors** to add, remove and amend **attributes, relationships and links** belonging to a single Entity.

If the user is a **basic editor**, only the properties coming from the **user's provenance** will be **editable**. If the user is an **advanced editor**, the **whole Prism** (meaning all properties) will be **editable**.

We will show the use cases related to this scenario, observing the client interaction with the Share-VDE server in relation to editing **one property at a time** (the recommended way to implement the edit feature).

However, the server supports submitting changes for multiple properties at the same time as well, with some limitations for transient values broadcast.



# Edit a cluster: create/edit/delete

A user with the “**editor**” role enters an entity’s page.

Publication

## Apache Solr Essentials

Online resource. Published in English in 2015 by Packt Publishing, Limited.

ISBN: 9781794396902  
Identifier: p1341054884566258

Notes: Description based on publisher supplied metadata and other sources.  
Bibliographic level: Monograph  
Cataloging source: MARC  
Descriptive cataloging form: ISBD  
Encoding level: Abbreviated  
Extent: 1 online resource (255 pages)  
Record status: Corrected or revised  
System control number: (MARC)IEC1373264

Borrow a copy at UPenn Libraries: More options ▾

Related agents Subjects

Filter agents... Type

2 results

No.	Name	Type	Start year	End year	Location	Relationship
1	Andrea Gazzarini	Person				author
2	Packt Publishing, Limited	Organization				publisher

Create new attribute/relationship

+ Add new

Title:

Is leader:

Language:

Share-VDE GraphQL Server

Edit an attribute/relationship

Title:

Is leader:

Language:

Share-VDE GraphQL Server

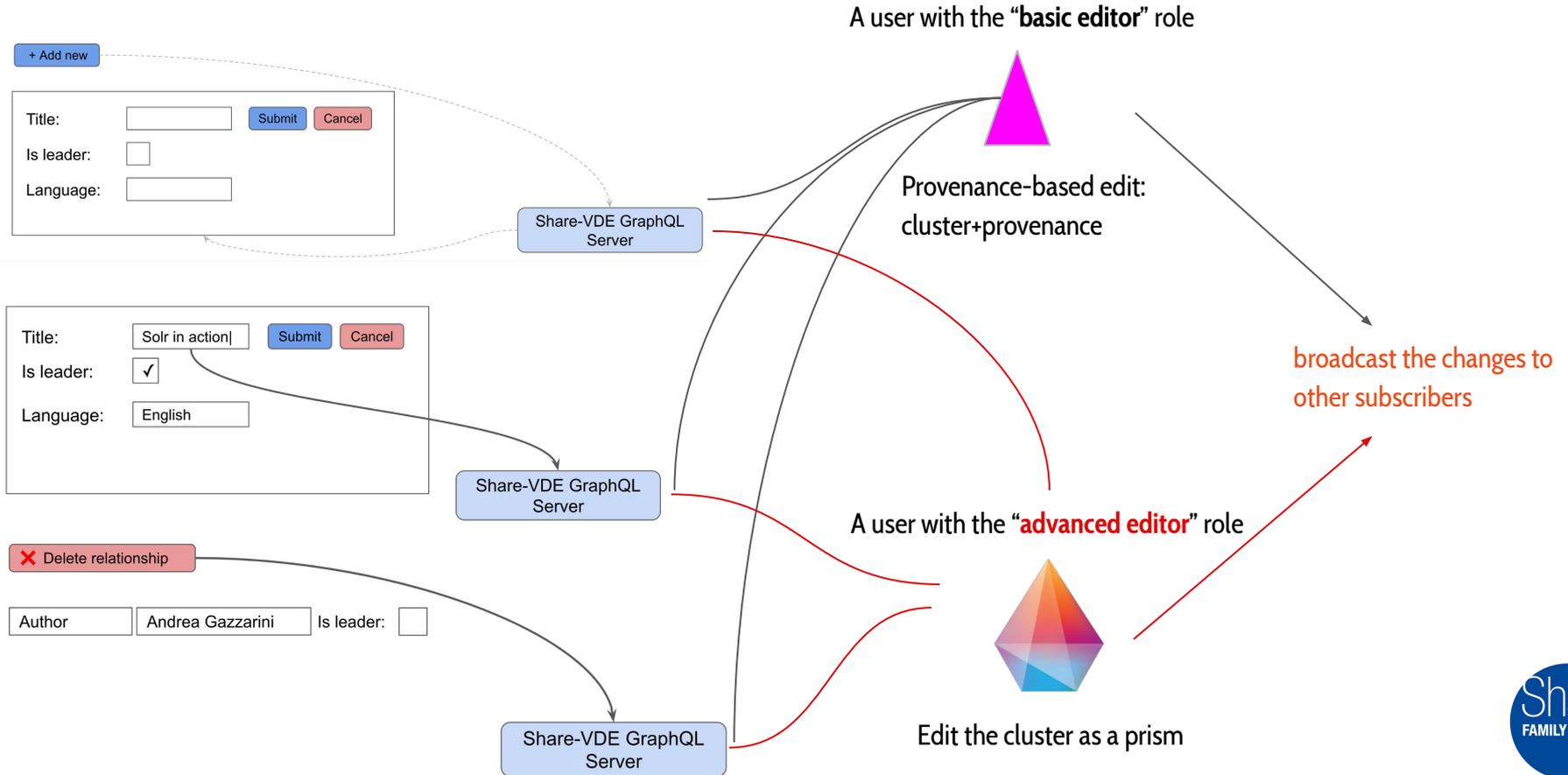
Delete an attribute/relationship

Author  Is leader:

Share-VDE GraphQL Server



# Edit a cluster: create/edit/delete



# Edit - An editor enters an Entity's page

A user with the “editor” role enters an entity's page.

