

# Analyzing Genomic Mappings with the GOLAM Framework<sup>\*</sup>

Lorenzo Baldacci, Matteo Golfarelli, Simone Graziani, and Stefano Rizzi

DISI – University of Bologna,  
V.le Risorgimento 2, 40136 Bologna, Italy  
{lorenzo.baldacci|matteo.golfarelli|simone.graziani2|stefano.rizzi}@unibo.it

**Abstract.** The huge amount of data made available by the technologies for sequencing the human genome calls for sophisticated and automated analysis techniques. In this direction, the GOLAM framework has been developed within the scope of the GenData 2020 Project to enable multi-resolution analysis and mining of matches between genomic regions. In this paper we show how the GOLAM framework can be used for OLAP analysis of the mappings between regions and samples resulting from queries expressed in GMQL against repositories of genomic data. In particular we show the multidimensional schema that represents mappings and we briefly address the main research challenge implied by GOLAM, i.e., on-demand ETL.

**Keywords:** OLAP, ETL, Genomics, Data Warehouse

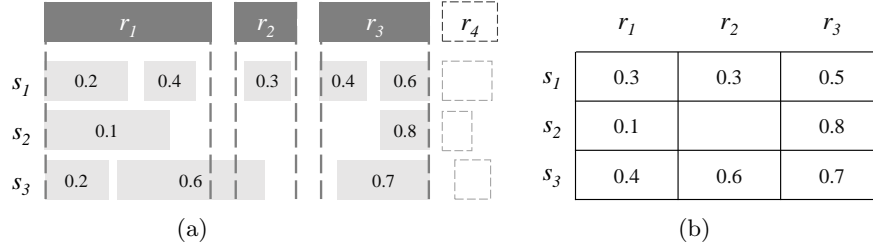
## 1 Introduction

High-throughput technologies that enable rapid, intelligent, and parallel acquisition of experimental genomic data are quite mature, and sequencing and genetic screening are becoming faster and cheaper every year, trending towards even more promising improvements. An individual’s full genome sequencing takes little time (1-10 days) and very little money in an average laboratory. Such speed and low cost of sequencing have led to an explosion in the rate of genomic data generated, therefore the research challenges are moving towards the extraction of biological meaning from this mass of data on the one hand, the storage, retrieval, security, and presentation of information to biologists on the other.

This is the context for the GenData 2020 Project, in whose framework our work is being developed. GenData 2020 aims at managing genomic data through an integrated data model, expressing the various features that are embedded in the produced bio-molecular data, or in their correlated phenotypic data that can be extracted from clinical databases, or in the information inferred by applying data analysis to them. This goal is achieved by enabling viewing, searching, querying, and analyzing over a worldwide-available collection of shared genomic

---

<sup>\*</sup> Partially supported by the GenData 2020 PRIN Project (<http://gendata.weebly.com>) funded by the Italian Ministry of Education, Universities, and Research.



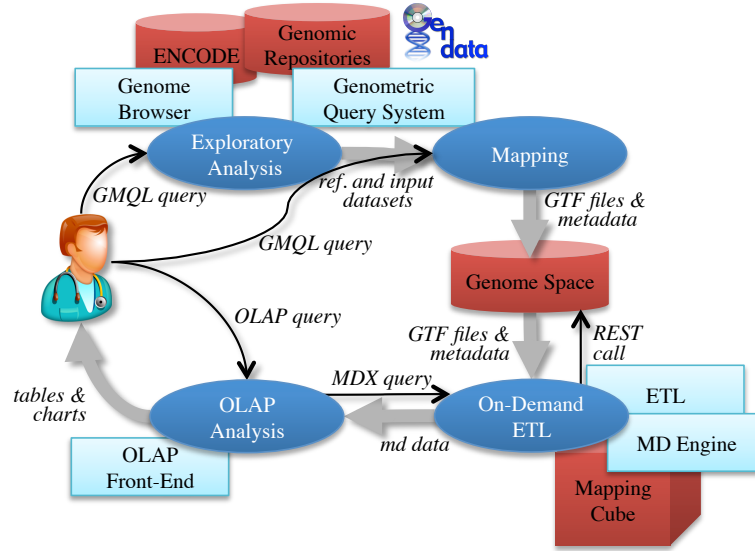
**Fig. 1.** Mapping between reference regions  $r_1, r_2, r_3$  and input samples  $s_1, s_2, s_3$  (a), and resulting genome space where each cell is computed as the average of the corresponding region values (b)

data. The reference framework for storing and querying data in GenData is Hadoop; in particular, an ad hoc language called GMQL (GenoMetric Query Language [8]) based on Pig Latin has been devised for managing genomic data.

