META-BASE: a Novel Architecture for Large-Scale Genomic Metadata Integration

Anna Bernasconi, Arif Canakoglu, Marco Masseroli and Stefano Ceri

Abstract—The integration of genomic metadata is, at the same time, an important, difficult, and well-recognized challenge. It is important because a wealth of public data repositories is available to drive biological and clinical research; combining information from various heterogeneous and widely dispersed sources is paramount to a number of biological discoveries. It is difficult because the domain is complex and there is no agreement among the various metadata definitions, which refer to different vocabularies and ontologies. It is well-recognized in the bioinformatics community because, in the common practice, repositories are accessed one-by-one, learning their specific metadata definitions as result of long and tedious efforts, and such practice is error-prone.
In this paper, we describe META-BASE, an architecture for integrating metadata extracted from a variety of genomic data sources, based upon a structured transformation process. We present a variety of innovative techniques for data extraction, cleaning, normalization and enrichment. The result is a repository that already integrates several important sources, and a general, open and extensible pipeline that can easily incorporate any number of new data sources.

Index Terms—Data Integration, Genomic Datasets, Metadata Management, Open Data, Rule-Based Languages, Bioinformatics.

1 Introduction

Genomic research is showing a variety of initiatives for the production of high-value biological and clinical datasets, stored in open repositories and available to the research community for secondary research use. Some examples include the Encyclopedia of DNA Elements (ENCODE [36]), The Cancer Genome Atlas (TCGA [43]) and its successor Genomic Data Commons (GDC [14]), Roadmap Epigenomics Project (REP [18]), 1000 Genomes [35], GTEx [19], and many others. Metadata is an essential ingredient of genomic repositories; it describes the experimental conditions, the cell lines or tissues, the donors with their demography, phenotypes, and treatments, and the process of extraction of stored genomic signals with the used technological devices. By inspecting metadata, it is possible to locate the datasets that better fit for formulating queries over the genome; these in turn can answer important questions in modern biology and precision medicine.

Unfortunately, while we observe a good convergence in the definition of data formats and protocols for genomic information, no agreement for a common metadata format has been reached so far: metadata of distinct repositories often disagree on their entities, attributes and values, and have no associated conceptual representations. In earlier work, we developed a conceptual approach to metadata integration and presented the Genomic Conceptual Model (GCM), which mediates over the most important and complex data sources [3]. This paper is focused on the process required to generate the GCM content, specifically on a novel architecture for metadata ingestion and on the resulting repository:

- We describe **META-BASE**, a novel architecture for the integration of genomic datasets; the architecture is deployed as a generic pipeline of six progressive steps for data integration, applicable to arbitrary genomic data sources providing semi-structured metadata descriptions. Two steps are assisted by tools that help the designer in the progressive creation and adaptation of data management rules, with the general objective of minimizing the cognitive effort required from integration designers.
- The pipeline generates the **META-BASE repository**, a very large integrated repository of tertiary genomic datasets. In this paper, we focus on the integration of three data sources featuring complex metadata: ENCODE [36], GDC [14], and Roadmap Epigenomics [18]. In addition, META-BASE integrates other sources, whose conceptual complexity is much simpler: genomic annotation data from GENCODE [8] and RefSeq [32]; topologically associating domains (TADs) [34] from GEO (https://www.ncbi.nlm.nih.gov/geo/); epigenomic data from Cistrome [26]. The META-BASE repository will continue to grow in the next years, responding to biological and clinical needs.

Every step of the META-BASE pipeline produces a data ingestion program that can be applied to data sources after an initial design; these programs need to be adapted only in case of structural changes of the data sources. The process is extensible, as the designer who wants to add a new source has just to add new definitions and rules to the data integration framework.

Within the data enrichment step of the META-BASE pipeline, we also use some selected ontological sources for improving value matching, which is extended from exact match to semantic match inclusive of the use of synonyms, hyponyms and hyperonyms; they enable simple value conversion strategies, which capture some value mismatches that may occur in different repositories.

- *The authors are with Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via Ponzio 34/5, 20133, Italy. E-mail: first.last@polimi.it*

Queries upon the META-BASE repository can be used for producing as a result the URIs of the relevant data in the source repositories; scientists can build over them an arbitrary genomic computation, using any bioinformatics system and resource. In this way, the META-BASE repository provides a conceptual entry point to the supported genomic data sources. In addition, the META-BASE pipeline and repository feed an architecture for genomic data processing, defined in [25], providing portable and scalable genomic data management on powerful servers and clusters[1]; in such distinct environment, metadata can be queried together with their respective datasets using GenoMetric Query Language (GMQL), a high-level domain-specific query language [23].

The most innovative aspects of our work are: from a computer science perspective, providing an end-to-end pipeline whose steps make novel use of rewrite rules for data cleaning, mapping, normalization, enrichment and integrity verification; from a biological perspective, the partitioning schemes for each data source and the selection of the ontologies providing enrichment for specific GCM attributes.

Paper organization. Section 2 overviews the GCM, which is at the base of this work, and provides a motivating example. Section 3 describes our generalized approach to metadata management and integration. Section 4 describes the pipeline to extract metadata from original sources and to prepare it for integration. Section 5 shows the integration process towards the final META-BASE repository, which includes ontological enrichment. Section 6 discusses the effectiveness of our approach. Section 7 describes the architecture of the system. Section 8 overviews related work, and Section 9 mentions future developments and concludes the paper.

2 Background

In [3] we originally presented the Genomic Conceptual Model (GCM), an Entity-Relationship model used to describe metadata of genomic data sources; its current version[2] is shown in Fig. 1. The main objective of GCM is to recognize the common organization for a limited set of concepts that are supported by most genomic data sources, although with very different names and formats.

GCM is organized as a star-schema centered the Item entity, representing an elementary experimental file of genomic regions and their attributes. Files are typically used by biologists for data extraction, analysis and visualization operations. Four hierarchical dimensions describe: 1) the *biological* elements involved in the experiment: the sequenced sample, its preparation, its donor; 2) the *technology* used in the experiment, including the specific technique; 3) the *management* aspects of the experiment: the projects/organizations behind its preparation and production; 4) the *extraction* parameters used for internal selection and organization of items.

1. Based on Apache Spark http://spark.apache.org/ and Apache Hadoop http://hadoop.apache.org/

2. Several adaptations of the model w.r.t. what was presented in [3] have been performed afterwards.

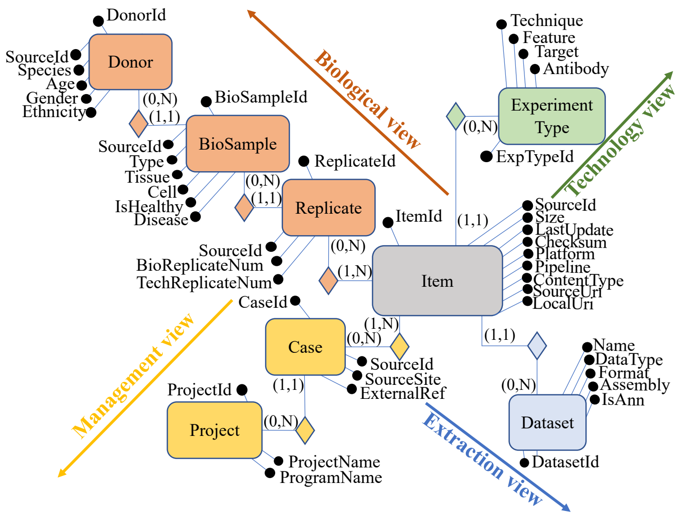
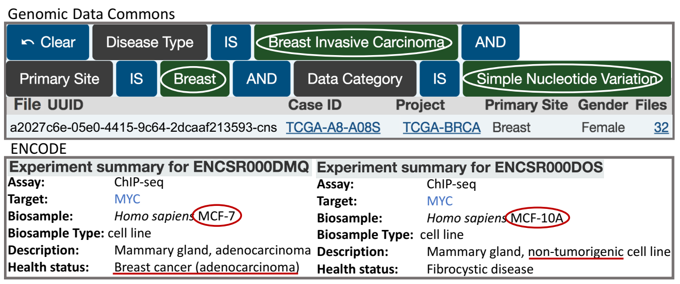


Fig. 1. Genomic Conceptual Model.

The **Central Entity** Item features the *SourceId* attribute, which identifies it uniquely on the source system, along with its *Size*, *LastUpdate*, and *Checksum*. *Platform* and *Pipeline* are respectively used to provide references to the methods and parameters used for production and processing of sequenced raw data (to which sometimes biologists resort for reprocessing) and its processed data. *ContentType* accepts values such as "peaks"/"hotspots"/"exon quantifications" when the contained regions are experimental, or "gene"/"transcript"/"promoter" when they are annotations. The Item is physically available for download at the *LocalUri* and, in its original form, at the *SourceUri*.

The **Biological View** consists of the chain of entities Item-Replicate-BioSample-Donor, representing the biological elements that contribute to the Item production. An Item is associated with one or more Replicates, each originated by a BioSample, each derived from a Donor. Donor is identified by a *SourceId*. It represents an individual (characterized by *Age*, *Gender* and *Ethnicity*) or strain of a specific organism (*Species*) from which the biological material was derived or the cell line was established. BioSample is identified within possibly multiple original sources by *SourceId*. Its characterizing *Type* expresses values such as "cell line", "tissue", or "primary cell", depending on the kind of material sample used for the experiment. *Cell* includes information of (single) cells in their natural state, immortalized cell lines, or cells differentiated from specific cell types. *Tissue* includes information regarding a multicellular component in its natural state, or the provenance tissue of the *Cell*(s) of a biosample. *IsHealthy* denotes a healthy (normal/control) or non-healthy (e.g., tumoral) sample, and *Disease* stores information about the disease investigated with the sample. Replicate is useful to model cases where an assay is performed multiple times on similar biological material. If repeated on separate biological samples, the generated items are biological replica of a same experiment; if repeated on two portions from the same biological sample (treated for example with same growth, excision, and knockdown), the items are technical replicates. This occurs only in some epigenomic data sources (such as ENCODE and Roadmap Epigenomics) that perform assay replication.



File UUID	Case ID	Project	Primary Site	Gender	Files
a2027c6e-05e0-4415-9c64-2dcaaf213593-cns	TCGA-A8-A08S	TCGA-BRCA	Breast	Female	32

Experiment summary for ENCSR000DMQ	
Assay:	ChIP-seq
Target:	MYC
Biosample:	*Homo sapiens* MCF-7
Biosample Type:	cell line
Description:	Mammary gland, adenocarcinoma
Health status:	Breast cancer (adenocarcinoma)

Experiment summary for ENCSR000DOS	
Assay:	ChIP-seq
Target:	MYC
Biosample:	*Homo sapiens* MCF-10A
Biosample Type:	cell line
Description:	Mammary gland, non-tumorigenic cell line
Health status:	Fibrocystic disease

Fig. 2. Example of Web interfaces of data sources: GDC and ENCODE.

