# DELIVERABLE D5.2

D5.2 - Methodologies for deployment and usage of the COMSODE publication platform (ODN), tools and data

<table>
<thead>
<tr>
<th>Project</th>
<th>Components Supporting the Open Data Exploitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acronym</td>
<td>COMSODE</td>
</tr>
<tr>
<td>Contract Number</td>
<td>FP7-ICT-611358</td>
</tr>
<tr>
<td>Start date of the project</td>
<td>1st October 2013</td>
</tr>
<tr>
<td>Duration</td>
<td>24 months, until 31st September 2015</td>
</tr>
</tbody>
</table>

| Date of preparation | 29/9/2014 |
| Author(s) | Carlo Batini, Marco Cremaschi, Angela Locoro, Andrea Maurino, Matteo Palmonari, Anisa Rula, Blerina Spahić, Claudio Venturini. |
| Responsible of the deliverable | Carlo Batini |
| Email | batini@disco.unimib.it |
| Reviewed by | Martin Nečaský |
| Status of the Document | Final |
| Version | 2.2 |
| Dissemination level | PU public |
## History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
<th>Revised by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2014-07-02</td>
<td>The initial index of the document</td>
<td>Andrea Maurino</td>
</tr>
<tr>
<td>0.2</td>
<td>2014-07-12</td>
<td>A first mostly-motivational draft</td>
<td>Carlo Batini</td>
</tr>
<tr>
<td>1.0</td>
<td>2014-08-07</td>
<td>First draft of the final version</td>
<td>Carlo Batini</td>
</tr>
<tr>
<td>1.1.</td>
<td>2014-08-16</td>
<td>Second draft version of the document</td>
<td>Carlo Batini</td>
</tr>
<tr>
<td>2.0</td>
<td>2014-09-13</td>
<td>Third version of the document</td>
<td>Carlo Batini</td>
</tr>
<tr>
<td>2.1</td>
<td>2014-09-17</td>
<td>Integration and publication on google drive</td>
<td>Andrea Maurino</td>
</tr>
<tr>
<td>2.2</td>
<td>2014-09-24</td>
<td>Selection of the catalogue record schema, dataset description, designing data schemas and ontologies</td>
<td>Angela Locoro, Blerina Spahiu, Matteo Palmonari</td>
</tr>
</tbody>
</table>
# Table of Contents

1 Executive summary .......................................................................................................................... 8
2 Deliverable context .......................................................................................................................... 9
  2.1 Purpose of deliverable .................................................................................................................. 9
  2.2 List of attachments ................................................................................................................. 10
  2.3 Related Documents ................................................................................................................. 11
3 Methodology used .......................................................................................................................... 12
  3.1 Methodology .............................................................................................................................. 12
  3.2 Partner contributions ................................................................................................................. 13
4 Methodology overview ................................................................................................................... 14
  4.1 Objectives of the methodology ............................................................................................... 14
  4.2 Structure of the methodology ................................................................................................. 14
5 Using the ODN Platform, DPUs and Pipelines ............................................................................. 17
  5.1 Generalities on the ODN platform ............................................................................................ 17
  5.2 Pipeline walkthrough ............................................................................................................... 19
    5.2.1 Creating the pipeline .......................................................................................................... 19
    5.2.2 Building the pipeline skeleton using DPUs and links ..................................................... 20
    5.2.3 Configuring the instances of the DPUs in a pipeline ...................................................... 23
      5.2.3.1 Configuring the DPU e-httpDownload ......................................................................... 23
      5.2.3.2 Configuring the t-tabular DPU ..................................................................................... 24
      5.2.3.3 Configuring the t-sparql DPU ....................................................................................... 28
    5.2.4 Pipeline execution ............................................................................................................. 30
    5.2.5 Verify, preview and execute ............................................................................................. 30
6 Techniques for phases ..................................................................................................................... 31
  6.1 Preparation of publication ......................................................................................................... 31
    6.1.1 Data sources access configuration ..................................................................................... 31
    6.1.2 Definition of the catalogue record schema, selection of the target data catalogues and description of the datasets ................................................................. 32
      6.1.2.1 Definition of the catalogue record schema ................................................................. 32
      6.1.2.2 Selection of the target catalogue ................................................................................. 34
      6.1.2.3 Dataset Cataloguing with the T-Metadata DPU ......................................................... 35
        6.1.2.3.1 Metadata supported by T-Metadata DPU ............................................................... 35
        6.1.2.3.2 How to describe a dataset using the T-Metadata DPU ...................................... 36
      6.1.2.4 Relation between this section and other deliverables .............................................. 46
6.1.3 Designing data schemas and ontologies for datasets published as 3*, 4* and 5* data 46

6.1.4 Determining entity identifiers and conventions for URLs of entities in datasets published as 4* and 5* data .......................................................... 54

6.1.5 Definition of the approach to the dataset publication ........................................ 56
   6.1.5.1 Approaches to anonymization of datasets ..................................................... 56
   6.1.5.2 Approach to updates of datasets ................................................................. 57

6.1.6 Design and implementation of the ETL procedures ........................................... 58
   6.1.6.1 Designing extractors ...................................................................................... 58
   6.1.6.2 Designing transformers ................................................................................. 59
   6.1.6.3 Designing loaders .......................................................................................... 60

6.1.7 Testing of the ETL procedures .......................................................................... 61

6.1.8 Licensing ............................................................................................................. 63

6.2 Realization of publication ............................................................................................. 64
   6.2.1 Initial publication of the dataset ........................................................................ 64
   6.2.2 Data Cataloguing .............................................................................................. 64

7 Techniques for data quality management ........................................................................ 65
   7.1 Data quality dimensions and quantitative vs subjective measurement of data quality 66
   7.1.1 Introduction to data quality dimensions .......................................................... 66
   7.1.2 Quantitative measurement ................................................................................. 67
   7.1.3 Subjective measurement ................................................................................... 68

   7.2 Data Quality Management activities ...................................................................... 68

   7.3 Requirements Collection and Analysis .................................................................. 68
   7.3.1 Requirements collection - Case study .............................................................. 69
   7.3.2 Requirement analysis ......................................................................................... 69

   7.4 Assessment ............................................................................................................. 70
   7.4.1 Standardization ................................................................................................. 70
   7.4.2 Accuracy ............................................................................................................ 71
      7.4.2.1 Accuracy of values ....................................................................................... 71
         7.4.2.1.1 Accuracy of a single value ................................................................ 71
         7.4.2.1.2 Syntactic accuracy of a value in CSV/relational model ................. 72
         7.4.2.1.3 Syntactic accuracy of an attribute/table in CSV/relational model .... 73
         7.4.2.1.4 Syntactic accuracy of a value, attribute, CSV/relational table measured with Jaccard distance .............................................................. 74
      7.4.2.2 Syntactic accuracy in RDF ......................................................................... 75
      7.4.2.3 Semantic accuracy ....................................................................................... 78
7.4.3 Completeness ................................................................. 78
  7.4.3.1 Metadata completeness ............................................. 78
  7.4.3.2 Completeness for CSV/relational values .................. 78
  7.4.3.3 Completeness in CSV/relational models ................. 79
7.4.4 Consistency ................................................................. 81
  7.4.4.1 Consistency in the relational model ......................... 81
  7.4.4.2 Consistency in the RDF model ...................... 82
7.4.5 Currency ................................................................. 83
7.4.6 Tradeoffs between value dimensions ..................... 83
7.5 Schema dimensions ........................................................ 84
  7.5.1 Relational model ......................................................... 84
  7.5.2 XML model .............................................................. 84
7.6 Data quality improvement .................................................. 85
  7.6.1 Generalities on Data quality improvement ............ 86
  7.6.2 A set of activities for the data quality improvement task 88
  7.6.3 A detailed case study .................................................. 88
    7.6.3.1 Quality improvement of accuracy ..................... 88
    7.6.3.2 Quality improvement of completeness ............. 89
    7.6.3.3 Quality improvement of consistency ............... 91
    7.6.3.4 QI-D Quality improvement through deduplication . 91
    7.6.3.5 Quality improvement of currency ..................... 93
8 References ....................................................................... 100
Attachment IV - Questionnaire for user perception of the quality of data ........................................ 102
List of Figures

Figure 3.1: Entity Relationship metaschema of issues dealt with in D5.1 and in this deliverable .................................................. 12
Figure 4.1: Input-output description of the D5.2 methodology .......................................................... 14
Figure 4.2: The high level conceptual view of the methodology described in the deliverable .......... 14
Figure 4.3: Format transformations suggested in COMSODE ....................................................... 15
Figure 5.1: The COMSODE reference architecture .................................................................. 17
Figure 5.2: Implementation of an Open Data Node (ODN) ........................................................ 17
Figure 5.3: A pipeline section ................................................................................................. 19
Figure 5.4: Pipeline property specification ............................................................................. 20
Figure 5.5: Building a pipeline .......................................................................................... 22
Figure 5.6: Pipeline resulting from the specification of flow of data ............................................ 22
Figure 5.7: Definition of the correct mapping ......................................................................... 23
Figure 5.8: Configuration dialog ......................................................................................... 24
Figure 5.9: Configuration dialog window ................................................................................ 28
Figure 5.10: DPU configuration dialog ................................................................................... 30
Figure 6.1: T-Metadata DPU in Transformers Folder ............................................................. 36
Figure 6.2: The pipeline for the Italian Dataset number 21 ....................................................... 37
Figure 6.3: The T-Metadata DPU configuration (part I) .............................................................. 38
Figure 6.4: List of the possible choices for “Media Type” field .................................................... 39
Figure 6.5: The T-Metadata DPU Configuration (part II) ......................................................... 41
Figure 6.6: How to select a license ......................................................................................... 42
Figure 6.7: How to add a new license .................................................................................... 42
Figure 6.8: The configuration for the “Available sources” field ................................................ 42
Figure 6.9: The T-Metadata DPU configuration (part III) ......................................................... 43
Figure 6.10: The configuration for the “Available keywords” field ........................................ 44
Figure 6.11: The configuration for the “Available themes” field ............................................. 44
Figure 6.12: The configuration for the “Available languages” field ........................................ 44
Figure 6.13: The configuration for the “Available publishers” field ....................................... 44
Figure 6.14: The statistics calculated by this DPU .................................................................. 45
Figure 6.15: A screenshot of the t-tabular DPU interface panel for the assignment of schema entities to csv data........................................ 53
Figure 6.16: A screenshot of the t-tabular DPU interface panel for the assignment of schema entities to multiple csv columns................................................................. 54
Figure 6.17: User interface for debugging pipelines ................................................................. 62
Figure 6.18: RDF data generator ......................................................................................... 63
Figure 7.1: The conceptual schema of the data quality management cross cutting activity ........ 65
Figure 7.2: A relation Movies with data quality problems ...................................................... 66
Figure 7.3: The triples present some of the RDF data referring to the current clubs ............... 67
Figure 7.4: Dimensions for which quantitative metrics are provided ....................................... 67
Figure 7.5: Data Quality Management activities .................................................................... 68
Figure 7.6: CSV table in input to standardization .................................................................. 70
Figure 7.7: Output of the standardization step (changed values are in gray) ......................... 71
Figure 7.8: A CSV Table .................................................................................................... 72
Figure 7.9: Edit distance used to quantify syntactic accuracy .................................................. 73
Figure 7.10: A second example of edit distance evaluation ..................................................... 74
List of Tables

Table 2-1: Deliverable objectives and their fulfilment ................................................................. 10
Table 5-1: selected DPUs for the pipeline ..................................................................................... 21
Table 5-2: Mapping table ............................................................................................................... 25
Table 5-3: publication mapping table ............................................................................................ 28
Table 6-1: Access and read dataset ............................................................................................... 31
Table 6-2: List of the statistics computed by this DPU and the related representation .............. 45
Table 6-3: List of available Vocabularies and Ontologies ............................................................. 47
Table 6-4: Technique and related DPUs table ............................................................................... 56
Table 6-5: Anonymization techniques .......................................................................................... 57
Table 6-6: DPUs for the extraction step ......................................................................................... 58
Table 6-7: Description of the features available on the Show detail tab of the E-RdfFromSparqlEndpoint DPU ............................................................................................................ 58
Table 6-8: DPUs for the transformation substep .......................................................................... 59
Table 6-9: DPUs for the load substep ........................................................................................... 61
Table 6-10: Considered licenses .................................................................................................. 63
Table 7-1: The result of the RDF alerts validator for the description of the resource http://live.dbpedia.org/resource/Ronaldinho ................................................................. 77
1 Executive summary

The purpose of this deliverable is to specialize the methodology described in Deliverable D5.1 (D5.1 in the following) in order to deploy and use the COMSODE publication platform (ODN), tools and data. In this deliverable, tools are techniques available as Data Processing Units (DPU) of the ODN platform as delivered according to Deliverable D3.2, together with techniques related to phases 2 and 3 of the methodology described in D5.1 and the data quality activities. With this deliverable users of the COMSODE software components will be able not just to install and run the platform but also to build on top of it and publish their own datasets.

This methodology provides answers to questions such as how to identify and use Data processing Units (DPU) available in the Open Data Node (ODN) platform, and how to identify and use other techniques made available on the Web or defined in the literature, in order to: reuse well known codebooks/vocabularies/ontologies, transform (e.g. anonymize) data before being published, identify descriptive metadata that should be published together with the dataset (such as name, format, source), etc. We recall that the COMSODE methodology described in D5.1 is intended mainly for data owners and publishers (mainly public bodies) and it is made of five main building blocks:

- Phases;
- Cross-cutting activities;
- Tasks and subtasks
- Artefacts
- Roles and practices

They represent the stages of the open data publication process and reflect the lifecycle of an open dataset. The following phases of the open data publication process were defined:

(P01) Development of open data publication plan,
(P02) Preparation of publication,
(P03) Realization of publication,
(P04) Archiving.

There are also some activities that should be performed in every phase of the open data publication process, that are the cross-cutting activities. There are four cross-cutting activities in the methodology:

(CA01) Data quality management;
(CA02) Communication management;
(CA03) Risk management;
(CA04) Benefits management.

Among phases and cross-cutting activities, DPUs in the ODN platform and techniques available on the Web or defined in the literature refer mainly to phase P02 Preparation for publication, phase P03 Realization of publication, and the cross-cutting activity CA01 Data quality Management, that are considered the most suitable activities to be supported by these tools.

Both phases and cross-cutting activities are further divided into sets of tasks. Tasks represent steps in the publication process. For each task, practices are described in order to provide more detailed guidelines on how the tasks should be performed. Furthermore, we model practices as subtasks of tasks, so to represent all methodological issues dealt with in COMSODE within a common conceptual framework. As to artefacts and roles, there is no specific contribution in this deliverable w.r.t. what discussed in D5.1.
2 Deliverable context

2.1 Purpose of deliverable

The purpose of this deliverable is to specialize the methodology described in Deliverable D5.1 in order to use the techniques available as DPUs of the ODN platform as delivered according to Deliverable D3.2, and including techniques related to phases 2 and 3 of the methodology described in D5.1 and the data quality activities. In this deliverable we assume that a unique dataset is in input to the COMSODE life cycle, while in Deliverable 5.4 more than one datasets will be considered in input. As a consequence, several activities such as e.g. data set integration will be discussed in D5.4.

The methodology described in this deliverable provides answers to questions such as how to identify and use DPUs available in the ODN platform, and how to identify and use other techniques made available on the Web or defined in the literature, in order to: reuse well known codebooks/vocabularies/ontologies, transform (e.g. anonymize) data before being published, identify descriptive metadata that should be published together with the dataset (such as name, format, source), etc. The methodology described in this deliverable is intended for publishers who have already decided to publish at least some of their datasets as open data. Therefore, it focuses mainly on the publication process and the dataset lifecycle, and, more specifically on phases, tasks, practices (here called subtasks) and cross-cutting activities, for which techniques have been developed or documented. Such techniques pertain to three classes:

- DPUs implemented and are in a stable version,
- Tools available as open source applications.
- Techniques documented with a specification and in several cases under development, not yet in a stable version at the date (30/9/2014).

Techniques of the first and second type are also called tools in the deliverable.

In summary, while D5.1 contents, answers to questions such as “what datasets should I publish as open data?” and “what and why I must publish as open data?”, D5.2 contents answer to question “how I can publish open data?”. Such question is addressed in this deliverable in the following way. We first address the issues related to the usage of the COMSODE publication platform, DPUs and pipelines, and an example of pipeline is shown. Then we recall and sometimes expand the description of phases 2 and 3, and associate to each of them the corresponding techniques. With reference to cross-cutting activity “Data Quality management” we extend methodological issues related to requirement collection, assessment and improvement of data quality that were mentioned in D5.1. Since data quality may concern many quality dimensions and the COMSODE life cycle may refer to several data formats, we focus on most relevant dimensions represented by accuracy, completeness, consistency and currency, and, with reference to data formats, we focus mainly on CSV and RDF representations. Also for data quality management, at the end of each methodological step (e.g. accuracy assessment of CSV data sets) we identify corresponding techniques that can be used in such step. In order to clarify the methodology, we decided to provide meaningful and understandable examples and case studies for this part.

The following table 2.1 summarizes the deliverable objectives in more detail and provides information about how these objectives are addressed in the methodology.
Table 2-1: Deliverable objectives and their fulfillment

<table>
<thead>
<tr>
<th>Deliverable objective</th>
<th>Fulfillment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe a methodology for deployment and usage of the COMSODE publication platform</td>
<td>Description of techniques subdivided in:</td>
</tr>
<tr>
<td>(ODN), tools and data</td>
<td>DPUs</td>
</tr>
<tr>
<td></td>
<td>Open Source tools</td>
</tr>
<tr>
<td></td>
<td>Documented</td>
</tr>
<tr>
<td>Usage of the COMSODE</td>
<td>Description of steps to:</td>
</tr>
<tr>
<td>publication platform, DPUs and Pipelines</td>
<td>Use the ODN platform</td>
</tr>
<tr>
<td></td>
<td>Create a pipeline</td>
</tr>
<tr>
<td></td>
<td>Construct the skeleton of the pipeline using the DPUs</td>
</tr>
<tr>
<td></td>
<td>Configure the DPUs in order to specify their behavior</td>
</tr>
<tr>
<td>Preparation of datasets for publication</td>
<td>Techniques for Phase <em>(P02)</em> Preparation of publication</td>
</tr>
<tr>
<td>Publication of the datasets</td>
<td>Techniques for Phase <em>(P03)</em> Realization of publication</td>
</tr>
<tr>
<td>Data quality management</td>
<td>Techniques for Cross-cutting activity <em>(CA01)</em> Data Quality management</td>
</tr>
<tr>
<td>Describe Techniques for Identification of resources in the datasets</td>
<td>See below</td>
</tr>
<tr>
<td>Reuse of the well-known codebooks/ vocabularies/ ontologies</td>
<td>Task <em>(P02A03)</em> Description of the datasets</td>
</tr>
<tr>
<td>Transformation and anonymisation of the data</td>
<td>Techniques for Task <em>(P02A05)</em> Definition of the approach to the dataset publication</td>
</tr>
<tr>
<td></td>
<td>Task <em>(P02A06)</em> Design and implementation of the ETL procedures</td>
</tr>
<tr>
<td>Development and publication of descriptive metadata</td>
<td>Tasks <em>(P02A03)</em> Description of the datasets and <em>(P03A02)</em> Data cataloguing</td>
</tr>
<tr>
<td>Cross-cutting activity <em>(CA01)</em> Data quality management</td>
<td>Techniques for <em>(CA01A01)</em> Data quality requirements analysis</td>
</tr>
<tr>
<td></td>
<td><em>(CA01A02)</em> Quality assessment</td>
</tr>
<tr>
<td></td>
<td><em>(CA01A04)</em> Quality improvement</td>
</tr>
</tbody>
</table>

2.2 List of attachments

Four are the attachments related to this deliverable. The first called Techniques-Phases-Tools provides definition of phases, cross-cutting activities, tasks, subtasks (practices) and related DPUs and other techniques. The second one is a list of codified techniques used in the methodology. The third called Cataloging provides explanation of metadata supported by the T-Metadata DPU. The fourth one called Quality questionnaire provides a modification of servqual
(service quality model) approach to collect quality requirements on specific dataset and the third attachment.

