SEVENTH FRAMEWORK PROGRAMME

CAPACITIES

Research Infrastructures

INFRA-2009-1 Research Infrastructures

OpenAIREplus

Grant Agreement 283595

"2nd-Generation Open Access Infrastructure for Research in Europe

OpenAIREplus"

Specification of adaptation of content management services

Deliverable Code: D6.2

Document Description

Project	
Title:	OpenAIREplus, 2nd Generation Open Access Infrastructure for Research in Europe
Start date:	1st December 2011
Call/Instrument:	INFRA-2011-1.2.2
Grant Agreement:	**283595**

Document	
Deliverable number:	D6.2
Deliverable title:	Specification of adaptation of content management Services
Contractual Date of Delivery:	30th of June, 2012
Actual Date of Delivery:	
Editor(s):	
Author(s):	Paolo Manghi, Marko Mikulicic, Claudio Atzori, Michele Artini
Reviewer(s):	Mateusz Kobos (ICM), Lars Holm Nielsen (CERN)
Participant(s):	
Workpackage:	WP6
Workpackage title:	OpenAIREplus data model and content management services
Workpackage leader:	CNR
Workpackage participants:	NKUA, CNR, UNIBI, UNIWARSAW, CERN, DTU, EKT-NHRF, EMBL, KNAW-DANS, STFC
Distribution:	Public
Nature:	Deliverable
Version/Revision:	v1.2
Draft/Final:	Draft
Total number of pages: (including cover)	

File name:	D6.2 Specification of adaptation of content management services
Key words:	Content management, data model, big data

Disclaimer

This document contains description of the OpenAIREplus project findings, work and products. Certain parts of it might be under partner Intellectual Property Right (IPR) rules so, prior to using its content please contact the consortium head for approval.

In case you believe that this document harms in any way IPR held by you as a person or as a representative of an entity, please do notify us immediately.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of the OPENAIRE consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 27 Member States of the Union. It is based on the European Communities and the member states cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors. (http://europa.eu.int/)

OpenAIREplus is a project funded by the European Union

Table of Contents

Document Description **2**
Disclaimer **4**
Table of Contents **5**
Table of Figures **6**
Summary **7**
Log of Changes **8**
1 Introduction **9**
2 Data Flows **11**
3 High-level Architecture **12**
3.1 Caching Layer 13
3.2 Information Space Layer and Inference 14
3.3 Data Provision Layer 15
4 Inference **16**
4.1 De-duplication 16
4.2 Propagating inferred information to the data provision layer 17

Table of Figures

Figure 1 – OpenAIRE infrastructure data flow 11
Figure 2 - Information Space services 12

Summary

The deliverable describes how the services of the OpenAIRE infrastructure dedicated to collecting, storing, curating and indexing of content have been adapted to cope with management of data conforming to the data model (D6.1) of the OpenAIREplus infrastructure.

Log of Changes

Deliverable Version	Date	Changes (description, section and pages)
1.1	04.10.12	Introduction of protocol buffer
1.2	08.10.12	Adding notes to respond to reviews from ICM and CERN

1 Introduction

The OpenAIRE infrastructure has been running in production since 2009. Today it implements services whose aim is to measure the impact of EC OA mandates by collecting and interlinking publication metadata with EC funding information (i.e., CRIS-like metadata about projects, organizations, and people involved). Publications are collected (i.e. harvested) from repositories or instead provided by end-users through the OpenAIRE portal. The OpenAIRE project also realized and operates a so-called "orphan repository" where authors without a repository of reference can deposit their publication files and metadata.

The OpenAIREplus project (Dec 2011 – May 2014) continues and extends the scope of the OpenAIRE data infrastructure to promote and monitor Open Access to a wider audience and to more research output typologies. More specifically, it aims to grow a richer graph of data, covering material from all research disciplines and additional countries, and include projects from national funding schemes, non-peer-reviewed publications, and research datasets. Among its major objectives are experiments to interlink datasets and publications across different disciplines, by automatically inferring semantic relationships between them, by enabling end-users to construct "enhanced publications, and by interoperating with existing infrastructures, e.g., DataCite,[1] Mendeley,[2] ORCID,[3] EUDAT,[4] REIsearch[5]. The OpenAIRE orphan repository scope will also be extended from publications to include datasets. In addition, the OpenAIRE infrastructure has to integrate the whole DRIVER infrastructure information space, today harvesting and aggregating around 6,000,000 Open Access bibliographic records from 330 repositories.