The **Technology View** describes the technology used to produce the data ITEM. An ITEM is associated by means of a one-to-many relationship with a given EXPERIMENTTYPE, which includes the *Technique* (e.g., "ChIP-seq", "DNase-seq", "RRBS") and the *Feature*, which denotes the specific genomic aspect described by the experiment (e.g., "Copy Number Variation", "Histone Modification", "Transcription Factor"). When the *Technique* is "ChIP-seq", *Target* and *Antibody* are needed to further characterize the experiment.

The **Management View** consists of the chain of entities ITEM-CASE-PROJECT describing the organizational process for the production of items. CASE represents a set of items that have been collected within the same research study. *SourceId* and *ExternalRef* contain identifiers respectively taken from the main original source and other sources that contain the same data. The *SourceSite* represents the physical site where the material is analyzed. PROJECT represents a project, a program, or a single initiative responsible for the production of the item. It provides a single point of reference to find diverse data types generated in a same research context.

The **Extraction View** includes the entity DATASET, used to describe common properties of homogeneous items. Its attributes include a *Name*, useful to locate and organize data, the *DataType*, describing the specific kind of genomic data contained in the items of such dataset (e.g., "peaks", "copy number segments", "gene expression quantification"), the *Format*, which denotes the ITEM data file format (e.g., "bed", or more specific ones such as "narrowPeak" and "broadPeak"), and the reference genome alignment (*Assembly*). *IsAnn* distinguishes between experimental items (describing arbitrary genomic regions) and annotations (describing known genomic regions).

2.1 Motivating Example

To motivate our effort, we introduce an example that simulates the research of data suitable for a genomics project on two different sources. We focus on a simple situation, which can be appreciated even by a reader with limited biological background. Consider a comparison study between a human non-healthy breast tissue, affected by carcinoma, and a healthy sample coming from the same tissue type. A researcher in the field locates two portals having interesting data for this analysis. The results obtained after some browsing are reported in Fig. 2.

For the healthy data, the chosen source is GDC Data Portal, an important repository of human cancer genomic data. As it can be seen on the top of Fig. 2, typically more data files can be retrieved by composing a query that allows locating variation data on "Breast Invasive Carcinoma" from "Breast" tissue. By browsing several metadata information sections (sometimes hard to identify), the researcher can find files corresponding to "normal" (i.e., non-tumoral) tissue.

To compare such data with others from a diseased reference, the researcher considers additional datasets coming from cell lines, i.e., cell cultures that have been permanently established and made immortal. Cell lines are frequently used in place of primary cells to study biological processes, as the scientific community tends to accept the derived findings more readily. On ENCODE, the researcher chooses both a tumor cell line (bottom left of Fig. 2) and a normal cell line (bottom right of Fig. 2), to make a control comparison. "MCF-7" is a cell line from a diseased tissue affected by "Breast cancer (adenocarcinoma)", while "MCF-10A" is its widely considered non-tumorigenic counterpart.

Note that some external knowledge is necessary in order to find these connections, which cannot be obtained on the mentioned portals. Regarding the *disease*, note that "Breast Invasive Carcinoma" and "breast cancer (adenocarcinoma)" are related sub-types of "breast carcinoma" (as observed in the EFO and DOID ontologies [16]); this allows us to compare GDC's data with the dataset from ENCODE. For what concerns the *cell line*, researchers typically query specific databases (such as the cell line browser of the Catalogue Of Somatic Mutations In Cancer[3]) or dedicated forums to discover tumor/normal matched cell line pairs. This information is not encoded in a unique way over data sources and is often missing.

3 APPROACH

The six phases of the META-BASE approach can be seen in Fig. 3. Through downloading, metadata is imported at the repository site. During transformation, they are translated to raw attribute-value pairs, which are then cleaned, thereby producing a collection of clean metadata pairs for each source. The mapper extracts information from these pairs and adds it to GCM; GCM values are then normalized (resorting to generic term-ids that may take specific sets of values) and enriched (by means of external ontologies). Finally, the consistency of the database content is checked with respect to integrity constraints.

For exemplifying the META-BASE framework, we consider three important and complex data sources:

- *ENCODE* contains datasets related to the functional DNA sequences, which intervene at the protein/RNA levels, and to the regulatory elements that control gene expression.
- *GDC* contains datasets from TCGA program, related to many aspects of cancer genomics.
- *Roadmap Epigenomics (REP)* datasets related to epigenomic features in human normal tissues often involved in human diseases.

The above repositories are subject to rapid changes, as each source is a continuously evolving system. Luckily, most changes are additive and use already existing metadata in their descriptions. For this reason, we approach each source

3. https://cancer.sanger.ac.uk/cell_lines/

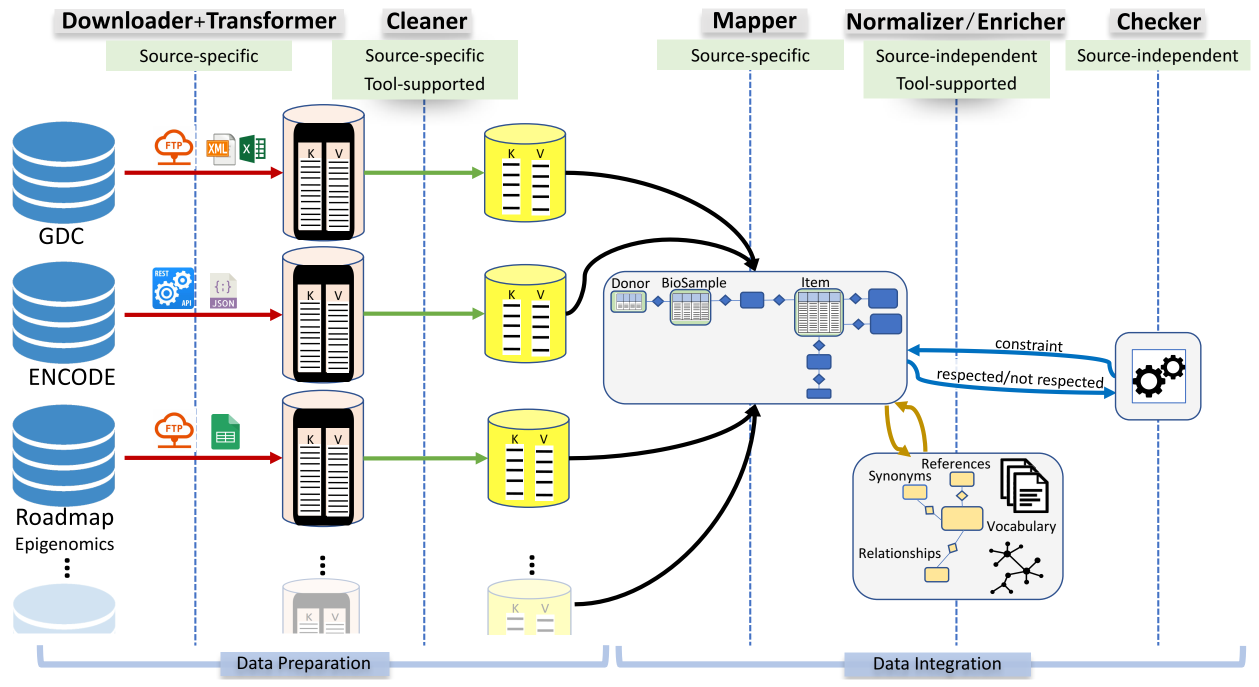


Fig. 3. The overall data preparation and integration process.

with an initial activity for the production of a source-specific set of metadata transformation rules, followed by periodic data integration sessions, where new items are discovered and their metadata are modeled.

At the same time, rules capture only a portion of the data integration semantics, as we allow for exceptions. Attributes that are not in common to most sources, while specific for few experiment types, are modeled as attribute-value pairs; the corresponding data is directly referenced from the ITEM entity.

4 Data Preparation

The META-BASE data preparation pipeline allows us to extract metadata from a set of selected data sources and arrange it for integration. Metadata is first downloaded in their original formats (Section 4.1); then, it is transformed into a ⟨key, value⟩ equivalent form (Section 4.2); finally, since it is still *raw metadata*, it is exposed to a cleaning process that aims to improve raw attribute names and to filter irrelevant metadata (Section 4.3) before data integration.

4.1 Data Download

The *Downloader* module produces files both for the genomic data and its metadata, in original source-specific format, at the processing site hosting our repository; it must be programmed or adapted for each source. In most target sources, several protocols or APIs are made available for data downloading, but they do not share any standard for the metadata description or format (e.g., XML, JSON, tab-delimited). Some sources provide a metadata file for each experiment. In other cases, a single metadata file describes a collection of experiments; also in such a case, we produce multiple metadata files, one for each experimental data file.

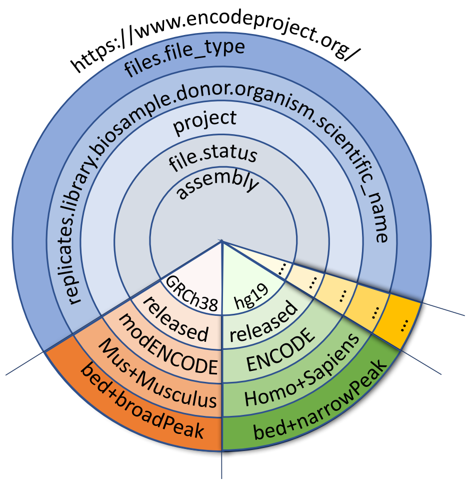


Fig. 4. Selection of portions from *ENCODE*. In the upper area we specify parameters names, in the two small bottom slices we specify example values, defining a partition of the source.

The main difficulty is to identify a specific data partitioning scheme at each source; in this way, each partition can be repeatedly accessed and source files that are added to or modified within the partition can be selectively recognized, avoiding the download for those source files that are unchanged. Table 1 illustrates three endpoints for data download used for the considered sources, with their protocol, request format, and example actual parameters for invocation. A partitioning scheme for the *ENCODE* data source is illustrated in Fig. 4, with a specific set of parameters used during download, corresponding to a partition.

Formalization. For a given source i, a *Downloader* is a method $\mathcal{D}_i$ for importing genomic data and metadata from a specific partition P_i of a data source. At each invocation of the method, a new set D_i of files (one for each data ITEM) is retrieved at the repository site, and associated with a sig-

TABLE 1
Endpoints for data download from sources and example invocations.