- Deliverable 5.2 - describes the overall concept of the methodology;
- Attachment I - provides definition of phases, cross-cutting activities, tasks, subtasks (practices) and related DPUs and other techniques.
- Attachment II - The list of codified techniques for the methodology
- Attachment III – Cataloging process for the metadata supported by the T-Metadata DPU
- Attachment IV - Quality questionnaire provides a modification of servqual (service quality model) approach to collect quality requirements on specific dataset

2.3 Related Documents

This document provides a personalization of the general methodology for publishing datasets as open data and it explains the concept of open data described in D5.1 to the use case of the ODN software platform. The descriptions of techniques enabling the most technical methodological tasks are also presented and discussed. The full list of described techniques as well as the list of available DPUs is provided as attachments to this document. The Attachment II spreadsheet provides basic description of which techniques can be used for a given phase, tasks or sub-tasks; inputs and outputs related to techniques and their nature are also described.

List of related documents from the project:
DOW, version date 2013-08-06, pages 25-26
Deliverable D5.1: Methodology for publishing datasets as open data
Deliverable D4.1: COMSODE publication platform - Open Data Node – for test
Deliverable D2.3: Architecture and design documentation for COMSODE development tasks
Deliverable D3.2: User requirements and techniques for data transformation, quality assessment, cleansing, data integration and intended data consumption of the selected datasets
Deliverable D3.3: Definition of ways to advertise the published data sets to Open Data catalogues
3 Methodology used

3.1 Methodology

As the first step, a high level concept of the methodology was designed, starting from the analysis of the ODN platform and of existing literature. This activity involved definition of the focus and the scope of the methodology. With the goal to clearly distinguish between the conceptual structure of D5.1 and the conceptual structure of this deliverable, we have produced the metaschema of the two methodologies, as shown in Figure 3.1.

Figure 3.1: Entity Relationship metaschema of issues dealt with in D5.1 and in this deliverable

The diagram adopts an Entity Relationship graphical representation (see for details [Batini et al. 1992]) where:

1. rectangles represent entities, namely classes of objects of interest (e.g. the entity Phase represents all phases of the COMSOIDE methodology described in D5.1)
2. diamonds represent relationships between entities
3. symbols (1,1) and (1,n) represent minimum and maximum cardinalities
4. arrows represent is-a relationships among entities
5. attributes and identifiers are omitted

Notice in Figure 3.1 that to each phase and cross-cutting activity several tasks may correspond; phases and cross-cutting activities are in many to many correspondences; tasks correspond to a unique phase or cross-cutting activity; to each task several subtasks
correspond and each subtask corresponds to a unique task. Furthermore, in Figure 3.1 it is clarified that techniques that are not documented techniques, are in many to many correspondence with tools, that in turn can be classified in terms of:

1. DPUs implemented,
2. Tools available as open source applications.

As a second step, we decided to develop in this deliverable a methodological path that assumes in input to the publication process a unique data set, while \( n > 1 \) datasets will be in input in the methodology described in Deliverable 5.4.

As a third step, we considered of high priority in the deliverable a clear description of how to use the ODN Platform, DPUs and Pipelines. We decided that this part of the methodology should be placed at the very beginning of the deliverable.

As a fourth step, based on analysis of the Description of Work (DoW) a set of topics to be covered by the methodology were identified which were further extended based on the experience of the authors. It was decided to focus mainly on the publication process, and specifically on the core activities of the COMSODE project, namely phases P02, P03 and the cross cutting activity CA01.

As a fifth step, each of the tasks and of subtasks was analyzed to create the correspondence with techniques of the three classes described above. Relevant artefacts (inputs and outputs) and related data types were associated to each technique. As to requirements discussed in D3.2 referring to quality assessment and data cleaning, a deeper investigation on documented techniques reported in the literature and available tools, led us to focus on a subset of quality dimensions mentioned in D3.2, more specifically to accuracy, completeness, consistency and currency dimensions.

As a sixth step the final version of the methodology and this deliverable was prepared.

### 3.2 Partner contributions

The responsible partner for the deliverable is UNIMIB who is also the main author of the deliverable. Other partners of the project consortium provided feedback and comments during the development of the methodology.
4 Methodology overview

4.1 Objectives of the methodology

An input output specification of the methodology is shown in Figure 4.1.

The inputs are:
- the dataset to be published, with its own characteristics described in metadata and in the schema
- DPUs made available in the Unified View environment for “Extract, Transform and Load” phases described in D3.2.
- Open source tools and documented Techniques for other tasks of Phases 2 and 3 and for Data quality management.

4.2 Structure of the methodology

In Figure 4.2 we see the conceptual view of the methodology described in the deliverable.
As Figure 4.2 shows, the (unique) dataset in input to the COMSODE life cycle has an associated dataset format, which can be of one of the following types:

1. CSV
2. Relational
3. HTML
4. XML
5. RDF

Other formats are also supported, such as Excel or JSON, but without loss of generality we discuss in the following the five formats above.

During the COMSODE life cycle, the dataset is subject to several kinds of analyses and transformations of various types, which will be described in detail in the following. E.g. we may have in input an html webpage, which can be transformed in CSV file, further analyzed to assess the level of accuracy of its data, for example by means of openrefine\(^1\), further improved as to accuracy comparing data values with values in a reference domain, and so on. Focusing on transformations, they can be broadly classified in two types, namely format transformations and other types (e.g. the data quality improvement transformation mentioned in previous example). A relevant decision concerns, given the format of the data set in input, how to intertwine format transformations with analyses and other types of transformations, considering that in the current version of the ODN platform RDF is the target format for publication of open data. Figure 4.3 clarifies this methodological issue.

![Figure 4.3: Format transformations suggested in COMSODE](image)

The following recommendations, applicable to the current version of the ODN platform, hold:

- If the dataset in input is in RDF format, keep this format in the whole life cycle and, consequently, use techniques supporting analyses and transformations for this format.
- If the dataset in input is in CSV format, according to user skills and available tools it is possible to perform transformations in the original format or transform it into RDF to fully exploit the features of ODN platform.
- If the dataset is in HTML, transform as soon as possible the format into CSV, in such a way to be able to apply the rich set of techniques that are provided for CSV available in external open sources tools.
- If the dataset is in XML, depending on the skill of the user and the type of analyses and transformations to be performed, there are two choices, a. to transform the format into CSV, and b. to transform the format into RDF.
The relational model can be seen as an extension of the CSV model with integrity constraints represented by keys, tuple constraints and referential integrity constraints. If the dataset in input is in relational format, you may choose to keep data in relational, or else transform in CSV, keeping in mind that the relational model is more powerful and consequently more complex than the CSV model, so that several relational DBMS tools can be used, whose rationale needs more comprehension effort by the user of COMSODE.

The rest of the deliverable is organized as follows. In Section 5 we provide detailed instructions about the use of the ODN Platform, DPU invocation and parameters setting and on pipelines, namely clusters of DPUs that can be used to build a coarse grain functionality needed by the user. Section 6 deals with techniques that can be used in tasks and subtasks of phases 2 and 3. Section 7 deals with techniques used in data quality management cross-cutting activity. As to data quality management, we deepened the contents provided in D5.1, in the aspects related to data quality dimensions for the different datasets formats.
5 Using the ODN Platform, DPUs and Pipelines

5.1 Generalities on the ODN platform

In Deliverable D5.1 it has been recommended to choose several software tools which implement components of a reference architecture depicted in Figure 5.1.

The COMSODE project is presently developing an implementation of the reference architecture that is called Open Data Node (ODN). The main objective of this section is to highlight the recommendations, practices and procedures of correct usage of the ODN.

Figure 5.1: The COMSODE reference architecture

Figure 5.2: Implementation of an Open Data Node (ODN)
As described in deliverable D2.3, the implementation of an ODN consists of the following modules (Figure 5.2):

- ODN/UnifiedViews
- ODN/Storage
- ODN/Publication
- ODN/InternalCatalog
- ODN/Catalog
- ODN/Management

In particular in this section we show how to use the UnifiedViews module. UnifiedViews is an Extract, Transform, Load (ETL) and data enrichment tool which enables the configuration of the so called ETL procedures. UnifiedViews enables to extract source data (datasets) from the respective data source and transform them through various techniques so that they can be published as a (linked) open dataset. The result of the transformation is stored in the database managed by ODN/Storage module.

The ODN/UnifiedViews module is responsible for:

1. Extracting data provided by data publishers from data sources;
2. Transforming data to machine readable format; such transformations include enriching the data, assessing and improving data quality;
3. Storing the machine readable data in the database managed by ODN/Storage.

UnifiedViews allows users to define and execute data processing activities (called pipelines) by means of a graphical user interface; the core components of every data processing activity are data processing units (DPUs) that are used to execute specific actions on data.

Other modules of the ODN platform are less relevant for the goal of this deliverable and aren’t further described here. Interested readers can refer to deliverable D2.3.

In the following paragraphs we present an example to create a pipeline using UnifiedViews. The example is an excerpt of the pipeline needed to fulfill the transformation and linking requirements for the publication of the dataset IT_X_16, as defined in Attachment A – Requirements and Techniques of deliverable D3.2. To summarize, the example loads a dataset from a CSV file, which represents the set of preschools of the municipality of Rome. The dataset is then converted in RDF, and the represented concepts are aligned with the ones represented by the standard ontologies FOAF (http://www.foaf-project.org) and Schema (http://www.schema.org). Finally the result is loaded into an RDF store for consumption.
In order to create this transformation process we create a pipeline. In UnifiedViews a pipeline is a set of DPUs that are linked with each other. Every link (called edge) defines the flow of data from one DPU to another. These linked DPUs form paths through which the data can flow. Therefore we say that a pipeline is data-flow oriented.

5.2 Pipeline walkthrough

The proposed task is composed of three subtasks:

1. Creating the pipeline.
2. Constructing the skeleton of the pipeline using the available DPUs.
3. Configuring each DPU in order to specify their behavior.

5.2.1 Creating the pipeline

UnifiedViews allows the creation of multiple pipelines. The pipelines configured are visible in the Pipelines section, as shown in Figure 5.3

![Figure 5.3: A pipeline section](image)

For each pipeline UnifiedViews shows the time of the last execution, its duration, and the outcome. You can also take several actions on pipelines, such as starting the execution, debug it, planning automatic executions using the scheduling features, editing and deleting.

In order to create a new pipeline, the user must click the Create pipeline button. A Pipeline details window appears (Figure 5.4) where pipeline properties can be specified. In this case, write a name and a description, then click Save. It is also possible to choose the visibility of the pipeline as well.

Pipeline visibility can be:

- **Private**: the pipeline is accessible only by its creator.
- **Public (read only)**: anyone can access and run the pipeline, but it can be edited or deleted only by its creator.
Public: anyone can access, run, edit and delete the pipeline.

Figure 5.4: Pipeline property specification

5.2.2 Building the pipeline skeleton using DPUs and links

The DPU (Data Processing Unit) is the minimal unit inside a pipeline. Each DPU is designed to accomplish a specific function or to implement a specific technique, such as converting or normalizing a dataset, computing a data quality metric, enriching data etc. DPUs are divided in three main categories: Extractors, Transformers, and Loaders. A wide variety of DPUs is already available, but the platform can be extended with third-parties custom DPUs, and allows any developer to create new DPUs from scratch using the template provided. In Appendix 1 there is a list of available DPUs.

A link is a graphical representation of the data flow between two DPUs, with an origin and a destination. The data that flow through the link constitute the Output Data of the origin DPU, and the Input Data of the destination DPU. A link has only one origin and one destination, but more than one link could leave a DPU output or enter a DPU input. When more than one link originates from a DPU, the same output data is forwarded to the input of each destination DPU. Likewise, when links originating from different DPUs outputs end in a single DPU input, the data flows are merged and the target DPU receives the data as a single stream.

The pipeline for the example of this section requires the following four steps:

1. Download the CSV file containing the dataset.
2. Convert the dataset into RDF triples.
3. Align the RDF triples with terms defined in the standard ontologies Schema and FOAF.
4. Load the output into a triple store for consumption.

The pipeline employs a different DPU for each step, as shown in the following Table 5.1. In this case there is a one-to-one match between the above mentioned steps and DPUs although this is not always the case.

Table 5-1: selected DPUs for the pipeline

<table>
<thead>
<tr>
<th>Step</th>
<th>Step type</th>
<th>DPU id</th>
<th>DPU name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Extraction</td>
<td>E-DWNLD</td>
<td>e-httpDownload</td>
</tr>
<tr>
<td>2</td>
<td>Transformation</td>
<td>T-TABULAR</td>
<td>t-tabular</td>
</tr>
<tr>
<td>3</td>
<td>Transformation</td>
<td>T-SPARQL</td>
<td>t-sparql</td>
</tr>
<tr>
<td>4</td>
<td>Loading</td>
<td>L-SPARQL</td>
<td>l-rdfToSparql</td>
</tr>
</tbody>
</table>

To build the pipeline we drag the DPUs needed from left pane and drop them onto the canvas on the right, and then we link them with edges:

1. In the left pane, locate the DPU e-httpDownload under the Extractors category.
2. Drag and drop it onto the workspace on the right-hand side.
3. Repeat the operation for the other DPUs. DPUs for step 2 and 3 are located under the Transformers folder, while the DPU l-rdfToSparql is located under the Loaders folder.

The result is shown in Figure 5.5. Adding a DPU to the pipeline leads to create a new instance of that DPU. It is possible to add the same DPU more than once.
Now we can link DPUs in order to define how the data must flow through them. To create a new link we have to:

1. Select the starting DPU.
2. Click on Create new edge. Now you can see a black line that comes out from the DPU and follows the mouse pointer.
3. Click on the target DPU.

In order to define the data flow of the example we have to repeat the above steps to form the following links: from e-httpDownload to t-tabular, from t-tabular to t-sparql, and from t-sparql to l-rdfToSparql. The pipeline now appears as in Figure 5.6.

When a link is created, UnifiedViews automatically selects the output and input to be connected. If there is more than one output or input, UnifiedViews cannot automatically choose: this condition is signaled by means of a red arrow. To map the output with the correct input, click on the red link and then on the Details button (the cog icon). A new dialog window appears, in which the user can define the correct mapping (Figure 5.7).
In this example we have to map the only output of t-tabular with the input of l-rdfToSparql named “input”. To do this:

1. Select the output named “triplifiedTable” from the list of outputs of the source DPU.
2. Select the input name “input” from the list of inputs of the target DPU.
3. Click the Map button (see Figure 5.7).

Figure 5.7: Definition of the correct mapping

The pipeline skeleton is now complete, but each DPU must be properly configured. Before continuing, you can save your work by clicking on Save or Save & Close, at the bottom of the workspace canvas.

5.2.3 Configuring the instances of the DPUs in a pipeline

Every DPU can have a set of configuration parameters that specify its behavior. Such settings depend on the functionality of the DPU and the category to which it belongs. However, every instance of the DPUs in a pipeline can have a Name, and a custom Description, that describe the purpose of the specific DPU instance in the context of the pipeline. The configuration values are always specific to the instance of the DPU, not to the DPU itself.

To change the settings of a DPU instance, click on it and select Show detail (the cog icon). A dialog window appears.

5.2.3.1 Configuring the DPU e-httpDownload

In this example the e-httpDownload DPU downloads the CSV file containing the dataset and saves it on the local file system.
The configuration dialog appears as in Figure 5.8. To configure the DPU follow these steps:

1. In the Name field you can replace the default name with a more representative name for the DPU instance, e.g. “Download”.

2. Optionally tick the Use custom description checkbox to enable the Description field, in which you can type a custom description of the purpose of the DPU in the pipeline.

3. In the URL field, specify the URL from where the CSV file will be downloaded.

4. In the Target - file name and location in output field, specify the path where the downloaded file will be saved.

5. The Max attempts at one download field defines how many times the DPU should try to download the file, in case of connection failures.

6. The Interval between downloads field specifies the time interval (in milliseconds) between each download. You may leave it as default.

### 5.2.3.2 Configuring the t-tabular DPU

The purpose of t-tabular in the example is to convert the dataset to RDF. Each row in the example dataset represents a preschool located in the Municipality of Rome. The DPU t-tabular converts each row of the input CSV file into an RDF resource, identified by an URI. Thus we convert each row into a Preschool resource. Each attribute in the CSV file is treated as an RDF property, so that the DPU outputs a triple for each attribute of each record.

The following table illustrates the mapping between the columns of the CSV file and the RDF properties.
These are the steps needed to configure the DPU instance through the configuration dialog window (Figure 5.9):

1. In the **Name** field you can replace the default name with a more representative name for the DPU instance, e.g. “Convert to RDF”.

2. In the **Description** field you can choose to leave the default description or to provide another one. Tick the **Use custom description** checkbox in case you decide to change the description.

3. Choose the type of the file in input by selecting the appropriate radio button: CSV, XLS or DBF. In this example select CSV.

4. If needed adapt the settings related to the format of the CSV.
   
   1. **Quote char** field: the character that is used in the CSV file to quote strings. In this example it is a double quote.
   
   2. **Delimiter char** field: specifies the character that separates the values of the attributes in each row (usually a comma or a semicolon). In this example it is a semicolon.
5. If there are irrelevant lines at the beginning of the CSV file you can instruct the DPU to ignore them by specifying how many lines to skip in the field **Skip n first lines**. In this example the file doesn’t contain irrelevant lines so we set them to 0.

6. If the CSV file includes a row representing the labels of the columns tick the **Has header** checkbox.

7. You can define which column of the CSV file represents a key for the dataset, if it exists. If the CSV file includes a header line, specify the label of the key column in the **Key column** field. If there is not a header line you can specify the ordinal number of the column. In this example there is no key column in the dataset.

8. In the **Resource URI base** field define the base URI to use in order to generate the URI for every resource produced in output. In this example we use http://COMSODE.eu/preschool.

   **Note**: the form of the URI generated for the resources produced in output depends on whether you have specified a key column or not. In presence of a key column, a unique identification value is available for each input row. In this case the generated URI will have the form \( \text{(Resource URI Base)}/\{\text{Unique key}\} \). For example, suppose that value of the parameter **Resource URI base** is “http://COMSODE.eu/preschool”. If the value of the key column for a given row is “323”, then the generated URI for this RDF resource would be “http://COMSODE.eu/preschool/323”.

   Differently, if a key column is not specified, as for the dataset of this example, the generated URI will be based on the path to the input file, the number of the sheet being processed (in case of XLS spreadsheets), and the number of rows. This is important because many input tables can be processed during one execution of a pipeline. The considered path is relative to the working directory reserved to the execution of the pipeline. For example suppose that the input file is stored at the path “dpu_382/tmp/preschools2011” inside the working directory of the current execution. In this case the generated URI for the 10th row could be “http://COMSODE.eu/preschool/preschools2011/Sheet_1/10”.