To this aim, in the course of OpenAIREplus, the following actions have been necessary:

- **Data Model**: the OpenAIRE data model has been extended to add dataset entities to the existing publications, persons, organizations, data sources and projects, and to add new relationships between them. Moreover, new data source entities have been introduced beyond repositories, including CRIS systems, data archives, and entity registries (e.g., ORCID, OpenDOAR, CORDA).
- **Services**: the OpenAIRE infrastructure services have been adapted and extended to cope with:
    1. The collection of data from a wider range of data sources (not only institutional repositories, CORDA and OpenDOAR);
    2. The size of the incoming data, which thanks to the integration of DRIVER repositories and dataset repositories will reach tens of millions of objects;
    3. The need of efficiently processing such content in order to infer information out of it; e.g., de-duplication, relationship inference, subject classification, etc.

In this deliverable we shall describe the new architecture envisaged for the Opener infrastructure in order to satisfy the requirements of OpenAIREplus. Section 2 will introduce

[1] *DataCite*, www.datacite.org
[2] *Mendeley,* http://www.mendeley.com/
[3] *ORCID*, http://about.orcid.org/
[4] *EUDAT*, www.eudat.eu
[5] *REIsearch*, http://www.reisearch.eu

the data flows within the infrastructure that is how content is collected and processed before being delivered to the public. Section 3 describes the main changes reported in the infrastructure in order to cope with the data flow requirements of OpenAIREplus. Finally, section 4 focuses on the inference steps and on how the infrastructure makes it possible to keep inference a flexible and efficient mechanism in between data collection and data provision.

2 Data Flows

The high-level architecture is described in Figure 1. The OpenAIREplus information space, which conforms to the data model described by D6.1, is populated by means of three main workflows:

- **Manual provision**: end-users can *claim* associations between publications/datasets they deposited in remote repositories and the projects that funded such results; alternatively, while browsing the information space through the portal, they can send feedback (e.g. corrections) to data curators, so as to enrich and refine the information space; finally, data curators can *validate* any piece of information in the information space (be it inferred, collected, claimed, feedback-ed) in order to make it persistent;
- **Automated provision**: data curators operate the collection (e.g. harvesting) of content from a set of registered data sources; data sources can be of the following typologies: publication repository, dataset repository, CRIS system, and entity registry (e.g., ORCID, OpenDOAR);
- **Semi-automated provision**: inference algorithms are run over the information space (as obtained by manual and automated provision) in order to add, remove or update its content; in principle, such algorithms may require interactions with humans to refine their outcome.