ENCODE	List of `file_id`	Protocol HTTP GET: https://www.encodeproject.org/metadata/?type=Experiment&⟨`params`⟩/metadata.tsv
	Example `params`	assembly=hg19 & file.status=released & project=ENCODE & ... & files.file_type=bed+narrowPeak
	Download file	https://www.encodeproject.org/files/⟨`file_id`⟩/@@download/⟨`file_id`⟩.bed.gz
GDC	List of `file_id`	Protocol HTTP POST: https://api.gdc.cancer.gov/files with ⟨`params`⟩ in Payload
	Example `params`	field:cases.project.project_id-value:["TCGA-ACC"], field:files.data_type-value:["Copy Number Segment"], ...
	Download file	https://gdc-api.nci.nih.gov/data/⟨`file_id`⟩
REP	`dir` paths	Protocol FTP: http://egg2.wustl.edu/roadmap/data/byFileType/peaks/consolidated/`dir`
	Example `dir`	broadPeak
	Download file	http://egg2.wustl.edu/roadmap/data/byFileType/peaks/consolidated/⟨`dir`⟩/⟨`file_name`⟩.⟨`dir`⟩.gz

nature $\langle dataset\_name, source, endpoint, parameters\rangle$; parameters include the timestamp t_h of the download operation. In this way, future invocations of $\mathcal{D}_i$ at time $t_k > t_h$ will be used to download information from P_i and then start a data integration session by tracking the changes that occurred to P_i at the data source between time t_h and t_k.

Method. Each download module first connects to the data source servers and retrieves the list of the identifiers of the files that belong to the partition to be downloaded (corresponding to the ITEMs of the conceptual model). Many sources provide (semi-)programmatic methods to translate a query composed on their portal visual interface into an API request or a downloadable list of files corresponding to the search; otherwise, this step has to be programmed.

For each ITEM, the downloader typically retrieves the *Size*, *LastUpdate* and *Checksum*, denoting properties of the data file; these are provided by most sources.[4] We match these values with data that is stored in GCM, using the *SourceId* unique identifier. The matching allows us to pinpoint:

- New items: they are stored as genomic data files and their metadata are processed by invoking the pipeline discussed in this section.
- Matching items, having same *Size*, *LastUpdate* and *Checksum* values as their local values stored in the conceptual model. In this case, we reprocess just the metadata by invoking the pipeline discussed in this section, but we avoid the download of region data, which is typically much bigger in size. If any one of the metadata values is different, we then download also the genomic data files.
- Missing items, i.e., items whose identifier was present at the previous invocation but it is no longer present: these items are deprecated, the genomic data and metadata is copied to an archive, which can only be inspected by archive lookups (but they are no longer retrieved by standard queries).

A downloader task splits originally downloaded metadata into files that correspond each to a single experimental region data file (e.g., ITEM). Eventually, we collect into the set of files D_i all the metadata relative to new or changed items; these downloaded files are then used in the next steps of the pipeline. In parallel, the corresponding genomic data files are stored in the GMQL data repository (see Section 7).

Example. The *ENCODE* Web portal, described in [6], supports a faceted searching system that can be used to evaluate alternative options for metadata retrieval. Each search option produces different JSON objects. After careful analysis, we selected the option of retrieving the JSON file associated with an *experimental study* in "embedded" mode, as it includes compact information about all data files, replicates, and biosamples involved in it. Therefore, this was selected as the *ENCODE* metadata reference endpoint (see Table 1): when a downloader is invoked, it retrieves the data and metadata files partitioned by experiment.

```
{"accession": "ENCSR635OSG",
 "assembly": ["hg19"],
 "award": {
           "pi": {
                 "lab": {"name": "michael-snyder",...},
           ...},
 ...},
 "dbxrefs": [],
 "files": [
     {"accession": "ENCFF134AVY",
      "biological_replicates": [1],...},
     {"accession": "ENCFF429VMY",
      "biological_replicates": [1,2],
      "file_type": "bed narrowPeak",...},
      ...
 ],
 "replicates": [
     {
         "@id": "/replicates/4874c170-7124-4822-a058-4bb/",
         "biological_replicate_number": 1,
         "library": {
             "biosample": {
                 "donor": {"age": "6",...},
                 "health_status": "healthy",
             ...},
         ...},
         "antibody": {"lot_id": "940739",...},
         ...
     },
     {
         "@id": "/replicates/d42ff80d-67fd-45ee-9159-25a/",
         "biological_replicate_number": 2,
         "library": {
             "biosample": {
                 "donor": {"age": "32",...},
                 "health_status": "healthy with non-
                      obstructive coronary artery disease",
             ...},
         ...},
         "antibody": {"lot_id": "940739",...},
         ...
     }
 ],
 ...}
```

Listing 1. Excerpt from example JSON file retrieved for *ENCODE* experiment ENCSR635OSG.

As it can be observed in Listing 1, the information associated with the specific experiment with accession ENCSR635OSG is a hierarchically structured JSON file, including several embedded elements: information about the whole experimental study, arrays of "files" elements (a list of items included in the experimental study) and of "replicates" elements, along with other information.

After retrieval, the identifiers of the items belonging to the considered partition are recorded together with their size, last update date and checksum. Then, a downloader task separates the information retrieved for an experiment

4. If some of them are unavailable, we either compute them at the source or accept a less precise matching by using fewer parameters.

into several metadata files, each containing the information about a specific item of the experimental study. From the excerpt of Listing 1, two JSON files are created for items ENCFF134AVY and ENCFF429VMY, where the former one has one replica while the latter one has two. In this way, all following data preparation steps apply to metadata files that are in one-to-one correspondence with data ITEMs.

4.2 Data Transformation

The *Transformer* module takes as input the metadata files resulting from the previous phase and transforms them into key-value pairs (consistent with the Genomic Data Model [24]).

Formalization. A *Transformer* is a source-specific method $\mathcal{T}_i$. When applied to each file in D_i downloaded from a given source i, it produces a file in T_i of $\langle key, value\rangle$ pairs, compatible with the GDM format.

Method. The *Transformer* process downloads files with an adaptation strategy that depends on their format: (i) hierarchical formats (JSON, XML, or equally expressive) require applying a flattening procedure that creates for each value a pair formed by a key (composed as the concatenation of all JSON/XML elements from the root to the element corresponding to the selected value) and the value itself; (ii) tab-delimited formats (CSV or Excel/Google Sheet) require pivoting tab-delimited columns into rows (which corresponds to creating key-value pairs); (iii) completely unstructured metadata formats, collected from Web pages or other documentation provided by sources, need case-specific formatting. The output of a transformer is a lists of key-value pairs, added to the set T_i. We wrote transformers for the most used formats for origin metadata. Additional ones can be easily added.

Algorithm 1 Transformer Procedure

```
function TRANSFORMATION($D_i, T_i$)
    for each $d \in D_i$ do
        switch $d$ do
            case $d$ is hierarchical
                $t \leftarrow flattenPaths(d)$
            case $d$ is tab-delimited
                $t \leftarrow pivot(d)$
            case $d$ is unstructured
                $t \leftarrow manualFormatting(d)$
        $T_i \leftarrow T_i + newTransFile(t)$
    end for
    return $T_i$
end function
```

Example. The output of data transformation for *ENCODE* is shown in Listing 2; it is obtained by considering as input the portion of the JSON file from Listing 1, which describes the information extracted for a specific item with accession ENCFF429VMY (with two replicates) of experiment ENCSR635OSG. First-level elements are translated directly to $\langle key, value\rangle$ pairs (e.g., $\langle$accession, ENCSR635OSG$\rangle$); nested elements are flattened (e.g., `"name"` inside `"lab"`, inside `"pi"`, inside `"award"` becomes `award__pi__lab__name`, where double underscore `__` is used to separate levels of nesting); arrays are translated in one $\langle key, value\rangle$ pair for each value in the array (e.g., see `file__biological_replicates`); empty arrays are not translated (e.g., `"dbxrefs"`).

```
accession ENCSR635OSG
assembly hg19
award__pi__lab__name michael-snyder
file__accession ENCFF429VMY
file__biological_replicates 1
file__biological_replicates 2
file__file_type bed narrowPeak
replicates__1__@id /replicates/4874c170-7124-4822-a058-4bb/
replicates__1__biological_replicate_number 1
replicates__1__library__biosample__donor__age 6
replicates__1__library__biosample__health_status healthy
replicates__1__antibody__lot_id 940739
replicates__2__@id /replicates/d42ff80d-67fd-45ee-9159-25a/
replicates__2__biological_replicate_number 2
replicates__2__library__biosample__donor__age 32
replicates__2__library__biosample__health_status
    healthy with non-obstructive coronary artery disease
replicates__2__antibody__lot_id 940739
```

Listing 2. Excerpt from example transformed file corresponding to *ENCODE* file accession ENCFF429VMY.

Note that several replicates can be associated with each file; in such a case, a progressive naming scheme tracks the replicate to which each $\langle key, value\rangle$ pair relates. In the specific example, the file has two biological replicates, each with five associated key-value pairs (in Listing 2 other pairs are omitted for brevity). All elements in the replicate element with id `4874c170-7124-4822-a058-4bb` are transformed into keys that start with `"replicate__1__"`. Vice versa, elements in replicate `d42ff80d-67fd-45ee-9159-25a` are transformed into keys that start with `"replicate__2__"`.

4.3 Data Cleaning

After the transformation step, a typical key is a long string, e.g., `replicates__1__library__biosample__donor__age`. As this information applies to an ITEM, a much simpler attribute name can be derived, e.g., `donor__1__age`. Such name is later used to map values in the conceptual schema, and is a much simpler key.

The *Cleaner* module applies transformation rules to complex attribute names, so as to simplify them. Left parts of rules use the formalism of *regular expressions:* they recognize the strings that compose a complex attribute. Then, an action encoded in the form of *pattern matching replacement strategy* builds a simpler string. The use of regular expressions brings a simple formalization of cleaning algorithms through language containment and language-recognizing automata. Rules are source-specific, as they depend on the specific way in which attribute names are encoded at each source; after an initial design, they are applied to each transformed file. Rules may require adjustments when the attribute encoding changes, or new attributes are created. We provide a tool for rule design and ordering, which assists designers in rule creation and maintenance.[5]

Informally, rules consist of an antecedent, recognizing an input string, and a consequent, transforming it into a simpler output string. The rule's antecedent is a regular expression matching a sequence of keys; it contains parentheses, which group parts of regular expressions in order to either apply a quantifier or to restrict alternation to the entire group, and positionally identify the rule's

5. https://github.com/DEIB-GECO/Metadata-Manager/wiki/Rule-Base-Generator

parameters, used in the rule's consequent as numbered capturing groups.[6] Some parameters are typed, e.g., `[0-9]` denotes a sequence of digits; some keys may be equivalently used, e.g., `(age|sex)` denotes an alternative. The consequent can contain strings of characters or special "dollar" symbols, which positionally refer to the content of the antecedent's variables. The consequent can be empty, in which case no cleaned key is generated for the transformed key, and the corresponing pair is removed. For illustration purposes, rules are indicated with the notation *antecedent ⇒ consequent*.