9. In the **Encoding** field specify the character set to read the input file. By default it is UTF-8. In this example you can leave the default value.

10. The **Rows limit** field specifies how many input rows to convert. Set the value -1 to convert all the input rows.

11. The field **Class for a row object** allows defining the class to which all resources produced in output belong. In this example every row represents a preschool, so you can assign the value Preschool.

12. Now we can define the mappings between the columns of the input files and the RDF properties. Every time a new row is processed, the DPU produces a resource
representing it, and an RDF triple for each of the properties mapped. Such triples have the resource as subject, the property as predicate, and the column value as object. The procedure to add a mapping is described by the following steps. To better clarify each step we show how to map the column named “Nome” of the example dataset. Repeat these steps for each of the columns of the CSV file, following the mapping rules defined by the previous table.

a. Click the **Add mapping** button: a new empty row is appended to the list of mappings. Mappings can be specified in different ways.

b. In the **Column name** field you can specify either the label or the ordinal number of the column.

c. Select the data type for the value of the property (string, integer, double, etc.). For this example select String.

d. In the **Language** field you can define the ISO code of the language of the values for the property. In the example the dataset is in Italian, so we can type “it”.

e. Finally, specify the URI representing the property in the field **Property URI**. For our example mapping use: http://COMSODE.eu/preschool#name.

13. By default the DPU produces a triple for each value of the columns of every row in input, even for blank values (e.g. NULL values in the original data source). To avoid the generation of such RDF triples tick the **Ignore blank cells** checkbox. For the example check it.

14. If the dataset includes a header row it is possible to generate a triple for each of the mapped properties that specify its label. To do this tick the **Generate labels** checkbox.

15. In this example we also add a column with the ordinal number of the rows processed. This defines an additional property, so that a new RDF triple is produced for every row processed, that associates the generated resource with its row number. To do this tick the **Generate row column** checkbox.
5.2.3.3 Configuring the t-sparql DPU

The purpose of t-sparql in the example is to align the terms of the RDF triples produced by t-tabular with terms of standard ontologies such as Schema\(^3\) and FOAF\(^4\). In this example we show only how to implement the matching with terms of the Schema ontology. The reader could easily extend the example for mapping also with FOAF terms. The following table lists the mappings that will be implemented for the publication of this dataset.

### Table 5-3: publication mapping table

<table>
<thead>
<tr>
<th>Original term type</th>
<th>Original term</th>
<th>Mapping description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Preschool</td>
<td><em>Preschool class (<a href="http://schema.org/Preschool">http://schema.org/Preschool</a>)</em></td>
</tr>
<tr>
<td>Property</td>
<td>name</td>
<td><em>name property (<a href="http://schema.org/name">http://schema.org/name</a>)</em></td>
</tr>
</tbody>
</table>
A new resource of type PostalAddress (http://schema.org/PostalAddress) is created, and the address is associated to that resource through the streetAddress (http://schema.org/streetAddress) property.

| Property | address
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>telephone</td>
<td>telephone property</td>
</tr>
</tbody>
</table>

The mapping is implemented through the SPARQL query reported below. It employs the CONSTRUCT ... WHERE syntax in order to match the RDF triples containing terms to be mapped, and create new ones with the target terms.

PREFIX c:<http://COMSODE.eu/preschool>
PREFIX s:<http://schema.org/>

CONSTRUCT {
  ?s a s:Preschool .
  ?s s:name ?n .
  ?s s:telephone ?t .
  ?s s:address ?pa .
}

WHERE {
  ?s a c:Preschool .
  ?s c:name ?n .
  OPTIONAL { ?s c:telephone ?t } .
  OPTIONAL { ?s c:address ?a }
}

To properly configure t-sparql for the ontology alignment, open the DPU configuration dialog (Figure 5.10) and follow these steps:

1. In the Name field replace the default name with a more representative name for the DPU instance, e.g. “Ontology alignment”.
2. In the Description field you can choose to leave the default description or to provide another one. Tick the Use custom description checkbox in case you decide to change the description.
3. A single t-sparql DPU instance can perform more than one SPARQL query. To add a new query, click the Add query tab button. In this example we need only one query.
4. Type the SPARQL query reported above in the Query field.
5.2.4 Pipeline execution

To execute a pipeline, select one of the existing pipelines from the pipeline lists (see Figure 5.3) and select the run button ( ).

5.2.5 Verify, preview and execute

Before launching the execution of the pipeline, you can check that everything is properly configured by clicking the Validate button, placed at the bottom of the workspace. This verifies the syntactical correctness of the pipeline, and looks for unreachable DPUs and missing connections. If everything is in order, you are ready to preview the output.
6 Techniques for phases

6.1 Preparation of publication

6.1.1 Data sources access configuration

This task provides a methodology for automating the access to existing datasets; either ready to be published or those that need a preparatory phase in order to become ready for publication, as also described in task “Determination of target level of openness” in D5.1. In order to provide a minimum set of tools for accessing sources in different formats, this phase is centered on the definition and automation of a “read-only” access to remote as well as local sources. As to this issue, we recommend in case of CSV files to represent in the first row the header of each column, to support the definition of the schema for the dataset and subsequent mappings to shared vocabularies.

The following table gives a checklist of the main locations, formats and modalities to “access and read” datasets:

<table>
<thead>
<tr>
<th>location</th>
<th>format</th>
<th>Accessing and Reading modalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>Relational Database</td>
<td>User account with read-only privileges to query the data or accessing periodical dumps</td>
</tr>
<tr>
<td></td>
<td>Data files (pdf, excel, CSV)</td>
<td>Tools to read them locally</td>
</tr>
<tr>
<td></td>
<td>RDF</td>
<td>SPARQL based tools to query the data</td>
</tr>
<tr>
<td>Remote</td>
<td>Relational Database</td>
<td>Periodical scripts of database dumps, providing data in different formats that can be accessed (e.g. via HTTP or other remote access protocols) and saved locally.</td>
</tr>
<tr>
<td></td>
<td>Main formats:</td>
<td>Application interface (e.g. Web) providing a read-only user account and methods to query the data available in remote servers (e.g. via REST services), and to choose different formats for local download.</td>
</tr>
<tr>
<td></td>
<td>- .sql files</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- CSV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- XML</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Json</td>
<td></td>
</tr>
</tbody>
</table>
In this section we provide a set of recommendations to support users in preparing and publishing the dataset description using the ODN platform. These recommendations cover the phases P02A02 and P02A03 of the general methodology for open data publishing described in the deliverable 5.1. These phases are aimed at creating the metadata needed to describe a dataset based on the COMSODE methodology. The dataset description will be primarily stored in the ODN Data Catalogue. However, it can also be useful for data publishers or owners to upload and advertise their datasets in other data catalogues. For this reason, the dataset description has to be compatible with cataloguing facilities used in the more prominent platforms to publish Open Data, such as CKAN. It is therefore a goal of COMSODE to define cataloguing methodologies that maximize the interoperability with other open data publishing platforms.

A dataset description consists in a set of metadata associated with the dataset and defined using RDF and one or more metadata schemas, which specify the attributes used to describe the dataset and their semantics (e.g., the name of the dataset can be specified using a predicate “title” whose values will be of type literal). In COMSODE we require that metadata schemas have to be defined by shared vocabularies/ontologies recommended by W3C working groups for dataset cataloguing (e.g., to describe the name of a dataset the property “title” of the Dublin Core vocabulary is used).

### 6.1.2.1 Definition of the catalogue record schema

As described in P02A03-03 of Deliverable 5.1 (Documentation of best practices), there exist several ontologies or vocabularies proposed to catalogue a dataset. The datasets to be published in ODN should be compliant with catalogues that use the DCAT\(^5\) and VoID\(^5\) vocabularies. DCAT and VoID are RDF\(^7\) vocabularies that provide constructs for the specification of metadata used for dataset descriptions, which are designed to facilitate interoperability between data catalogs, increase discoverability and enable applications easily to consume metadata from multiple catalogs published on the Web. For these reasons, the user of the ODN platform is guided to use DCAT and VoID for the representation of metadata. In this section, we provide an overview of these two vocabularies.

**DCAT overview.** Data Catalogue Vocabulary (DCAT) is an RDF based schema proposed to describe generic datasets and support their discovery and usage. DCAT defines three main classes:

- `dcat:Catalog` which represents a catalog of dataset description
- dcat:Dataset which represents a dataset in a catalog.
- dcat:Distribution which represents an accessible form of a dataset.

In DCAT it is not obligatory that the dataset has to be available to be downloaded as a file. If the dataset is available somewhere, and accessible via an API, we can define the dataset as an instance of dcat:Dataset and the API as an instance of the dcat:Distribution class. Another important class in DCAT is dcat:CatalogRecord which represents a dataset entry in the catalog. It is important to make a distinction between dcat:Dataset and dcat:CatalogRecord. While dcat:Dataset represents the dataset itself, dcat:CatalogRecord represents the record that describes a dataset in the catalog. The use of the CatalogRecord is optional and the main reason to use it is to capture provenance information about dataset entries in a catalog. A data catalog conforms to DCAT if:
- It is organized into datasets and distributions
- An RDF description of the catalog itself and its datasets and distributions is available

DCAT does not make any assumption about the format of the dataset described in the catalog. In this way we can use other complementary vocabularies, such as VoID properties, and in particular DCAT makes extensively use of Dublin Core Metadata Terms.

VoID overview. Vocabulary of Interlinked Datasets (VoID) is an RDF based schema that describes linked dataset and allow their discovery and usage. In VoID a dataset is a set of RDF triples (expressions in form of subject-predicate-object) that are published, maintained or aggregated by a single user. An RDF link is a RDF triple whose subject and object are in different datasets. VoID is also used to describe RDF links between datasets. A linkset is a collection of such RDF links between two datasets. In VoID the dataset is modelled as an instance of the void:Dataset class. This instance is a single RDF resource that represents the entire dataset. In order to access a concrete triple contained in the dataset we use access information such as the address of a SPARQL endpoint. A linkset is an instance of the void:Linkset class which is a subclass of void:Dataset. A linked dataset is a meaningful collection of triples, which deal with certain topic, are stored in a certain server, and published by a certain publisher. VoID reuse existing vocabularies, e.g., Dublin Core Metadata, to represent some metadata in an interoperable manner.

We describe more in detail the metadata covered by VoID, which shed light on the kind of information that have to be provided by the user in the cataloguing process. VoID covers four categories of metadata, described in the following.

- General dataset metadata describe the content of the dataset, the licenses and provenance information. This information helps the potential users of a dataset to decide whether the dataset is appropriate for their purposes or not. Dublin Core Metadata Terms, which contain several useful properties, are used to represent these metadata. General metadata can be further classified in four subcategories as follows.
  - Content – The metadata that specifically describe the content of the dataset such as, dataset name (in English or in local language), description, domain, primary and secondary topic.
  - Technical Features – This kind of information is used to express technical features of a dataset, such as the current data format in which the dataset is available.
  - Provenance – Provenance describe the sources that are involved in the production and publication of a dataset. This information is very important
because it helps consumers to evaluate if they can trust the information they use and its origin. For more information about provenance please refer to this link: http://www.w3.org/2005/Incubator/prov/wiki/What_Is_Provenance.

➢ History – History is referred to the information that is related to the creation, modification, release or update date of the dataset. This information is important to the user or the consumer of the data, because they need to know how fresh are the data they are using or plan to use and when was the last modification or the last update of the dataset.

➢ Licensing – Licensing information states under what rights is the dataset published. It also, gives information to the consumers about the terms of use of the dataset

- **Access metadata** describe methods of accessing the RDF triples of the dataset.
- **Structural metadata** provide high-level information about the schema and internal structure of a dataset and can be helpful when exploring or querying datasets. Structural metadata can be further classified in three subcategories as follows.
  - The vocabularies used in the dataset – Every RDF dataset can use one or more vocabularies or ontologies. The vocabulary provides the terms (classes and properties) used to organize the data.
  - Statistics about the size of the dataset – VoID provides a number of properties for expressing numeric statistics about a dataset, such as the number of RDF triples it contains, or the number of entities it describes, the total number of classes, or properties it has, etc.
  - Relevant Partitions – A partition of the dataset can be described more in detail. VoID provides a vocabulary to define a partition, e.g., the partition of a dataset consisting of all the triples that have a given predicate, and to describe some features of these partitions. There could be different reasons that a dataset might be subdivided, for example its part have different provenance, different publication dates, are accessible through different SPARQL endpoint, have different topics or can be downloaded separately in different RDF dumps. In additions, partitions can be useful to describe more in detail a feature of a conceptual partition, e.g., the number of triples that have a given predicate.

- **Describing links (for RDF datasets)** – Links from/to other datasets can be represented in the dataset. VoID support the description of these links.

Some of these metadata can be used with literals and others with non-literal values. A literal is an assignment to an explicit value. A literal can be a string, a Boolean, a real number, a list of numbers, a function etc. A non-literal value is a value which is a physical, digital or conceptual entity. A non-literal can be an identifier, a statement, an expression, an URI etc.

### 6.1.2.2 Selection of the target catalogue

The dataset description will be stored in the ODN Data Catalogue. Since the dataset description is represented using shared vocabularies for data cataloguing, it is possible to publish the description obtained with the ODN platform also in external data catalogue. A description of data catalogues of interest, as well as of the metadata schema required by these catalogues can be found in Deliverable 3.3, Section 7, 8 and 9.
6.1.2.3 Dataset Cataloguing with the T-Metadata DPU

T-Metadata is a DPU aimed to support a user of the COMSODE platform to describe a dataset by annotating it with a set of relevant metadata. T-Metadata is used to publish every dataset in COMSODE.

Technically, T-Metadata implements a content transformation technique, which enriches a dataset with its metadata. Some metadata are provided by the user using a web form, some other metadata, e.g., statistics about the dataset are computed automatically. T-Metadata gets the RDF file of the dataset, computes some statistics about the data like the number of triples, subjects, predicates and objects, and prompt a web form to fill in with additional metadata. The automatically computed statistics and the input of the user are transformed into metadata that are represented using DCAT and VoID vocabularies.

The metadata that require user input are specified when the pipeline for the data publication process is designed by the user. The user input is collected in the configuration panel of the DPU. Then, when the pipeline is run, T-Metadata computes statistics and creates the dataset description with all the metadata.

In the next subsections we will describe in detail the metadata supported by T-Metadata and a guide to use this DPU to complete the cataloguing process.

6.1.2.3.1 Metadata supported by T-Metadata DPU

As mentioned above to describe a dataset we can use a set of attributes. The T-DPU considers only a subset of these attributes. We have described in details this subset.

A detailed explanation of metadata supported in the T-Metadata DPU is given in the Attachment III. In this document, we explain how the spreadsheet can be used as a guide to cataloguing, we describe the main principles underlying these recommendations and we discuss the most frequent problems that users face during the cataloguing process (e.g., metadata that require literal values vs metadata that require non-literal values). As a running example to describe the steps needed to catalogue a dataset we use the dataset (IT_X_21) selected from the datasets to be published in the COMSODE project listed in D2.2.

The spreadsheet is composed of the following columns:

- **Attribute Name**: an intuitive name for the attribute of the dataset described by the metadata
- **Metadata Classification**: the category of the metadata (based on the classification in Section 6.1.2.1)
- **Description**: a description of the attribute
- **Predicate**: a predicate of a well-known vocabulary recommended to represent the attribute
- **Vocabulary**: the vocabulary in which the predicate is defined
● **Type of Range:** specify if the predicate requires a literal or non-literal value (a URI) according to the schema of the vocabulary
● **Range:** the class or data type specified as range of the predicate in the schema of the vocabulary
● **Example:** an example that shows more in detail how the predicate can be used for representing the attribute.
● **Importance:** the level of importance of the attribute that is represented, evaluated on the basis of the usage of the attribute in existing data catalogues (to be incrementally completed during the next project phases).
● **Additional guidelines:** additional recommendations for the usage of the predicate (to be incrementally completed during the next project phases)

6.1.2.3.2 How to describe a dataset using the T-Metadata DPU

T-Metadata can be found in the Transformers Folder / DPU group, as shown in Figure 6.1

![Figure 6.1: T-Metadata DPU in Transformers Folder](source: attachment)

We will see how the example of the dataset 21 in the Attachment III could be implemented in the platform.

Below is the picture of the pipeline of the selected dataset. We will not go into details to describe all the DPUs that constitute this pipeline but we will focus only in the T-Metadata DPU. As you can see from this picture this DPU gets in input RDF data that are created by the t-tabular DPU, which transforms tabular data (DBF, CSV with headers, CSV without headers, XLS, XLSX) into RDF data (see Section 5.2.3.2). However, we show the pipeline in which the t-tabular DPU is used also because for a limited number of metadata, their specification should be consistent with some parameters of the pipeline under usage (for example, the Media Type field in T-Metadata should use the “application/zip” choice because the output of this DPU will be stored in a .zip file.)
Figure 6.2: The pipeline for the Italian Dataset number 21

What we will get if we click on the T-Metadata is the configuration of this DPU as in Figure 6.3:
We have to fill in each field with the needed value. In the upper part we can declare information about the DPU, such as the name (which is filled in by default), if it has a parent, in the second field we have to write the name of the parent of this DPU. After, we can write a general description about T-Metadata. If we check the box “Use the custom description” we use the default description for this DPU and we don’t have to fill in all the above mentioned fields. These parameters are not related to the dataset description and we recommend that the user keep the default configuration.
In the **Output graph name** field we have to write the URI that will be used as name for the RDF description of the dataset. This URI should be dereferenced upon request, so as to return the RDF dataset description when a call to that URI is posted. We recommend to use the name of the dataset (in our example: “beni-confiscati-sportelli-anticrisi”) and then “/metadata” to name the file that contains its metadata in RDF. Thus, in our case, we can write: “http://comsode.disco.unimib.it/resource/dataset/beni-confiscati-sportelli-anticrisi/metadata”

The next field we have to fill in is **COMSODE Dataset ID** which will be used as part of the URI. For the example we are considering, the ID of this dataset is “IT_X_21”.

After, we have to fill in the **Dataset URI** field, which requires the name and location of the dataset for which we are specifying the metadata. In our example, we can use: “http://comsode.disco.unimib.it/resource/dataset/beni-confiscati-sportelli-anticrisi”

In **Distribution URI**, we have to add the URI of the distribution file: “http://comsode.disco.unimib.it/resource/dataset/beni-confiscati-sportelli-anticrisi/distribution”

In **Data dump URL**, we have to write the URL where the data in ZIP archive will be downloadable. “http://comsode.eu/dump/it_x_21-beni-confiscati-sportelli-anticrisi.zip”

In **Media Type**, we have to choose one of the available choices. If you click on the arrow on the left of this icon, you can see a list of the possibilities to fill in this field:

![List of the possible choices for “Media Type” field](image)

The “application/zip” is used when the output of this DPU will be stored in a .zip file. We have to be careful when we enter this link, because it has to be written in the same way as it is in the field Zip file path/name (with extension) of the DPU T-Zipper. We can use one of the other options if the output is different from .zip. In the considered example we have to choose the “application/zip”.

The next field to fill in is **Sparql Endpoint URI**, which provides information about the URI of the Sparql Endpoint. In our example, we can write “http://comsode.eu/sparql” because this dataset will be querable using this particular address

In the field **Contact Point URL** we have to enter the address of a web page where it is possible to contact the organization/person responsible for the data publication. In our example, we can write “http://dati.comune.milano.it/contattaci.html”

In the **Original language – RDF language tag** we have to write the language of the dataset. This dataset is in the Italian language so in this case we can write “it”.

