*Briefings in Bioinformatics*
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Subject Section

# Federated sharing and processing of genomic datasets for tertiary data analysis

**Arif Canakoglu*, Pietro Pinoli*, Andrea Gulino, Luca Nanni, Marco Masseroli, Stefano Ceri**

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy.

*Co-primary and corresponding authors..

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

## Abstract

**Motivation:** With the spreading of biological and clinical uses of Next-Generation Sequencing (NGS) data, many laboratories and health organizations are facing the need of sharing NGS data resources, and easily accessing and processing comprehensively shared genomic data; in most cases, primary and secondary data management of NGS data is done at sequencing stations, and sharing applies to processed data. Based on the previous single-instance GMQL system architecture, here we review the model, language, and architectural extensions that make the GMQL centralized system innovatively open to federated computing.
**Results:** A well-designed extension of a centralized system architecture to support federated data sharing and query processing. Data is federated thanks to simple data sharing instructions. Queries are assigned to execution nodes; they are translated into an intermediate representation, whose computation drives data and processing distributions. The approach allows writing federated applications according to classical styles: centralized, distributed, or externalized.
**Availability:** The federated genomic data management system is freely available for non-commercial use as an open source project at: *http://www.bioinformatics.deib.polimi.it/FederatedGMQLsystem/*
**Contact:** {arif.canakoglu, pietro.pinoli}@polimi.it

**Author biographies:**
**Arif Canakoglu** is Postdoctoral Fellow at Politecnico di Milano. His research interests include data integration, data-driven genomic computing, big data analysis and processing on cloud and artificial intelligence applications of genomic data.
**Pietro Pinoli** is a Research Assistant in Computational Biology at Politecnico di Milano. His research activity focuses on the management, modeling and analysis of omics data.
**Andrea Gulino** is a Ph.D. student in Computer Science and Engineering at Politecnico di Milano. His research interests include cloud computing and system architectures, with specific focus on optimisation.
**Luca Nanni** is a Ph.D. candidate in Computer Science and Engineering at Politecnico di Milano. His research interests include genome organization, machine learning and interactive data analysis of genomic data through cloud computing.
**Marco Masseroli** is Associate Professor at Politecnico di Milano. He is the author of more than 200 scientific articles, which have appeared in international journals, books and conference proceedings.
**Stefano Ceri** is Professor at Politecnico di Milano. He authored over 350 publications (H-index 78) and 15 books in English. He received two advanced ERC Grants and the ACM-SIGMOD Innovation Award.

**summary:**

- The growth of genomics data over the last decade has been astonishing with many independent consortia and institutes producing and releasing genomics data;
- we review Federated GMQL, a system for querying distributed genomics datasets across many instance connected through the Web;
- with respect to its competitors it provides a more complete query language;
- we review advanced feature of the system that allows to automatically distribute the computation while preserving privacy constraints.

## 1 Introduction

The rate of growth of genomic data over the last decade has been astonishing, with the total amount of produced sequence data doubling approximately every seven months [1]. Many international consortia are involved in coordinated efforts for collecting data through principled processes and for providing world-wide open access to them, including the Encyclopedia of DNA elements (ENCODE, [2]), The Cancer Genome Atlas (TCGA, [3]), and Roadmap Epigenomics [4]; meanwhile, important large-scale sequencing projects are conducted at the national level, including the 100,000 Genomes Project in UK [5], the FinnGen project in Finland [6], and the AllOfUs effort of NIH in the United States [7]. Availability of very large, well-organized datasets has the potential of boosting biological and clinical research, provided that resources are easily accessible in a uniform and integrated manner. The recently presented GenoMetric Query Language (GMQL) [8, 9] is a language and data processing system for supporting queries over big heterogeneous datasets. GMQL is focused on tertiary genomic data management, i.e., querying data describing genomic regions and their metadata – such as genome annotations, variants, gene and miRNA expression levels, peaks of DNA methylation and transcription factor bindings, copy number alterations, chromatin states and so on; instead, GMQL is not concerned with primary or secondary genomic data management, whose focus is on genomic region extraction from raw data as effect of alignment algorithms and of algorithms that identify (call) relevant regions of the genome. The query language takes advantage of interoperability provided by the Genomic Data Model [10]; the data management system maps high-level queries for extracting and manipulating genomic regions and their metadata into Apache Spark queries [9], efficiently executed on arbitrary servers and clusters.

We hereby review the novel Federated GMQL, an advanced technology for interconnecting and querying several computing sites, each running an instance of the GMQL system. The proposed technology is supported by light-weight administration protocols: a new GMQL site can join an existing federation by involving only a subset of its resources, through agreements that do not require the intervention of a global administrator. Groups of sites can be formed arbitrarily in order to effectively assign accessibility rules to their resources, and data protection rules can protect sensitive data from being transferred to other sites due to the choice of query processing strategy.

This technology can be very useful in a variety of emerging application contexts. Most institutions, who are newcomers in genomic computing, today invest most of their resources on mastering primary and secondary data management pipelines at their premises, but we expect them to move soon to tertiary data management in a collaborative style, sharing their results within collaborations and comparing them to a wealth of open access information already available. Teams of professionals who share genomic data within biological or clinical projects can easily build a GMQL federation, typically in the context of funded research both at regional and international level. GMQL instances providing access to integrated resources can be added to federated databases – one is the GMQL site at CINECA, a Supercomputing Center of Italian Universities, providing access to the GenoSurf collection, which integrates processed data from ENCODE, TCGA, Roadmap Epigenomics and other sources [11].

### 1.1 Federated genomic data management system comparison

Federated data management is a widely used and overloaded term in the context of genomic computing, so the first objective of this Section is to provide a precise definition of its meaning in the scope of our work and useful to identify other related projects.

A large body of research uses the term *federated database* to denote semantic data integration efforts [12–15]. In this context, federation occurs among heterogeneous data stores, e.g., describing gene expressions and gene or protein sequences, typically driven by ontological mediation [12]. The main focus of these systems is the ontological mapping between terms used in different contexts, which is helped by linked data and their semantic relationships [14].