Rule Example. An example of a rule is:
`replicates(__[0-9]__)library__biosample__(donor)__(age|sex)(.*)` ⇒ `$2$1$3$4`.
When `replicates__1__library__biosample__donor__age` is considered as the input key, `$2$1$3$4` stands for a concatenation of the content of the second variable `donor`, with the first one `__1__`, with the third one `age` and finally with the fourth one (i.e., anything that follows the third parenthesis) - in this case an empty string. As a result, the rule produces the string `donor__1__age`.

Formalization. A *Cleaner* is a source-specific method $\mathcal{C}_i = \langle \mathrm{T}_i, \mathrm{C}_i, \mathcal{RB}_i \rangle$. For every transformed metadata file in T_i, it converts the key k of each key-value pair $\langle k, v \rangle$ from its transformed syntax to a cleaned version k' of pair $\langle k', v \rangle$. If k' is empty, a related pair is not produced. Files in C_i contain cleaned key-value pairs and are produced by running the rule engine $\mathcal{C}_i$ over T_i using the set of rules $\mathcal{RB}_i$.

Method. The description of the method requires the definition of relationships between rules and of rule base.

Definition 1. *(Rule Equivalence, Containment, and Partial Overlap) Given two rules $r, r' \in \mathcal{RB}_i$, their antecedents $r.a$ and $r'.a$, and the corresponding generated languages $\mathcal{L}(r.a)$ and $\mathcal{L}(r'.a)$:*

- *r is* equivalent *to r' when $\mathcal{L}(r.a) = \mathcal{L}(r'.a)$;*
- *r is* contained *in r' when $\mathcal{L}(r.a) \subset \mathcal{L}(r'.a)$;*
- *r* partially overlaps *r' when $\mathcal{L}(r.a) \not\subset \mathcal{L}(r'.a)$, $\mathcal{L}(r'.a) \not\subset \mathcal{L}(r.a)$, and $\mathcal{L}(r.a) \cap \mathcal{L}(r'.a) \neq \emptyset$*

Definition 2. *(Rule Base) The $\mathcal{RB}$ Rule Base is a list of rules such that rule r precedes rule r' in $\mathcal{RB}$ if either 1) r is contained in r', or 2) r partially overlaps r' and the user gives priority to r over r'.*

By effect of the above definitions, rules that are more specific precede more general rules. When the intersection of languages recognized by the rules is non-empty, the user can specify the desired order in which the rules should appear in the RB. When the intersection is empty, the rules' order in the RB corresponds to the order of insertion.

Building a *Cleaner* requires building the Rule Base (Algorithm 2), by calling the function to insert a rule in the right order (Algorithm 3), which is based on the comparison between pairs of rules performed by the function COMPARE (Algorithm 4). When the Rule Base is prepared, it is applied to the transformed files, in particular to the keys from the $\langle key, value \rangle$ pairs in T_i (Algorithm 5). After the consolidation of cleaning rules, a rule base can be repeatedly applied to transformed data, until major changes occur at the sources.

6. The replacement strategy specified by a rule is implemented using the `java.util.regex` library (https://docs.oracle.com/javase/8/docs/api/java/util/regex/package-summary.html), supporting full regular expressions.

Algorithm 2 Rule Base Creation
```
1: function RBCREATION($RB, SK, AK$)
2:     $UK \leftarrow AK - SK$
3:     while $UK$ is not empty do
4:         $newRule \leftarrow getRuleFromUser()$
5:         if $userApprSimul(RB, newRule)$ then
6:             RULEINSERTION($RB, newRule$)
7:             $matched \leftarrow matchAll(RB, UK)$
8:             $SK \leftarrow SK + matched$
9:             $UK \leftarrow UK - matched$
10:        end if
11:    end while
12: end function
```

Algorithm 2 takes as input RB, which stores the information about rules in their order (Def. 1), SK, the set of seen keys, and AK, the set of all keys retrieved from the files of a given source. It first finds the unseen keys UK (those that have not been considered for rule creation yet). Then, until all unseen keys have been considered, the user is asked to insert new rules and approve (or not) the simulated effect of the incremented RB on all keys. When the user is satisfied with the results, the rule is actually added to the RB and the sets of keys are updated accordingly.

Algorithm 3 Rule addition in Rule Base
```
1: function RULEINSERTION($RB, newRule$)
2:     for $r$ in $RB$ do
3:         $res \leftarrow$ COMPARE($newRule, r$)
4:         if $res$ is EQUIVALENT then
5:             if $userPref(newRule, r) = newRule$ then
6:                 $replaceRule(newRule, RB, indexOf(r))$
7:             end if
8:             return $RB$
9:         else if $res$ is CONTAINED then
10:            $addRule(newRule, RB, indexOf(r))$
11:            return $RB$
12:        else if $res$ is PARTIALLY_OVERLAPS then
13:            if $userPriority(newRule, r) = newRule$ then
14:                $addRule(newRule, RB, indexOf(r))$
15:                return $RB$
16:            end if
17:        end if
18:    end for
19:    $addRule(newRule, RB, RB.size)$
20:    return $RB$
21: end function
```

Adding a new rule to the Rule Base means inserting it in the right position with respect to the order defined in Def. 2. This is accomplished by Algorithm 3, which iterates over the RB list and, based on the comparison between each pre-existing rule with the one to be added, determines the insertion position (in an "insertion sort" manner).

Comparing rules means evaluating the containment relationship between the languages generated by their antecedents, as described by Algorithm 4. Several procedures

Algorithm 4 Order comparison between rules

```
1:  function COMPARE($r, r'$)
2:      $\mathcal{A}_r \leftarrow NFA2DFA(RegEx2NFA(r.a))$
3:      $\mathcal{A}'_r \leftarrow NFA2DFA(RegEx2NFA(r'.a))$
4:      if $\mathcal{L}(\mathcal{A}_r) = \mathcal{L}(\mathcal{A}'_r)$ then
5:          return EQUIVALENT
6:      else if $\mathcal{L}(\mathcal{A}_r) \subset \mathcal{L}(\mathcal{A}'_r)$ then
7:          return CONTAINED
8:      else if $\mathcal{L}(\mathcal{A}_r) \not\supset \mathcal{L}(\mathcal{A}'_r) \wedge \mathcal{L}(\mathcal{A}_r \cap \mathcal{A}'_r) \neq \emptyset$ then
9:          return PARTIALLY_OVERLAPS
10:     end if
11: end function
```

exist to convert regular expressions into equivalent Non-deterministic Finite Automata (NFA); we use the Brics Java library [27] for automata implementations, which is based on Thompson's construction algorithm [41]. Then, NFA need to be converted into equivalent Deterministic Finite State Automata (DFA) $\mathcal{A}_r$ and $\mathcal{A}_{r'}$ – this can be done with the Rabin-Scott powerset construction [33]. Later, the two languages are checked for equivalence, containment and partial overlapping (by using the automaton constructed from the cross-product of states that accepts the intersection of the languages). Algorithm 5 describes the *Cleaner* as application of the Rule Base to the input dataset; rules are applied in the order in which they appear in the Rule Base.

Algorithm 5 Application of Rule Base to keys

```
1:  function CLEANER($RB$,$\mathrm{T}_i$)
2:      $\mathrm{C}_i \leftarrow []$
3:      for each $\langle key, value \rangle \Leftarrow \mathrm{T}_i$ do
4:          $newKey \leftarrow matchFirst(key, RB)$
5:          if $nonEmpty(newKey)$ then
6:              $add(\mathrm{C}_i, \langle newKey, value \rangle)$
7:          end if
8:      end for
9:      return $\mathrm{C}_i$
10: end function
```

Example. Table 2 shows the cleaning of a set of transformed *ENCODE* keys. It assumes an initial set of transformed keys from T_i; for each key, the user produces cleaning rules, driven by Algorithm 2. Eventually, the method produces a rule base made of a list of 7 rules; their application to keys in T_i produces the set of cleaned keys in C_i.

For instance, rule (2) deletes the key: `replicates__1__library__biosample__sex`. Rule (3), applied to the key: `replicates__1__library__biosample__biosample_type`, dictates that the key must be rewritten by concatenating the content of the second parenthesis (i.e., `biosample`) with the content of the first (i.e., `1`), and with the content of the fourth (i.e., `type`), obtaining at the end `biosample__1__type`.

TABLE 2
Example of cleaning process.

Transformed keys in $\mathbf{T}_i$

replicates__1__library__biosample__donor__age	32
replicates__1__library__biosample__donor__age_units	year
replicates__1__library__biosample__donor__sex	male
replicates__2__library__biosample__donor__age	4
replicates__2__library__biosample__donor__age_units	year
replicates__2__library__biosample__donor__sex	female
replicates__1__library__biosample__sex	male
replicates__1__library__biosample__biosample_type	tissue
replicates__1__library__biosample__health_status	healthy, CAD
file__biological_replicates	1
file__technical_replicates	1_1
file__assembly	GRCh38
file__file_type	bed narrowPeak
replicates__1__biological_replicate_number	1
replicates__1__technical_replicate_number	1
replicates__2__biological_replicate_number	2
replicates__2__technical_replicate_number	1
assembly	hg19

↓ **RuleBase RB_i**

```
(1) replicates(__[0-9]__)library__biosample__(donor)__(age | sex)(.*)
                                                      ⇒ $2$1$3$4
(2) replicates__[0-9]__library__biosample__sex.* ⇒
(3) replicates(__[0-9]__)library__(biosample)__(biosample_)?(.*)
                                                        ⇒ $2$1$4
(4) file__(biological | technical)_replicates ⇒
(5) (file__)(file_)?(.*) ⇒ $1$3
(6) (replicate)s(__[0-9]__)(.*) ⇒ $1$2$3
(7) assembly ⇒
```

↓ **Cleaned keys in $\mathbf{C}_i$**

donor__1__age	32
donor__1__age_units	year
donor__1__sex	male
donor__2__age	4
donor__2__age_units	year
donor__2__sex	female
biosample__1__type	tissue
biosample__1__health_status	healthy, CAD
file__assembly	GRCh38
file__type	bed narrowPeak
replicate__1__biological_replicate_number	1
replicate__1__technical_replicate_number	1
replicate__2__biological_replicate_number	2
replicate__2__technical_replicate_number	1

5 Data Integration

The META-BASE data integration process consists of three phases. During data mapping, described in Section 5.1, cleaned metadata is mapped into a global relational schema that embodies the conceptual schema presented in Section 2. Data mapping is a simple syntactic transformation; the following phase of value normalization and enrichment, described in Section 5.2, produces homogenized data equipped with appropriate ontological term labels, references, hyponyms, hypernyms and synonyms. Finally, the integrity constraint checker, discussed in Section 5.3, provides methods for specifying and enforcing integrity constraints that describe legal values in the META-BASE repository. For a high-level workflow of the data integration process refer again to Fig. 3.