Publication Jules Verne  
Book, Published in English in 1961 by Dobson.

ISBN: 04037074  
Wikidata: Q191022390003  
Notes: (0) annotations  
First published year: 1961-01-01  
Wikidata label: /en/fr/ShareVDE  
Copyright owner: US  
Genre: (2) sci-fi  
Extent: 120 pages  
Responsibility statement: Illustrated by Margaret Allen  
Series statement: [Topic to read \(sharevde.com\)](#)

Explore the DIME at the British Library

Related agents

Filter agents: [ ] [ ] Type: [ ]

2 results

No.	Name	Type	Start year	End year	Location	Relationship
1	Catherine Owen Dore	Person	1911			author

The application establishes a GraphQL subscription to Share-VDE to keep the page updated on Entity's changes (status, attributes, relationships, links).

Share-VDE GraphQL Server

When the Entity changes due to someone else modifying it, the change is notified to our user's browser as well, and the application updates the related field.

# Edit - An editor adds a new property (1 / 4)

The user adds a new attribute (attribute, relationship or link)

+ Add new

Title:

Is leader:

Language:

Initially, the client application uses the Dictionary API to request the Entity's Dictionary description, so to know what the Entity does support in terms of properties (attributes, relationships, links). This will guide the client in building a consistent UI.

1. The application sends a mutation to the GraphQL server notifying a new (still transient) property has been added.

Share-VDE GraphQL Server

3. The client must not let the user enter values in the new field before getting that response from the server.

2. No persistence is made by the server, as the change is still transient, but it responds with the empty attribute containing the server-assigned identifier.

# Edit - An editor adds a new property (2 / 4)

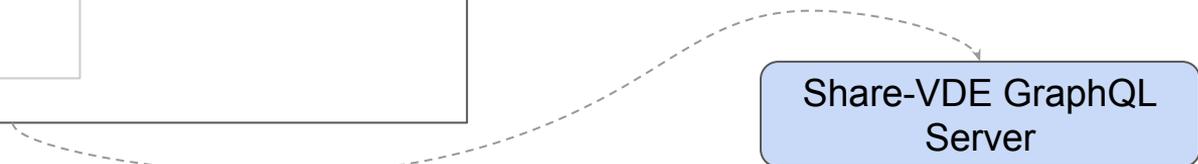
The user starts making changes to the new attribute.

The mechanics, despite some intrinsic differences between the property types, are the same for relationships and links as well.



The screenshot shows a web form with three input fields and two buttons. The first field is labeled 'Title:' and contains the text 'Solr in'. To its right are two buttons: a blue 'Submit' button and a red 'Cancel' button. The second field is labeled 'Is leader:' and contains a checked checkbox. The third field is labeled 'Language:' and contains the text 'En|'. Below this field is a dropdown menu with the following options: 'Armenian', 'English', and '...'. The form is enclosed in a light gray border.

At a certain time interval (e.g. 1s), a mutation is sent to the GraphQL server to let it **broadcast the changes to other subscribers**. On the first mutation sent, the server switches the cluster+provenance binomial status to “EDIT”.



A dashed line originates from the bottom of the form and curves to the right, ending in an arrowhead pointing to a blue rounded rectangle labeled 'Share-VDE GraphQL Server'.

Share-VDE GraphQL Server

No persistence is made by the server, as the change is still transient.

# Edit - An editor adds a new property (3 / 4)

The user may now hit the field's **Cancel** button

Title:

Is leader:

Language:

A mutation is sent to the GraphQL server to set the previous value back and **broadcast the reset to subscribers.**

Share-VDE GraphQL Server

If no other user with an overlapping provenance is editing the Prism (a.k.a. Cluster), the Prism' status shifts back to "SAVED"

# Edit - An editor adds a new property (4 / 4)

The user can now hit the field's **Submit** button

Title:	<input type="text" value="Solr in action"/>	<input type="button" value="Submit"/>	<input type="button" value="Cancel"/>
Is leader:	<input checked="" type="checkbox"/>		
Language:	<input type="text" value="English"/>		

The client sends a mutation to make the server persist the change.

Share-VDE GraphQL Server

If no other user with an overlapping provenance is editing the Prism (a.k.a. Cluster), the Prism' status is updated to "SAVED"

# Edit - A user edits an existing property (1 / 3)

The user edits a property.

Title:

Is leader:

Language:

At a certain time interval (e.g. 1s), a mutation is sent to the GraphQL server to let it **broadcast the change to other subscribers**. On the first mutation sent, the server switches the cluster+provenance binomial status to “EDIT”.

Share-VDE GraphQL  
Server

**No persistence is made by the server, as the change is still transient.**

# Edit - A user edits an existing property (2 / 3)

The user may now hit the field's **Cancel** button

Title:	<input type="text" value="Solr in action"/>	<input type="button" value="Submit"/>	<input type="button" value="Cancel"/>
Is leader:	<input checked="" type="checkbox"/>		
Language:	<input type="text" value="English"/>		

A mutation is sent to the GraphQL server to set the previous value back and **broadcast the reset to subscribers.**

Share-VDE GraphQL Server

If no other user with an overlapping provenance is editing the Prism (a.k.a. Cluster), the Prism' status shifts back to "SAVED"

# Edit - A user edits an existing property (3 / 3)

The user can now hit the field's **Submit** button

Title:

Is leader:

Language:

The client sends a mutation to make the server persist the change.