---

Figure 6.4: List of the possible choices for “Media Type” field
The next field to fill in is **Title in original language**, in which information about the title of the dataset in its original language should be specified. In our example, we can write “Beni confiscati alla mafia e sportelli anticrisi”.

In the field named **Title in English** we have to write the title of dataset in English. So we can write: “Confiscated mafia assets and cash crisis”.

In **Description original language** we have to write a description of the dataset, in the original language. We have to give general information about the content of the dataset. In our example, we can write: “Il presente dato alfanumerico reusable contiene la localizzazione (per via e numero civico) dei beni immobili confiscati alla criminalità organizzata, gestiti direttamente o assegnati a terzi”.

In **Description in English** we have to rewrite the general information written in the field above but in English. In our example: “These figures contain reusable alphanumeric paging (for street and number) of property confiscated from organized crime, directly managed or assigned to third parties”.

In the field **Modified** we have to write down the date of the modification of the dataset. The date should be formatted as dd/mm/yyyy. You can write down the date or you can choose it, clicking in the calendar at the right corner of this field. If the date of modification is the current date you are running the pipeline, you can check the box below “Always use current data instead”. In our example, the value for the modified field is “03/01/2013”.

**Periodicity** is another field which holds information about the periodicity with which the dataset is being updated. If you click on the arrow at the end of the right corner you can see some of the periodicity options you may fill in. In the considered example we choose to fill “Annual”
The next field we have to complete is **Available licenses**. Here we have to enter the information about the license associated with the usage of the dataset. This field is composed by 4 sections. The first section is the Available licenses. In this section are listed all the available licenses so far. If the license is in this list you can select it and then click the arrows in the red square. So the right license is now in the right section, in the Selected licenses.
Figure 6.6: How to select a license

If the license associated with your dataset is not listed in the “available licenses” section you can write it down in the field showed with blue square in the picture below. After, you have to click the plus button which is showed in the red square. Then, the license you added is now in the Selected licenses section.

Figure 6.7: How to add a new license

The next field is Available example resources where you can add information about the available example resources. The procedure of adding a new resource, if it is not available is as the example above, in case of the available licenses.

In the field of Available sources you can add information about the available sources. T-Metadata keeps track of resources assigned to other datasets, which are listed in the left-hand side of the selection panel. If the resource that the user want to specify is not listed among the available resources, the user can specify a new resource using the box below the selection panel, as already described for licenses in Figure 6.7.

Figure 6.8: The configuration for the “Available sources” field
The next field to fill in is the Available keywords. In this field we can write some keywords, which can be as representatives of the content of the dataset. If the keyword that the user wants to specify is not listed among the ones that are available (because used in the past), the user can specify a new keyword using the box below the selection panel, as already described for licenses in Figure 6.7. In our example we can add “assets, confiscated assets, crime, mafia, beni confiscati”. After adding all the keywords, this field will look like the Figure 6.10.
The following field to fill in is the **Available themes**. Here we have to enter information about the themes covered in the dataset. The user can add new themes as in the other selection panels (see Figure 6.7). In order to use shared vocabularies, we will use dbpedia for the theme field. So we can fill in this field as in the figure below writing


In the field **Available languages** we can enter information about the available languages of the dataset. The user can add new available languages as in the other selection panels (see Figure 6.7). In our example, the dataset is available only in Italian language so we can write:

http://id.loc.gov/vocabulary/iso639-1/it.html as in the Figure 6.12:

In the field **Available authors** we can enter information about the authors of the dataset. In the field Available publishers we can enter information about the publishers of this dataset, e.g., an organization. An URI that describe the publishers should be used as value of this attribute. In our example, we can write: “http://dati.comune.milano.it/”

After the configuration of each DPU that belong to the pipeline, we debug and run it as described in Section 5.2.1.
As a result of this DPU, as mentioned above, we can calculate some statistics. The computed statistics are represented in the dataset description. In Table 6.2 we list the statistics computed by T-Metadata and we explain how they are represented in the dataset description.

Table 6-2: List of the statistics computed by this DPU and the related representation

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique subject count</td>
<td>We use void:distinctSubjects to represent the total number of distinct subjects in the dataset. In other words, the number of distinct URIs or blank nodes that occur in the subject position of triples in the dataset.</td>
</tr>
<tr>
<td>Unique property count</td>
<td>We use void:properties to represent the total number of distinct properties in the dataset. In other words, the number of distinct property URIs that occur in the predicate position of triples in the dataset.</td>
</tr>
<tr>
<td>Unique object count</td>
<td>We use void:distinctObjects to represent the total number of distinct objects in the dataset. In other words, the number of distinct URIs, blank nodes, or literals that occur in the object position of triples in the dataset.</td>
</tr>
<tr>
<td>Number of triples</td>
<td>We use void:triples to represent the total number of triples contained in the dataset.</td>
</tr>
</tbody>
</table>

After the whole pipeline run, a summary page is returned in the ODN. The summary page contains also a link to the statistics computed by the T-Metadata DPU. In our example, we get the statistics shown in Figure 6.14.

**Basic data metrics**

- Unique subject count 640
- Unique property count 27
- Unique object count 815
- Number of triples 3056

Figure 6.14: The statistics calculated by this DPU
6.1.2.4 Relation between this section and other deliverables

In curating this section we considered inputs from Deliverable 2.2 Criteria for the selection of datasets, Deliverable 3.1 - Final version of the selected dataset list, Deliverable 3.2 – Summary report on user requirements and techniques for data transformation, quality assessment, cleansing, data integration and intended data consumption of the selected datasets, and Deliverable 3.3 Definition of ways to advertise the published datasets to Open Data Catalogues.

In D2.2 the criteria for the selection of the datasets are defined. In this deliverable, a number of metadata for describing the dataset and support the selection of the ones to be published in COMSODE has been defined.

In particular we considered the descriptions of the datasets in D3.1. In curating this section we considered inputs from use as running example also considered Deliverable 3.1 - Final version of the selected dataset list, and the metadata used for describing the dataset.

However, the recommendation in this document may provide different guidelines for some specific metadata, based on a further investigation of the specifications of VoID and DCAT with reference to the Deliverable 3.2 – Summary report on user requirements and techniques for data transformation, quality assessment, cleansing, data integration and intended data consumption of the selected datasets.

6.1.3 Designing data schemas and ontologies for datasets published as 3*, 4* and 5* data

A data schema describes the structure of the dataset and partly also the semantics of data instances. In case of 3* and 4* data, a data schema for a dataset is optional but recommended. In D5.1 some preparation and publication formats are recommended for these kinds of datasets. For data represented in CSV we recommend to describe the content of each column with a label in a natural language and enclose these labels in the first row of the table.

A data schema of a dataset published as 5* data (represented in RDF) is specified as a so called vocabulary or ontology, which is defined in one of the RDF-compatible languages, like RDFS and OWL2. Intuitively, these languages will define the meaning of classes (e.g., foaf:Person), datatypes (e.g., xsd:integer) and properties (e.g., dcterms:author) used to define the types of resources and literals, and the predicates used in the RDF data set. The choice of the terms “vocabulary” and “ontology” when referring to a resource might depend on several pragmatic and contextual reasons (e.g., the expressivity of the adopted language, the depth of the modeling choices, and so on), which are not relevant to the purpose of this document. Therefore, in the following we will use the terms vocabulary and ontology, or the term vocabulary/ontology interchangeably.

A complete specification of the RDFS and OWL2 languages can be found at http://www.w3.org/TR/rdf-schema/ and http://www.w3.org/TR/owl2-overview/. An introductory guide to create a schema (an ontology) with RDFS and OWL2 can be found at http://www.linkeddatatools.com/introducing-rdfs-owl. Protégé® is a widely adopted, free, and open source modeling tool for the design of ontologies in OWL2. We recommend that the users
who want to define an ontology to publish their data and do not have a strong background in ontological modeling use RDFS or a small subset of OWL2 that is close in meaning to the expressivity of RDFS. This subset may include the specification of: subclass relations (one class is subclass of another class), subproperty relations (one property is subproperty of another property), and domain and range restrictions (all the resources that occur as subjects in triples with a given property belong to a given class, i.e., the domain of the property; all the resources that occur as objects in triples with a given property belong to a given class or datatype, i.e., the range of the property).

For example, let us consider a very simple data set that represent information about people and the organization they are member of. Let myonto: be the name space of the URI where the ontology myonto is represented. The schema of these data can be represented in RDFS or OWL2 ontology by:

- introducing a class named myonto:Person,
- introducing a class named myonto:Organization
- introducing a property named myonto:memberOf
- introducing a domain and a range restriction that specifies that myonto:Person and myonto:Organization are respectively the domain and the range of the property myonto:memberOf

Although RDFS is an ontology modeling language simpler than OWL2, for a user who wants to design the ontology with Protégé it might be more convenient to produce a OWL2 ontology (with a limited number of simple constructs) instead of a RDFS ontology, because an ontology designed with the Protégé’s user interface is represented in OWL2.

**Definition of a data schema and reuse of existing vocabularies/ontologies.** When defining a schema for a dataset, it is important to bear in mind that several well-known and widely adopted vocabularies/ontologies exist, which can be used in the definition of the schema. For example, if a dataset provides information about people and the organization they are member of, it is possible to use the predicate foaf:member_of to represent the relation between a person and its organization. Reusing existing vocabularies/ontologies is a good practice and makes it easy for third parties to understand and integrate the content of the published data. At the same time, reusing a piece of a vocabulary requires attention because the user might misuse classes/predicates defined in the vocabulary by breaking their semantics. For example, since foaf:Organization is defined as the range of foaf:member_of in FOAF, one should avoid using foaf:member_of with literal values.

**Resources for schema definition.** In what follows, a summary table reports catalogues and search engines where it is possible to find existing vocabularies/ontologies. The table also reports prominent domain specific and cross-domain vocabularies/ontologies that can be reused to model a schema of a 5* dataset. We also include widely adopted taxonomies and concept schemes that can be helpful for annotating data instances.

### Table 6-3: List of available Vocabularies and Ontologies

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Name</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online vocabulary and ontology catalogues and</td>
<td>Linked Open Vocabularies</td>
<td><a href="http://lov.okfn.org/dataset/lov/">http://lov.okfn.org/dataset/lov/</a></td>
<td>Catalogue of recommended ontologies with classification of the ontologies into domains</td>
</tr>
<tr>
<td>Search Engines</td>
<td>Description</td>
<td>URL</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Sindice</td>
<td>Search engine in the Linked Open Data space which can also be used to search for ontologies used in the indexed datasets</td>
<td><a href="http://sindice.com">http://sindice.com</a></td>
<td></td>
</tr>
<tr>
<td>Swoogle</td>
<td></td>
<td><a href="http://swoogle.umbc.edu/">http://swoogle.umbc.edu/</a></td>
<td></td>
</tr>
<tr>
<td>Watson</td>
<td></td>
<td><a href="http://watson.kmi.open.ac.uk/WatsonWUI/">http://watson.kmi.open.ac.uk/WatsonWUI/</a></td>
<td></td>
</tr>
<tr>
<td>BioPortal</td>
<td>A catalogue of ontologies from the domain of life sciences</td>
<td><a href="http://bioportal.bioontology.org/">http://bioportal.bioontology.org/</a></td>
<td></td>
</tr>
<tr>
<td>W3C Semantic Web Ontologies</td>
<td>A list of ontologies maintained by W3C</td>
<td><a href="http://www.w3.org/standards/semanticweb/ontology">http://www.w3.org/standards/semanticweb/ontology</a></td>
<td></td>
</tr>
<tr>
<td>EU ISA Programme ontologies</td>
<td>A list of ontologies produced by EU ISA initiative</td>
<td><a href="https://joinup.ec.europa.eu/community/semic/og_page/studies#core-vocabularies">https://joinup.ec.europa.eu/community/semic/og_page/studies#core-vocabularies</a></td>
<td></td>
</tr>
<tr>
<td>Ontology Design Pattern Initiative</td>
<td>A list of Domain ontologies with related patterns</td>
<td><a href="http://ontologydesignpatterns.org/wiki/Community:Domain">http://ontologydesignpatterns.org/wiki/Community:Domain</a></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Widely Adopted Ontology and Metadata Vocabularies/Languages</th>
<th>DCMI Metadata Terms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCMI Metadata Terms</td>
<td><a href="http://dublincore.org/documents/dcmi-terms/">http://dublincore.org/documents/dcmi-terms/</a></td>
<td>Dublin Core schema</td>
</tr>
<tr>
<td>Simple Knowledge Organization System (SKOS)</td>
<td><a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a></td>
<td>SKOS schema</td>
</tr>
<tr>
<td>RDF Schema</td>
<td><a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a></td>
<td>RDF schema</td>
</tr>
<tr>
<td>OWL2</td>
<td><a href="http://www.w3.org/TR/owl2-overview/">http://www.w3.org/TR/owl2-overview/</a></td>
<td>Owl schema</td>
</tr>
<tr>
<td>Functional Requirements for Bibliographic Records (FRBR)</td>
<td><a href="http://purl.org/vocab/frbr/core#">http://purl.org/vocab/frbr/core#</a></td>
<td>For representing publications and documents</td>
</tr>
<tr>
<td>Open Annotation Core Data Model</td>
<td><a href="http://www.openannotation.org/spec/core/">http://www.openannotation.org/spec/core/</a></td>
<td>For representing annotations to publications and documents</td>
</tr>
<tr>
<td>Domain specific ontologies</td>
<td>GoodRelations</td>
<td><a href="http://purl.org/goodrelations/v1#">http://purl.org/goodrelations/v1#</a></td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>RegOrg</td>
<td><a href="http://www.w3.org/TR/vocab-regorg/">http://www.w3.org/TR/vocab-regorg/</a></td>
<td>For representing organizations registered in national registries</td>
</tr>
<tr>
<td>PublicContractsOntology</td>
<td><a href="http://purl.org/procurement/public-contracts#">http://purl.org/procurement/public-contracts#</a></td>
<td>For representing public contracts</td>
</tr>
<tr>
<td>LEX Ontology</td>
<td><a href="http://purl.org/lex#">http://purl.org/lex#</a></td>
<td>For representing legal documents</td>
</tr>
<tr>
<td>Friend-of-a-Friend</td>
<td><a href="http://xmlns.com/foaf/spec/">http://xmlns.com/foaf/spec/</a></td>
<td>For representing people and their relationships</td>
</tr>
<tr>
<td>GeoNames</td>
<td><a href="http://www.geonames.org/ontology/documentation.html">http://www.geonames.org/ontology/documentation.html</a></td>
<td>For representing geospatial semantic information</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multi-domain and standard upper ontologies</th>
<th>schema.org</th>
<th><a href="http://schema.org">http://schema.org</a></th>
<th>A set of ontologies which cover different aspects and domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opencyc</td>
<td><a href="http://www.cyc.com/platform/opencyc">http://www.cyc.com/platform/opencyc</a></td>
<td>A general knowledge base, including Dbpedia, Wordnet and foaf terms</td>
<td></td>
</tr>
<tr>
<td>DBpedia Ontology</td>
<td><a href="http://wiki.dbpedia.org/Ontology">http://wiki.dbpedia.org/Ontology</a></td>
<td>The ontology used to organize the DBpedia knowledge base</td>
<td></td>
</tr>
<tr>
<td>YAGO2 schemas</td>
<td><a href="https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yagnag/yago/downloads/">https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yagnag/yago/downloads/</a></td>
<td>The schemas (types and relations) used to organize the YAGO knowledge base</td>
<td></td>
</tr>
<tr>
<td>SUMO – Suggested Upper Merge ontology</td>
<td><a href="http://www.ontologyportal.org/">http://www.ontologyportal.org/</a></td>
<td>A generalist ontology with many domain ontologies attached under it</td>
<td></td>
</tr>
<tr>
<td>DOLCE</td>
<td><a href="http://www.loa.istc.cnr.it/old/DOLCE.html">http://www.loa.istc.cnr.it/old/DOLCE.html</a></td>
<td>Descriptive ontology for linguistic and Cognitive engineering</td>
<td></td>
</tr>
<tr>
<td>formal ontology</td>
<td>med.de/ontologies/gfo/</td>
<td>processes and objects: An ontology supporting information retrieval, analysis and integration in scientific and other domains</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>BFO – Basic formal ontology</td>
<td><a href="http://www.ifomis.org/bfo/">http://www.ifomis.org/bfo/</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Most reused taxonomies</th>
<th>EUROVOC</th>
<th>Multilingual thesaurus of European Union</th>
</tr>
</thead>
<tbody>
<tr>
<td>NACE</td>
<td><a href="http://ec.europa.eu/eurostat/ramon/ontologies/nace.rdf">http://ec.europa.eu/eurostat/ramon/ontologies/nace.rdf</a></td>
<td>Nomenclature of Economic Activities</td>
</tr>
<tr>
<td>NUTS</td>
<td><a href="http://ec.europa.eu/eurostat/ramon/rdfdata/nuts2008/">http://ec.europa.eu/eurostat/ramon/rdfdata/nuts2008/</a></td>
<td>Nomenclature of territorial units for statistics</td>
</tr>
<tr>
<td>ISO 4217</td>
<td><a href="http://www.iso.org/iso/home/standards/currency_codes.htm">http://www.iso.org/iso/home/standards/currency_codes.htm</a></td>
<td>Currency codes</td>
</tr>
<tr>
<td>ISO 3166-1</td>
<td><a href="http://www.iso.org/iso/country_codes.htm">http://www.iso.org/iso/country_codes.htm</a></td>
<td>Country codes</td>
</tr>
<tr>
<td>UN/CEFACT</td>
<td><a href="http://www.unece.org/fileadmin/DAM/cefact/recommendations/rec20/rec20_rev3_Annex3e.pdf">http://www.unece.org/fileadmin/DAM/cefact/recommendations/rec20/rec20_rev3_Annex3e.pdf</a></td>
<td>units of measurement</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Taxonomies</th>
<th>Data Portals</th>
<th>World Health Organization portal</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>NACE</td>
<td><a href="http://open-data.europa.eu/cs/data/dataset/eurovoc">http://open-data.europa.eu/cs/data/dataset/eurovoc</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUTS</td>
<td><a href="http://ec.europa.eu/eurostat/ramon/ontologies/nace.rdf">http://ec.europa.eu/eurostat/ramon/ontologies/nace.rdf</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISO 4217</td>
<td><a href="http://www.iso.org/iso/home/standards/currency_codes.htm">http://www.iso.org/iso/home/standards/currency_codes.htm</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISO 3166-1</td>
<td><a href="http://www.iso.org/iso/country_codes.htm">http://www.iso.org/iso/country_codes.htm</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Recommendations for schema definition.** To define a schema for a 5* dataset and to make the published data interoperable with other data published on the Web a user should carry out three main cognitive tasks.

1. Understand the types of entities described in the dataset and their mutual relationships
2. Choose whether to create brand new classes and properties or to reuse existing ontologies by referencing / customizing / expanding their concepts and properties [Schaible et al., 2014]

3. Define the ontology

4. Align the new schema classes and / or properties (if any) to other existing schemas [Wölger et al., 2011; Atencia et al. 2014]

Point 1 and 2 are discussed in the following paragraphs, whereas point 3 is only introduced and will be discussed in detail in D5.4.