In other cases, the term federated databases is used to denote a technological framework for speeding up computations by splitting a query workload to several execution engines. Among them, [16] applies a data partitioning scheme across database clusters for speeding up similarity search across 3D models of proteins. The Top-Fed TCGA project [17] transforms the TCGA data into the Resource Description Framework (RDF) format, links it to relevant datasets in the Linked Open Data (LOD) cloud, and further proposes an efficient data distribution strategy to host the resulting 20.4 billion triple data via several SPARQL endpoints.

Federated GMQL is not focused either on semantic data heterogeneity or on execution engine optimization; each of these aspects is addressed in previous GMQL work. Specifically, interoperability of region data is guaranteed by the adoption of the Genomic Data Model in GMQL [10], while interoperability of metadata descends from the use of a common conceptual model for genomic sources [18], resulting in the GenoSurf repository [11]. Effective parallel execution of GMQL queries is guaranteed by several physical optimizations, including [19] and [20].

Therefore, Federated GMQL restricts the interpretation of a federated system to its essence: a technology for supporting queries over several GMQL instances, each managed by an independent organization and supporting a local data storage. This technology provides a step forward in the wide spreading of distributed data management technology for tertiary genomic data management, seen as a much needed future evolution of genomic computing [21, 22].

According to the above interpretation, the most relevant project on federated biodata management is BioMart [23], a community-driven effort to provide unified access to worldwide distributed biomedical databases. According to the BioMart paradigm, each data source is managed, updated and released independently, but the project provides a data-agnostic, cross-platform collection of graphical and programmatic interfaces for presenting a unified view of the sources, so that they appear to be a single, integrated database. A technical description of BioMart is provided in [24]. The BioMart implementation is based upon Structured Query Language (SQL) and supported by several SQL engines, including MySQL, PostgreSQL, Oracle, DB2 and MS SQL. The main difference of Federated GMQL with BioMart is that the former exposes the full power of GMQL, a high-level declarative language allowing the expression of queries over genomic regions and metadata, and is implemented on Spark, which scales better than relational databases on the cloud [25], [26].

The general need for responsible data sharing is emphasized in several documents of the Global Alliance for Genomics and Health (*https://www.ga4gh.org/*); they recall the *1948 Universal Declaration of Human Rights* that highlights not only data protection but also the right to benefit from the fruits of scientific and medical advances by virtue of having access to larger data sets. This international organization provides several technical solutions for data interoperability, including workflow execution services and best practices. Among them, the *Beacon Project* [27] is an open technical specification for sharing genetic variant data, supporting queries asking for the presence or absence of a specific allele. The *BRCA Exchange* [28] is a publicly accessible Web portal that provides a simple interface for patients, clinicians, and researchers to access curated, expert interpretations of BRCA1/2 genetic variants. The CanDIG project [29], funded by the health department of Canada within the Global Alliance framework, emphasizes differential privacy and the possibility of analyzing datasets without exposing sensitive individual data. Researchers at participating sites can analyze national datasets while keeping data private and under local control.

## 2 Modelling federated data and queries

### 2.1 Preliminaries

The Genomic Data Model (GDM) [10], used by the GMQL system, is based on the notion of genomic region; it mediates existing data formats, and covers also metadata of arbitrary structure. Thanks to these features, GDM is capable of supporting data interoperability, by modeling semantically heterogeneous data. Datasets are the unit of data definition in GDM; datasets consist of several samples, each corresponding to a pair of files respectively storing genomic region data and metadata of a given experimental condition or patient.

So far, GMQL has been available for download and use on a single instance (either a server or a cluster); installation provides a GMQL engine connected to a data repository, and query processing occurs at the installed server. Such a system (engine + repository) is named a *GMQL instance*. In each such a system, account management is under the responsibility of a single data administration authority (called ADMIN). *PRIVATE* accounts are login-protected and can manage their own datasets, which are not visible from other accounts and are stored in a private personal space of the system. In addition, a GMQL instance can support *GUEST* accounts. Datasets are either PRIVATE or PUBLIC; the former ones are created or deleted by individual users, the latter ones are managed by the GMQL instance administrator, who is responsible for their creation and deletion; also guest accounts can access public datasets. Several versions of the same public dataset may be kept, each featuring a timestamp added to the dataset name (so as to guarantee a consistent view of the datasets used within a project).

We next review and discuss the enhancements made to the GDM model and GMQL language to support data federation. Model and query language extensions are designed aiming at minimal syntactic changes; yet, they enable writing applications according to a variety of options, which are explored in the example application Section 4.

### 2.2 Federated GDM

In *Federated GDM*, a subset of the PUBLIC datasets in a GMQL instance is upgraded by the instance repository administrator and declared as FEDERATED; federated datasets may become visible to other GMQL instances.

Federated GMQL repositories are supported by means of data sharing agreements. They are facilitated by *GMQL groups*, named collections of GMQL instances, each controlled by an administrator, who can dynamically add (or drop) GMQL instances to (from) the group, and make agreements about the group; GMQL-ALL is a group including all GMQL instances currently interconnected.

Data sharing agreements refer to a target dataset, which is set to *federated status* at a given GMQL instance (referred as the *owner instance*) by the instance administrator; every agreement is established with a recipient administrator, either of another GMQL instance or of a GMQL group. As effect of the agreement, the target dataset becomes visible to all private users of the recipient GMQL instance or group. When a new GMQL instance is added to a GMQL group, all private users of the new GMQL instance immediately benefit of all the agreements that the group administrator has done in the past.