Share-VDE GraphQL Server

If no other user with an overlapping provenance is editing the Prism (a.k.a. Cluster), the Prism' status is updated to "SAVED"

# Edit - An editor deletes a property

The user deletes a property.

 Delete relationship

Author  Is leader:

The change is not transient, i.e. it is immediately persisted.

Share-VDE GraphQL  
Server

The GraphQL server persists the change, the mutation triggers the change broadcast to other subscribers.

The Prism status is brought back to "SAVED".

# Merge - Overview (1 / 2)

---

The **Merge Clusters** operation is available for users with the “**advanced editor**” role to convey **one or more** source **Entities into one**, picking the source Cluster(s) properties that must be ported.

The user picks **two or more Clusters** to merge, then designates the **destination** one. The remaining Clusters are automatically marked as “**source**”. Such **destination** or **source** traits are sealed by dedicated statuses “**MD**” (Merge Destination) and “**MS**” (Merge Source).

After that, the user can choose which properties to copy to the destination Entity.

## Merge - Overview (2 / 2)

---

After picking all the properties to put in the destination Cluster, the user **confirms the merge** and contextually **requests a review action** by **designating a reviewer**. The destination Cluster shifts its status to “RN” (Review Needed).

The reviewer may **approve** the merge; at that point the destination Cluster shifts its status to “SV” (**Saved**), while source Clusters acquire the status “IN” (**Invalidated**).

**Invalidated Clusters will remain in the system**, but they won't be indexed anymore, i.e. they will not appear in search results.

The reviewer may even **reject** the merge; in that case the destination cluster shifts back to the “MD” status; the reviewer provides some **rejection notes** to guide the editor.



# Merge one or more clusters

A user with the “**advanced editor**” role enters an entity’s page.

Publication: Jules Verne  
Book, Published in English in 1961 by Dobson.

#BX: 64637014  
Identifier: p18765327956932  
Name: Jules Verne  
Encoding level: Advanced  
Biographical level: Hierarchy  
Cataloging source: UK  
Dimension: 2D cm.  
Extent: 100 pages  
Responsibility statement: Illustrated by Margaret Ann  
Date statement: 1961 in two photographic issues?

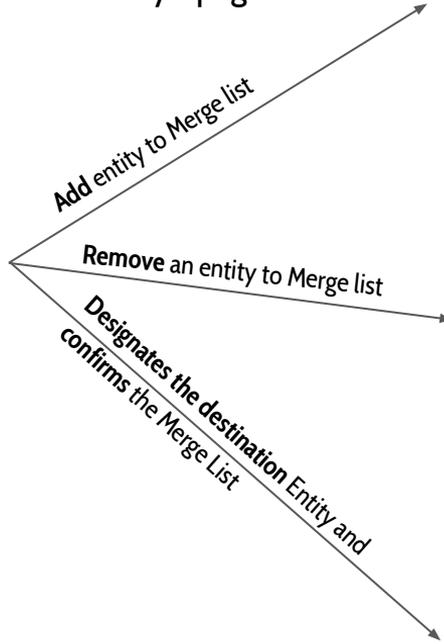
Explore the 998 of the British Library [More options](#)

Related agents

Enter agents:  Type

2 results

No.	Name	Type	Start year	End year	Location	Relationship
1	Catherine Owen Peare	Person	1911			author



People

- Jules Verne (16455887799)
- Julius Verneau (1234567383322)
- ...

Opuses

- ...

Share-VDE GraphQL Server

People

- Jules Verne (16455887799)
- Julius Verneau (1234567383322)
- Julius Vernell (187634562)

Share-VDE GraphQL Server

Choose destination:

- Jules Verne (16455887799)
- Julius Verneau (1234567383322)
- Julius Vernell (187634562)

Share-VDE GraphQL Server



# Approve/Reject the Merge



The designated **Reviewer** is notified about the assignment.

The designated **Reviewer** analyzes the merge result and decides to **approve** or **rejects** it

The **Reviewer** analyzes the merge result and decides to **approve** it

The **Reviewer** analyzes the merge result and decides to **reject** it



Share-VDE GraphQL Server



Share-VDE GraphQL Server

The Editor gets **notified** about the rejection



Share-VDE GraphQL Server



Rejection notes (optional):



# Merge - An editor enters an Entity's page

A user with the “advanced editor” role enters an entity’s page.

The screenshot shows the entity page for Jules Verne. It includes a header with the name and a brief description: "Book, Published in English in 1961 by Dobson." Below this, there are several metadata fields such as ISBN, Wikidata ID, and Wikisource ID. A "Related agents" section is visible, containing a search bar and a table with 2 results. The table has columns for ID, Name, Type, Start year, End year, Location, and Relationship. The first result is "Catherine Owen Dore" with the relationship "author".

ID	Name	Type	Start year	End year	Location	Relationship
1	Catherine Owen Dore	Person	1911			author

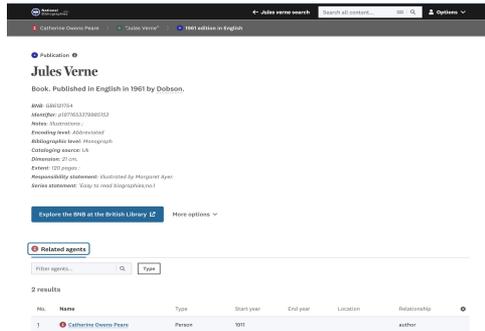
The application establishes a GraphQL subscription to Share-VDE to keep the page updated on Entity's changes (status, attributes, relationships, links).