In the definition of the schema for a dataset, a publisher can adopt different strategies depending on his/her willingness to reuse existing vocabularies/ontologies. We can consider the following two reuse strategies at the opposite ends of a reuse vs. definition-from-scratch spectrum:

A) A maximum reuse strategy, where the publisher trusts the most popular vocabularies/ontologies and is interested in assuring maximum interoperability for future integration with other datasets. According to this strategy, the schema will contain the maximum number of reusable predicates in existing popular ontologies;

B) A minimum reuse strategy, where the publisher minimizes the reuse of external ontologies in order to strictly adhere to the specific domain nomenclature / dialect. According to this strategy, the schema will contain the minimum number of reusable terms (e.g. only the most general and common ones) from existing ontologies in favor of highly customized brand new predicates.

Furthermore, if we consider the process of schema definition, a publisher can follow two main methodologies to define schema:

i. a top-down strategy (ontology first, high-quality schema, higher cost), where the publisher first defines the ontology (with eventually aggregation of existing vocabulary predicates) and publishes it, and then he uses the published ontology to define a schema for the dataset;

ii. a bottom-up strategy (mapping first, low-quality schema, low cost), where the publisher defines the schema at the same time when he/she transforms the data in RDF, by specifying the predicates used in the RDF transformation. After the data are published with the selected predicates and classes, the publisher better specifies the ontology/vocabulary of the schema by subsequent transformations, mappings and other operations.

The problem of aligning a data schema to other schemas (Step 4 in schema definition) will be discussed more in detail in D5.4. However, a brief discussion of possible strategies to schema alignment is useful because these strategies interplay with the strategy adopted to define a schema (e.g., minimum reuse vs. maximum reuse, and top-down vs bottom-up). Well-known schema alignment representation strategies to schema alignment are the following (the strategies are not mutually exclusive):

1. Implicit alignment at source schema: the source schema is aligned by definition. For example, the schema reuses or specializes predicates or classes of a widely adopted vocabulary/ontology.

2. Transformation with substitution: the triples including predicates / classes of the source schema, i.e., the instance data, are replaced with triples including predicates and classes from the destination schema.

3. Data Addition: triples with predicates/classes of the destination schema (the one to which the dataset schema is aligned to) are added to the triples of the source dataset.
4. Mapping Addition: schema mappings are explicitly added to the dataset, while instance data are represented with predicates/classes of the source schema.

Several possible approaches to the definition of a schema can be devised stemming from the combination of the above reuse strategies, methodologies, and alignment strategies. In the following we describe three schema definition techniques, and we explain how the ODN user can implement them using, e.g., the t-tabular DPU when transforming a tabular data into a RDF data.

**T-TOPMIN**: “top down” and “minimum reuse”. With this technique the ODN user defines the schema first and can adopt one of the following schema alignment strategies: “implicit alignment at source schema” or “data addition” or “mapping addition”. To implement the T-TOPMIN technique it is necessary to use an open source tool such as Protégé® in order to create a schema with brand new classes and properties, import existing schemas to one own schema, and manually linking classes/predicates, e.g., using the <rdfs:subClassOf> predicate. When the ontology that represents the schema is created and published, it will be possible to use the classes and predicates of the new ontology when the data are transformed in RDF with the t-tabular DPU.

**T-BOTMIN**: “bottom-up” and “minimum reuse”. With this technique the ODN user defines the schema when he publish the data, e.g., using a data transformation DPU such as t-tabular. Then he can use another data transformation DPU to adopt one of the following schema alignment strategies: “transformation with substitution” or “data addition” or “mapping addition”. To implement the T-BOTMIN technique the t-tabular and t-sparql DPUs can be used. As shown in the following screenshot, an entity may be defined by specifying it in the Resource URI base field of the t-tabular DPU:
Figure 6.15: A screenshot of the t-tabular DPU interface panel for the assignment of schema entities to csv data.

Multiple entities may be defined also in the same form:
Figure 6.16: A screenshot of the t-tabular DPU interface panel for the assignment of schema entities to multiple csv columns.

A more detailed exemplification of the use of the t-tabular DPU is contained in Section 5.2.3.2 of this Deliverable. The bottom-up mapping between terms of different schemas may be executed by exploiting the t-sparql DPU, according to the example reported in Section 5.2.3.3 of this Deliverable.

**T-BOTMAX**: “bottom-up” and “maximum reuse”. With this technique the ODN user defines the schema when he publish the data, e.g., using a data transformation DPU such as t-tabular. In this process he/she adopts an “implicit alignment at source schema” strategy. To adopt this strategy, when he/she specify the mapping from columns in the tabular data source to RDF predicates in the configuration panel of t-tabular (see Section 5.2.3.2 for further details), he/she selects only predicates defined in existing vocabularies/ontologies.

### 6.1.4 Determining entity identifiers and conventions for URLs of entities in datasets published as 4* and 5* data

This recommendation merges the two separate ones of D5.1, namely: “Determining entity identifiers” and “Convention for URLs of entities in datasets published as 4* and 5* data”.

Datasets may represent real-world entities in the following forms:
- rows in tables
- elements in XML documents
- objects in JSON documents
- RDF entities
Furthermore, for 4* and 5* data, each entity identifier should be designed as a URLs according to some conceptualization and standard.

Rules for constructing entity identifiers may be summarized as follows:

- To have only one (identifying) attribute for the entity;
- To consider informative identifiers (such as for example real world codifications or real systems identifiers conventions) and avoid using meaningless artificially given identifiers (e.g. database records auto increment id), unless they are the only ones at disposal;
- To describe each identifier in the data schema of the dataset.

Regarding entity URLs design for 4* and 5* datasets, some rules may be summarized in the following list:

1. They should be in form of a machine readable representation of the entity, as the entity identifier;
2. They should be stable (do not change during the whole lifecycle of the entity);
3. They should link the entity from other related entities in the same or other;
4. In case of 5* data, they should be dereferenceable. (i.e. when a client requests the URL via the HTTP protocol it receives a machine readable representation of the entity in RDF format serialized in an appropriate format - e.g., TTL or JSON-LD - from the server).

URL design patterns have been proposed in the literature, depending on the entity type, on some aspects regarding the URL main functions (e.g. linking, naming, multi-representations, and so on), and on clusters of problems and solutions that should be addressed by URL foundation mechanisms [Abbas & Ojo, 2013]. The 8 patterns regard the following aspects of URLs design:

- Dereferenceability
- Human readable aspect
- Uniqueness
- Immutability
- Stability
- Longevity
- Multiple representations aspect
- Quality

Besides the above URL design pattern conceptualization, well known standards for URL construction constitute some major guidelines [Berners-Lee, 1998; Davidson, 2009; W3C, 2012; Archer, 2013]. Basic pattern guidelines for standard URLs construction may be also found in D5.1, in paragraph “Convention for URLs of entities in datasets published as 4* and 5* data".
6.1.5 Definition of the approach to the dataset publication

6.1.5.1 Approaches to anonymization of datasets

Anonymization of sensible data such as personal information or business secrets needs to be tackled in order to prevent the risk of publishing protected data while preparing dataset access, dataset linking and publication. Anonymization techniques belong to the more general problem of Privacy-Preserving Data Publishing [Fung et al. 2010], and range from the removal of sensible data attributes from a dataset (referred to as Projection in D5.1), to averaging of single information (Aggregation in D5.1), and reduction or removal of links (referred to as Link removal in D.5.1). In particular, in the present methodology, anonymization is treated according to the following approaches:

Projection, where particular attributes with protected data are removed from the dataset, e.g., in case of tabular files, it means removing a column or columns. This approach corresponds to the techniques listed below:

- **T-PDR** Personal Data Removal (name and address)
- **T-RINR** Record Identification Number Removal

These techniques are implemented in the following DPUs:

<table>
<thead>
<tr>
<th>Technique ID</th>
<th>DPU ID</th>
<th>DPU description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-PDR</td>
<td>T-XSLT</td>
<td>Transform input XML files to output RDF files according to XSLT template</td>
</tr>
<tr>
<td>“</td>
<td>T-SPARQL</td>
<td>Perform a series of SPARQL Construct queries for linkage and ontology alignment</td>
</tr>
<tr>
<td>T-RINR</td>
<td>T-SPARQL-SELECT</td>
<td>Extract a CSV file from RDF data using SPARQL SELECT query</td>
</tr>
</tbody>
</table>

Aggregation, whose main approaches aim to data perturbation (e.g. by means of generalization, suppression or permutation, and randomization of values), in order to minimize the chances of identifying individual records while maximizing the accuracy of data querying. In particular, differential privacy [Dwork, 2006; McSherry & Talwar, 2007] focuses on data randomization techniques of statistical databases, in order to release statistical information about a database without revealing information about its individual entries; K-anonymity, L-diversity, and T-closeness [Ghinita et al. 2009; Cormode & Srivastava 2009; Iyengar 2002], as well as d-presence [Nergiz et al., 2007] are sophisticated approaches to replace single values with averaged data, and to partition them into equivalence classes, so that the information for each person contained in the dataset cannot be distinguished from that of a class of individuals whose information also appear in the same dataset release. Techniques that support these approaches are the following:

- **T-DPRI** Differential Privacy
T-KAN          K-Anonymity
T-LDIV            L-diversity
T-TCLLO       T-closeness
T-DPRE         δ-presence
T-SAR         Specific attribute randomization

These techniques are implemented by the following open source tools available online:

**Table 6-5: Anonymization techniques**

<table>
<thead>
<tr>
<th>Technique ID</th>
<th>Tool Name</th>
<th>Tool Language</th>
<th>Tool input</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-KAN</td>
<td>Arx(^{10}) and UTD Anonymization toolbox(^{11})</td>
<td>Java</td>
<td>CSV files</td>
</tr>
<tr>
<td>T-LDIV</td>
<td>“ “</td>
<td>“</td>
<td>“</td>
</tr>
<tr>
<td>T-TCLLO</td>
<td>“ “</td>
<td>“</td>
<td>“</td>
</tr>
<tr>
<td>T-DPRE</td>
<td>Arx</td>
<td>“</td>
<td>“</td>
</tr>
<tr>
<td>T-SAR</td>
<td>Data-anonymization(^{12})</td>
<td>Ruby</td>
<td>All RDBMS databases</td>
</tr>
</tbody>
</table>

Link removal or linkability reduction removes the external links that may hinder data-privacy before publishing the dataset. In the literature, this approach is also referred to as “Pseudonymisation” [e.g. Narayanan & Shmatikov, 2009], as it is not a method of anonymisation, but it is merely a security measure intended to reduce the linkability of a dataset with the original identity of a data subject.

**6.1.5.2 Approach to updates of datasets**

This task provides suggestions for subsequent modifications, evolutions and newer versions of the published datasets, with the possibility to maintain older versions of them or to make available only their last update. This also depends on the user needs (e.g. history and versioning services are useful for statistical purposes or for the alignment of the last user version of the dataset with the latest published official version). The update is also related to the possibility of representing the modification of data through the exploitation of suitable metadata that may record the history of a dataset and through which, for example, reconstructing datasets evolution. This topic will be further investigated in deliverable D5.4.
6.1.6 Design and implementation of the ETL procedures

6.1.6.1 Designing extractors

The DPUs in the Extract phase collect input data from data sources; they provide inputs to the pipeline. Table 6.6 lists all the DPUs developed so far in UnifiedViews. The requirements referring to DPUs are described in Deliverable 3.2, we consider in Table 6.6 only those DPUs implemented in UnifiedViews.

<table>
<thead>
<tr>
<th>DPU Id</th>
<th>DPU name</th>
<th>DPU name in UnifiedViews</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-FS</td>
<td>File system extractor</td>
<td>E-FilesFromLocal</td>
<td>File system</td>
<td>File</td>
<td>Extract a file or directory from a file system</td>
</tr>
<tr>
<td>E-DWNLD</td>
<td>Generic downloader</td>
<td>e-httpDownload</td>
<td>URL</td>
<td>File</td>
<td>Downloads a single file from URL given its configuration</td>
</tr>
<tr>
<td>E-DTGNRT</td>
<td>Rdf Data Generator</td>
<td>E-rdfDataGenerator</td>
<td>RDF data</td>
<td></td>
<td>Generates specified number of unique triples to RDF data. Usually used for testing purposes</td>
</tr>
<tr>
<td>E-SPARQL</td>
<td>Universal SPARQL extractor</td>
<td>E-RdfFromSparqlEndpoint</td>
<td>SPARQL endpoint</td>
<td>Query result</td>
<td>Executes a given SPARQL query on a SPARQL endpoint</td>
</tr>
<tr>
<td>E-SLKLNK</td>
<td>Silk Linker</td>
<td>E-SilkLinker</td>
<td>Silk configuration file</td>
<td>RDF data</td>
<td>Creates links between RDF resources based on the Silk Link Specification Language</td>
</tr>
<tr>
<td>E-UPL2FLS</td>
<td>Upload to files</td>
<td>E-UploadToFiles</td>
<td>Files</td>
<td>Files</td>
<td>Upload files that will be used during the pipeline execution</td>
</tr>
</tbody>
</table>

Table 6-7: Description of the features available on the Show detail tab of the E-RdfFromSparqlEndpoint DPU.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the DPU. Note: This name has to be unique in a single transformation.</td>
</tr>
<tr>
<td>Description</td>
<td>Description allows you to clarify the purpose of the DPU. Select the</td>
</tr>
</tbody>
</table>
6.1.6.2 Designing transformers

The DPUs in the Transformation phase are shown in Table 6.8.

Table 6-8: DPUs for the transformation substep

<table>
<thead>
<tr>
<th>DPU Id</th>
<th>DPU name</th>
<th>DPU name in UnifiedViews</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-FLFLT</td>
<td>Files Filter</td>
<td>T-FilesFilter</td>
<td>Files</td>
<td>Files</td>
<td>Perform a files filtering</td>
</tr>
<tr>
<td>T-FLRNM</td>
<td>Files Renamer</td>
<td>T-FilesRenamer</td>
<td>Files</td>
<td>Files</td>
<td>Perform a files renaming</td>
</tr>
<tr>
<td>T-FILE2RD</td>
<td>Files to RDF Transformer</td>
<td>T-FilesToRdf</td>
<td>RDF file</td>
<td>RDF data</td>
<td>Extract RDF data from RDF files</td>
</tr>
<tr>
<td>T-MDT</td>
<td>Metadata Form</td>
<td>T-Metadata</td>
<td>RDF data</td>
<td>RDF data</td>
<td>Compute metadata including cataloguing and statistic like metadata</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>T-RDFVALDT</td>
<td>RDF data validator</td>
<td>T-RdfDataValidator</td>
<td>RDF data</td>
<td>RDF data</td>
<td>Perform a validation for RDF data</td>
</tr>
<tr>
<td>T-RDFMRG</td>
<td>RDF Merger</td>
<td>T-RdfMerger</td>
<td>RDF data</td>
<td>RDF data</td>
<td>Merges RDF data</td>
</tr>
<tr>
<td>T-RDF2FIL</td>
<td>RDF to File</td>
<td>T-RdfToFiles</td>
<td>RDF data</td>
<td>RDF file in one of the formats: TTL, RDF/XML, N-Triples, N3, TRIX, TRIG</td>
<td>Put RDF data to RDF file</td>
</tr>
<tr>
<td>T-SPARQL</td>
<td>SPARQL</td>
<td>T-SPARQL</td>
<td>RDF data</td>
<td>RDF data</td>
<td>Perform a series of SPARQL Update queries for content transformation, linkage and ontology alignment</td>
</tr>
<tr>
<td>T-SPARQL-SELECT</td>
<td>SPARQL SELECT -&gt; CSV</td>
<td>T-SPARQLSelect</td>
<td>RDF data</td>
<td>CSV</td>
<td>Extract a CSV file from RDF data using SPARQL SELECT query</td>
</tr>
<tr>
<td>T-UNZIP</td>
<td>Unzipper</td>
<td>T-UnZipper</td>
<td>Zip data</td>
<td>Any format of the file</td>
<td>Extract file from the ZIP archive</td>
</tr>
<tr>
<td>T-XSTL</td>
<td>XSLT</td>
<td>T-XSLT</td>
<td>XML data + XSLT template</td>
<td>RDF data</td>
<td>Transform input XML files to output RDF files according to XSLT template</td>
</tr>
<tr>
<td>T-ZIP</td>
<td>Zipper</td>
<td>T-Zipper</td>
<td>Files</td>
<td>Zip data</td>
<td>Zip input files</td>
</tr>
<tr>
<td>T-HTMLCSS</td>
<td>HTML CSS</td>
<td>UK-T-HtmlCss</td>
<td>HTML</td>
<td>RDF data</td>
<td>Parse a HTML using CSS selector</td>
</tr>
<tr>
<td>T-TABULAR</td>
<td>Tabular File</td>
<td>t-tabular</td>
<td>CSV, XLS, DBF, ODF</td>
<td>RDF data</td>
<td>Transform the tabular data to basic RDF representation</td>
</tr>
</tbody>
</table>

6.1.6.3 Designing loaders

The DPUs in the Load phase are shown in Table 6.9.
Table 6-9: DPUs for the load substep

<table>
<thead>
<tr>
<th>DPU Id</th>
<th>DPU name</th>
<th>DPU name in UnifiedViews</th>
<th>Input</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-FL2FS</td>
<td>File to Local File</td>
<td>L-FilesToLocalFiles</td>
<td>Files</td>
<td>PATH</td>
<td>Loads files to the specified local host directory</td>
</tr>
<tr>
<td></td>
<td>System</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L-FL2SCP</td>
<td>File to SCP</td>
<td>L-FilesToScp</td>
<td>Files</td>
<td>URL</td>
<td>Perform file uploading using SCP protocol</td>
</tr>
<tr>
<td>L-FL2SPARQL</td>
<td>Files to SPARQL</td>
<td>L-FilesToSparql</td>
<td>Files</td>
<td>RDF data</td>
<td>Loads RDF data stored in Files to the specified remote SPARQL endpoint</td>
</tr>
<tr>
<td>L-FL2VIRTUOSO</td>
<td>Files to Virtuoso</td>
<td>L-FilesToVirtuoso</td>
<td>Files</td>
<td>RDF data</td>
<td>VirtuosoLoader issues Virtuoso internal functions to load directory of RDF data</td>
</tr>
<tr>
<td>L-RDF2SPARQLENDPT</td>
<td>RDF to SPARQL Endpoint</td>
<td>L-RdfToSparqlEndpoint</td>
<td>RDF data</td>
<td>RDF data</td>
<td>Loads RDF data to the specified remote SPARQL endpoint</td>
</tr>
<tr>
<td>L-SOLR</td>
<td>Solr</td>
<td>UK-L-Solr</td>
<td></td>
<td></td>
<td>Import given CSV into given Solr instance</td>
</tr>
</tbody>
</table>

6.1.7 Testing of the ETL procedures

The testing of the ETL procedures consists of two phases, the first implies the validation of the pipeline consistency, and the second is the verification of the produced data. As indicated previously, a first validation of the pipeline can be carried out through the Validate button. This verifies the syntactical correctness of the pipeline, and looks for unreachable DPUs and missing connections. UnifiedViews allows debugging the pipeline through a user interface which providing a detailed report of the execution of DPUs, see Figure 6.17.
To verify the correctness of the ETL procedures it is necessary to analyze data produced by them. In this context it is almost impossible to predict all the possible types of errors that can occur with real data. A possible solution is the creation of test data and test scenarios. Each implemented ETL procedure must be tested with test data according to testing scenarios. The production of test data can be performed by sampling data from the original dataset. In this case, after the execution of the ETL procedures it is necessary to verify that the output generated data meet the requirements, through an analysis of these data, or else by making a comparison with a representative manually produced dataset. In alternative, when datasets involved are in RDF, data can be generated using the E-RdfDataGenerator DPU, see Figure 6.2.
6.1.8 Licensing

Based on the type of dataset to open (e.g. a public sector dataset), the part of the dataset that should be licensed (e.g. the data schema, the data content, both of them) and according to national or international legislation for copyright protected work, some preliminary questions are issued and an analysis of salient legal features is conducted. A subset of seven licenses were identified and described in D5.1 as suitable candidates for a licensing methodology of Public Sector Body, see the following Table 6.10.