5.1 Data Mapping

The *Mapper* module is in charge of the integration at the schema-level of a set of cleaned keys produced for each source. The method extends the work proposed in [3], where we first introduced local-to-global mappings using a classical Datalog syntax. The current *Mapper* is part of a broader integration workflow in which metadata is made available as lists of $\langle key, value \rangle$ pairs; mapping rules build relational rows from such pairs.

The global relational schema $\mathcal{G}$ is obtained as straightforward mapping from the conceptual schema in Fig. 1.

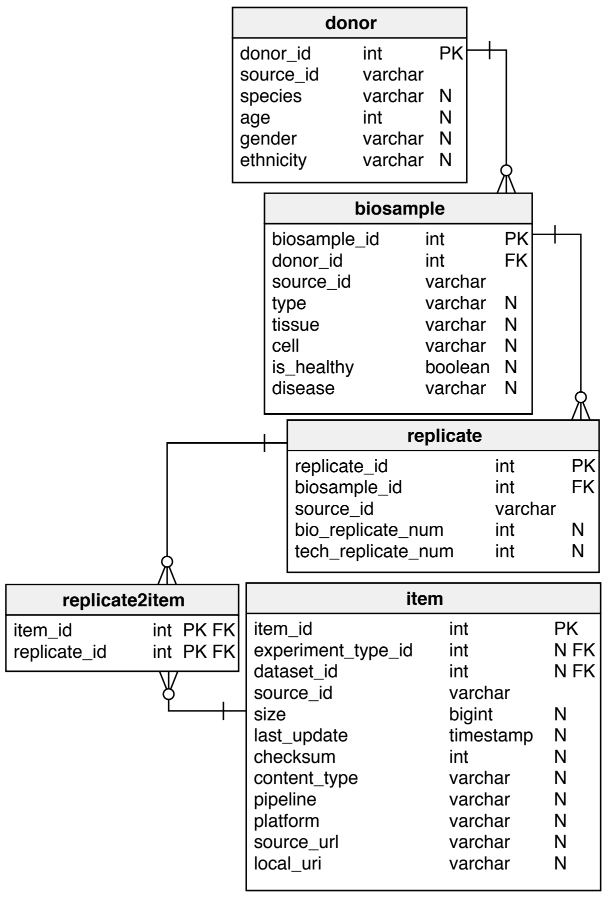


Fig. 5. Logical schema of the GCM biological view.

It contains the central entity table ITEM, a set of entity tables DONOR, BIOSAMPLE, REPLICATE, PROJECT, CASE, DATASET, EXPERIMENTTYPE, which model as well 1:N relationships, and two relationship tables ITEM2REPLICATE and ITEM2CASE, which model the two N:N relationships. Fig. 5 shows the logical schema of the biological view, where *PK* denotes attributes forming the table's primary key, *FK* denotes foreign keys, *N* denotes nullable attributes, multiplicity in the edge denotes a many mapping, a circle on the edge denotes an optional mapping, and a cut on the edge denotes a mandatory mapping.

Every source is represented by a set of cleaned files C_i, each of which contains a set of $\langle key, value\rangle$ pairs, where the keys are produced by the *Cleaning* phase. Mapping rules assemble several values extracted from the key-value pairs into rows of the relations in $\mathcal{G}$. Their format recalls deductive rules: each table of $\mathcal{G}$ corresponds to several rules for each source, whose head is a predicate named as the table and with the same arity as the table's grade; the body lists several attribute-value pairs such that attribute names are matched to cleaned keys of files in C_i. The semantics of mapping rules is also similar to that of deductive rules: if all the attribute names of the body are matched to keys in C_i (in deductive terms they *unify*), then the values corresponding to those keys are assembled by the rule into relational rows.

It is possible to apply to values a set of predefined syntactic transformations (*SynTr*), defined in Table 3, which can be freely composed in the left side of mapping rules; transformations can be easily extended. For example, to put into lowercase letters two values that have been first concatenated with a space, the expression $\texttt{LCase}(\texttt{Conc}(value_1, value_2, \text{“ ”}))$ can be used to generate a value for a specific position of a row.

TABLE 3
Example syntactic transformations for mapping rules.

$\texttt{Conc}(s_1, s_2, c)$	concatenates s_1 and s_2 using c as separation string
$\texttt{Alt}(s_1, s_2)$	outputs s_1 if present and not null, else s_2
$\texttt{Rem}(s_1, s_2)$	removes the occurrences of string s_2 from s_1
$\texttt{Sub}(s_1, s_2, s_3)$	substitutes occurrences of s_2 in s_1 with the new s_3
$\texttt{Eq}(s, p)$	outputs `true` when s is equal to p, else `false`
$\texttt{ATD}(a)$	converts a, a number followed by space and unit of measurement, into the correspondent number of days
$\texttt{LCase}(s)$	converts string s into its lower case version
$\texttt{Int}(n)$	casts number n to its correspondent Integer format
$\texttt{Id}(...)$	generates synthetic *id* for faster indexing of table t from specified arguments

Formalization. A *Mapper* is a source-specific method $\mathcal{M}_i = \langle \mathrm{C}_i, \mathcal{MB}_i\rangle$. For every metadata cleaned file f in C_i, it assembles several values v present in the pairs $\langle k, v\rangle$ of f into rows of the tables of $\mathcal{G}$. Rows are produced by running the rule engine $\mathcal{M}_i$ over C_i using the mapping rules in $\mathcal{MB}_i$. A *mapping rule* is a declarative rule of the form: $\text{ENTITY}(SynTr(v_1), ..., SynTr(v_i), ..., SynTr(v_N)) \rightsquigarrow \{\langle k_1, v_1\rangle, ..., \langle k_i, v_i\rangle, ..., \langle k_N, v_N\rangle\} \subseteq f$, where every v_i in the LHS of the rule also appears in the rule RHS (i.e., rule evaluations are finite), in a positive form (i.e., rules are safe).

Method. Once mapping rules are fully specified, the method consists simply in applying the rules to each file f in C_i of a data source, in arbitrary order. Note that every file associated with a data source as produced by the cleaning method may have several versions for the same key, numbered from 1 to n_f; each rule is applied for every version, and associates with each version a distinct row. When a version is present (e.g., in rules for DONOR, BIOSAMPLE and REPLICATE of Table 4), we denote such version by generically naming the keys in the rule's body using j, and then generating a rule for each value of j. For each v_i in the rule LHS, if the corresponding $\langle k_i, v_i\rangle$ in the rule RHS exists in f, then we add $SynTr(v_i)$ to the result, i.e., a tuple in the ENTITY specified in the rule LHS.

Example. Table 4 illustrates all the rules that are required to build the relational schema shown in Fig. 5 for the *ENCODE* data source. Note that $\texttt{Oid}_t$ is the notation used for the *ObjectIdentifier* of table t, which is a unique accession retrieved from the source.

5.2 Data Normalization and Enrichment

During this step, specific values of the global schema are associated with controlled terms, lists of synonyms and hypernyms, and external references to reference ontologies. We consider nine *semantically enrichable* attributes of the global schema: *Technique, Feature* and *Target* of experiment types, *Disease, Tissue* and *Cell* of bio samples, *Ethnicity* and *Species* of donors, and *Platform* of items.

The adoption of a specific knowledge base for each semantically enrichable attribute provides us with *value normalization*, as we consider the values of reference knowledge bases as a restricted vocabulary. Using external knowledge

TABLE 4
Mapping rules for biological view of *ENCODE* source.

$\mathrm{C}_{ENC} = \{\mathrm{C}_i \mid i = ENCODE\}, \forall f \in \mathrm{C}_{ENC}, j \leq n_f$
$\textsc{Donor}(\texttt{Id}(\texttt{Oid}_D), \texttt{Oid}_D, v_1, \texttt{ATD}(\texttt{Conc}(v_2, v_3, \text{`` ''})), v_4, v_5) \rightsquigarrow$ $\{\langle \textit{donor\_\_j\_\_accession}, \texttt{Oid}_D\rangle,$ $\langle \textit{donor\_\_j\_\_organism}, v_1\rangle,$ $\langle \textit{donor\_\_j\_\_age}, v_2\rangle,$ $\langle \textit{donor\_\_j\_\_age\_units}, v_3\rangle,$ $\langle \textit{donor\_\_j\_\_sex}, v_4\rangle,$ $\langle \textit{donor\_\_j\_\_ethnicity}, v_5\rangle\} \subseteq f$
$\textsc{BioSample}(\texttt{Id}(\texttt{Oid}_B), \texttt{Id}(\texttt{Oid}_D), \texttt{Oid}_B, \text{``tissue''}, v_2, \mathrm{NULL}, \texttt{Eq}(v_3, \text{``healthy''}), v_3) \rightsquigarrow$ $\{\langle \textit{biosample\_\_j\_\_accession}, \texttt{Oid}_B\rangle$ $\langle \textit{donor\_\_j\_\_accession}, \texttt{Oid}_D\rangle,$ $\langle \textit{biosample\_\_j\_\_type}, \text{``tissue''}\rangle$ $\langle \textit{biosample\_\_j\_\_term\_name}, v_2\rangle$ $\langle \textit{biosample\_\_j\_\_health\_status}, v_3\rangle\} \subseteq f$
$\textsc{BioSample}(\texttt{Id}(\texttt{Oid}_B), \texttt{Id}(\texttt{Oid}_D), \texttt{Oid}_B, \text{``cell line''}, \mathrm{NULL}, v_2, \texttt{Eq}(v_3, \text{``healthy''}), v_3) \rightsquigarrow$ $\{\langle \textit{biosample\_\_j\_\_accession}, \texttt{Oid}_B\rangle$ $\{\langle \textit{donor\_\_j\_\_accession}, \texttt{Oid}_D\rangle,$ $\langle \textit{biosample\_\_j\_\_type}, \text{``cell''}\rangle$ $\langle \textit{biosample\_\_j\_\_term\_name}, v_2\rangle$ $\langle \textit{biosample\_\_j\_\_health\_status}, v_3\rangle\} \subseteq f$
$\textsc{Replicate}(\texttt{Id}(\texttt{Oid}_R), \texttt{Oid}_R, \texttt{Oid}_B, v_1, v_2) \rightsquigarrow$ $\{\langle \textit{replicate\_\_j\_\_uuid}, \texttt{Oid}_R\rangle,$ $\langle \textit{biosample\_\_j\_\_accession}, \texttt{Oid}_B\rangle,$ $\langle \textit{replicate\_\_j\_\_bio\_rep\_num}, v_1\rangle,$ $\langle \textit{replicate\_\_j\_\_tech\_rep\_num}, v_2\rangle\} \subseteq f$
$\textsc{Item}(\texttt{Id}(\texttt{Oid}_I), \texttt{Id}(v_1, v_2, v_3), \texttt{Id}(\texttt{Oid}_{DS}), \texttt{Oid}_I, v_4, v_5, v_6, v_7, v_8,$ $\texttt{Conc}(\text{``www.encodeproject.org''}, v_9, \text{``/''}), \texttt{Conc}(\text{``www.gmql.eu...''}, \texttt{Oid}_I, \text{``/''})) \rightsquigarrow$ $\{\langle \textit{assay\_term\_name}, v_1\rangle,$ $\langle \textit{target\_\_investigated\_as}, v_2\rangle,$ $\langle \textit{target\_\_label}, v_3\rangle,$ $\langle \textit{dataset\_name}, \texttt{Oid}_{DS}\rangle,$ $\langle \textit{file\_\_accession}, \texttt{Oid}_I\rangle,$ $\langle \textit{file\_\_size}, v_4\rangle,$ $\langle \textit{file\_\_date\_created}, v_5\rangle,$ $\langle \textit{file\_\_md5sum}, v_6\rangle,$ $\langle \textit{file\_\_pipeline}, v_7\rangle,$ $\langle \textit{file\_\_platform}, v_8\rangle,$ $\langle \textit{file\_\_href}, v_9\rangle\} \subseteq f$
$\textsc{Item2Replicate}(\texttt{Id}(\texttt{Oid}_I), \texttt{Id}(\texttt{Oid}_R)) \rightsquigarrow \{\langle \textit{file\_\_accession}, \texttt{Oid}_I\rangle,$ $\langle \textit{replicate\_\_j\_\_uuid}, \texttt{Oid}_R\rangle\} \subseteq f$