Share-VDE GraphQL Server

When the Entity changes due to someone else modifying the same Entity, the change is notified to our user's browser as well, and the application updates the related field.

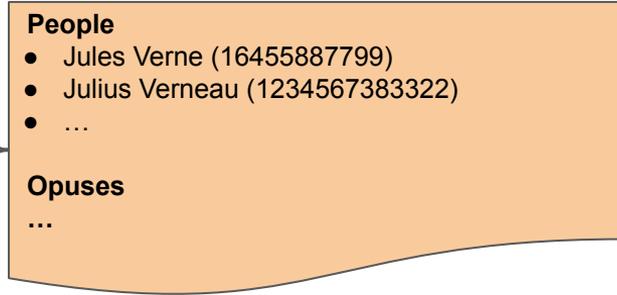
# Merge - An Entity is added to the Merge List

While browsing entities, the user **adds** them to the Merge List



The screenshot shows a web interface for a book entry. The book is titled "Jules Verne" and is published in English in 1963 by Dobson. Below the book information, there is a section for "Related agents" with a search bar and a "Type" dropdown. A table with 2 results is displayed below the search bar.

No.	Name	Type	Start year	End year	Location	Relationship
1	Catherine Owens-Pearce	Person	1971			author



The diagram shows a light orange rounded rectangle representing a Merge List. It is divided into two sections: "People" and "Opuses".

**People**

- Jules Verne (16455887799)
- Julius Verneau (1234567383322)
- ...

**Opuses**

...

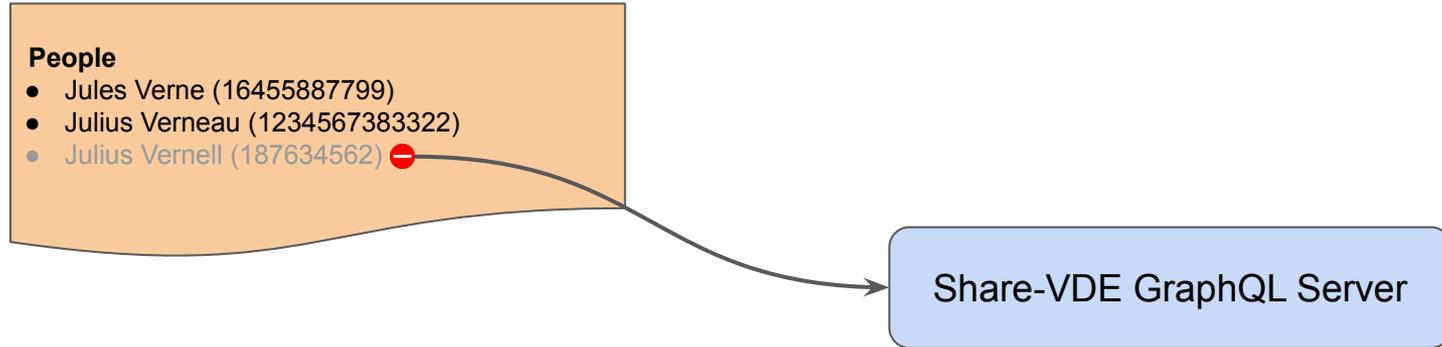
The Merge List is **subdivided** by Entity type.  
It works just like a shopping cart.

Share-VDE GraphQL Server

A **mutation** is sent to inform the server about the action, so to let it set the entities status to "ML" (**Merge List**).

# Merge - Entity is removed from the Merge List

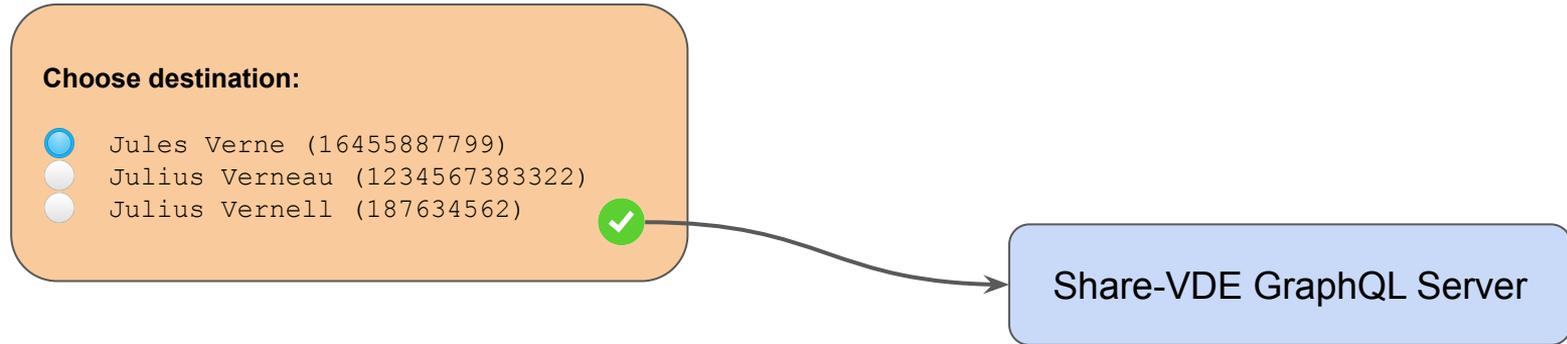
While looking at the Merge List, the user decides to **remove** an Entity from it.