One of the analysis services envisioned in GenData 2020 is related to multi-resolution analysis of matches between genomic *regions* (i.e., segments of the genome, represented as grey boxes in Figure 1.a), in which the user (typically, a biologist or a geneticist) knows the regulatory mechanisms of a gene and wants to discover other genes with similar regulation mechanisms. To support this need, in [2] we presented *GOLAM* (Genome On-Line Analytical Mining), a framework in which the detection of similar genome areas is automated using similarity search techniques, and the resulting matches are analyzed using OLAP and mining techniques on an RDBMS. The reason why we chose to use a “traditional” DBMS for OLAP is that, though in principle OLAP front-ends and Hadoop-based query engines can be coupled, none of the engines in the Hadoop ecosystem present the features necessary to efficiently support OLAP analysis since they either work in a batch fashion (e.g., Hive) or they do not allow efficient select-anywhere capabilities. Recently, some new engines (e.g., Cloudera Impala) are trying to fill the gap between Big Data and OLAP and some papers investigated the related research issues [3, 11, 14]. Nonetheless the Hadoop ecosystem is still mainly used for storing operational data, computing analytics, and executing ETL tasks aimed at extracting multidimensional data to feed traditional OLAP systems.

In this work we discuss a different use of the GOLAM framework, aimed at multi-resolution analysis of the mappings between regions and *samples* (i.e., sets of regions and correlated metadata resulting from an experiment). Mappings are computed by issuing a GMQL query based on the *Map* operation against some repositories of genomic data such as ENCODE,<sup>1</sup> and represent a powerful and biologically meaningful way to relate experimental data to known annotations

<sup>1</sup> ENCODE, the *Encyclopedia of DNA Elements*, is a public repository (accessible via ftp) created and maintained by the US National Human Genome Research Institute to identify and describe the regions of the 3 billions base-pair human genome that are important for different kinds of functions [12].



**Fig. 2.** The GOLAM approach

[8]. Figure 1 shows an example of a Map operation and the resulting genome space. As sketched in Figure 2, mappings are computed after an exploratory analysis conducted by a biologist; the query output, called *genome space*, comes in the form of a set of GTF (*Gene Transfer Format*, <http://genome.ucsc.edu>) files and related metadata, and is stored on the Hadoop platform.

Unfortunately, the genome space generated by most biologically-relevant queries is too large (up to hundreds of millions of elements) to enable a traditional ETL process to load it into a multidimensional mapping cube on an RDBMS for OLAP analysis. To solve this problem, GOLAM relies on a process called *on-demand ETL*, which incrementally extracts data from the genome space, transforms them into multidimensional form, and loads them into the mapping cube *as these data are required to answer OLAP queries*.

In this paper we give an overview of how genome mappings are analyzed within the GOLAM framework. First we discuss the related literature in Section 2, then in Section 3 we describe the main components of GOLAM with an emphasis on on-demand ETL. Section 4 draws the conclusions.

## 2 Related Work

Although some publications advise the usage of OLAP as a tool for information discovering, only in a few cases it has actually been employed in the biological domain. In [1] the authors used OLAP technologies to analyze gene expression data, gathered from tests done on soybean roots, with the purpose of discovering

information to develop soybean cultivars resistant to the SCN pest. In [4], important correlations between deletion patterns in the chromosome Y and patient populations have been found through the application of OLAP and data mining techniques. In both studies, the fact of interest differs from ours; while we focus on approximate genome area matches, [1] uses gene expression data—the result of DNA microarray experiments—and [4] applies OLAP analysis to the result of clustering performed on deletion patterns in the genome.

As to conceptual and logical modeling of data warehouses for genomic data, the authors of [7] propose models for three gene expression data spaces (*sample*, *annotation*, and *gene expression*) based on star and snowflake schemata. A more exhaustive study of the challenges and requirements of a biomedical multidimensional model is presented in [13], along with a new schema called *BioStar*. This last work in particular differs from ours because we propose a true multidimensional data model based on ENCODE and GenData, while BioStar is a general framework to be used as a way to model various types of biomedical data.