An example of Federated GDM is shown in Figure 1. The federated system consists of four GMQL instances, respectively called CINECA (located at CINECA in Bologna, IT), DEIB and GeCo (located at Politecnico di Milano, in Milano, IT), and IMN (located at Mario Negri Institute in Milano, IT). Group-A includes CINECA and GeCo, Group-B includes IMN and GeCo. DEIB has four datasets: DS1 is private, DS2 is public but not federated, DS3 and DS4 are public and federated, with DS4 located at CINECA. DS5 is private at CINECA, DS6 is public and federated at GeCo. This situation could be generated by the next sequence of agreements:

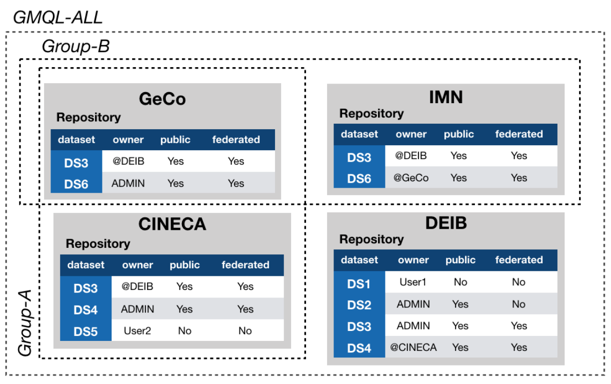
- DS3 is owned by DEIB and then shared with all the instances of the GMQL-ALL group. This produces the effect shown in Figure 1, where DS3 is marked as federated at all sites.
- DS4 is owned by CINECA and then shared with DEIB. This produces the effect shown in Figure 1, where DS4 is marked as federated at CINECA and DEIB.
- DS6 is owned at GeCo and then shared within Group-B. This produces the effect shown in Figure 1, where DS6 is marked as federated at all nodes of Group-B, i.e., IMN and GeCo.

### 2.3 Federated GMQL

GMQL is a language consisting of algebraic operations that apply to GDM datasets; a *Federated GMQL query* adds to the GMQL code tags that indicate where each operation is executed. Consider a simple example GMQL query consisting of four operations: the first one selects an experiment dataset, the second one selects RefSeq genes from an annotation dataset, the third one joins the two selected datasets and keeps only the genomic regions of the first dataset that intersect at least one genomic region of the second dataset, and the fourth one materializes the result. The GMQL code of this query is:

```
myData = SELECT() Experiments;
genes = SELECT(annotation_type == "gene" AND
               provider = "RefSeq"
              ) HG19_BED_ANNOTATIONS;
onGenes = JOIN(distance < 0;
               output: right) genes myData;
MATERIALIZE onGenes INTO result;
```

The following is a federated version of the same query, where Experiments is a private dataset, available at DEIB, and RefSeq gene annotations are in a federated dataset available at GeCo. The join operation occurs at CINECA. The query is launched by the private dataset owner, at DEIB. In all federated queries, results are returned to the user who initiates the query; therefore, the MATERIALIZE operation also occurs at DEIB.



GeCo

| dataset | owner | public | federated |
|---|---|---|---|
| DS3 | @DEIB | Yes | Yes |
| DS6 | ADMIN | Yes | Yes |

IMN

| dataset | owner | public | federated |
|---|---|---|---|
| DS3 | @DEIB | Yes | Yes |
| DS6 | @GeCo | Yes | Yes |

CINECA

| dataset | owner | public | federated |
|---|---|---|---|
| DS3 | @DEIB | Yes | Yes |
| DS4 | ADMIN | Yes | Yes |
| DS5 | User2 | No | No |

DEIB

| dataset | owner | public | federated |
|---|---|---|---|
| DS1 | User1 | No | No |
| DS2 | ADMIN | Yes | No |
| DS3 | ADMIN | Yes | Yes |
| DS4 | @CINECA | Yes | Yes |

**Fig. 1.** Example Federated GMQL system, with four GMQL instances and two groups; for each GMQL instance, we show visible datasets.

```
myData = SELECT(at:DEIB) Experiments;
genes = SELECT(annotation_type == "gene" AND
               provider = "RefSeq";
               at:GeCo) HG19_BED_ANNOTATIONS;
onGenes = JOIN(distance < 0;
               output: right;
               at:CINECA) genes myData;
MATERIALIZE onGenes INTO result;
```

By contrasting the latter query with the previous one, note that in Federated GMQL each operation is allocated to a given GMQL instance. The allocation of every SELECT operation is forced at the location storing the dataset to which the SELECT applies, where it performs also the data loading into the server's memory from the files. All other operations over any accessible federated dataset can be executed at any federated GMQL instance; the tagging of execution GMQL instances of the operations prescribes also the data transfer in case required. In the above example there are three possibilities:

1. If the JOIN is allocated at `CINECA`, as in the above example, experiment data are moved from `DEIB` to `CINECA`, annotations are moved from `GeCo` to `CINECA`, and results from `CINECA` to `DEIB`.
2. If the JOIN is instead allocated at `GeCo`, experiment data are moved from `DEIB` to `GeCo`, and then the result of the JOIN is moved back to `DEIB`.
3. Finally, if the JOIN is instead allocated at `DEIB`, just annotations are moved to `DEIB` and all processing takes place at that instance.

### 2.4 Execution policies and data protection

Federated GMQL supports default policies for generating distributed query executions without asking users to indicate the execution site of each operation. Policies can be specified as compiler directives prior to a query, as follows:

```
@policy <policy_name> [<parameter>]
```

The indication of a specific allocation site for any operation overrides the assignment generated by the policy. In addition, it is possible to protect datasets, by imposing that no query execution strategy will move them to a different site. Protections can also be specified as compiler directives, as follows:

```
@protected <dataset_name>
```

The effect of query execution policies and of dataset protection is explained in Section 4.

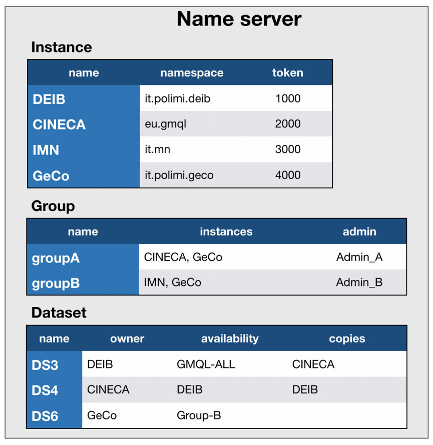
## 3 Executing federated queries

### 3.1 Name server

A GMQL federation hosts a unique *Name server*, implemented as a REST API. Figure 2 shows the information available at the Name server to reflect the construction of the federated datasets and groups discussed in Section 2.2. Every GMQL instance must be registered at the Name server; the Name server registry stores its `NAME`, its `NAMESPACE` (which uses a reverse-DNS notation, e.g., `it.polimi.deib`) and a `TOKEN` for authentication with the Name server, carried at each Name server request.