bases (rather than creating a new one) is essential in the biomedical domain, where specialized ontologies are already available and their use boosts interoperability.

This process is supervised and requires a preliminary selection of the most suitable ontologies to describe each semantically enrichable attribute of the global schema.

5.2.1 Ontology Selection

The choice of attribute-specific ontologies took into account the rules for selecting a bio-ontology given in [20]. We used four different services to evaluate the best ontologies for nine ontological attributes from GCM. These are: (a) BioPortal[7] [44], (b) Ontology Recommender[8] [22], (c) Ontology Lookup Service[9] (OLS, [16]), and (d) Zooma.[10] For each semantically enrichable attribute, we searched all values using the four services, and computed the best score for recommended ontologies.[11] Finally, for each pair attribute-ontology, we considered both the best matching scores and the coverage (number of values of given attribute that were successfully annotated, i.e., matched to an ontological term).

The results of our selection are shown in Table 5, where, for each semantically enrichable attribute, we indicate the preferred ontology and three normalized indicators. COVERAGE indicates the percentage of attribute values that are found in the ontologies. SCORE is an average matching score of all the annotated attribute values weighted by ontology acceptance. SUITABILITY is a measure of how much an ontology set is adequate for an attribute. Note that a second preferred ontology is added when the first one did not reach 0.85 coverage. In this case, indicators refer to the union of the ontologies.

TABLE 5
Choice of reference ontologies for semantically enrichable attributes.

Attribute	Pref. ontologies	Coverage	Score	Suitability
Technique	OBI, EFO	0.857	0.486	0.490
Feature	NCIT	1.000	0.854	0.893
Target	OGG	0.950	0.747	0.948
Disease	NCIT	0.978	0.784	0.802
Tissue	UBERON	0.957	0.753	0.937
Cell	EFO, CL	0.953	0.644	0.577
Platform	NCIT	1.000	0.909	0.950
Ethnicity	NCIT	0.962	0.907	0.912
Species	NCBITaxon	1.000	0.667	1.000

5.2.2 Process

The *Normalizer* is supported by an interactive tool that: 1) calls external services to annotate values with concepts from controlled vocabularies or dedicated ontologies; 2) asks for user feedback when annotations have a low matching score; users can either accept one of the proposed solutions, or manually specify new annotations.

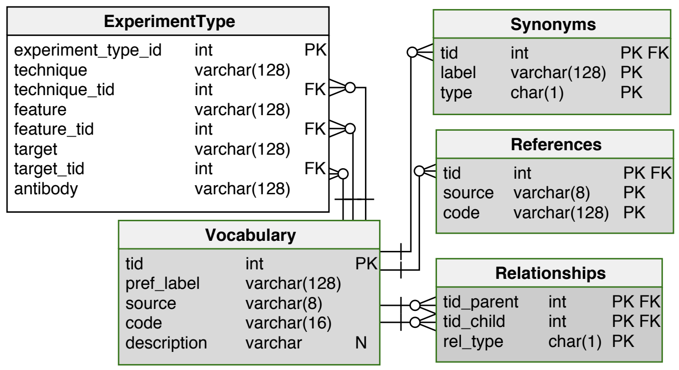


Fig. 6. Relational schema of the Local Knowledge Base $\mathcal{K}$, including links to attributes of EXPERIMENTTYPE from the global schema $\mathcal{G}$.

The result of the normalization is contained within the relational database $\mathcal{K}$, called *Local Knowledge Base*, illustrated in Fig. 6, populated from ontologies and referenced from the global schema $\mathcal{G}$. Specifically, we maintain the tables:

1) VOCABULARY: contains the term identifier and the term *preferred label*, in addition to the ontology providing the label, the code used for the label in that ontology and an optional description.
2) REFERENCES: for a given term, contains references to equivalent labels extracted from other ontologies (in the form of a pair ⟨*Source, Code*⟩).
3) SYNONYMS: contains other labels that can be used as synonyms of the preferred label in the chosen ontology.
4) RELATIONSHIPS: contains ontological hierarchical relationships between terms and the type of the relationships (either `generalization` or `containment`).

7. http://bioportal.bioontology.org/
8. https://bioportal.bioontology.org/recommender/
9. https://www.ebi.ac.uk/ols/
10. https://www.ebi.ac.uk/spot/zooma/
11. Recommender provides numerical scores, Zooma provides tags for indicating the annotation quality. For BioPortal and OLS we computed a score by considering the number of words with exact match in each ontology.

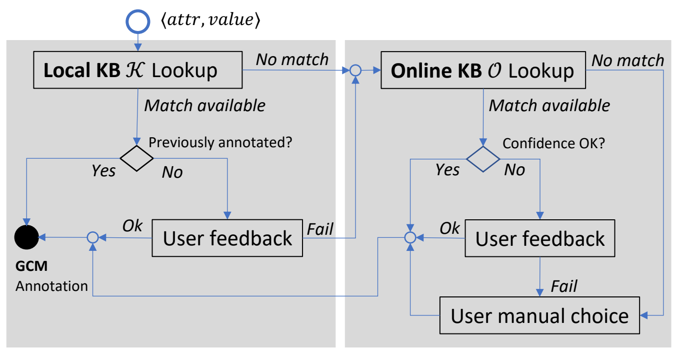


Fig. 7. Iterative supervised normalization and enrichment procedure.

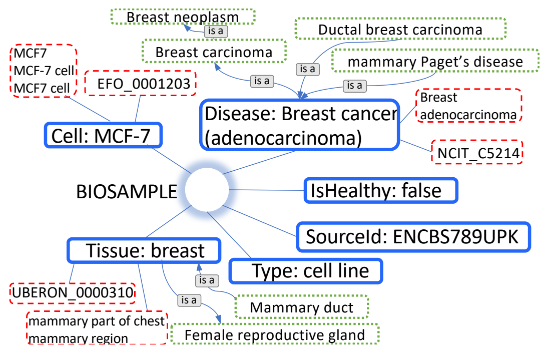


Fig. 8. Normalization and enrichment of a BIOSAMPLE tuple.

The system provides the unfolding of the hierarchies as an internal materialized view over the table RELATIONSHIPS, used for faster query processing.

Formalization. The *Normalizer* is a source-independent method $\mathcal{N} = \langle A, \mathcal{O} \rangle$. A is the set of sematically enrichable attributes of the global schema $\mathcal{G}$. For each attribute a in A and each possible value of a, $\mathcal{N}$ generates the corresponding entries in the Local Knowledge Base $\mathcal{K}$, extracted from the preferred ontologies of a in $\mathcal{O}$.

Method. Value normalization and enrichment is a supervised procedure illustrated in Fig. 7. The workflow is executed for all values of semantically enrichable attributes, and consists of two parts: 1) For each such value, the system initially looks for a suitable term in the vocabulary of the Local Knowledge Base; if a match is available, and the term was already annotated in the past, the procedure is completed. When the match is successful but annotations are lacking, a user's feedback is requested. 2) Terms that do not match with the vocabulary, or whose annotations are not approved by the user, are then searched within the specific ontologies associated with the attribute, as defined in Section 5.2.1. If matches are of high confidence (i.e., matching score), the procedure is completed; if the confidence is low, user feedback is requested. When feedback is negative or there is no match, users are asked to provide a new vocabulary term.

Example. Fig. 8 shows a tuple of the BIOSAMPLE global table. Solid line nodes include normalized attribute values, dashed line nodes represent some of the synonyms, dotted line nodes represent hierarchies, labeled by the relevant ontology (only a small subset is represented for brevity).

5.3 Integrity Checker

At the end of the integration process, we introduce integrity constraints, which define dependencies between values of the global schema $\mathcal{G}$. Preliminary versions of dependencies, called contextual and dependent features, were introduced in [3]. We consider pairs of attributes ($A_S \in R_S$ and $A_E \in R_E$), where R_S and R_E denote the starting and ending tables in the $\mathcal{G}$ global schema, connected by a join path in $\mathcal{G}$. Given that $\mathcal{G}$ is an acyclic schema, there is just one join path between any two tables in $\mathcal{G}$.

Definition 3. *A dependency rule between attributes $R_S.A_S$ and $R_E.A_E$ of $\mathcal{G}$ is an expression of the form: $Boolean(R_S.A_S) \rightarrow Boolean(R_E.A_E)$, where $Boolean(A)$ is a Boolean expression over an attribute A of $\mathcal{G}$. The interpretation of the dependency rule is that: 1)* **when** *the Boolean expression in the left part of the rule is true for a value $v_S \in A_S$, 2)* **if** *there exists one value $v_E \in A_E$ such that $\langle v_S, v_E \rangle$ are connected by the join path between R_S and R_E, 3)* **then** *the Boolean expression on the right part of the rule must be true for v_E. Boolean expressions include as special cases the predicates* IS NULL *or* IS NOT NULL.

Dependencies can be defined during the lifetime of the META-BASE repository; they are manually defined and their identification is not assisted by a tool. Table 6 shows some examples of dependency rules. For example, the first rule indicates that if the *Species* of a DONOR is "Homo sapiens" and the donor is connected to a DATASET through the only possible path in $\mathcal{G}$, then the *Assembly* of the dataset must be one of "hg19", "hg38", or "GRCh38". Dependency rules allow including in the GCM relevant attributes that are not common to all data types. For example, attributes *Target* and *Antibody* of EXPERIMENTTYPE are of great interest in ChIP-seq experiments, but are not significant in other experiments. Thus, a rule can specify that when *Technique* is not "ChIP-seq", then these attributes are null.