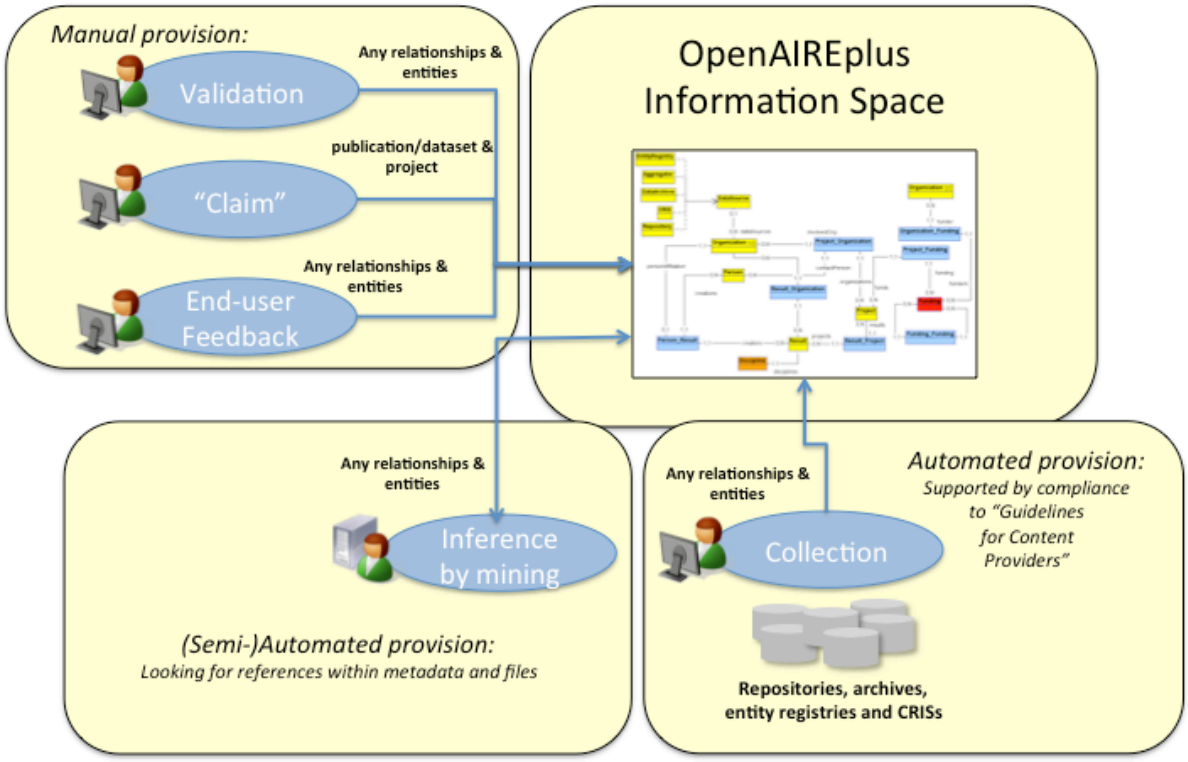

Figure 1 – OpenAIRE infrastructure data flow

3 High-level Architecture

The current OpenAIRE architecture (as designed and realized in the OpenAIRE project) is mainly organized in three service layers:

1. Cache layer: where publication metadata harvested from repositories is preserved into dedicated metadata store (MDStore) services;
2. Information Space layer: where information space entities and relationships are stored into a relational database service; the database is fed with records fetched from the MDStores and from external sources useful to the project, namely CORDA (European Commission CRIS system for FP7 projects) and OpenDOAR (directory of repositories admitted by the infrastructure);
3. Data provision layer: where a full-text index service layer is constantly fed with publications complete of their project information, as extracted from the relational database service with join query.

The main differences between such a scenario and the one targeted in the OpenAIREplus project are the following requirements:

1. Size of the information space: the numbers of entities, mainly publications, datasets and relative authors, increases of at least 2000 times; such an increase, possibly growing in time, makes the usage of (open source) relational databases less reliable and more complicated (e.g. keeping replicas, index maintenance, join query efficiency issues, building materialized views);
2. Running inference algorithms: due to their potentially exponential complexity, such algorithms need to access and navigate the content of the information space in an efficient way, which cannot be achieved with relational databases.

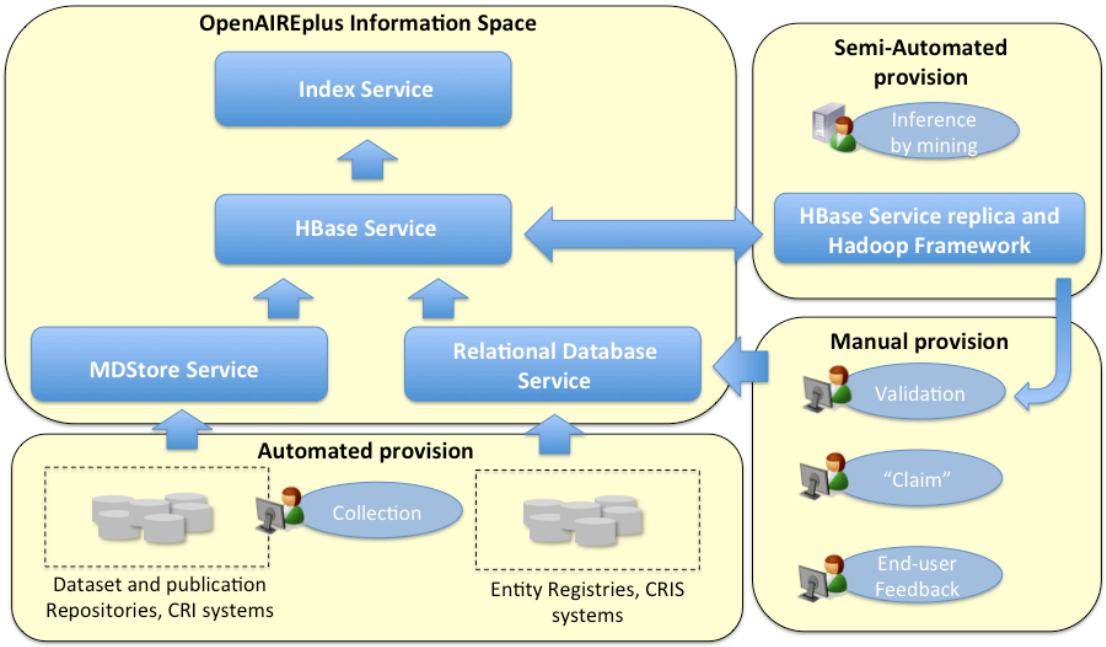

Figure 2 - Information Space services

To overcome such problems, in OpenAIREplus the infrastructure architecture has been upgraded and new services were developed. Figure 2 shows the high-level architecture of the services required to store content into the OpenAIREplus information space in a way that satisfies the data flow processes described in the previous section. In the following sections we shall introduce the three service layers of the OpenAIREplus infrastructure responsible for supporting collection and caching of native entities and information inference.