Every group is also registered at the Name server, listing the group `NAME`, the GMQL `INSTANCES` that constitute the group, and the identity `ADMIN` of the group administrator, so as to facilitate multilateral agreements.

After the establishment of an agreement, any recipient can also store a physical copy of the target dataset; datasets are read-only, therefore the


Name server

Instance

| name | namespace | token |
|---|---|---|
| DEIB | it.polimi.deib | 1000 |
| CINECA | eu.gmql | 2000 |
| IMN | it.mn | 3000 |
| GeCo | it.polimi.geco | 4000 |

Group

| name | instances | admin |
|---|---|---|
| groupA | CINECA, GeCo | Admin_A |
| groupB | IMN, GeCo | Admin_B |

Dataset

| name | owner | availability | copies |
|---|---|---|---|
| DS3 | DEIB | GMQL-ALL | CINECA |
| DS4 | CINECA | DEIB | DEIB |
| DS6 | GeCo | Group-B | |

**Fig. 2.** Description of the example Federated GMQL system at the Name server.

copy will remain identical to the target until it is deleted. The owner GMQL instance is responsible for the target dataset; therefore, its administrator must be informed of any copy creation or deletion. Physical copies enable a user to write several equivalent versions of the same query, by deciding the site where the dataset will be accessed, with potential different performances. Moreover, data replication enhances its availability; hence, it is commended that each dataset has at least one physical copy.

Every federated dataset has a dedicated Name server entry, carrying its `NAME`, the `OWNER` instance, the `AVAILABILITY` to GMQL instances and groups, and the GMQL instances storing its `COPIES`, if any. Read or write accesses to the Name server are performed by administrators. The federation authority manages GMQL instances, group administrators manage groups, dataset administrators manage the federated datasets.

Interaction with the Name server is required by each federated GMQL instance in order to support schema and metadata browsing on a target dataset prior to query execution, as well as data access during query execution; the goal is to enable direct communication between the user's GMQL site and the dataset owner GMQL site.

Assume that a Web user, carrying an authorization token obtained at login, is working at site `DEIB`. During the opening of the Web interface, the names of the federated datasets visible to the site users are loaded; their properties (i.e., schema, cardinality of samples, metadata attributes and their distinct values) can be inspected on the Web interface as if the federated datasets were local.

The resource authentication protocol for supporting these functions is shown in Figure 3. There, the user is located at `DEIB`, so user's requests are managed by the `DEIB` GMQL instance. A first request is rooted to the Name server, and produces a list of datasets that can be seen at user site (i.e., `DEIB`), whose names can be displayed; `DEIB` uses its token for direct interaction with the Name server. At this point, the user may selectively request an access to the information of any federated dataset; assume that the request concerns dataset `DS` stored at site `CINECA`. Such request is again routed to the local authorization system, which interacts with the Name server in order to get another token, this time enabling a direct communication of `DEIB` with `CINECA`. Once the token is received by `DEIB`, direct communication between `DEIB` and `CINECA` is initiated. On the first interaction, `CINECA` verifies that the request token was indeed generated by the Name server; then, it can respond to a sequence of requests regarding, e.g., `DS` schema or metadata, including those requests coming from the use of the Web interface Metadata browser to respond to exploratory queries.

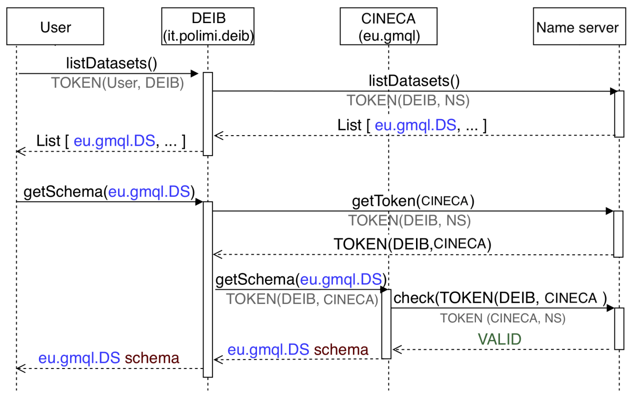


**Fig. 3.** Authentication protocol to obtain information about dataset DS, involving the user’s process, the authentication servers at the query and DS owner sites, and the Name server.

## 3.2 Intermediate Representation splitting

As discussed in [9], a GMQL query is encoded as an Intermediate Representation (IR) that omits all syntactic details and does not encapsulate any of the execution engine peculiarities. Technically, an IR is a directed acyclic graph (DAG), where nodes represent elementary parametric operations (e.g., a filter on region data or a filter on metadata) and edges represent the flow of the execution. Nodes can be clustered in two categories: nodes manipulating metadata (in red in Figure 4) and nodes manipulating region data (in blue in Figure 4). Execution takes place by a recursive descent of the DAG.

The execution of Federated GMQL queries requires *IR splitting* into a *DAG of DAGs*, such that each DAG contains only operations executed at a given GMQL instance. Every Federated GMQL query is associated with a given splitting strategy, as each dataset and operation is explicitly allocated to a given GMQL instance. The new operations StoreFM and StoreFR are used to store, respectively, metadata and region data fragments resulting from a local computation to the local file system; correspondingly, the operations LoadFM and LoadFR load metadata and region data fragments to the Spark engine of the involved GMQL instance. Data transmissions occur between pairs of store and load operations. Execution in the current federated system is still a recursive descent, hence each query operates on one GMQL instance at a time; this choice allows to better control query processing, but execution is sequential rather than parallel.