TABLE 6
Examples of dependency rules, including a description of the join path connecting the two attributes used in the left and right parts of the rule.

$\langle e_S, e_E \rangle$ in (DONOR·BIOSAMPLE·REPLICATE·ITEM·DATASET)
e_S.*Species* = "Homo sapiens" $\rightarrow$ e_E.*Assembly* $\in$ [hg19, hg38, GRCh38]
$\langle e_S, e_E \rangle$ in (DONOR·BIOSAMPLE)
e_S.*Gender* = "Male" $\rightarrow$ e_E.*Disease* $\neq$ "Ovarian cancer"
e_E.*Tissue* $\neq$ "Uterus"
e_S.*Gender* = "Female" $\rightarrow$ e_E.*Disease* $\neq$ "Prostate cancer"
$\langle e_S, e_E \rangle$ in (PROJECT·CASE)
e_S.*ProgramName* = "ENCODE" $\rightarrow$ e_E.*SourceId* = "ENCSR.*"
$\langle e_S, e_E \rangle$ in (PROJECT·CASE·ITEM)
e_S.*ProgramName* = "ENCODE" $\rightarrow$ e_E.*SourceId* = "ENCFF.*"
e_S.*ProgramName* = "TCGA" $\rightarrow$ e_E.*SourceId* = "^[0-9a-z]{8}-([0-9a-z]{4}-){3}[0-9a-z]{12}$"
e_S.*ProgramName* = "ENCODE" $\rightarrow$ e_E.*SourceUrl* is not null
e_S.*ProgramName* = "TCGA" $\rightarrow$ e_E.*SourceUrl* is not null
$\langle e_S, e_E \rangle$ in (BIOSAMPLE)
e_S.*Type* = "tissue" $\rightarrow$ e_E.*Tissue* is not null
e_S.*Type* = "cell line" $\rightarrow$ e_E.*Cell* is not null
$\langle e_S, e_E \rangle$ in (DATASET·ITEM)
e_S.*IsAnn* = `true` $\rightarrow$ e_E.*ContentType* is not null
$\langle e_S, e_E \rangle$ in (PROJECT·CASE·ITEM·DATASET)
e_S.*ProgramName* = "ENCODE" $\rightarrow$ e_E.*Name* = ".*ENCODE.*"
e_S.*ProgramName* = "Roadmap Epigenomics" $\rightarrow$ e_E.*Name* = ".*ROADMAP_EPIGENOMICS.*"

6 Validation

Performing a complete evaluation of the integration approach is hard from many perspectives: 1) reproducing queries on the source just within a partition of interest is not always possible; 2) query interfaces at sources may be different from ours (free-text search vs attribute-based search); 3) we cannot generate all possible queries; 4) manual check of results is very time consuming. Thus, to show the effectiveness of our approach, we performed an evaluation on a restricted number of meaningful example queries.

Table 7 reports the number of results from seven queries on human processed files, as found either in the META-BASE repository or in individual sources. Numbers do not exactly sum up, but they pinpoint analogies/differences to guide a manual verification of corresponding instances. Specifically, "H3K27me3" and "MCF-7", typically searched over epigenomics related sources, return a number of matches in our system comparable with the sum of matches in the integrated sources; "fat" and "breast", simple tissue specifications, are easily matched in all sources. Instead, disease and platform information, such as "breast cancer" and "Illumina", benefit from our enrichment procedure. The numbers of matches must be correctly interpreted; e.g., "RNA-seq" in META-BASE is associated with one third of the matches of GDC, but every META-BASE item is derived from three GDC files; hence, there is a full correspondence.

Note that each query to META-BASE repository is targeted to a specific attribute; thus, it finds items that are correctly related to the query, as we checked. By manually inspecting the data retrieved by the search interfaces of the analyzed sources, we noted some false positives. For example, the string "fat" is matched by Cistrome with genes like "NFAT5" and by ENCODE with identifiers such as "ENCSR582FAT". This explains most discrepancies. In summary, our metadata integration process enables an enhanced search functionality without losing matches.

TABLE 7
Validation of the metadata integration process through seven queries.

Attribute	Query	**META-BASE**	ENCODE	GDC	REP	Cistrome
Target	H3K27me3	1,802	649	-	381	1,440
Cell	MCF-7	2,428	1,411	-	-	1,246
Tissue	fat	130	212	-	57	10
Tissue	breast	11,746	236	20,448	94	1,970
Disease	breast cancer	11,858	-	-	-	70
Platform	Illumina	38,088	2	12,359	-	-
Technique	RNA-seq	11,491	-	33,279	56	-

The semantic normalization process has been validated by six experts in molecular biology. They have been asked to evaluate a random set of 200 matches achieved by the supervised procedure between metadata values and ontological controlled terms (either *preferred label* or *synonym*) equipped with their descriptions. Overall, 92% of results were considered adequate.

7 Overall Architecture

META-BASE is part of a broad architecture, whose main purpose is providing a cloud-based environment for genomic data processing.

The overall system architecture is presented in Fig. 9. In the left part of the figure we show the META-BASE pipeline discussed in Sections 4 and 5; the whole pipeline is configured using parameters provided as a single XML configuration file. Each dataset (on the left) is progressively downloaded, transformed and cleaned. The data mapping method transforms cleaned attribute-value pairs into the global database $\mathcal{G}$. The normalization and enrichment method adds references from the semantically enrichable attributes to the Local Knowledge Base $\mathcal{K}$, which is implemented by relational tables. Interactive access to the META-BASE repository is provided by a user-friendly interface (not discussed in this paper), that uses the acyclic structure of the global schema to support simple conjunctive queries at the center of Fig. 9), in a style that is similar to DeepBlue's query interface [1].

As shown in the right part of Fig. 9, the META-BASE repository can also be queried using the GMQL System [25], which supports integrated data managment on the cloud; the system is accessed through Web Services as a common point of access from a variety of interfaces, including a visual user interface, programmatic interfaces for Python [28] and R/Bioconductor, and workflow-based interfaces for Galaxy and FireCloud. The implementation is executed using the Apache Spark engine, deployed either on a single server or a cloud-based system.

8 Related work

Several general projects are focused on offering integrated access to biomedical data and knowledge extracted from heterogeneous sources, including BioKleisli [5] (for providing read access to complex structured data), BioMart [40] (for biomedical databases), NIF [12] (in the field of neuroscience), and DATS [38] (for scientific datasets in general). A survey of the data integration challenges in life sciences with a particular focus on *omics* subjects, therefore genomics in the first place, is provided in [10].

Many works in the literature use conceptual models in the genomics – and more in general biomedical – field. However, they employ conceptual models' expressive power to explain biological entities and their interactions [42], [37], [31], [30]. Instead, we propose in addition an architecture that uses a conceptual model for driving data integration.

Some of the large genomics consortia have also provided data models to organize metadata (see the BioProject database [2], ENCODE Data Coordination Center [13], or Genomic Data Commons [14]). However, these models do not provide so far general integration frameworks that cover aspects falling outside the specific focus of the consortium. Perhaps the most comprehensive approach is provided by DeepBlue [1], the data integration environment of the BluePrint Consortium, which is however focused only on epigenomic data (i.e., study of cell epigenetic modifications).

DNADigest [17] investigates the problem of locating genomic data to download for research purposes. The study is also documented more informally in a blog.[12] This work differs from our since, in addition to allow the dynamical and collaborative curation of metadata, they only provide means to locate raw data, while we provide data to be used by our genomic data management system (see Section 7).

12. https://blog.repositive.io/

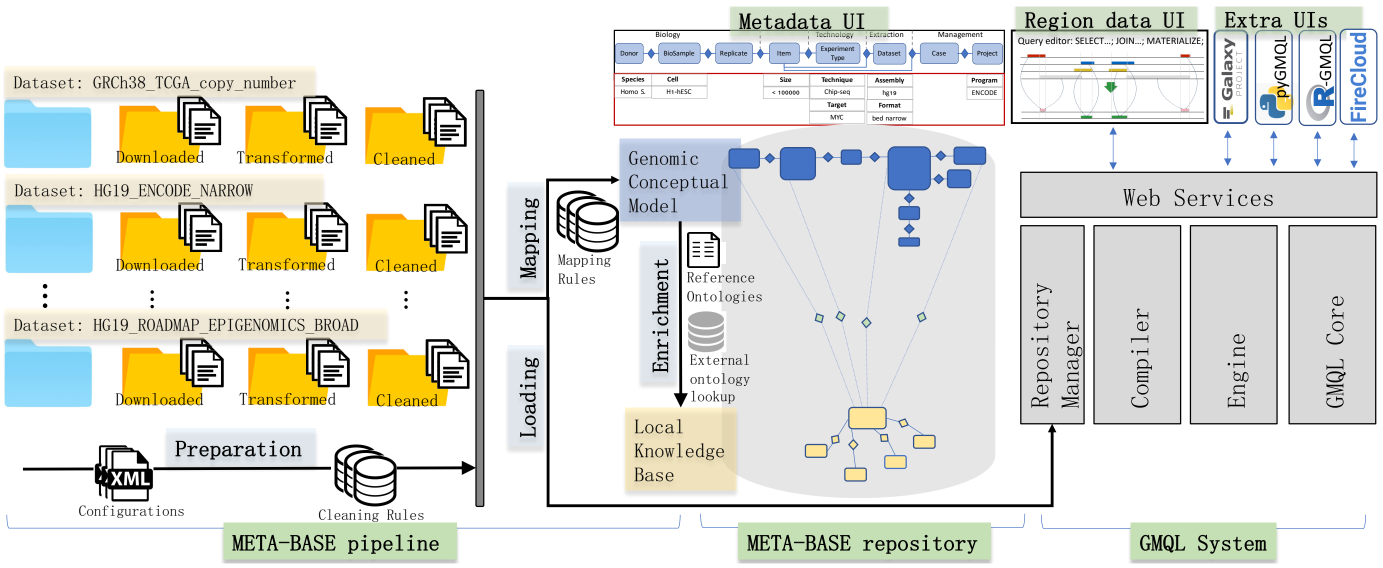


Fig. 9. Overall architecture for genomic data processing.

As to the process of semantic enrichment of metadata, many works have tackled the problem of recognizing ontological concepts to annotate data. Among others, we mention [21], [15], [39]. [4] is a very interesting survey (even if dated) on the use of ontologies in biomedical data management and integration. [9] focused particularly on GEO. [21] considers the problem of metadata authoring by using ontology-based recommendations; the authors use BioPortal services, focusing on metadata preparation and on manual creation of experiments documentation. [7] reports semantic metadata enrichment for the ENCODE dataset.