A **mutation** is sent to inform the server about the action, so to bring back the Entity's status to "SV" (Saved).

# Merge - The user confirms the Merge List

When all of the Entities of interest are in the Merge List, the user **designates the destination** Entity and **confirms** the Merge List

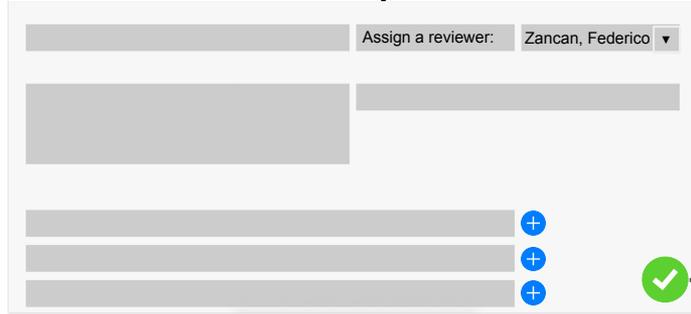


A **mutation** is sent to inform the server about the action.

All the **non-destination** Entities are given the “MS” (**Merge Source**) status.  
The **destination** Entity acquires the “MD” (**Merge Destination**) status.

# Merge - Properties are added to the destination

The user chooses what properties coming from source Entities must be **added** to the **destination**, **designates a reviewer**, and then **confirms the operation**



The screenshot shows a user interface for assigning a reviewer. At the top, there is a dropdown menu labeled 'Assign a reviewer:' with 'Zancan, Federico' selected. Below this, there are several rows of grey boxes representing properties, each with a blue plus sign to its right. A green checkmark icon is positioned to the right of the bottom row of properties.

**NOTE: While adding “foreign” properties to the destination entity, the user has the option to mark them as the leader form.**

**This operation will override any previous leader form for the same property already present in the destination.**

Share-VDE GraphQL Server

A **mutation** is sent to the server, containing all the new properties for the destination Cluster.

The destination Cluster is given the “RN” status (**Review Needed**).

The designated reviewer is notified about the assignment.



# Merge - The Reviewer approves the Merge

The designated **Reviewer** analyzes the merge result and decides to **approve** it



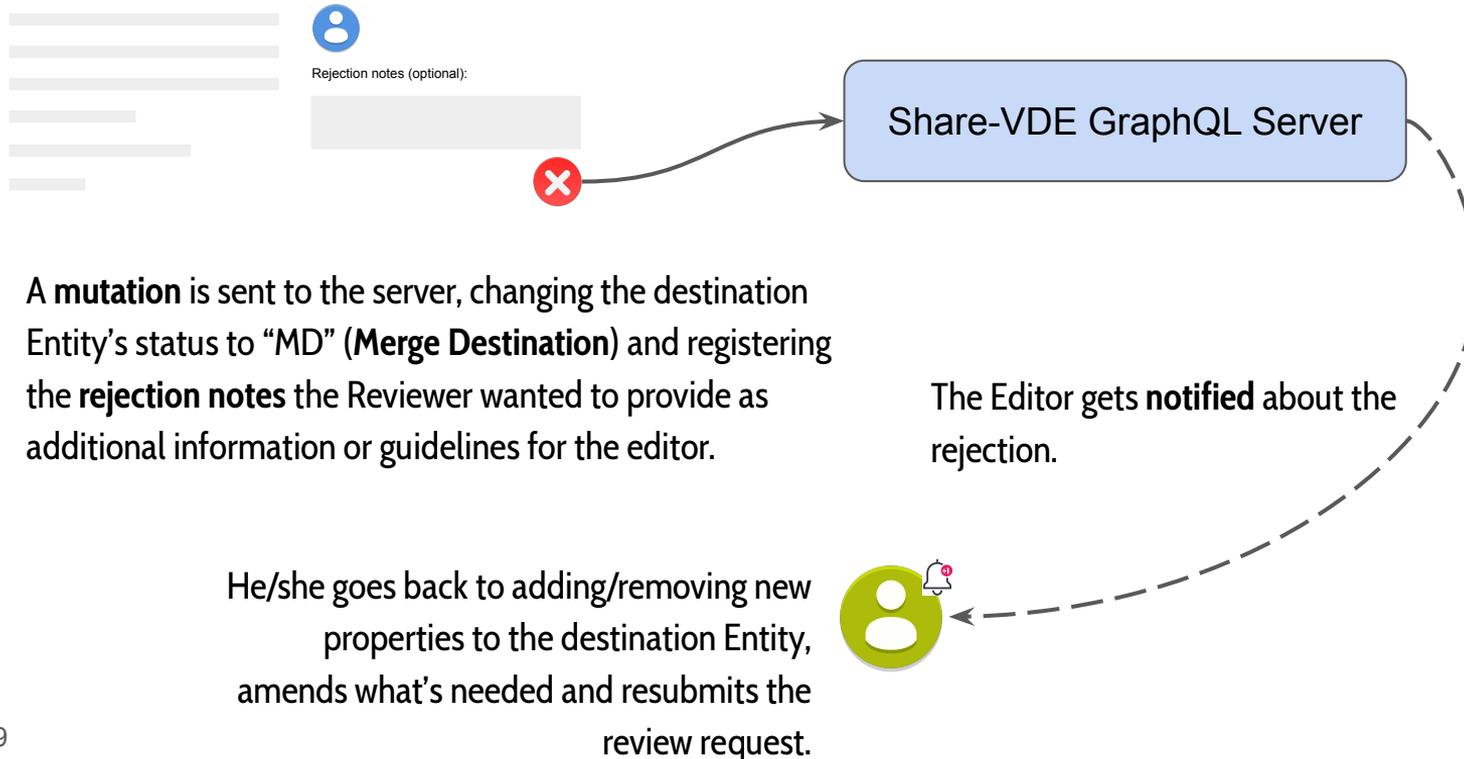
A mutation is sent to the server, changing the destination Entity's status to "SV" (**Saved**)