Figure 4 illustrates the DAG splitting corresponding to the running example query; the query is decomposed into 7 DAGs, each involving either metadata or region data (except the materialization operation). The query is initiated at the DEIB instance; after selection, both metadata and region data of the Experiments dataset are sent to CINECA. Next, after selection at GeCo, metadata and region data of the HG19_BED_ANNOTATIONS dataset are also sent to CINECA. Join processing occurs at CINECA separately for data and metadata, and results are sent to the DEIB instance. The last operation, which stores the query result, applies to both metadata and region data.

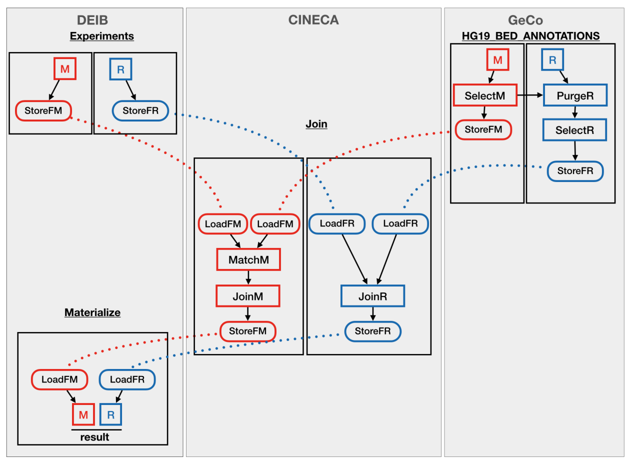


**Fig. 4.** DAG splitting: each DAG contains only operations assigned to a given GMQL instance of the Federated GMQL system. Each GMQL object consists of regions (R, in blue) and metadata (M, in red); each sub-object is processed by specific operations, which are executed at specific GMQL instances (DEIB, CINECA, GeCo); the query is launched and materialized at DEIB. The dataflow of operations includes precedence between operations and data transmissions, represented as dotted lines.

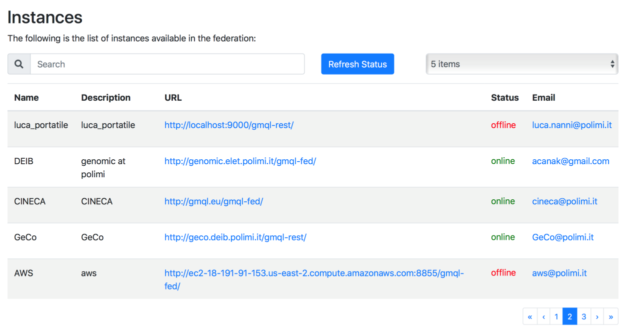
## 3.3 Supporting data federation

During execution, the Name server provides information about the status of Federated GMQL instances, by periodically trying to connect to each registered instance (by default, every 120 seconds). Such status is displayed on the Web interface of the Name server (Figure 5); it is changed at each registration of a new instance and can be refreshed. Each site administrator can configure GMQL instances by setting the maximum number of local resources that can be dynamically allocated to external queries.

The Web interface of a Federated GMQL instance is illustrated in Figure 6; it has few changes w.r.t. the Web interface of GMQL, as presented in [9]. Specifically, in the *Datasets* section (upper left corner), federated datasets are added to private and public datasets; each federated dataset is prefixed by the name of the GMQL instance that owns it. In the *Query editor* section (upper right corner), operations are tagged with the name of the GMQL instance where they are executed. Other panels, illustrated in [9], remain unchanged; in particular, the *Metadata browser* section (lower left corner) helps browsing metadata and filtering samples also of federated datasets, whose features are extracted from the repository of the owner GMQL instance after executing the protocol discussed in Section 3.1.

## 4 Applications

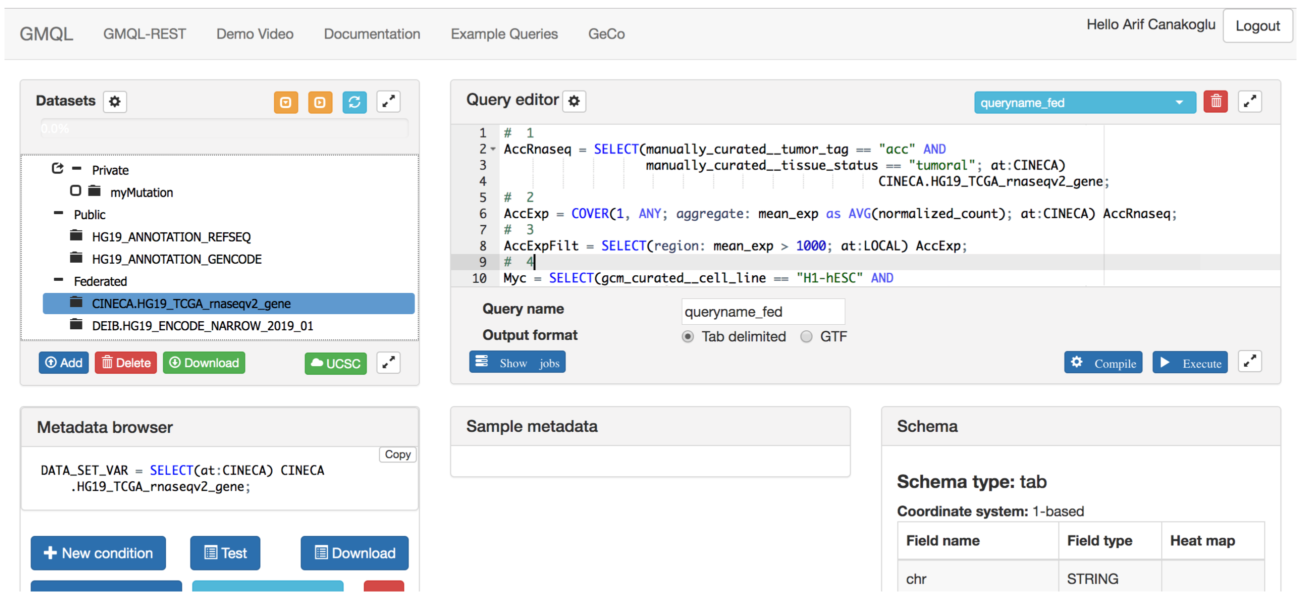
We review Federated GMQL at work according to two different usage scenarios, built with the same query. In the first scenario, a biologist connects her local GMQL instance, storing an experimental dataset of mutations, to the GenoSurf repository, storing two datasets about gene expressions and MYC binding sites, stored at CINECA. In the second scenario, the three datasets are distributed in several ways to the GMQL instances (CINECA, GeCo, DEIB), so as to discuss various execution options illustrated in Section 2.4. Statistics about the input datasets are shown in Table 1.