The choice of the ontologies to be used for semantic enrichment, discussed in Section 5.2.1, has been addressed in a number of articles; among them, [29] and [45]. The latter one presents the *FAIR* principles, which define the characteristics that contemporary data resources, tools and infrastructures should exhibit in order to be Findable, Accessible, Interoperable and Reusable by third parties. In Section 5.2.1 we presented the BioPortal [44], Ontology Recommender [22], Ontology Lookup Service [16] and Zooma systems for ontology lookup. We decided not to use HeTOP [11], another annotation service, as it was less precise for the specific annotations that we need.

9 Discussion and Conclusions

Genomic metadata integration is a complex process. We have designed our solution by breaking the process into several tasks and by associating powerful abstractions to each task. The *Cleaner*, *Mapper* and *Checker* modules contain new methods, all based on the interplay of different kinds of rules. The Cleaner rules are inspired to grammar-based transformations (where order matters), the Mapper rules are expressed in a Datalog-like formalism (order independent), the Checker rules are expressed as logical dependencies (order independent). Each rule-based method is provided with a formal description. The iterative process used in normalization and enrichment, where the designer's feedback is needed in order to validate or suggest annotations, has never been applied to genomic metadata. The significance of our approach stands in providing a single framework where the interplay of the three kinds of rules and of the effective interaction with the designer drives the whole process.

The META-BASE repository currently includes datasets from the ENCODE project ($\approx$ 21 million metadata $\langle key, value \rangle$ pairs in 26,111 items from 4 datasets), GDC (TCGA program, with $\approx$ 18M pairs, $\approx$ 100K items, 7 datasets), Roadmap Epigenomics ($\approx$ 200K pairs, $\approx$ 3K items, 6 datasets), Cistrome ($\approx$ 80K pairs, $\approx$ 6K items, 2 datasets), GENCODE and RefSeq annotations ($\approx$ 1.3K pairs, 105 items, 4 datasets), and TADs from GEO (272 pairs, 14 items, 2 datasets). In total, our framework has imported $\approx$ 40M $\langle key, value \rangle$ pairs, which correspond to $\approx$ 3.1K distinct keys. The addition of new sources to the repository is ongoing; our current commitment is to extend the data integration process by adding relevant sources one-by-one, in an incremental fashion. The resulting META-BASE repository is an important resource for supporting biological and clinical research.

Acknowledgments

This research is funded by the ERC Advanced Grant 693174 GeCo (data-driven Genomic Computing). The authors would like to thank the contributions of Alessandro Campi and of the master students Andrea Colombo, Federico Gatti, Riccardo Mologni, and Jorge Ignacio Vera Pena.

References

[1] F. Albrecht et al. Deepblue epigenomic data server: programmatic data retrieval and analysis of epigenome region sets. *Nucleic Acids Research*, 44(W1):W581–W586, 2016.

[2] T. Barrett et al. BioProject and BioSample databases at NCBI: facilitating capture and organization of metadata. *Nucleic Acids Research*, 40(D1):57–63, 2012.

[3] A. Bernasconi et al. Conceptual modeling for genomics: Building an integrated repository of open data. In H. C. Mayr, G. Guizzardi, H. Ma, and O. Pastor, editors, *Conceptual Modeling*, pages 325–339, Cham, 2017. Springer International Publishing.

[4] O. Bodenreider. Biomedical ontologies in action: role in knowledge management, data integration and decision support. *Yearbook of Medical Informatics*, page 67, 2008.

[5] S. B. Davidson et al. Biokleisli: A digital library for biomedical researchers. *International Journal on Digital Libraries*, 1(1):36–53, 1997.

[6] C. A. Davis et al. The encyclopedia of DNA elements (ENCODE): data portal update. *Nucleic Acids Research*, 46(D1):D794–D801, 2017.

[7] J. D. Fernandez et al. Ontology-based search of genomic metadata. *IEEE/ACM transactions on computational biology and bioinformatics*, 13(2):233–247, 2016.

[8] A. Frankish et al. GENCODE reference annotation for the human and mouse genomes. *Nucleic Acids Research*, 2018.

[9] C. B. Giles et al. ALE: Automated Label Extraction from GEO metadata. *BMC Bioinformatics*, 18(14):509, 2017.

[10] Gomez-Cabrero et al. Data integration in the era of omics: current and future challenges. *BMC Systems Biology*, 8(Suppl 2):I1, 2014.

[11] J. Grosjean et al. Health multi-terminology portal: a semantic added-value for patient safety. *Studies in Health Technology and Informatics*, 166:129, 2011.

[12] A. Gupta et al. Federated access to heterogeneous information resources in the Neuroscience Information Framework (NIF). *Neuroinformatics*, 6(3):205–217, 2008.

[13] E. L. Hong et al. Principles of metadata organization at the ENCODE data coordination center. *Database*, 1:10, 2016.

[14] M. A. Jensen et al. The NCI Genomic Data Commons as an engine for precision medicine. *Blood*, 130(4):453–459, 2017.

[15] C. Jonquet et al. A system for ontology-based annotation of biomedical data. In *International Workshop on Data Integration in The Life Sciences*, pages 144–152. Springer, 2008.

[16] S. Jupp et al. A new Ontology Lookup Service at EMBL-EBI. In J. Malone et al., editors, *Proceedings of SWAT4LS International Conference 2015*, pages 118–119, 2015.

[17] N. V. Kovalevskaya et al. DNAdigest and repositive: connecting the world of genomic data. *PLoS Biology*, 14(3):e1002418, 2016.

[18] A. Kundaje et al. Integrative analysis of 111 reference human epigenomes. *Nature*, 518(7539):317–330, 2015.

[19] J. Lonsdale et al. The Genotype-Tissue Expression (GTEx) project. *Nature Genetics*, 45(6):580, 2013.

[20] J. Malone et al. Ten simple rules for selecting a bio-ontology. *PLoS Computational Biology*, 12(2):e1004743, 2016.

[21] M. Martínez-Romero et al. Fast and accurate metadata authoring using ontology-based recommendations. In *AMIA Annual Symposium Proceedings*, volume 2017, page 1272. American Medical Informatics Association, 2017.

[22] M. Martínez-Romero et al. NCBO Ontology Recommender 2.0: an enhanced approach for biomedical ontology recommendation. *Journal of Biomedical Semantics*, 8(1):21, 2017.

[23] M. Masseroli et al. GenoMetric Query Language: a novel approach to large-scale genomic data management. *Bioinformatics*, 31(12):1881–1888, 2015.

[24] M. Masseroli et al. Modeling and interoperability of heterogeneous genomic big data for integrative processing and querying. *Methods*, 111:3–11, 2016.

[25] M. Masseroli et al. Processing of big heterogeneous genomic datasets for tertiary analysis of Next Generation Sequencing data. *Bioinformatics*, page bty688, 2018.

[26] S. Mei et al. Cistrome Data Browser: a data portal for ChIP-Seq and chromatin accessibility data in human and mouse. *Nucleic Acids Research*, 45(D1):D658–D662, 2016.

[27] A. Møller. dk.brics.automaton – finite-state automata and regular expressions for Java, 2017. `http://www.brics.dk/automaton/`.

[28] L. Nanni et al. Exploring genomic datasets: from batch to interactive and back. In *ExploreDB'18, Houston, TX, USA*, 2018.

[29] D. Oliveira et al. Where to search top-k biomedical ontologies? *Briefings in Bioinformatics*, page bby015, 2018.

[30] A. L. Palacio et al. A method to identify relevant genome data: Conceptual modeling for the medicine of precision. In *International Conference on Conceptual Modeling*, pages 597–609. Springer, 2018.

[31] N. W. Paton et al. Conceptual modelling of genomic information. *Bioinformatics*, 16(6):548–557, 2000.

[32] K. D. Pruitt, T. Tatusova, and D. R. Maglott. NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Research*, 35(suppl_1):D61–D65, 2006.

[33] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[34] S. S. Rao et al. A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. *Cell*, 159(7):1665–1680, 2014.

[35] Consortium 1000Genomes. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, 2010.

[36] Consortium ENCODE. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57–74, 2012.

[37] J. F. R. Román et al. Applying conceptual modeling to better understand the human genome. In *International Conference on Conceptual Modeling*, pages 404–412. Springer, 2016.

[38] S.-A. Sansone et al. DATS, the data tag suite to enable discoverability of datasets. *Scientific Data*, 4:170059, 2017.

[39] N. H. Shah et al. Ontology-driven indexing of public datasets for translational bioinformatics. *BMC Bioinformatics*, 10(2):S1, 2009.

[40] D. Smedley et al. The BioMart community portal: an innovative alternative to large, centralized data repositories. *Nucleic Acids Research*, 43(W1):589–598, 2015.

[41] K. Thompson. Programming techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422, 1968.

[42] L. Wang, A. Zhang, and M. Ramanathan. Biostar models of clinical and genomic data for biomedical data warehouse design. *International Journal of Bioinformatics Research and Applications*, 1(1):63–80, 2005.

[43] J. N. Weinstein et al. The Cancer Genome Atlas pan-cancer analysis project. *Nature Genetics*, 45(10):1113–1120, 2013.

[44] P. L. Whetzel et al. Bioportal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. *Nucleic Acids Research*, 39(suppl_2):W541–W545, 2011.

[45] M. D. Wilkinson et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3:160018, 2016.

Anna Bernasconi earned her Masters in Computer Engineering in 2015 from Politecnico di Milano and University of Illinois at Chicago. Currently PhD candidate at Politecnico di Milano, she works in the field of data-driven genomic computing. Her research interests include bioinformatics data and metadata integration methodologies to support complex biological queries answering. She focuses on open data repositories, biomedical vocabularies and ontologies.

Arif Canakoglu received his PhD Degree in Computer Engineering in 2016 from Politecnico di Milano and he is currently a Post-Doc fellow at Politecnico di Milano, working in the field of data-driven genomic computing and machine learning applications in the bioinformatics area. His research interests include databases, ontologies, big data processing, cloud computing, machine learning, and bioinformatics.

Marco Masseroli is Associate Professor at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) of Politecnico di Milano, Italy. His research interests are in the area of bioinformatics, focused on distributed Internet technologies, biomolecular databases and ontologies to effectively retrieve, analyze, and integrate genomic information with clinical and genomic data. He is the author of more than 200 publications in international journals, books and conference proceedings.

Stefano Ceri is Professor at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) of Politecnico di Milano. His research has been generally concerned with extending database technology; he has authored over 350 publications. He received two advanced ERC Grants, on Search Computing and on Data-Driven Genomic Computing (GeCo, 2016-2021). He received the ACM-SIGMOD Innovation Award (2013) and is an ACM Fellow.