Source Entities pass from the "MS" (**Merge Source**) to the "IN" (**Invalidated**) status. Although they **remain** in the system, and their **URIs** are still **valid**, they are **not** part of **search results** anymore.

If visited, their pages show off in a greyed-out fashion.

# Merge - The Reviewer rejects the Merge

The designated **Reviewer** analyzes the merge result and decides to reject it



# Split - Overview (1 / 2)

---

The **Split** Cluster operation is available for users with the “**advanced editor**” role to let them **move** one or more **properties** between two clusters.

The user picks the “**Giver**” and “**Receiver**” Clusters. The giver Cluster takes the “**SG**” (**Split Giver**) status, while the receiver takes the “**SR**” (**Split Receiver**) status.

The user can then **choose** the **giver’s properties** to be moved to the **receiver**.

When satisfied with the choice, the user **confirms** the **split** and contextually **requests** a **review** action by **designating** a **reviewer**. The receiver Cluster shifts its status to “**RN**” (**Review Needed**).

## Split - Overview (2 / 2)

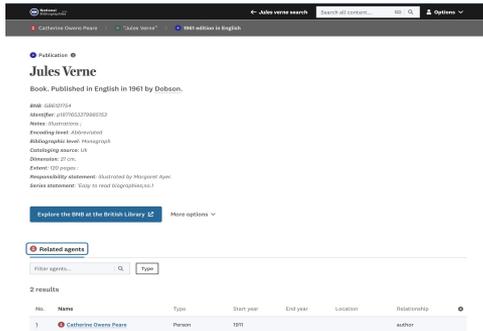
---

The reviewer may **approve** the **split**; at that point the giver and receiver Clusters shift their status to “SV” (**S**aved).

The reviewer may even **reject** the **split**; in that case the receiver cluster shifts back to the “SR” (**S**plit **R**eceiver) status; the reviewer provides some **rejection notes** to guide the editor.

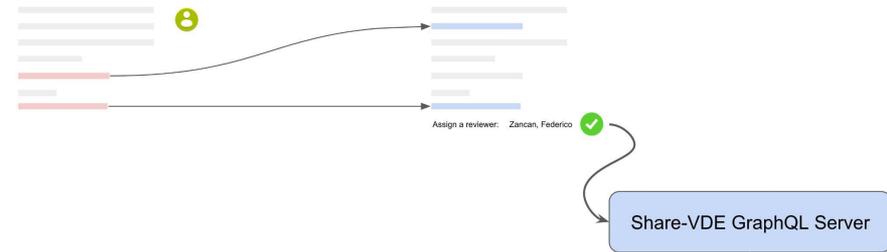
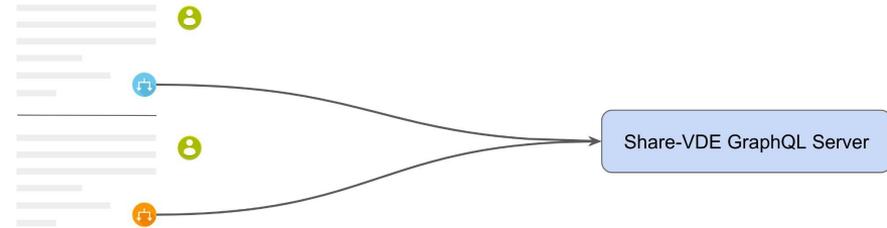
# Split clusters

A user with the “**advanced editor**” role enters an entity’s page.



The user activates the split context, designating the **giver** and the **receiver**

The user moves the giver's properties to the receiver



The designated reviewer is notified about the assignment



# Approve/Reject the Split



The designated **Reviewer** is notified about the assignment.

The designated **Reviewer** analyzes the **split** result and decides to **approve** or **rejects** it

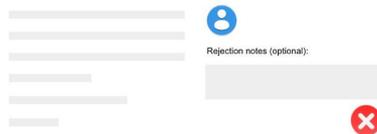


The Reviewer analyzes the properties that have been moved and decides to **approve**

The **Reviewer** analyzes the split result and decides to **reject** it



Share-VDE GraphQL Server



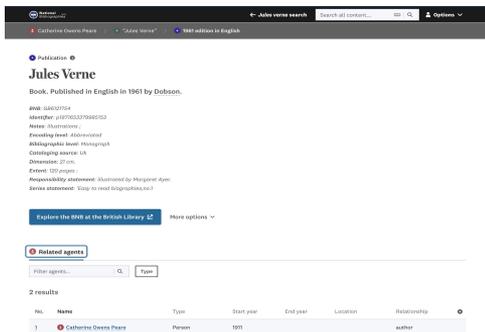
Share-VDE GraphQL Server

The Editor gets **notified** about the rejection



# Split - An editor enters an Entity's page

A user with the “advanced editor” role enters an entity's page.