Table 6-10: Considered licenses

<table>
<thead>
<tr>
<th>License name</th>
<th>Restrictions</th>
<th>Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>License</td>
<td>Description</td>
<td>Yes/No</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>CC13 or OKF14</td>
<td>Grant usage for the maximum extent</td>
<td>yes</td>
</tr>
<tr>
<td>CC015</td>
<td>Grant usage for the maximum extend allowed by national legislation</td>
<td>yes</td>
</tr>
<tr>
<td>CC:BY16</td>
<td>Copyright to DB schema / items or sui generis right (SGDR) to license provider</td>
<td>yes</td>
</tr>
<tr>
<td>CC:BY:NC17</td>
<td>Data extraction and use restrictions for commercial purposes</td>
<td>no</td>
</tr>
<tr>
<td>CC:BY:ND18</td>
<td>Re-use restrictions</td>
<td>no</td>
</tr>
<tr>
<td>CC:BY:SA19</td>
<td>Constraint to license derivative results under the same license</td>
<td>no</td>
</tr>
<tr>
<td>Own license</td>
<td>It is recommended to keep principles set in CC:BY or CC0</td>
<td>no</td>
</tr>
</tbody>
</table>

Licenses for datasets should be represented through metadata descriptions, as recommended in section "Missing CKAN metadata fields for distribution" of D3.3 of the present project.

### 6.2 Realization of publication

#### 6.2.1 Initial publication of the dataset

According to Deliverable 5.1, the prepared ETL procedures are executed and the first version of a dataset is published. First, the dataset is prepared and published, according to tasks P02A05 and P02A06. Testing and validation of the results follows as described in deliverable D5.1. After the dataset is published, it is also checked that the dataset is available.

#### 6.2.2 Data Cataloguing

According to Deliverable 5.1 this task is in charge of publishing the catalogue records in data catalogues previously selected. Metadata related to dataset descriptions are filled in by means of techniques described in Section 6.1.2 and Section 6.1.3. Advertising open datasets to open data catalogs is described in deliverable D3.3.
7 Techniques for data quality management

Data quality management deals with all activities of COMSODE that consider accuracy, completeness, consistency and currency of data. The conceptual schema of such activities is shown in Figure 7.1, where the two relationships named “refers to” put in evidence the strong connection among techniques, the main focus of this deliverable, and dimensions, datasets and data set types considered in the life cycle, and tasks in which techniques may be used.

Figure 7.1: The conceptual schema of the data quality management cross cutting activity

The conceptual schema highlights that all dimensions are considered for all data sets in the requirement and collection task, while subsequent assessment and improvement tasks focus only on a subset of them, with the goal of identifying the most critical issues.

In the following we first provide in Section 7.1.1 a general introduction to the concept of data quality dimensions and their measurement, then in Section 7.2 we propose the general structure of the data quality management activities. In the last three sections we discuss the three tasks of data quality management, requirements analysis and collection in Section 7.3, quality assessment in Section 7.4 and quality improvement in Section 7.5.
7.1 Data quality dimensions and quantitative vs subjective measurement of data quality

7.1.1 Introduction to data quality dimensions

Data are normally considered to be of poor quality if typos are present or wrong values are associated with a concept instance, such as an erroneous birth date or age associated with a person. However, data quality is more than simply data accuracy. Other significant dimensions such as completeness, consistency, and currency are necessary in order to fully characterize the quality of data. In Figure 7.2 we provide some examples of these dimensions. The relation in the figure adopts the relational model of data [Atzeni & De Antonellis 1993] and describes movies, with title, director, year of production, number of remakes, and year of the last remake. The example is from [Batini & Scannapieco, 2006].

<table>
<thead>
<tr>
<th>Id</th>
<th>Title</th>
<th>Director</th>
<th>Year</th>
<th>#Remakes</th>
<th>LastRemakeYear</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Casablanca</td>
<td>Weir</td>
<td>1942</td>
<td>1</td>
<td>NULL</td>
</tr>
<tr>
<td>2</td>
<td>Dead Poets Society</td>
<td>Curtiz</td>
<td>1989</td>
<td>0</td>
<td>NULL</td>
</tr>
<tr>
<td>3</td>
<td>Rman Holiday</td>
<td>Wylder</td>
<td>1953</td>
<td>0</td>
<td>NULL</td>
</tr>
<tr>
<td>4</td>
<td>Sabrina</td>
<td>NULL</td>
<td>1964</td>
<td>0</td>
<td>1985</td>
</tr>
</tbody>
</table>

![Figure 7.2: A relation Movies with data quality problems](image)

In the figure, the cells with data quality problems are shaded. At first, only the cell corresponding to the title of movie 3 seems to be affected by a data quality problem. In fact, there is misspelling in the title, where “Rman” stands for “Roman”, thus causing an accuracy problem. Nevertheless, another accuracy problem is related to the exchange of the director between movies 1 and 2; Weir is actually the director of movie 2 and Curtiz the director of movie 1. Other data quality problems are a missing value for the director of movie 4, causing a completeness problem, and a value NULL for the LastRemakeYear of movie 1 when the #Remakes is 1, causing a currency problem. Finally, there is a consistency problem for movie 4, since the value of LastRemakeYear cannot be different from NULL, because the value of #Remakes is 0.

The above examples of dimensions concern the quality of data represented in a relational table. When the format of data changes, e.g. we use a CSV table, where keys and other constraints that can be defined in the relational model are not defined, or an RDF format, quality dimensions adapt accordingly to the format.

Consider an example in linked open data where the RDF triples are retrieved as of May 2013. The first and third triple in Figure 7.3 have high currency while the second triple does not.
represent an up-to-date result with respect to the real world at the time the data was retrieved. The object of the second triple is not updated to the current club.

<table>
<thead>
<tr>
<th></th>
<th>currentclub</th>
<th>Clube Atlético Mineiro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ronaldinho</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Antonio Cassano</td>
<td>currentclub</td>
<td>Internacional</td>
</tr>
<tr>
<td>Cristiano Ronaldo</td>
<td>currentclub</td>
<td>Real Madrid C .F.</td>
</tr>
</tbody>
</table>

Figure 7.3: The triples present some of the RDF data referring to the current clubs

Quality in linked open data presents several challenges that are not present in other data models. For instance, temporal annotations are not always available due to the autonomous information providers and to the third party applications. Further, the original information is represented by the RDF data model used in linked open data which is based on binary relationships. In case of n-ary relationship information, the binary relationship representation will lose some of the linking relations. In alternative, some of the datasets may adopt approaches for representing n-ary relationship information. However, the assessment process needs to take into consideration these challenges typical of the RDF data model.

7.1.2 Quantitative measurement

In the previous paragraph we have informally introduced terms as accuracy, completeness, consistency, and currency. Although we did not provide so far metrics to associate a measure to dimensions, it should be clear that such metrics can be defined according to various methods. E.g. we can say that the second, third and fourth tuple of the relation Movies in Figure 7.2 have completeness equal to 0.75, since one of the four fields in the tuple has a null value, representing absence of information. Such methods of measurement and related metrics are called quantitative in the following. In Figure 7.4 we show all types of dimensions that will be defined in Section 7.3 for the different types of data formats managed in the COMSODE methodology.

<table>
<thead>
<tr>
<th>Dimension/ Format</th>
<th>Metadata</th>
<th>CSV</th>
<th>Relational</th>
<th>Html</th>
<th>XML</th>
<th>RDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Completeness</td>
<td>A measure for each metadata instance - global weighted measure</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Consistency</td>
<td>Functional dependencies</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Time related</td>
<td>Currency</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Scheme dimensions</td>
<td>-</td>
<td>-</td>
<td>Boyle Codd Normal Form</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Valid</td>
<td></td>
<td>Well-formed</td>
</tr>
</tbody>
</table>

Figure 7.4: Dimensions for which quantitative metrics are provided
7.1.3 Subjective measurement

Above metrics allow us to associate a quantitative measure to quality dimensions; such measure is “objective” since it is based on mathematical formulas and does not take into account the processes that make use of data and related users. Users of data are characterized by different levels of culture and skill; further, they make use of data for different purposes. Their perception on the quality of data is often subjective, and their own evaluation of the quality may be different from the result of objective measures previously discussed. So, we have to introduce a second type of method for measuring quality dimensions based on user perception. In Attachment IV the reader can find a questionnaire for the evaluation of user perception of data quality, adapted from well-known questionnaires for quality of service, see e.g. [Zeithalm et al 1990].

7.2 Data Quality Management activities

Main activities for data quality management are shown in Figure 7.5.

1. REQUIREMENTS COLLECTION – Collect general requirements, achieve from users specific requirements and subjective dataset quality evaluation and possible causes of errors using the COMSODE questionnaire.
2. REQUIREMENTS ANALYSIS - Analyze requirements and select priority quality dimensions.
3. QUANTITATIVE ASSESSMENT - Perform quantitative assessment on selected quality dimensions, choosing suitable quality metrics and related techniques.
4. IMPROVEMENT - Choose suitable improvement activities and related techniques. Apply chosen improvement techniques.

Figure 7.5: Data Quality Management activities

We now proceed to analyze specific activities.

7.3 Requirements Collection and Analysis

Requirements correspond to the needs and goals of users on the final outcome of the data quality management process. They are of two types. The first group of general requirements has been collected and listed in Deliverable 3.2. A deeper investigation on documented techniques reported in the literature and available tools led to focus on requirements referring to a subset of quality dimensions mentioned in D3.2, more specifically to accuracy, completeness, consistency and currency dimensions.

A second group of requirements specific to the dataset is collected delivering a questionnaire to users in the Requirements collection and analysis task; the goal of this activity is to acquire from users the perception on the quality of the data set, and to collect hints on possible causes of errors in data introduced during administrative/business processes. The questionnaire is provided in Appendix 2, and corresponds to the technique QR-QUP Questionnaire to acquire user perception on the quality of data.
Questionnaires filled by users are analyzed to put in evidence most critical situations and define priorities among quality dimensions to be further analyzed in the quantitative assessment task.

To better clarify the goal of this activity in the data quality process, we assume that the questionnaire is delivered in the context of the following case study.

### 7.3.1 Requirements collection - Case study

The following box describes the case study.

| In many countries, businesses in their life cycle have to interact with several central administrations (e.g. Social security, Social Insurance) and local administrations (e.g. Municipalities) to request services. The interactions are needed for several business events such as starting a new business or closing down a business, which involves registering the business, e.g., with the Chamber of commerce; evolving a business, which includes variations in legal status, board composition and senior management, number of employees, as well as the launching of a new location. Each administration usually manages information on businesses that in several cases is common information to all administrations (e.g. the name of the business), in other cases is related to the specific function of the administration. Since every business reports independently to each administration, the copies of common information have different levels of accuracy and currency. |

We may select a sample of internal and external users and deliver them the questionnaire. *Internal users* are public administration employees that manage administrative procedures related to businesses, while *external users* are employees or delegates of businesses that interact with public administrations.

### 7.3.2 Requirement analysis

We assume that results of interviews with internal and external users can be summarized as follows.

- Internal users are frustrated by the fact that businesses contacted frequently complain about multiple letters, messages, or telephone calls. This is a sign of the presence of duplicate objects in the databases.
- Internal users involved in tax frauds do not succeed in matching businesses when they perform cross-queries on several databases. This is an indication of loose matching of records in databases.
- Final users (businesses) contacted by phone interviews are burdened by the fact that for a long time after the communication of variations, e.g., of the address (``several months'' is typical), they do not receive letters or messages from agencies at the new address. Conversely internal users receive a huge amount of messages back from addresses that correspond to unknown businesses. This in an indication of the lengthy period it takes to perform updates in the database.
From the results of interviews, we conclude that we have to focus on the following quality dimensions and metrics:

- presence of duplicate objects in single databases, classified as inaccuracy;
- delay in the registration of updates, a case of low currency

The above priority quality dimensions direct subsequent assessment and improvement activities in such a way to save effort and focus on most effective choices in the data quality management process. Anyhow, notice that this activity may request a significant effort, so that the recommendation is to consider as a complementary tool for gathering requirements.

### 7.4 Assessment

We address first an activity that is preliminary to assessment; the standardization that it is worthwhile to perform on the data set so to homogenize values to common formats. Subsequently we will consider assessment for accuracy, completeness, consistency and time related dimensions. Such dimensions are defined for data values; in the final section we will deal with schema dimensions.

#### 7.4.1 Standardization

We assume to start the activity assuming as dataset the CSV table shown in Figure 7.6.

![Figure 7.6: CSV table in input to standardization](image)

In this step we have to homogenize heterogeneous values referring to the same value items in tuples. This activity is supported by a technique that is specific for each domain:

**QA-S (Domain) Standardization of values**

In the Figure 7.7, standardization can be performed only on attributes such as Month of Birth and Day of Birth, for which heterogeneities refer to numeric vs alphanumeric values.
Figure 7.7: Output of the standardization step (changed values are in gray)

More specifically, for Month of Birth we have mixed numeric and textual values in the input data set; we may decide to standardize into numerical values, so “September” is transformed into “9” and “June” into “6”.

7.4.2 Accuracy

Accuracy refers to the extent to which entities and facts are correct, that is, the degree to which they correctly represent the real-life phenomenon. Accuracy refers in the following to data values and to the schema, while accuracy of metadata will be addressed in D5.4. Furthermore, when accuracy refers to (data) values, it can be defined for data in different formats. In the following we address accuracy of values in the different formats (Section 7.4.2.1), and we shortly address accuracy of data schemas in Section 7.4.2.2, the interested reader can find further details on accuracy of schemas in [Batini et al. 1994]. Value accuracy can be defined for a single value or for a set of values, corresponding to an entire dataset or a part of it. We deal first with accuracy of a single value.

7.4.2.1 Accuracy of values

Value accuracy can be defined for a single value or for a set of values, corresponding to an entire dataset or a part of it. We deal first with accuracy of a single value.

7.4.2.1.1 Accuracy of a single value

Accuracy of a value \( v \) is defined as the closeness between \( v \) and a value \( v' \) considered as the correct representation of the real world phenomenon that \( v \) aims to represent. As an example, if the name of a person is “John” the value \( v = "John" \) is accurate, while the value \( v1 = "Jhn" \) is inaccurate.

Since usually we do not know, and it could be very costly to know, “the correct representation of the real world phenomenon that \( v \) aims to represent”, two kinds of accuracy can be identified, namely syntactic accuracy and semantic accuracy. Syntactic accuracy is defined when the
vocabulary or reference domain of values is known (e.g. the domain of Italian first names, the domain of Czech municipalities), and is the degree to which values correctly represent the domain values of underlying vocabularies or reference domains, while semantic accuracy is defined as the degree to which data values correctly represent the real world facts.

7.4.2.1.2 Syntactic accuracy of a value in CSV/relational model

To introduce syntactic accuracy in CSV data, look at Figure 7.8 that represents in the CSV data model a group of European professionals in the area of open data.

![Figure 7.8: A CSV Table](image)

It is easy to discover that the city “Milan” has two different representations in tuples 6 and 7. We also easily assess that also the city “Prague” has different representations in tuples 1 and 3, and the country “Ital” in tuple 6 has a missing character, we arrive to this conclusion since Italy is the European country whose name is the most similar to “Ital”.

From the above example we may say that syntactic accuracy of a data value can be measured through a distance function, which quantifies “the degree to which values correctly represent the domain values of underlying vocabularies or reference domains” The accuracy measure depends on the functional definition of the distance and on the underlying vocabulary or reference domain.

To introduce the first distance function we will use, look at Figure 7.9. In this figure the value v = “Mrio” is supposed to be an Italian first name. To establish a distance function we
may compare the string “Mrío” with the lookup table of all alphabetic strings representing Italian first names. In such comparison we may apply an unnormalized edit distance. The unnormalized edit distance between two character strings v and v’ is the minimum number of insertions, deletions, and substitutions of characters needed to transform v in v’. In Figure 7.9 the edit distances between “Mrío” and some names in the reference domain is evaluated. The closest name to “Mrío” is “Mario” and the level of inaccuracy is 1, which corresponds to the distance between Mrío and Mario.

Here and in the following we aim to define metrics whose values are normalized in a domain [0.1]. This is useful to be able to compare different quality dimension measures for the same data. To do so, we can normalize the above measure evaluating the ratio between the distance and the maximum possible distance, which corresponds to the maximum number of edit operations on a value in the domain. Assuming that in the domain of names the longest name is made of six characters as in Figure 7.9, the (normalized) edit distance in our case is:

\[
\text{Syntactic accuracy edit distance (“Mrío”, Reference domain of italian names)} = 1 - \frac{1}{6} = 0.833
\]

The syntax accuracy of alphanumeric values adopting the edit distance is measured by the following technique:

Q-SAVED (Syntactic accuracy of a value measured with edit distance)

7.4.2.1.3 Syntactic accuracy of an attribute/table in CSV/relational model

When we move from single data values to entire data sets of values referring to an attribute (e.g. the First name attribute in Figure 7.8) or an entire table, the syntactic accuracy of the data set can be measured as the average accuracy of values of the attribute or of the table, leading to techniques

Q-SAAED Syntactic accuracy of values of an attribute measured with edit distance,
Q-SATED Syntactic accuracy of values of a table measured with edit distance.
Notice that in the case of Figure 7.8 we have to measure the syntactic distance of first names of professional in different countries referring to reference domains distinct for the different countries.

Another alternative and more conservative measure of syntactic accuracy is the following.

\[
\text{accuracy of a data set} = \frac{\text{(number of tuples such that all values in the tuple have edit distance = 0)}}{\text{total number of tuples in the data set}}
\]

That leads to the techniques

- Q-SAAED2: Syntactic accuracy of values of an attribute measured with edit distance
- Q-SATED2: Syntactic accuracy of values of a table measured with edit distance.

### 7.4.2.1.4 Syntactic accuracy of a value, attribute, CSV/relational table measured with Jaccard distance

Edit distance can lead to measures of syntactic accuracy that do not match with our intuition and practical situations.

Consider Figure 7.10. In this case, if we evaluate using edit distance the syntactic accuracy of the two values to be assessed, assuming as reference domain the table in the right hand side, we obtain among others the following values:

\[
\begin{align*}
\text{Syntactic accuracy}_{\text{edit distance}}^{<\text{AT&T, AT&T Corporation}>} &= 1 - \frac{12}{16} = 0.25 \\
\text{Syntactic accuracy}_{\text{edit distance}}^{<\text{IBM Corporation, AT&T Corporation}>} &= 1 - \frac{5}{12} = 0.6
\end{align*}
\]

where “IBM Corporation” is closer that “AT&T” to “AT&T Corporation”, leading to an unintuitive conclusion. We can introduce a second metric, where values are no longer seen as alphabetic strings, but else as groups of items, separated by blank characters. This different view of values leads to a new distance called **Jaccard distance**, which can be evaluated with a two steps procedure:

1. Split value \( v_i \) and values in the reference domain in a set of items, we call e.g. \( T_i \) the list of items corresponding to \( v_i \); call \( C(T_i) \) the cardinality of the set.
2. The Jaccard distance between two values $v_1$ and $v_2$ is evaluated as $1 - \frac{C(T_1 \cap T_2)}{C(T_1 \cup CT_2)}$.