Instances

The following is the list of instances available in the federation:

| Name | Description | URL | Status | Email |
|---|---|---|---|---|
| luca_portatile | luca_portatile | http://localhost:9000/gmql-rest/ | offline | luca.nanni@polimi.it |
| DEIB | genomic at polimi | http://genomic.elet.polimi.it/gmql-fed/ | online | acanak@gmail.com |
| CINECA | CINECA | http://gmql.eu/gmql-fed/ | online | cineca@polimi.it |
| GeCo | GeCo | http://geco.deib.polimi.it/gmql-rest/ | online | GeCo@polimi.it |
| AWS | aws | http://ec2-18-191-91-153.us-east-2.compute.amazonaws.com:8855/gmql-fed/ | offline | aws@polimi.it |

**Fig. 5.** Monitoring of GMQL instances



```
# 1
AccRnaseq = SELECT(manually_curated__tumor_tag == "acc" AND
                   manually_curated__tissue_status == "tumoral"; at:CINECA)
                                                   CINECA.HG19_TCGA_rnaseqv2_gene;
# 2
AccExp = COVER(1, ANY; aggregate: mean_exp as AVG(normalized_count); at:CINECA) AccRnaseq;
# 3
AccExpFilt = SELECT(region: mean_exp > 1000; at:LOCAL) AccExp;
# 4
Myc = SELECT(gcm_curated__cell_line == "H1-hESC" AND
```

Query name: queryname_fed
Output format: [✓ Tab delimited] GTF

```
DATA_SET_VAR = SELECT(at:CINECA) CINECA
    .HG19_TCGA_rnaseqv2_gene;
```

**Schema type:** tab
**Coordinate system:** 1-based

| Field name | Field type | Heat map |
|---|---|---|
| chr | STRING | |

**Fig. 6.** Extension of the GMQL Web interface to support data federation. Note that datasets include federated ones and query operations include the `at` parameter; these are the only changes.

Table 1. Descriptive statistics of the datasets used in the example application.

| Dataset | Data type | Regions | Samples | Size |
|---|---|---|---|---|
| TCGA_RNASEQ | RNA-seq | 201M | 9,825 | 21 GB |
| ENCODE_NARROW | ChIP-seq | 1,998M | 12,601 | 140 GB |
| MUTATION | Mutations | 13K | 91 | 155 MB |
| ResGenes | Annotations | 223 | 1 | 1 MB |

## 4.1 First scenario: Remote repository access from local system

Suppose a biologist is interested in Adenoid Cystic Carcinoma (ACC), an uncommon form of malignant neoplasm that arises within secretory glands of the head and neck. She collected several mutation samples in her private `MUTATION` dataset, and she likes to know how they overlap with highly expressed genes that in turn overlap with some binding enrichment peaks of the MYC transcription factor.

In the GMQL query below, `LOCAL` denotes the local system used by the biologist. Names and values are shortened, the exact query is in Supplementary Materials and part of it is shown in Figure 6.

```
1. AccRNAseq = SELECT(tumor == "acc" AND
                      tissue == "tumoral";
                      at:CINECA) TCGA_RNASEQ;
2. AccExp = COVER(1,ANY;
                  aggregate: mean as AVG(RSEM);
                  at:CINECA) AccRNAseq;
3. AccExpFilt = SELECT(region: mean > 1000;
                       at:CINECA) AccExp;
4. Myc = SELECT(cell_line == "H1-hESC" AND
                target == "MYC-human" AND
                type == "peaks";
                at:CINECA) ENCODE_NARROW;
5. GeneMyc = JOIN(dist < 0;
                  output: left_distinct;
                  at:CINECA) AccExpFilt Myc;
6. myMutation = SELECT(at:LOCAL) MUTATION;
7. myMutationMerge = MERGE(at:LOCAL) myMutation;
8. GeneMycMut= MAP(count_name: mut_count;
                   at:LOCAL
                   ) GeneMyc myMutationMerge;
9. ResGenes = SELECT(region: mut_count > 0;
                     at:LOCAL) GeneMycMut;
10. MATERIALIZE ResGenes INTO ResGenes;
```

The query initially extracts available gene expression data of patients affected by ACC from the public `TCGA_RNASEQ` dataset available on GenoSurf at CINECA, which stores data originally from TCGA, and computes the mean of the expressions of each gene in the considered patients. As the scientist is interested in genes whose expression falls within the highest quartile, the query extracts the genes whose average expression is above the correspondent threshold of 1,000 RSEM (about 5,000 genes).

Then, the query extracts ChIP-seq experiment samples with binding enrichment regions of the MYC transcription factor in H1-hESC human embryonic stem cells ( in order to be able to see also if proliferation mechanisms are conserved in ACC; these are selected from the `ENCODE_NARROW` dataset, originally from ENCODE, also stored in the GenoSurf repository at CINECA. The query then produces a set of genes of interest, given by those genes of the first set that also intersect with one or more regions of binding enrichment of MYC.

Finally, the query locally counts the total number of mutations of TCGA patients affected by ACC overlapping with each of the selected genes. The top mutated genes extracted from the result are shown in Table 2; they include LRP1, a gene involved in tumorigenesis [30], on average highly expressed and mutated in 9 out of the 91 patients considered.