Finally, the problem of on-demand ETL is discussed and motivated in the area of scientific data in [6]. An approach similar to ours, named *lazy ETL*, is roughly described but no details are provided about the subsumed architecture, the way the data to be loaded are identified, and the way extraction is optimized. Furthermore, the lazy-ETL process can only include SQL queries, and all the dimension data must be loaded in advance. The need for a pay-as-you-go model for data loading emerges when data-enthusiasts and data scientists are involved [10]. These use cases ask for a more flexible, and possibly on-demand, definition of the ETL process. Our solution addresses the on-demand requirement assuming that one or more wrappers are available for extracting data or that they can be defined on-the-fly as well.

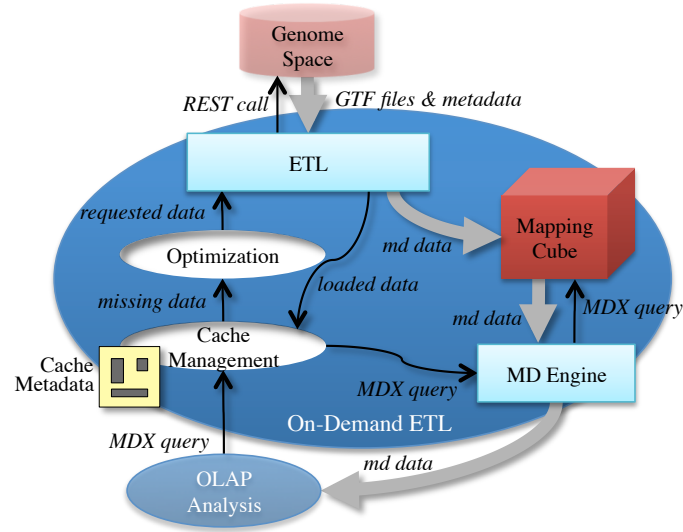
### 3 The GOLAM Framework

In this section we present the GOLAM framework for OLAP analyses of mappings between genome regions resulting from biological experiments. With reference to Figure 2, the main phases and components of the framework can be described as follows; the whole workflow is iterative, so as to give biologists the possibility of dynamically changing or refining their mapping query, and compare the new results with those obtained at previous iterations.

1. **Exploratory Analysis.** During this phase the biologist explores, manipulates, and transforms the data in the genomic repositories aimed at determining sequences of regions of interest within a chromosome. This task can be carried out using a genome browser, and is facilitated by the GMQL language that provides a set of operators to filter, aggregate, and compare datasets (typically based on predicates expressed on metadata such as the patients sex and phenotype, or the tissue and chromosome from which the region was extracted). The goal is to prepare two datasets—*reference* and *input*, respectively—on which the GMQL Map operation will be executed. The reference dataset includes a set of regions belonging to a single sample,

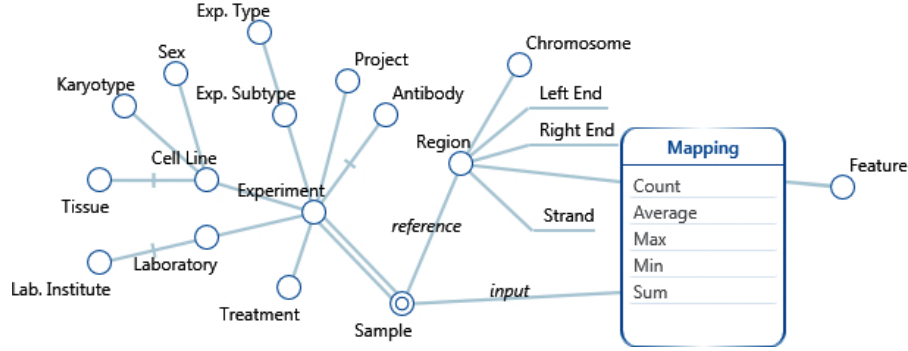
while the input dataset normally includes several samples. The initial selection of both datasets is driven by a biological enquiry and is based on prior expert knowledge; however, some further iterations are often necessary to better focus the search and obtain more meaningful results.