3.1 Caching Layer

Data sources registered to the infrastructure are automatically accessed (chon job configurations) via the access interfaces they provide in order to collect the relative "information packages" (see D6.1) – examples are: XML bibliographic records returned by repository OAI-PMH interfaces, or replies to HTTP requests to OpenDOAR entity registry. Information packages are in turn "unpackaged" in order to store the entities they contain across two types of storage services: Relational Database service and metadata store (MDStore) service. Such services will store data conforming to the OpenAIREplus data model but according to different logical models, namely the relational data model and a metadata record data model – the service encodes the notion of "metadata record" as an XML file consisting of a header section and a body section (the structure reflects that of OAI-PMH records). "Unpackaging" consists in parsing the packages to identify the subparts corresponding to data model entities and relationships and properly ingest them into the aforementioned storage services according to their internal model representation.

3.1.1 The Relational Database Service

The Relational Database Service is used to store native information (as collected from data sources) about "static" entities, i.e. whose size does not vary much in time and does not reach critical sizes. Examples are project entities as collected from entity registries (e.g. CORDA) and national CRIS systems. In addition, the relational database service will behave as a special OpenAIRE data source that contains the entities that were claimed, feedback-ed, or validated by users and data curators. For example, record relative to publications linked to records relative to projects, as inferred by publication PDF mining, will be stored in the relational database service.

To this aim, the Service will host a relational database that fully respects the OpenAIREplus data model, according to a standard mapping Entity-Relationship to Relational model. In particular, records relative to publications, results, organizations, persons, projects, and data sources will also refer to the end-user that has created them through claim or feedback or to the inference algorithm that has created them (see D6.1 for details).

3.1.2 The MDStore Service

The MDStore Service (based on MongoDB) is used to store native information about publication and dataset repositories, whose metadata records conform to Dublin Core (DRIVER or OpenAIRE compliancy) and DataCite respectively, and may I be used to store records obtained from CRIS systems. All such data sources may provide massive amounts of information packages (e.g., OAI-PMH records), the collection time may take hours or days, and their size typically increases in time.

The records collected from publication and dataset repositories will be straightforwardly stored into MDStores, while records collected from CRIS systems may need a form of "unpackaging" depending on the output format. In fact, CRIS systems may export XML

information packages whose content is a compound object whose entry point is a publication or a project together with their relationships to related entities and properties of such entities. In the first case, the packages need to be unpackaged and stored as a publication or a dataset in an MDStore, in the second case they need to be unpackaged to store projects in the relational database service. We may encounter hybrid scenarios, where XML packages need to be unpackaged to have their entities stored across the two services.

3.2 Information Space Layer and Inference

Both Relational Database and MDStore services work as "caches" for the information collected from data sources or provided by users. Such information is regarded as authoritative in the sense its level of trust is generally very high. However, as mentioned in the introduction, the main novelty of OpenAIREplus is that of introducing an information inference layer, capable of generating further information intended to disambiguate, enrich or correct the native information collected within such services. To this aim, the OpenAIRE infrastructure has been added an HBase Service offering distributed storage capabilities over an extendible cluster of machines. On top of such a cluster, a Hadoop Service allows for the execution of map-reduce algorithms in order to exploit parallelism and deliver the required efficiency. The HBase storage contains the materialization of the OpenAIREplus information space, obtained by feeding content from MDStores and Relational Database properly unpackaged into entities and relationships that conform to the OpenAIREPlus data model.

The storage representation of the information space into an HBase Service is based the relative column store data model. Object of main entities are stored as rows in the HBase cluster. The following columns characterize rows:

- The identifier of the row;
- The entity type of the object;
- The original object, as provided by the MDStore (XML record) or the Relational Database (XML encoding of relational record);
- A list of relationships to other objects in the information space: the column cell may contain the properties of such relationships, if any, e.g. class, scheme;
- A property stating if the object has been collected from data sources or added by inference services;
- A property stating if the object has been deleted by inference services;
- An XML encoding of the object to be used for full-text indexing.