Table 2. Top 4 mutated genes with their average expression and mutation counts.

| Gene | Expression (RSEM) | # Mutations |
|---|---|---|
| EEF1B2 | 1,055.31 | 11 |
| LRP1 | 3,535.41 | 9 |
| SYNE2 | 1,431.25 | 6 |
| VCAN | 1,838.68 | 5 |

## 4.2 Second scenario: Distributed access

The second scenario is an artificial one, built in order to review how Federated GMQL adapts to arbitrary data distributions. In such scenario, `TCGA_RNASEQ` is stored at `CINECA`, `ENCODE_NARROW` is stored at `DEIB`, and `MUTATION` is stored at `GeCo`. From this initial allocation, several execution strategy options are compared.

### 4.2.1 Distributed strategy

A distributed strategy consists in accessing datasets at their storage sites and then doing all unary operations (involving a single dataset) at such sites before transferring their results for doing binary operations (regarding two datasets). Since data sizes are typically reduced as effect of unary operations, this strategy typically minimizes the data transfers. For each binary operation we consider two allocations (at each of the operands sites); thus, with $n$ binary operations we need to consider $2^n$ alternatives. The example application has two binary operations (JOIN at line 5 and then MAP at line 8); thus, we need to consider 4 alternatives, shown in Figure 7(a). A default distributed policy for a GMQL query is generated by using the directive:

```
@policy <distributed>
```

This policy allocates each binary operation at the node of the *left operand* (thus, for the example application, it produces the DIST-2 strategy described in Table 3).

### 4.2.2 Centralized strategy

A centralized strategy consists in sending all the datasets to a single site, chosen within the query sites, where all processing occurs, except the initial SELECT operations; data transmission is typically heavier than in the distributed strategy. With $m$ GMQL instances involved in a query, there are $m$ alternative strategies; the three alternatives for the example application are shown in Figure 7(b). A centralized policy for a GMQL query is generated by using the directive:

```
@policy centralized (<GMQL-INSTANCE>)
```

### 4.2.3 Externalized strategy

The externalized strategy consists in sending all the datasets to an external server, where all processing occurs, except the initial selections and final materialization. The GMQL engine must run on the external server. An externalized policy for a GMQL query is generated by using the directive:

```
@policy externalize (<GMQL-INSTANCE>)
```

This strategy is recommended if the data to be transferred is relatively small, the processing is heavy and the external server guarantees better performance; in the reviewed case, Amazon Web Services (AWS) was used, with 10 dedicated nodes. The strategy is shown in Figure 7(b).

### 4.2.4 Best strategy

In the specific example application, the best strategy (which we determined with an exhaustive evaluation) is very similar to the third centralized strategy; the only difference is in operation 2, which is performed at `CINECA`, thereby reducing the size of the dataset that needs to be transmitted to `GeCo`. This strategy is shown in Figure 7(c).

### 4.2.5 Comparison

Execution times are reported in seconds in Table 3. In the example application, after the initial selections on TCGA and ENCODE datasets, sizes of intermediate results are small, thus parallelism at `CINECA` and `AWS` is not exploited. In general, the execution strategies with better performance use the single powerful server at `GeCo`, which has a dockerized installation that invokes Spark locally. In particular, the optimal execution strategy shows that anticipating selective operations of large datasets at the GMQL instance of origin reduces the need of storing intermediate datasets, and requires 7 minutes and 53 seconds. This execution time compares rather well with the execution time of the query running just at `CINECA` (6 minutes and 31 seconds), where GMQL is implemented on top of a small cluster, or just at `DEIB` (3 minutes and 38 seconds), where GMQL is implemented on a powerful server. We recall that the focus of Federated GMQL is accessibility, hence we expect centralized queries to be faster than distributed queries.

## 4.3 Data protection

Protecting private datasets from exposure to malicious attacks that could occur outside of the boundaries of their local site is an increasingly important aspect of distributed data management. In genomic applications, users often do not have the right to transfer private datasets, or any dataset derived from their processing, outside such boundaries.

A user can instruct the Federated GMQL system that some datasets are protected; in that case, a query execution strategy that allocates such datasets or any derived datasets at a non-local site produces an error. For instance, a user can mark the `MUTATION` dataset as `PROTECTED` as follows:

```
@protected MUTATION
```

Only four out of the nine strategies discussed in the previous section protect the `MUTATION` dataset, as shown in Table 3. Note that the centralized policy at the site of the user obviously protects all local datasets. It is also worth noticing that many cloud providers, including Amazon Web Services, Google Cloud and Microsoft Azure, meet security guidelines as required by genomic computations [31]; this would in principle allow a protected externalized execution of GMQL queries on the cloud.

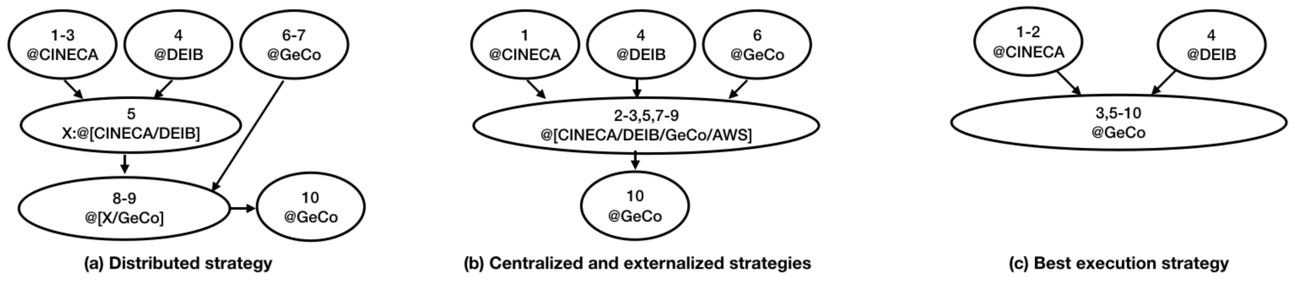


**Fig. 7.** Assignment of operations to GMQL instances according to distributed, centralized, externalized and best execution strategies. Numbers refer to query operations.