2. **Mapping.** The Map operation performs an overlap between the reference and the input datasets. An example of query based on Map is the one that maps the *peaks* (enriched regions) of each input sample to the *promoter* regions (a promoter is a DNA region that initiates the transcription of a particular gene) and, for each promoter, counts how many of such peaks co-localize with the promoter, saving the obtained results. The output of Map is a new dataset, called genome space, where each couple of a reference region  $r$  and an input sample  $s$  is associated to a set of numerical values each obtained by applying an aggregate function (e.g., the average) to a specific feature (e.g., the p-value) of the regions of  $s$  that overlap with  $r$ . Thus, a single mapping in the genome space is defined by a reference region, an input sample, the selected aggregate function, and the region feature used to aggregate. In practice, the genome space resulting from a Map operation is materialized as a set of GTF and metadata files —precisely, one GTF and one metadata file for each input sample. Note that, though using GMQL it is possible to materialize just a subset of the mappings of a genome space, the mappings related to a single input sample are necessarily contained in exactly one GTF file.
3. **On-Demand ETL.** This phase aims at incrementally loading the mappings belonging to the genome space, together with the related metadata, on a multidimensional cube. When a user formulates an OLAP query  $q$ , the front-end translates  $q$  into MDX [9] and sends it to the on-demand ETL component for processing. If all the multidimensional data necessary to answer  $q$  are already present in the mapping cube (i.e., they have been previously loaded), then they are sent to the front-end and shown to the user. Otherwise, the genome space is accessed via a REST call to fetch all the missing data, that are then put in multidimensional form and loaded on the mapping cube, so that  $q$  can be answered as above. Of course, from time to time, some data used for past OLAP queries must be dropped from the cube to make room for the data needed for new queries. The functional architecture of the on-demand ETL process is sketched in Figure 3. It includes the following subprocesses:
  - **Cache Management** takes the OLAP query  $q$  (received in MDX form from the OLAP front-end) and checks, using *cache metadata* that keep track of the data currently available in the cube, if  $q$  can be immediately answered or some data are missing. In the first case,  $q$  is sent to the multidimensional engine for processing. In the second case, the difference between the data required by  $q$  and the available data is computed (missing data) and handed to the optimization subprocess. The data currently stored in the cube, those required to answer  $q$ , and those missing are compactly represented as multidimensional intervals of coordinates.



**Fig. 3.** Functional breakdown of the on-demand ETL process

- **Optimization.** The interface for accessing the genome space is REST-based, and the extraction of missing data is done on a file basis. As previously mentioned, extracting *exactly* the mappings required by the OLAP query would be feasible, but it would require only a subset of the Map operation result to be materialized—which would add an extra processing cost to each OLAP query. So, the approach we are currently pursuing in GOLAM is to materialize the full genome space when the Map operation is executed, and then extract the GTF files that cover all the missing data needed to answer each OLAP query. In this scenario, the finest granularity offered for fetching data is the set of mappings of all the regions of a single reference sample against a single input sample (i.e., a selection on a subset of reference regions cannot be made when fetching). Based on this constraint and knowing the cost of each REST call in terms of data to be transferred, the optimization subprocess determines, through an approach based on hierarchical clustering, a set of requested data that cover all the missing data and has total minimum cost. The requested data are then fetched from the genome space and transformed into multidimensional form.
- 4. **OLAP Analysis.** Analysis sessions in biology are inherently exploratory and dynamic, so an OLAP approach is well suited since it provides powerful and flexible tools to explore data from different perspectives and levels of detail. So, at this stage, biologists can analyze mappings using classical OLAP operators such as roll-up, slice-and-dice, etc. Obviously, a primary role in ensuring the effectiveness of OLAP analysis is played by the multi-



**Fig. 4.** Multidimensional schema for the mapping cube

dimensional schema adopted for representing data. Figure 4 shows the schema we use to model mappings, based on the DFM notation [5]. Each mapping is associated to one region of a reference sample and one input sample, and is described by a set of statistical measures —count, average, max, min, and sum. Notably, except count, the other measures are computed on a specific feature of the regions found in the input sample. Since such features are not known a-priori, we use the **Feature** dimension to support a variable number of measures. For instance, measure **Max** of a mapping associated with feature p-value refers to the maximum p-value found in the mapped regions. As denoted in Figure 4 by a circle with a double border, reference and input samples share the same hierarchy. The **Sample** hierarchy mostly reflects the ENCODE data model [2]. An *experiment* is the result of a single biological analysis over a portion —called *cell line*— of a *tissue* extracted from a patient. In particular, the meaning of an experiment depends on its specific type (e.g., ChIP-Seq, DNA-Seq, etc.). Samples, which are the necessary input for a Map operation, are the result of one (or more) GMQL operations over a set of experiments.