The XML encoding of the object is the XML package to be returned to the portal by queries, in order for the portal to properly render the query hits on a query result page and on a detailed page of the individual hit. Such packages are obtained after a **consolidation step**, which processes the whole HBase storage to find and complete relationships between the objects (rows) and their related objects. Relationships whose *deletedByInference* flag is set to TRUE are not regarded in the process. As a result, packages group up the main entity object properties together with relationships (a combination of a reference and an intelligible string, e.g. name of a person and reference to the person) to other main entity objects in the index and to "children" of the object, i.e. objects of structural entities (such as "instance" objects of "result" objects).

Technical aside node #1 (protocol buffers): content of the columns relative to the original object as well as to the relationships are encoded using protocol buffer.

Technical aside node #2 (trinary relationships): according to this storage representation of the informationn space data model, only relationships with multiplicity 2 can be represented, although in the model some of the relationships have multiplicity 3; e.g. project-organization-person. The idea is that such relationships are encoded by picking two out of the three entities as "most important" in the relationship and treat the third as a property of the relationship. In the case of project-organization-person project and organization entities will be considered as the two relevant sides of the linking, while the person (i.e. the contact person for the organization in such project) will be considered as one property and represented with a reference and an intelligible string.

3.3 Data Provision Layer

The new infrastructure offers an index service layer, whose entries may represent objects of the main entity types of the OpenAIREplus data model together with their relationships. More specifically, in OpenAIREplus the portal will operate over an index containing records relative to the main entities, i.e., publications, datasets, persons, organizations, data sources, and project. Moreover, each entity will allow browsing to the entities associated with them by means of a relationship.

This will be possible thanks to the XML encoding presented above. The index will be fed with such records, containing, for each entity and depending on its type, the fields to be searched-on individually (advanced search) or as a whole (Google search), plus a field that contains the relative XML package. As a result of the query only the last field will be returned, in order for the portal to present the results in the most proper way.

The index will be fed with XML encodings of object rows whose *deletedByInference* value is set to FALSE. This excludes from the index those objects that have been removed by the inference process.

Technical aside note: the idea is that the portal will present results organized into different result sets, one per entity type. This will present users with a structured view of their hits and initiate from there navigation, by selecting one result and following its references to other objects, or apply further query refinements. Which main entity types will appear in the results is still to be decided; while publications, datasets, persons and projects will likely be there, the same may not hold for data sources and organizations.

4 Inference

As depicted in Figure 2, the overall materialization of the **native information space**, i.e. the space containing entities collected from external data sources or provided by users, is to be *cached* in MDStores and Relational databases and *materialized* in an HBase cluster.

Content in the HBase cluster can be delivered to the Data Provision Layer, i.e. to the index, to be made available to the OpenAIREplus portal or to consuming services. However, the motivation of using HBase as OpenAIRE back-end is that one of the main goals of the OpenAIREPlus project is to realize the tools for a team of data curators to enrich, refine, or disambiguate the native information space with inference algorithms before making it available to the public. In particular, such algorithms can be efficiently run exploiting a Hadoop Service deployed on top of the same cluster. Samples of such algorithms are:

- De-duplication of objects: automated merging of publications and persons (other main entities may be de-duplicated in the future is needed);
- Subject classification: result objects are assigned automatically one or more subject of pertinence;
- Title and author inference: as extracted from PDFs, when these are available;
- Citation inference: relationships between publications are identified, where possible, within the information space;
- Relationship inference: several algorithms will be run to identify relationships of various nature between objects; for example tracking and analysing user-behaviour patterns, or project-publication relationships by mining PDFs.

The OpenAIREplus data model equips entities and relationships with properties that allow distinguishing between native objects/relationships and those added or removed by the inference steps. This peculiarity enables a un-do and re-do strategy, supported by the efficiency of the Hadoop Services used to run inference algorithms. Algorithms can always be run in "reasonable time" and their results be dumped, to be calculated again in a subsequent run, without touching the native information space.

Once the native information space has been applied the required inference steps and the data curators are happy with the result, a final step of **consolidation** is necessary before sending the content to the data provision layer. The process, as explained above, produces the XML encoding of entities obtained by the updated information space.