The screenshot shows the entity page for Jules Verne. It includes a header with the name and a brief description, followed by various metadata fields like ISBN, Wikidata ID, and Wikisource ID. Below this is a section for 'Related agents' with a search bar and a table of results. The table has columns for ID, Name, Type, Start year, End year, Location, and Relationship. One result is visible: ID 1, Name Catherine Owen Date, Type Person, Start year 1911, and Relationship author.

ID	Name	Type	Start year	End year	Location	Relationship
1	Catherine Owen Date	Person	1911			author

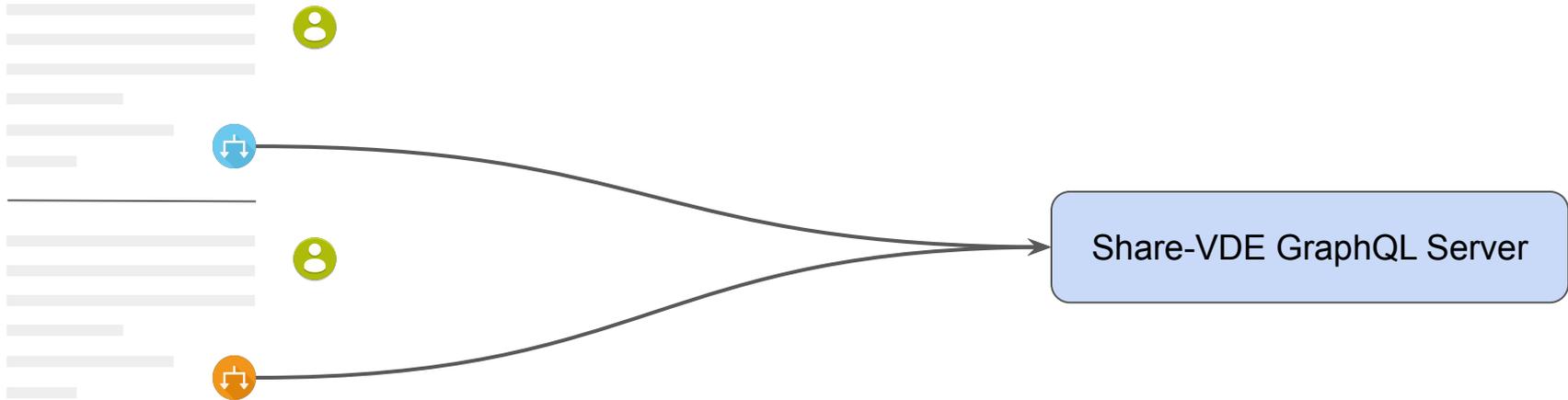
The application establishes a GraphQL subscription to Share-VDE to keep the page updated on Entity's changes (status, attributes, relationships, links).

Share-VDE GraphQL Server

When the Entity changes due to someone else modifying the same Entity, the change is notified to our user's browser as well, and the application updates the related field.

# Split - The user activates the Split

The user activates the split context, designating the **giver** and the **receiver**.

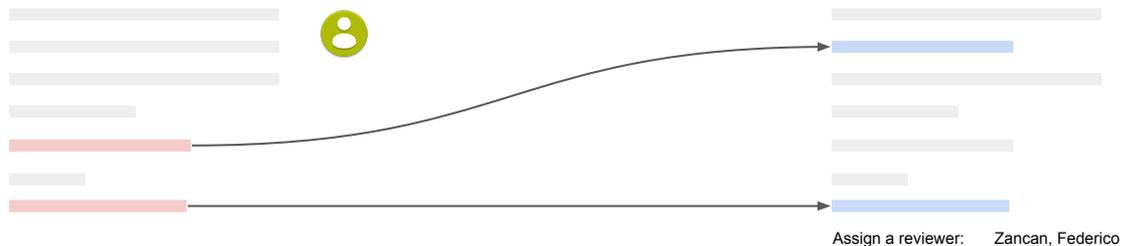


A **mutation** is sent to the server, changing the giver Entity's status to "SG" (**Split Giver**). A second mutation is sent to let the server mark the receiver Entity with the status SR (**Split Receiver**).

It is important to state that both the operations can even be contextual other than separate. The important thing is: the split operation starts when the split context is confirmed.

# Split - The user moves the properties

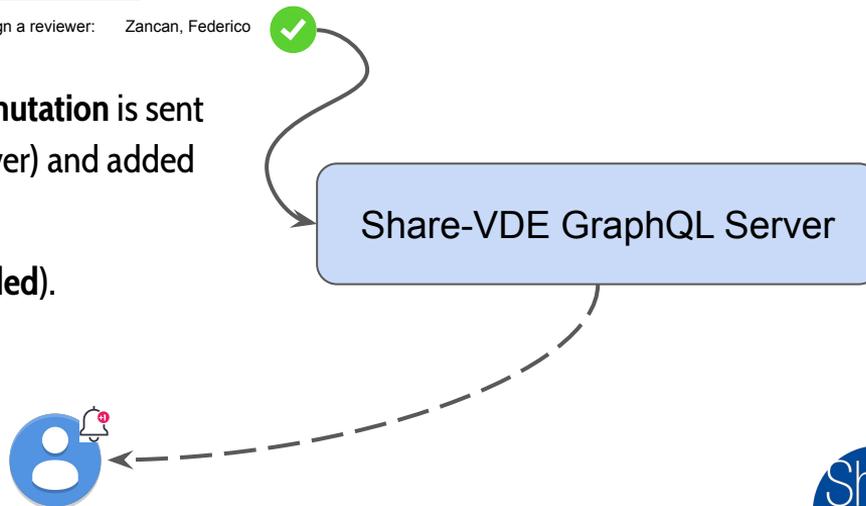
The user moves the giver's properties to the receiver.