As mentioned above, the size of the data being processed may be significant. To give an idea of the data volumes, we remark that a typical sample might include some hundreds of thousands of regions, while a Map operations usually takes one reference sample to be mapped against (up to) a couple of thousands of input samples. Since the Map operation maps each region of a reference sample against each input sample, the resulting set of mappings might amount to hundreds of millions of elements.

## 4 Final Remarks

The goal of the GOLAM framework is to address a problem in modern genomics, that of overcoming the current limitations of genome analysis methods by partially automating and speeding-up the analysis process on the one hand, and by

making it more flexible on the other. In this direction we presented the GOLAM framework, whose goal is to enable biologists to run multi-resolution analyses on genome region mappings. With reference to Figure 2, the OLAP Analysis component has been developed using open source tools; in particular, MySQL, Mondrian, and Saiku are used as RDBMS, multidimensional engine, and OLAP front-end, respectively. As to on-demand ETL, Talend Open Studio implements basic ETL (see Figure 3); the cache management subprocess has already been implemented in C++, while optimization is currently being implemented.

## References

1. Alkharouf, N., Jamison, C., Matthews, B.: Online analytical processing (OLAP): a fast and effective data mining tool for gene expression databases. *BioMed Research International* 2005(2), 181–188 (2005)
2. Baldacci, L., Golfarelli, M., Graziani, S., Rizzi, S.: GOLAM: a framework for analyzing genomic data. In: *Proc. DOLAP*. pp. 3–12. Shanghai, China (2014)
3. Cuzzocrea, A., Song, I.Y., Davis, K.C.: Analytics over large-scale multidimensional data: the big data revolution! In: *Proc. DOLAP*. pp. 101–104 (2011)
4. Dzeroski, S., Hristovski, D., Peterlin, B.: Using data mining and OLAP to discover patterns in a database of patients with Y-chromosome deletions. In: *Proc. AMIA*. pp. 215–219 (2000)
5. Golfarelli, M., Rizzi, S.: *Data Warehouse design: Modern principles and methodologies*. McGraw-Hill (2009)
6. Kargin, Y., Pirk, H., Ivanova, M., Manegold, S., Kersten, M.L.: Instant-on scientific data warehouses: Lazy ETL for data-intensive research. In: *Proc. BIRTE*. pp. 60–75. Istanbul, Turkey (2012)
7. Markowitz, V., Topaloglou, T.: Applying data warehouse concepts to gene expression data management. In: *Proc. BIBE*. pp. 65–72. Bethesda, Maryland (2001)
8. Masseroli, M., Pinoli, P., Venco, F., Kaitoua, A., Jalili, V., Paluzzi, F., Muller, H., Ceri, S.: Genometric query language: A novel approach to large-scale genomic data management. *Bioinformatics*, to appear (2015)
9. Microsoft: MDX reference. <http://msdn.microsoft.com/en-us/library/ms145506.aspx> (2015)
10. Morton, K., Balazinska, M., Grossman, D., Mackinlay, J.: Support the data enthusiast: Challenges for next-generation data-analysis systems. *Proc. VLDB* 7(6) (2014)
11. Moussa, R.: TPC-H benchmark analytics scenarios and performances on Hadoop data clouds. In: *Networked Digital Technologies*, pp. 220–234. Springer (2012)
12. Raney, B., Cline, M., Rosenbloom, K., Dreszer, T., Learned, K., Barber, G., Meyer, L., Sloan, C., Malladi, V., Roskin, K., Suh, B., Hinrichs, A., Clawson, H., Zweig, A., Kirkup, V., Fujita, P., Rhead, B., Smith, K., Pohl, A., Kuhn, R., Karolchik, D., Haussler, D., Kent, J.: ENCODE whole-genome data in the UCSC genome browser (2011 update). *Nucleic Acids Res.* (39), D871–D875 (2011)
13. Wang, L., Zhang, A., Ramanathan, M.: BioStar models of clinical and genomic data for biomedical data warehouse design. *International Journal of Bioinformatics Research and Applications* 1(1), 63–80 (2005)
14. Zhao, H., Ye, X.: A practice of TPC-DS multidimensional implementation on NoSQL database systems. In: *Performance Characterization and Benchmarking*, pp. 93–108. Springer (2014)