In our example we have:

<table>
<thead>
<tr>
<th>Syntactic accuracy Jaccard distance</th>
<th>Value</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;AT&amp;T, AT&amp;T Corporation&gt;</td>
<td>1/2 = 0.5</td>
<td></td>
</tr>
<tr>
<td>&lt;IBM Corporation, AT&amp;T Corporation&gt;</td>
<td>1/3 = 0.33</td>
<td></td>
</tr>
</tbody>
</table>

The adoption of Jaccard distance leads to the techniques

- Q-SAVJD - Syntactic accuracy of a value measured with Jaccard distance.
- Q-SAAJD - Syntactic accuracy of values of an attribute measured with Jaccard distance.
- Q-SAAJJD - Syntactic accuracy of values of a table measured with Jaccard distance.

A third metric that takes into account not only the internal structure of the string in terms of items, but also the presence of acronyms such as “JFK Airport” instead of “John Fitzgerald Kennedy Airport”; in this case we need to add to the reference domain of values a second table of correspondences between values and their acronyms, see Figure 7.11.

![Figure 7.11: Examples of correspondences acronym-full name](image)

A final metric we discuss assumes that $v$ is written according to a set of syntactic rules defined in a language of rules LR. In this case, a metric e.g. for the syntactic accuracy of values in a table is given by the following formula

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full name</th>
</tr>
</thead>
<tbody>
<tr>
<td>JFK</td>
<td>John Fitzgerald Kennedy</td>
</tr>
<tr>
<td>PA</td>
<td>Public Administration</td>
</tr>
<tr>
<td>SS</td>
<td>Social Security</td>
</tr>
</tbody>
</table>

Syntactic accuracy of values in a table based on a language rules $= 1 - \frac{\text{number of tuples that violate at least an accuracy rule}}{\text{total number of tuples in the data set}}$

This metric leads to techniques

- Q-SAVLR Syntactic accuracy of a value based on a language of rules
- Q-SAALR Syntactic accuracy of values of an attribute based on a language of rules
- Q-SATLR Syntactic accuracy of values in a table based on a language of rules

### 7.4.2.2 Syntactic accuracy in RDF

Syntactic accuracy issues in the RDF data model mostly refer to literals incompatible with datatype range or ill formed datatype literals. For example, a property `ex:dateOfBirth` has the
range xsd:date but there are triples where ex:dateOfBirth have xsd:integer in the predicate position. In the second case, consider that the datatype associated with a literal is xsd:gYear and it is possible to identify triples where xsd:dateTime literals are used instead.

Consider for instance the triple: dbpedia:Stephen_Fry dbpedia-owl:activeYearsStartYear "1981-01-01T00:00:00+02:00"^^xsd:gYear. In this case, the DBpedia ontology datatype property activeYearsStartYear has xsd:gYear as range. Although the datatype declaration is correct, it is formatted as xsd:dateTime. The expected value is "1981"^^xsd:gYear.

Techniques that capture incorrect entities or property values can be detected by validators or by explicit definition of allowed values for a certain datatype.

The RDF data model uses the RDF/XML syntax in most cases of linked datasets. The detection of syntax errors in an RDF document can be performed through a syntax validator; this corresponds to technique:

QRDFSV1 Syntactic accuracy in RDF through syntax validator.

Figure 7.12 shows a possible validator\(^20\) that checks the validity of a document either by inserting the RDF/XML document into the text field or by entering a URI. The validation result is a message which shows a successful validation and subsequently a graphical visualization of the data model in case the document does not contain any syntactic error or alternatively it shows an erroneous validation message. Along this line, an RDF validator is used to parse the RDF document and ensure that it is syntactically valid, that is, to check whether the document is in accordance with the RDF specification.

![Figure 7.12: RDF validator](image)

In alternative to the correctness of the RDF specification, the syntactic accuracy can be assessed also by checking the valid usage of the underlying vocabularies such as the detection of members of deprecated classes or properties; detection of ill-typed literals w.r.t. to their respective datatype. Possible causes of such errors are (i) malformed typos or (ii) members of incompatible datatypes.

QA-RDFS\(\text{SV2}\) - Syntactic accuracy in RDF - Detection of members of deprecated classes or properties.
Along this line, other types of validators exist such as the RDF Alerts\textsuperscript{21}, which is more general purpose and checks not only for syntax errors but also for undefined classes or properties, a typical use of core vocabularies, datatype errors and more. This leads to the techniques

QA-RDFA1 - Syntactic accuracy in RDF through RDF Alert - Undefined classes or properties
QA-RDFA2 - Syntactic accuracy in RDF through RDF Alert – Datatype errors

Table 7-1 shows a snippet of the validation result of the following soccer player: http://live.dbpedia.org/data/Ronaldinho. We can distinguish three types of alerts: note, warning and error, where each of them comes with a description of the problem. The types of syntactic accuracy shown in the table are: a. parsing errors (row 1), b. undefined classes and properties (rows 2 and 3), and c. erroneous definition of datatype literals (rows 4,5 and 6).

<table>
<thead>
<tr>
<th>row</th>
<th>alerts</th>
<th>results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>note</td>
<td>error parsing <a href="http://purl.org/dc/elements/1.1/description">http://purl.org/dc/elements/1.1/description</a> - org.semanticweb.yars2.rdfxml.RDFXMLParserBase$RDFXMLParseException: FATAL ERROR: Dangling text 'Properties in the /elements/1.1/ namespace' found outside of enclosing property tags. Line 17 column 62</td>
</tr>
<tr>
<td>2</td>
<td>warning</td>
<td>could not find a definition for Class <a href="http://dbpedia.org/class/yago/1999FIFAConfederationsCupPlayers">http://dbpedia.org/class/yago/1999FIFAConfederationsCupPlayers</a>... term does not dereference to an RDF vocabulary description?</td>
</tr>
<tr>
<td>3</td>
<td>warning</td>
<td>could not find a definition for Property <a href="http://dbpedia.org/property/years">http://dbpedia.org/property/years</a>... term does not dereference to an RDF vocabulary description?</td>
</tr>
<tr>
<td>4</td>
<td>error</td>
<td>plain literal &quot;134061969&quot;@de incompatible with range <a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a> of property <a href="http://dbpedia.org/ontology/individualisedPnd">http://dbpedia.org/ontology/individualisedPnd</a></td>
</tr>
<tr>
<td>5</td>
<td>error</td>
<td>literal &quot;Ronaldinho&quot;@en has language tag -- incompatible with range <a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a> of property <a href="http://dbpedia.org/ontology/commonName">http://dbpedia.org/ontology/commonName</a></td>
</tr>
<tr>
<td>6</td>
<td>error</td>
<td>literal &quot;Attacking midfielder / Forward&quot;@en has language tag -- incompatible with range <a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a> of property <a href="http://dbpedia.org/ontology/position">http://dbpedia.org/ontology/position</a></td>
</tr>
</tbody>
</table>

Other quality issues may be related to the accuracy e.g. of the date format. For instance: use of regular expressions to identify date information in date, dateTime, gYearMonth, gYear, gMonthDay, gDay and gMonth formats. This leads to technique

QA-RDFADate – Syntactic accuracy of Date
### 7.4.2.3 Semantic accuracy

Consider Figure 7.13 and assume that for some reason the true name corresponding to “Mrio” is “Maria”. Looking at the reference domain, there is no knowledge source that can help us to fix such an error, since the closests name to “Mrio” is “Mario”, not “Maria”. In general, discovering the true value may be very costly; a strategy we can afford is to get new knowledge and try to deduce, with a formal reasoning, which is the value in the reference domain that is consequent to such knowledge. In our case, we could know from some other knowledge source that “Mario” is a female, so we can decide to choose the female first name that is closer to “Mrio”.

![Figure 7.13: Example of semantic accuracy that does not fit with syntactic accuracy](image)

We do not address further semantic accuracy, so we do not associate techniques to this dimension.

### 7.4.3 Completeness

Completeness is defined as the degree to which all the required information is present in a dataset. Completeness is defined both for metadata and for data.

#### 7.4.3.1 Metadata completeness

In the ODN platform the metadata completeness can be measured by the following technique: Q-MC detects the percentage of the recommended metadata for the dataset description for which a value is specified. Metadata are discussed and described in Section 6.1.2.

#### 7.4.3.2 Completeness for CSV/relational values

Intuitively, the completeness of a table characterizes the extent to which the table represents the corresponding real world. Completeness in the relational model can be characterized with respect to:

(i) the presence/absence and meaning of null values, and
(ii) the validity of one of the two assumptions called open world assumption and closed world assumption. We now introduce the two issues separately.

In a model with null values, the presence of a null value has the general meaning of a missing value, e.g., a value that exists in the real world but for some reason is not available. In order to characterize completeness, it is important to understand why the value is missing. Indeed, a value can be missing either because it exists but is unknown, or because it does not exist at all, or because it may exist but it is not actually known whether it exists or not. We describe the three types of null values by means of an example.

![Figure 7.14: A Person relation with different null value meanings for the E-mail attribute](image)

Let us consider a Person relation with the attributes Name, Last name, BirthDate, an Email. The relation is shown in Figure 7.14. For the tuples with Id equal to 2, 3, and 4, the Email value is null. Let us suppose that the person represented by tuple 2 has no e-mail: no incompleteness case occurs. If the person represented by tuple 3 has an e-mail, but its value is not known then tuple 3 presents incompleteness. Finally, if it is not known whether the person represented by tuple 4 has an e-mail or not, incompleteness may or may not be the case. In the following, we will consider only the second type of Null value, so we assume that every time we find a null in a relation its meaning is “not known”.

In logical models for databases, such as the relational model, there are two different assumptions on the completeness of data represented in a relation instance r. The closed world assumption (CWA) states that only the values actually present in a relational table and no other values represent facts of the real world. In the open world assumption (OWA) we can state neither the truth nor the falsity of facts not represented in the tuples of r. We assume in the following both assumptions.

### 7.4.3.3 Completeness in CSV/relational models

We introduce now four techniques for completeness in CSV/relational data, defined according to the validity of CWA/OWA, and to the part of the table they apply. All of them make reference to the table T in Figure 7.15.
The four metrics are defined in Figure 7.16.

**Object completeness** - in this case we assume OWA; the table T represents m objects of a universe U, and we assume to know from some knowledge source that the number of objects of the universe is k. Object completeness is defined as “number m of tuples in the table / number k of objects of the real world that should be represented in the table”.

The other three completeness definitions make reference to CWA, so the number of objects is assumed k = m. See formulas in Figure 7.16.

The above metrics lead to the following techniques:
- Q-CO Object completeness
- Q-CT Tuple completeness
- Q-AC Attribute completeness
- Q-TC Table completeness

To make a comprehensive example, look at the Student-Exam Table of Figure 7.17. In Figure 7.18 we calculate values of metrics for object, tuple, attribute and table completeness. We make the assumption that students in the observed universe are 15.
7.4.4 Consistency

Consistency captures the violation of semantic rules defined over a dataset. It checks if a value \( v \) in input to the assessment is logically compatible with a set of semantic rules. Such semantic rules depend in general on the model.

7.4.4.1 Consistency in the relational model

In the relational model semantic rules correspond to integrity constraints. A common case in relational tables is the presence of functional dependencies between pairs of attributes or sets of attributes. A functional dependency between two attributes or groups of attributes A and B, written A \( \rightarrow \) B means that for every value or set of values in A, a unique value or set of values in B corresponds in the table. Looking at the table in Figure 7.6, we may define the following functional relationship:

\[
\text{Name of Toponym, Number, City} \rightarrow \text{Zip Code}.
\]

When functional dependencies are known, their consistency can be measured on a relational table using the technique:
7.4.4.2 Consistency in the RDF model

A straightforward way to check for consistency in RDF data model is to load the knowledge base into a reasoner and check whether it is consistent. However, for certain knowledge bases (e.g. very large or inherently inconsistent ones) this approach is not feasible. Moreover, most OWL reasoners specialize in the OWL 2 DL sublanguages as they are internally based on description logics. However, it should be noted that linked data does not necessarily conform to OWL DL and, therefore, those reasoners cannot directly be applied. Consider the triple: dbpedia:Drei_Flisse_Stadion dbpprop:seating Capacity "20"^^xsd:integer.

In Wikipedia, the seating capacity for this stadium has the value “20.000”, but in DBpedia the value displayed is only 20.

Some of the important metrics identified in the literature are:

- Detection of the re-definition by third parties of external classes/properties (ontology hijacking) such that reasoning over data using those external terms is not affected. The ontology hijacking detects the redefinition by analyzing defined classes or properties in datasets and looks for the same definition in its respective vocabulary;
- Detection of misuse of undefined classes and properties in a dataset by checking for their definitions in the respective referred vocabulary;
- Detection of misuse of owl:DatatypeProperty or owl:ObjectProperty through the ontology maintainer;
- Detection of misplaced classes or properties using entailment rules that indicate the position of a term in a triple;
- Detection of use of entities as members of disjoint classes.
- Detection of inconsistent values by the generation of a particular set of business rules for all properties in a dataset (Q-CN_1) through SPARQL queries.

A SPARQL rules constraint defines a condition that, if true, should be brought to the user's attention. The SPARQL ASK statement, which returns a boolean true or false value, is ideal for this. Consider for example, we're going to put a constraint on materials purchases so that SPIN22 alerts us to unpaid materials purchases that are more than 90 days old. In the following we report an example of SPARQL ASK statement:

```
# an invoice with no paidDate is > 90 days old
ASK WHERE {
  ?this :invoiceDate ?invoiceDate OPTIONAL {?this purchases:paidDate ?paidDate}
  FILTER (!bound(?paidDate)) && (spif:duration("d", ?invoiceDate, afn:now()) > 90)) .
}
```
Instead of an ASK query, a SPARQL rule using a CONSTRUCT statement can be used to create triples about the violated constraint. These triples can be used in applications built using the TopBraid platform.

All of the above constraints are implemented by the technique QARDFCON Consistency check for an RDF data set.

7.4.5 Currency

Time-Related Quality Dimensions capture important aspects of data regarding changes and updates in time (i.e., in the linked data model the dynamic nature of linked data). Considering the RDF model, important aspects of time-related quality dimensions are data freshness over time (currency) and data freshness over time for a specific task (timeliness). Related techniques are:

- QA-RDFTA Currency as Age of the document - the currency of a document is measured as the age of the document, that is computed as the difference between the current time (the observation time) and the time when the document was last modified.
- QA-RDFTC: Currency of a document - the currency of a triple/set of triples is measured as the average age of the documents describing the entities occurring in the triple.
- QA-RDFTC: Timeliness – Timeliness refers to the property of data of being produced on time for their usage. Timeliness is measured based on two components that are currency and volatility. For currency see case (a) above. Volatility is the time-lapse the data remain valid, which is measured by the frequency of change of the value for a property.

7.4.6 Tradeoffs between value dimensions

Data quality dimensions are not independent, i.e., correlations exist between them. If one dimension is considered more important than the others for a specific application, then the choice of favoring it may imply negative consequences for the other ones. In this section, we provide some examples of possible trade-offs. First, we briefly mention the related issue of intrinsic vs contextual dimensions. Coming back to currency and timeliness defined above, the difference between currency and timeliness is that currency is inherently an **intrinsic** dimensions of data, while currency is called **contextual**, namely its value depends on the specific task or decision we have to perform. For a thorough discussion on intrinsic vs contextual dimensions and metrics and the related fundamental issue of data quality as “fitness for use” see [Batini Scannapieco 2014].

As a first example, trade-offs may need to be made between timeliness and any one of the three dimensions: accuracy, completeness, and consistency. Indeed, having accurate (or complete or consistent) data may need checks and activities that require time, and thus timeliness is negatively affected. Conversely, having timely data may cause lower accuracy (or completeness/consistency). A typical situation in which timeliness can be preferred to accuracy, completeness, or consistency is given by most Web applications: as the time constraints (hence, timeliness) are often very stringent for Web data, it is possible that such data are deficient with respect to other quality dimensions. For instance, a list of courses published on a university Web site must be timely though some fields specifying courses could be missing in
case they are not yet assigned. Conversely, when considering an administrative application, accuracy, consistency, and completeness requirements are more stringent than timeliness, and therefore delays affecting timeliness are mostly admitted.

Another significant case of trade-off is between consistency and completeness [Ballou Pazer 2003]. Here the question is whether it is better to have less but consistent data, i.e., poor completeness, or to have more but inconsistent data. This choice is domain specific. As an example, statistical data analysis typically requires a significant and representative amount of data in order to perform the analysis; in this case, the approach is to favor completeness, tolerating inconsistencies, or adopting techniques to solve them. Conversely, when considering the publishing of a list of partial and total grades obtained by students as the result of an exam, it is more important to have a list of consistency checked grades than a complete one, possibly deferring the publication of the complete list.

7.5 Schema dimensions

Datasets are characterized, besides quality dimensions of values, by quality dimensions defined on the schema, which, consequently hold for every dataset that conforms to the schema. In the following we examine schema dimensions in the relational data model and in the XML data model.

7.5.1 Relational model

We have first to define the concept of key in a relational table. A key is a set of attributes KS such that if in any pair of tuples t1 and t2 values of attributes in KS are the same, then also values of other attributes are the same. In other words, values in the key uniquely identify tuples in a table. E.g. in the relation schema European professionals of Figure 7.8 the unique set of attributes having the property of key is the attribute Tuple#. In the specific table of values represented in Figure 7.8 other groups of attributes are characterized by the key property, e.g. the group First Name, Last Name, Year of Birth. In order for a group of attributes to be a key, the uniqueness property must hold for all tables that may correspond to the relation schema.

A relational schema is said to be in Boyce Codd normal form if all functional dependencies defined on the relational schema have as left hand side a key. The property of Boyce Codd Normal form is checked by the technique:

QA-SBCNF Checks if a relational schema is in Boyce Codd Normal Form.

7.5.2 XML model

XML is a semistructured language where users can choose an arbitrary complex set of tags. Two are the possible quality assessment techniques that can be applied to an XML document. The first technique is related to the syntactic correctness of the XML document as foreseen in the W3C standard24.
Thus well formed in relation to an XML document means that it has no syntax, spelling, punctuation, grammar errors, etc. in its markup. For example an XML document where there is the tag <foo> but no closing tag </foo> is not well formed. The related technique is QA-XMLSC Checks if an XML document is syntactically correct.