When confirming the **move** and the **reviewer designation**, a **mutation** is sent to the server, containing the properties removed (from the giver) and added (to the receiver).

The destination Cluster is given the "RN" status (**Review Needed**).

The **designated reviewer** is **notified** about the assignment.



# Split - The Reviewer approves the Split

---

The designated Reviewer analyzes the properties that have been moved and decides to give the **approval**



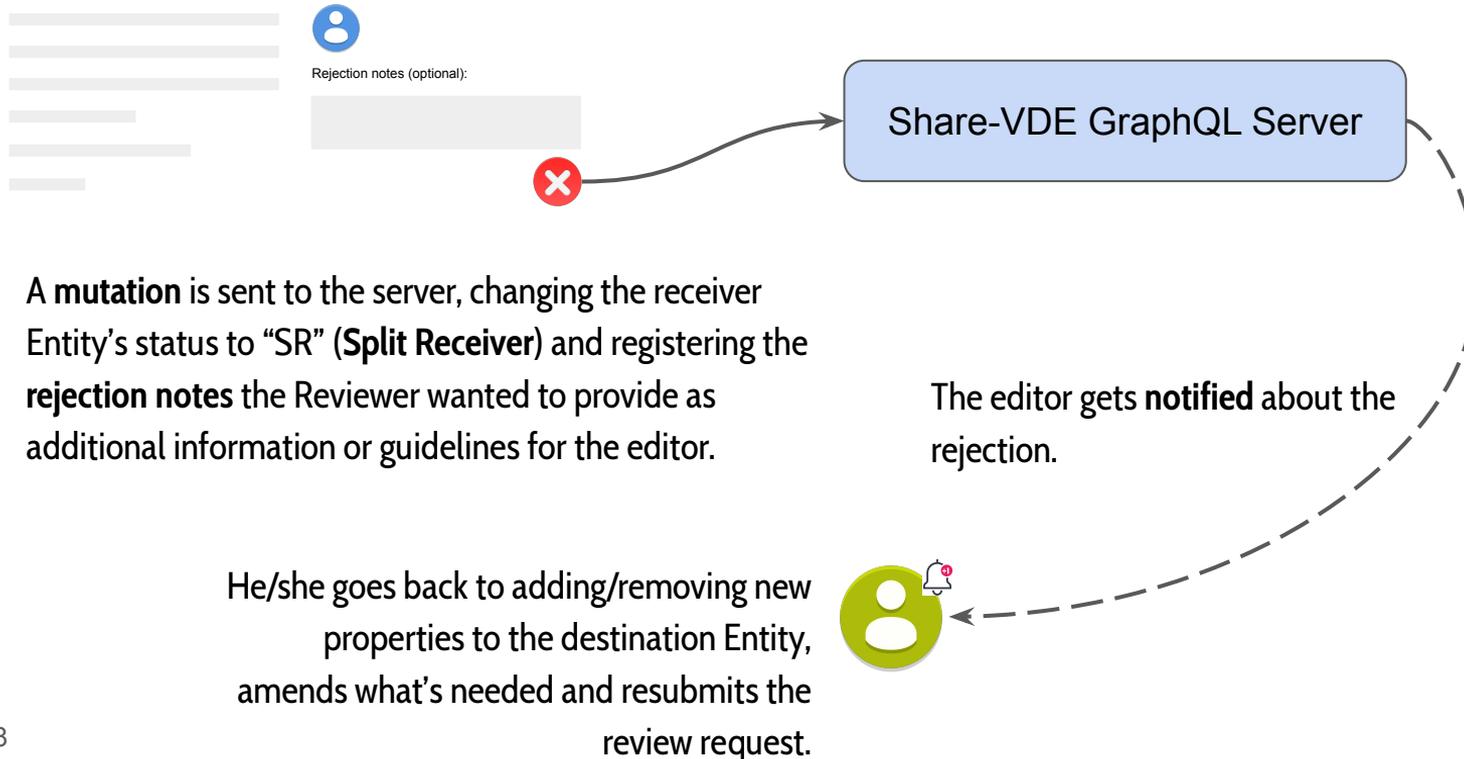
A **mutation** is sent to the server, changing the giver's and the receiver's statuses to "SV" (**Saved**).

The **giver** is now available to the world, **deprived** of the yielded properties.

The **receiver** is now available to the world, **enriched** of the given properties.

# Split - The Reviewer rejects the Split

The designated Reviewer analyzes the split result and decides to **reject** it



# J.Cricket - Postponed Features

---

- Create new Cluster
- Split cluster outputs n clusters ( $n \geq 2$ )
- Unauthorized users should be able to request changes to entities
- Ad-hoc alert system for misaligned clusters with respect to bibliographic records

# Next generation cataloguing

---

The J.Cricket editor is an example of how the Share family of initiatives is pursuing a new way of managing library cataloguing in a cooperative way:

- aggregation of data from multiple sources
- managed through standard protocols (linked data)
- in a collaborative and integrated environment
- that makes available open data and resources
- to end users and professionals (researchers, scholars etc.)
- for reuse in the library community and beyond