4.1 De-duplication

The de-duplication process requires a further in-depth explanation. The process automatically merges objects of the same entity type based on the fact they are similar according to a given similarity function. As a result, the merging action returns a new *representative* object created out of the merged objects following a "blending" strategy, which varies from entity type to entity type. For example, if two publication objects are considered the same and therefore merged into one, the new representative object will contain all references to authors, projects, subjects, instances featured by the two original objects. More generally, independently of the entity type, de-duplication of a set of objects marked as "equivalent" consists of the following high-level actions: creation of a representative object and adjustment of the information space.

4.1.1 Creation of a new representative object

Identifier As all objects in the information space, representative objects (obtained out of a merge action) may be "bookmarked" via their IDs by users or applications. Since such objects may be "undone" and regenerated with further runs of de-duplication, it is important to ensure the stability of their identifier in order not to invalidate the relative bookmarks. To this aim, a stable (stateless) ID for the new *representative* object is created out of the IDs of the objects to be merged: IDs are sorted in lexicographical ordering and the first ID is picked and prefixed with the string "merge_". This strategy minimizes the chances that the same merge action will generate a different ID in future de-duplication runs – this will happen only if a new object with a minor ID and equivalent to these will appear in the information space, or if the object with minor ID will be removed from the space.

Properties The representative object has properties inherited by the merged objects. Objects are sorted by their *trust* values and when this value is the same, by ID. The object properties of the first object "win" unless they are empty; in this case values for such properties are "shadowed" by the fields of the next record in the ordering, and so on. The property *addedByInference* of the object is set to TRUE.

Relationships The representative objects will inherit all relationships owned by the objects it merges. To this aim new relationships, with an *addedByInference* flag set to TRUE, will be created so as to shift all incoming relationships for the merged objects to the representative object. New relationships *mergedWith* will point from the original objects to the representative object.

4.1.2 Adjustment of the information space

The creation of the new representative object requires the "invalidation" of the merged objects. To this aim, such objects as well as their incoming relationships will be marked with *deletedByInference* set to TRUE. This tagging, together with the *addedByInference* tagging, will make it possible for:

- Delivering to the index only "valid" objects as resulting by the last sequence of inference actions;
- Removing all inference actions from the HBase cluster to restore the native information space as it was before inference took place.

4.2 Propagating inferred information to the data provision layer

In OpenAIREplus de-duplication will be heavily used to make sure changes applied by the inference algorithms are correctly applied before indexing takes place. This is why a correct understanding of the de-duplication process will allow designers of inference algorithms to produce results in such a way their intended actions are applied to the information space.

Consider the following examples:

- Subject classification algorithm. Such algorithm returns a set of subject terms to be assigned to a given object A (be it a publication or a dataset). This information can be added to A, hence to the information space by exploiting the behaviour of the de-duplication service. A new object B is added to the information space, whose ID is

obtained from the one of A in a stable manner (prefix+ID, the prefix string related with the inference process, e.g. "addSubject_"), whose persistent identifier is the same of A (hence they are equal 100%), but whose level of TRUST (the property TRUST establishes an assigned level of trustabilty of the information and is part of any object in the information space; see data model in D6.1) is minor of the one of the object (in fact, it is not necessary to copy properties or relationships of the original object). The next de-duplication run will merge A and B, let A win and include the new subjects from B.

- Inference of publication title. Let's assume that for any publication object A with an empty title, the algorithm extracts a title from the publication PDF (if available) to complete the metadata. A new object B is added to the information space, whose ID is obtained from the one of A in a stable manner (prefix+ID, the prefix string related with the inference process, e.g. "updateTitle_"), whose persistent identifier is the same of A (hence they are equal 100%), but whose level of TRUST is minor of the one of the object (this way it is not necessary to copy properties or relationships of the original object) and whose title property is set to the inferred title. The next de-duplication run will merge A and B, let A win and include the new shadow title property from B.

Playing with level of TRUST and creation of new records is therefore the way to ensure inferred information is propagated up to the data provision layer, while at the same time guaranteeing that the native information space can always be restored and all inferred information deleted. Not only, since inferred information is also accompanied by "provenance" information relative to the process/service that produced it, the removal of inferred information can be selective with respect to its origin.

A further relevant example is that of **validation actions** performed by data curators. Such actions consist in the official approval of information obtained out of the data source collection chain or from end-users (claims, feedbacks), e.g. inference of objects, inference of relationships between objects, merge of objects, inference of new values for object properties, etc. In practice, validation makes the target information persistent and trustable (i.e. prevailing to other equivalent information) to reflect the expertise of a data curator. Such actions are stored into the relational database in chronological order. Their factual application is translated into the addition of objects or relationships with a very high level of TRUST, or removal of objects and relationships, directly into the information space