A second and tighter technique validates XML documents against a DTD (document type definition) or an XSchema. If the validation process (typically realized by means of software tools) is passed, the XML document is well formed and valid. Consider the following XML document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
  <to>Miros</to>
  <from>Andrea </from>
  <heading>Reminder</heading>
  <body>Don't forget the weekly call</body>
</note>
```

Let note.dtd be described as follow:

```xml
<!DOCTYPE note [ 
  <!ELEMENT note (to,from,heading,body)> 
  <!ELEMENT to (#PCDATA)> 
  <!ELEMENT from (#PCDATA)> 
  <!ELEMENT body (#PCDATA)> 
]
```

The above XML document is well formed because no syntax errors are found, but it not valid because the tag <heading> is not foreseen as child of note tag. As shown in the example, it is important to underline that if a document is valid it is also well formed, while a well formed document is not necessarily valid. The corresponding technique is QA-XMLW Checks if an XML document is well formed.

### 7.6 Data quality improvement

In this section we consider as a motivating example the CSV dataset data represented in Figure 7.19, which is an extension of the dataset shown in Figure 7.8. At this stage, we only know that the data set refers to a group of professionals in the area of open data living in different countries. It is evident from the data set that data refer to similar individuals and properties of individuals, that some values are missing and some values are probably incorrect. So, we perceive that we need from one side to improve the accuracy and completeness of records in the data set, from the other side we need to find groups of records referring to the same entity of the real world, and fuse them, activity that is called deduplication.
The rest of the section first introduces in Section 7.6.1 several strategies for data quality improvement, while Section 7.6.2 describes activities for the improvement phase. Section 7.6.3 provides a set of techniques, described by means of the case study in Figure 7.19.

<table>
<thead>
<tr>
<th>Tuple</th>
<th>F.Name</th>
<th>L.Name</th>
<th>Y.OBIRTH</th>
<th>M.Birth</th>
<th>D.Birth</th>
<th>Toponym</th>
<th>Name.T</th>
<th>postn.T</th>
<th>Number.T</th>
<th>City</th>
<th>Zip.Code</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Miroslav</td>
<td>Konecny</td>
<td>1976</td>
<td>7</td>
<td>10</td>
<td>Street</td>
<td>St John</td>
<td>49</td>
<td>Prague</td>
<td>412770</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>2</td>
<td>Martin</td>
<td>Necasely</td>
<td>1975</td>
<td>7</td>
<td>8</td>
<td>Sq</td>
<td>Vienna</td>
<td>null</td>
<td>Bratislava</td>
<td>101278</td>
<td>Slovakia</td>
<td>Slovakia</td>
</tr>
<tr>
<td>3</td>
<td>Miroslav</td>
<td>Konecny</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>Str</td>
<td>Széchenyi</td>
<td>49</td>
<td>Prague</td>
<td>412778</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>4</td>
<td>Ciro</td>
<td>Bili</td>
<td>1949</td>
<td>June</td>
<td>7</td>
<td>Street</td>
<td>Desse</td>
<td>15</td>
<td>Roma</td>
<td>00198</td>
<td>Italia</td>
<td>Italia</td>
</tr>
<tr>
<td>5</td>
<td>Miroslav</td>
<td>Konecny</td>
<td>1976</td>
<td>7</td>
<td>null</td>
<td>Sq</td>
<td>Budapest</td>
<td>23</td>
<td>Wien</td>
<td>null</td>
<td>Austria</td>
<td>Austria</td>
</tr>
<tr>
<td>6</td>
<td>Anita</td>
<td>Rula</td>
<td>1962</td>
<td>September</td>
<td>7</td>
<td>Via</td>
<td>Seto</td>
<td>null</td>
<td>Milan</td>
<td>20...</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>7</td>
<td>Anita</td>
<td>Rula</td>
<td>1962</td>
<td>9</td>
<td>7</td>
<td>Via</td>
<td>Seto</td>
<td>null</td>
<td>Milan</td>
<td>2033</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>8</td>
<td>Anna</td>
<td>Ria</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>Via</td>
<td>Sarca</td>
<td>336</td>
<td>Milan</td>
<td>null</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>9</td>
<td>Carlo</td>
<td>Babil</td>
<td>1949</td>
<td>6</td>
<td>7</td>
<td>V.</td>
<td>Beato-Angelo</td>
<td>23</td>
<td>Milan</td>
<td>2033</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>10</td>
<td>Carlo</td>
<td>Babil</td>
<td>1949</td>
<td>June</td>
<td>7</td>
<td>Av.</td>
<td>Charles</td>
<td>null</td>
<td>Prague</td>
<td>412733</td>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>

Figure 7.19: A CSV dataset representing European professionals

7.6.1 Generalities on Data quality improvement

The goal of the improvement phase is to clean data in such a way to increase their quality w.r.t to values assessed in the previous phase. Since quality is multidimensional, specific activities are needed for the different qualities.

In the improvement phase we can adopt two general strategies, namely data-driven and process-driven. Data-driven strategies improve the quality of data by directly modifying the value of data through the comparison with other data considered of good quality. For example, obsolete data values are updated by refreshing a database characterized by higher currency.

Process-driven strategies improve quality by redesigning the processes that create or modify data. As an example, a process can be redesigned by including an activity that controls the format of data before storage. Two main techniques characterize process-driven strategies:

— **Process control** inserts checks and control procedures in the data production process when: (1) new data are created, (2) data sets are updated, or (3) new data sets are accessed by the process. In this way, a reactive strategy is applied to data modification events, thus avoiding data degradation and error propagation.

— **Process redesign** redesigns processes in order to remove the causes of poor quality, and introduces new activities that produce data of higher quality. If process redesign is radical, this technique is referred to as **business process reengineering** [Hammer and Champy 2001].

We do not go further in considering process driven strategies, which will be discussed in D5.4.
Data-driven strategies apply a variety of techniques to improve data quality, among them the most popular are:

1. **Improvement of specific quality dimensions.** More specifically:
   a. for accuracy - comparison of values with a reference domain. This is the kind of technique we have considered diffusely for syntactic accuracy in the quality assessment phase.
   b. for completeness, completion of incomplete data with specific techniques that exploit knowledge on data (see later).
   c. for consistency – identification of the correct data by exploitation of integrity constraints, functional dependencies or derived data (see later).

2. **Improvement of the global quality of the data set or of specific records**, that can be performed for all dimensions considered in the assessment phase. More specifically:
   a. *acquisition of new data*, which improves data by acquiring higher-quality data to replace the values that raise quality problems. This strategy can be used for all dimensions examined in the assessment phase.
   b. *record linkage* (also called object identification), which compares datasets with dirty values with a certified or higher quality source, identifying tuples/records in the two data sets that might refer to the same real-world object, and cleansing dirty values with corresponding higher quality values.
   c. *source trustworthiness*, which selects data sources on the basis of the quality of their data.
   d. *error localization and correction*, which identifies and eliminates data quality errors by detecting the values that do not satisfy a given set of rules, called *edits* in methodologies for statistical censuses. These techniques are mainly studied in the statistical domain. Compared to elementary data, aggregate statistical data, such as average, sum, max, and so forth are less sensitive to possibly erroneous probabilistic localization and correction of values. Techniques for error localization and correction have been proposed for inconsistencies, incomplete data, and outliers, namely values that significantly different from all other values in a dataset for elementary data and statistical data, see [Dasu and Johnson 2003]; [Batini and Scannapieco 2006].

3. **Deduplication**, which, corresponds to grouping of records of the dataset referring to the same entity in the real world.

Several techniques typical of data- and process- driven strategies are qualitatively compared in [Redman 1996] by discussing the improvement that each technique can achieve along different quality dimensions and the implementation cost of each technique. This comparison is performed from both a short-term and a long-term perspective. The comparison focuses on: (1) acquisition of new data, (2) record linkage, (3) error localization and correction, (4) process control, and (5) process redesign techniques. In general, in the long term, process-driven techniques are found to outperform data-driven techniques, since they eliminate the root causes of quality problems. However, from a short term perspective, process redesign can be extremely expensive [Redman 1996] [English 1999]. On the contrary, data-driven strategies are reported to be cost efficient in the short term, but expensive in the long term. They are suitable for one-time application and, thus, they are recommended for static data.
Several activities and related techniques are available in COMSODE to perform quality improvement. In Section 7.4.2 we will describe such activities and techniques.

7.6.2 A set of activities for the data quality improvement task

In the following we introduce activities related to:

1. **Quality improvement of single values or entire tuples in the table.**
   This activity involves all quality dimensions considered in the methodology, namely
   - syntactic accuracy
   - completeness,
   - currency and
   - consistency.

2. **Deduplication**, that has in input a table and produces in output a new table where tuples are subdivided in three groups:
   - pairs (or groups) of matching (M) tuples, namely tuples that correspond to the same entity in the real world;
   - pairs of non-matching (NM) tuples that match with distinct entities in the real world.
   - possible pairs (or groups) of tuples, for which we could not achieve a definite conclusion, and more investigation (and cost) is needed to assign them to matching or not matching.

In the following section we will provide more detail for the two activities. For a detailed description and comparison of methodologies for data quality assessment and improvement see [Batini et al 2009].

7.6.3 A detailed case study

We assume the case study is based on the CSV data set of Figure 7.19.

7.6.3.1 Quality improvement of accuracy

All values that do not exactly match with values in reference domains have to be corrected. To do so, we have to choose suitable reference data bases. E.g. as to the value “Saint Jon”, we may use either the service google Refine or else we have to access to the table of street names of Prague, made available from the reference database for addresses RUIAN – register of territorial identification, addresses and real estates.
Accuracy improvement is performed substituting the incorrect value with the one at lower distance in the reference table, where the distance can be evaluated after a previous inspection of values and their characteristics with the most suitable assessment technique among those ones described in Section 7.5. So, improvement techniques for accuracy are in one to one correspondence with assessment techniques described in Section 7.5. They are not listed here for brevity. The result of the accuracy improvement is shown in Figure 7.20, where we have corrected in tuple 1 “Prage” to “Prague”, in tuple 3 “Saint Jon” to “Saint Jon”, in tuple 6 “Mian” to “Milan” and “Ital” to “Italy”, in tuple 7 “Seso” to “Sesto”, and in tuple 11 “Slov” to “Slovakia”.

### 7.6.3.2 Quality improvement of completeness

While for improvement of accuracy we may adopt standard procedures described in Section 7.5, for completeness the context is much more complex, since we do not have an item to start from, having only a “null” value available. Luckily we have a context in which the null value appears. So, the most intuitive procedure is to perform a new acquisition of incomplete data; since this is usually a very costly activity, we can adopt several heuristics that are usually context dependent.
Figure 7.21: Output of the completeness improvement step

Assuming that in the table represented in Figure 7.21 the functional dependency

Name of Toponym, Number, City → Zip Code

holds, we may eliminate null values referring to Zip codes, under the assumption that Name of Toponym, Number, City are specified, by simply substituting them with the result of a query on a data source that produces in output the specified value. A similar approach can be followed for the substitution of null values referring to countries. The technique based on functional dependencies is

QI-COMFD Quality improvement of completeness based on functional dependencies

Data sources representing correspondences among addresses, zip codes and cities are available for many countries, so we assume to have access to them, leading to the table shown in Figure 7.22.

Other cases that are easily solved in completeness improvement are derived data. We say that an attribute A is an attribute derived from the set of attributes A1, A2, .. An when a mathematical formula exists that evaluates values in A from values in A1, A2, .. An. E.g. the Net value of a payment is the difference between the Gross value and Taxes. It is clear that if a null corresponds to Net value, we may calculate it trivially evaluating the formula. The corresponding technique is

QI-COMDD Quality improvement of completeness based on derived data

An exhaustive analysis of completeness improvement techniques, especially in statistical databases, can be found in [Dasu Johnson 2003].
7.6.3.3 Quality improvement of consistency

Functional relationships on derived values and other intra relational integrity constraints can be exploited, besides completeness, also for consistency improvement. We consider the data set of Figure 7.22; using the same functional dependency defined before for completeness, we can assess the consistency of Zip codes, coming to the conclusion that the Zip code in tuple 4 is wrong and has to be substituted with “00199”, see Figure 7.22.

![Table 7.22: Consistency check (changed values are underlined)](image)

The corresponding technique is QI-CONFD Quality improvement of completeness based on functional dependencies.

7.6.3.4 QI-D Quality improvement through deduplication

Deduplication: Once we have performed quality assessment and quality improvement, we are at a mature stage to address deduplication, which corresponds to the identification of all pairs or groups of tuples corresponding to the same real world object. In principle, any two pairs of tuples have to be compared for this activity. In our case study we have 11 tuples, so we should perform $11 \times 11 = 121$ comparisons. In real life datasets made of thousands and even millions of items, the number of comparisons grows quadratically ($n \times n$) with the number of items. So, there is no possibility to perform this activity manually. In this section we simplify the theory underlying deduplication, making use of examples to highlight more relevant concepts. A complete treatment of deduplication may be found in [Batini Scannapieco 2006] and [Batini Scannapieco 2014].

When comparing two tuples with the goal of deciding if they correspond to the same object of the real world, we have to consider that, although in previous activities we have eliminated errors and incompleteness, we cannot be sure that all values are “true”. So, in comparing tuples we have to adopt distance functions that coincide with those ones introduced for assessment of accuracy of single values.
To avoid $n^2$ calculations of distances, we can adopt some *blocking strategy* that selects a subset of pairs of tuples on the basis of common properties. E.g. in Figure 7.23 we can consider only groups of tuples referring to individuals of the same country.

Once decided the blocking strategy, we have to choose the distance formula for all pairs of tuples in the group fixed above. At this point we have to establish two thresholds $T_m$ and $T_{nm}$ such that we can to decide for each pair whether it corresponds to:

1. a matched pair, namely a pair for which we have enough confidence that they refer to the same object of the real world. This is the case in which the distance is $< T_m$. In Figure 7.24 we mark with a symbol $M_i$ such pairs.
2. a non-matched pair, namely a pair for which the distance is too high, so that the conclusion is that the pair of tuples corresponds to different objects of the real world. This is the case in which the distance is $> T_{nm}$. In Figure 7.25 we mark with $N_{Mi}$ tuples of this kind.
3. a pair for which we do not arrive to a definitive conclusion, so that we mark it with $P_{Mi}$, possible match. For these pairs we need some supplementary investigation, which has its cost.

The sets of matching, non-matching and possible matching pairs depend on the two thresholds $T_m$ and $T_{nm}$ we choose to discriminate among pairs. If they are close the one to the other, the number of $P_{Mi}$s will be low, but we risk to consider as $M$ pairs of tuples that do not represent the same object, such tuples are called *false positives* (symmetrical for $N_{Mi}$, resulting in *false negatives*). Such methods are called *error based*, to mean that there is an inherent error in any possible choice. It has been demonstrated that higher errors are made for large datasets by humans, so we can accept a statistical error in exchange of a low cost. The corresponding technique is

We can assume that with a certain choice of thresholds the resulting groups of $M_i$s, $N_{Mi}$s, and $P_{Mi}$s are as in Figure 7.23.
Notice that the unique pair of tuples that is considered PM is the pair (t4,t9). This corresponds to a distance function that assigns higher importance to First Names, Last Names and Dates of Birth, and lower importance to other attributes. In fact considering tuples 4 and 9, First names and Last names differ both in one character, the Dates of birth coincide while Addresses are completely different, and the only similarity is that they are both in Italy. Due to this contrasting context, we may perform a currency assessment on the two tuples.

7.6.3.5 Quality improvement of currency

In order to check if one of both tuples has low currency, we have to investigate in this case the production process of data, and look at the log of update transactions. This corresponds to the technique:

QI-LA Log analysis

The analysis of the log leads to the following figures:

- last update (t4, address_data) = 03/06/1993
- last update (t9, address_data) = 13/06/2013

If we assume to know that the average time between two updates of the same tuple is about 7 years, we can conclude that there is a high chance that address data of tuple t4 are out of date. We can perform an analysis of the historical log, and verify that a new update has been done on the tuple referring to address data, but it has incorrectly been associated to another tuple, due probably to an error in inputting the key; as a consequence, we correct out of date data, as shown in Figure 7.24.

![Figure 7.24: Currency improvement (modified values are underlined)](image)

In Figure 7.25 we see the final table, where we have identified four matching tuples and three non-matching tuples, and we have completely eliminated possible matching tuples.
With this last activity we have concluded the analysis of the methodology for data quality management.

8 References


[Archer 2013] Study on persistent uris with identification of best practices and recommendations on the topic for the member states and the European Commission


[Berners-Lee 1998] Cool URIs don't change.


[Davidson 2009] Designing uri sets for the uk public sector


[W3C 2013] best practices uri construction


Attachment IV - Questionnaire for user perception of the quality of data

**Instructions for filling the Questionnaire**

The aim of the questionnaire is to evaluate quickly and simply the perception that users of a dataset of the quality of data in the data set. Data quality is a multidimensional concept, meaning that cannot be expressed in terms of a unique definition, better in terms of several dimensions. In the following we will consider four dimensions, a. accuracy, b. completeness, c. consistency and d. relevance. For each one of the four quality dimensions we want to measure the user perception on the level of the quality dimension, where the level may be:

1. the *minimal* expected quality level that should be achieved by data according to user perception
2. the *maximal* expected quality level that should be achieved by data according to user perception
3. the *actual* quality level of data according to user perception

Notice that minimal and maximal level should realistically consider the organizational context where data are used, and the resources available for their management. The three above levels should be expressed in a scale of 7 values where 1 corresponds to value “very low” and 7 to value “very high”, while 4 corresponds to the average or “just acceptable”. It is also possible for the user to include a value “don’t know”, but such choice would be minimized.

Please follow the simple instructions below:

- Do not omit evaluations and minimize the [Don't know].
- Provide only one value for each item
- Do not think too much, so that your instinctive evaluation is captured.
- Notice that there are not right or wrong evaluation: we want to have a picture of your
overall evaluation of the your level of satisfaction on the information you use in your everyday administrative activity.

<table>
<thead>
<tr>
<th>SQ-A – Accuracy of information (data, documents, images, etc.) provided by the information system of the organization. Accuracy is the characteristic of data to &quot;say the truth&quot; on phenomena they describe.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SQ-COM – Completeness of information (data, documents, images, etc.) provided by the information system of the organization. Completeness means the characteristic of information to say &quot;everything is needed&quot; for the phenomenon they describe.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SQ-CON – Consistency of information (data, documents, images, etc.) provided by the information system of the organization. Consistency is the characteristic of information to be mutually coherent providing a knowledge that is stable, clear and not contradictory.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Q-R - Relevance of information (data, documents, images, etc.) provided by the information system of the organization. Relevance means the degree of pertinence of information for your activity.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>NSQ-CUR - Currency of information (data, documents, images, etc.) in the dataset. Currency means the fact that data are up-to-date and describe the most recent version of the reality they represent.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>NSQ- CA - The following answer is open, you are asked to comment based on your perception and experience on the causes of low quality of data. Think to situations in which you have found data manifestly wrong, incomplete etc. and try to remember or reconstruct on the reasons that lead to such low quality.</th>
</tr>
</thead>
</table>
The instructions to install the ODN platform are available at https://grips.semantic-web.at/display/UDDOC/Introduction

http://www.schema.org
http://www.foaf-project.org
http://www.w3.org/TR/vocab-dcat/
http://www.w3.org/TR/void/
http://www.w3.org/RDF/
http://protege.stanford.edu/
http://protege.stanford.edu/
http://arx.deidentifier.org/overview/
http://cs.utdallas.edu/dspl/cgi-bin/toolbox/index.php
https://github.com/sunitparekh/data-anonymization
http://creativecommons.org/
http://opendatacommons.org/
http://creativecommons.org/publicdomain/zero/1.0/legalcode
http://creativecommons.org/licenses/by/4.0/legalcode
http://creativecommons.org/licenses/by-nc/4.0/legalcode
http://creativecommons.org/licenses/by-nc-nd/4.0/legalcode
http://creativecommons.org/licenses/by-sa/4.0/legalcode
http://www.w3.org/RDF/Validator/
http://swse.deri.org/RDFAlerts/
http://spinrdf.org
See spinrdf.org for more information
http://www.w3.org/TR/2006/REC-xml11-20060816/