Table 3. Comparison of distributed (DIST-1..4), centralized (CENT-1..3), externalized (EXT) and BEST execution strategy. In distributed executions, J refers to execution node of JOIN, and M of MAP, operation. Four strategies, denoted as Protective, protect the `MUTATION` dataset. We provide (in seconds): execution times (ET) at the GMQL instances, total transmission time (Trans. time) and total time.

| | DIST-1 J,M: DEIB | DIST-2 J,M: CINECA | DIST-3 J: DEIB M: GeCo | DIST-4 J: CINECA M: GeCo | CENT-1 DEIB | CENT-2 CINECA | CENT-3 GeCo | EXT AWS | BEST |
|---|---|---|---|---|---|---|---|---|---|
| ET (GeCo) | 5 | 5 | 39 | 39 | 6 | 5 | 756 | 3 | 44 |
| ET (DEIB) | 170 | 85 | 104 | 90 | 1,116 | 85 | 91 | 89 | 85 |
| ET (CINECA) | 611 | 2,138 | 628 | 789 | 150 | 2,152 | 163 | 152 | 337 |
| ET (AWS) | - | - | - | - | - | - | - | 1,003 | - |
| Trans. time | 12 | 12 | 5 | 6 | 306 | 13 | 304 | 314 | 6 |
| Total time | 830 | 2,273 | 780 | 929 | 1,652 | 2,292 | 1,315 | 1,587 | 473 |
| Protective | no | no | yes | yes | no | no | yes | no | yes |

## 5 Discussion and conclusion

GMQL and the associated multi-source repository fully satisfy the FAIR principles [32]. *Findability* and *accessibility* descend from the fact that all datasets and samples are described by rich metadata that explicitly refers to genomic region data and are registered and indexed as a searchable resource. *Interoperability* descends from the use of a data model highlighting interoperability among heterogeneous data types (e.g., uniformly describing annotations, mutations, expressions and peaks of binding enrichment), whose metadata are normalized thanks to controlled vocabularies and references to global ontologies [18]. *Reusability* descends from the emphasis on data provenance, which is preserved throughout query computations, whereas reproducibility is enhanced by the possibility of using GMQL from within Jupyter notebooks [33]. Federated GMQL makes a step forward in enhancing the potential of FAIR compliance, by amplifying the possibility of data exploration, sharing and reuse across data center boundaries. Note that data federation can be motivated by workload distribution to increase performance, e.g., in [16]. In our system, instead, we are mostly concerned with providing sharing functionalities among cooperating users; workload is distributed among the federated GMQL instances involved in the executed query, depending on the chosen query strategy.

Future development can be in the direction of adding parallel execution and query optimization. Defining an optimal execution strategy is a classic distributed database problem, which can be solved analytically or by learning the best strategies after several executions. A quantitative model for evaluating the cost of binary operations was already developed in [20]. The three strategies (distributed, centralized, or externalized) discussed in Section 4.2 can be used as starting points for an effective exploration of alternatives.

How to formally define privacy-protecting policies for Federated GMQL queries that guarantee various levels of data protection, while at the same time enable increasingly expressive computations, can also be a nice extension. Various privacy policies reviewed in [34] can be considered and supported within the Federated GMQL framework.

## Resources

Federated GMQL is freely available and can be tested through REST or Web interfaces at:
http://www.bioinformatics.deib.polimi.it/FederatedGMQLsystem/.
The open source project is available in GitHub with documentation.

In addition to classic downloadable packages for the Name server and Federated GMQL instance software, requiring installation and configuration over the Spark engine, also a Docker container [35] image of a Federated GMQL instance is provided. Once installed, the Docker image contains a copy of both the Web server and the GMQL engine, and sees the data repository at the local file system. The Name server can also be installed through a Docker container. Availability of dockers enables easy installation of a new GMQL instance on any Unix-based platform.

## Supplementary material

Strategies for the reviewed example application query and the execution log for its BEST strategy are provided in the Supplementary Material (except the externalized strategy that requires AWS).

## Funding

This work was supported by the ERC Advanced Grant 693174 "Data-Driven Genomic Computing (GeCo)".

## References

[1] Zachary D. Stephens, Skylar Y. Lee, Faraz Faghri, et al. Big data: astronomical or genomical? *PLOS Biology*, 13(7):1–11, 2015.

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

[3] John N Weinstein, Eric A Collisson, Gordon B Mills, et al. The Cancer Genome Atlas Pan-Cancer analysis project. *Nature Genetics*, 45(10):1113–1120, 2013.

[4] Bradley E Bernstein, John A Stamatoyannopoulos, Joseph F Costello, et al. The NIH Roadmap Epigenomics Mapping Consortium. *Nature Biotechnology*, 28(10):1045–1048, 2010.

[5] Nayanah Siva. UK gears up to decode 100 000 genomes from NHS patients. *The Lancet*, 385(9963):103–104, 2015.

[6] FinnGen. Online at: `https://www.finngen.fi/en`, 2019. Accessed: 2-4-2020.

[7] AllOfUs. Online at: `https://allofus.nih.gov/`, 2019. Accessed: 2-4-2020.

[8] Marco Masseroli, Pietro Pinoli, Francesco Venco, et al. GenoMetric Query Language: a novel approach to large-scale genomic data management. *Bioinformatics*, 31(12):1881–1888, 2015.

[9] Marco Masseroli, Arif Canakoglu, Pietro Pinoli, et al. Processing of big heterogeneous genomic datasets for tertiary analysis of Next Generation Sequencing data. *Bioinformatics*, 35(5):729–736, 2019.

[10] Marco Masseroli, Abdulrahman Kaitoua, Pietro Pinoli, et al. Modeling and interoperability of heterogeneous genomic big data for integrative processing and querying. *Methods*, 111:3–11, 2016.

[11] Arif Canakoglu, Anna Bernasconi, Andrea Colombo, et al. GenoSurf: metadata driven semantic search system for integrated genomic datasets. *Database*, 2019:pii: baz132, 2019.

[12] Ana Claudia Sima, Tarcisio Mendez de Farias, Erich Zbinden, et al. Enabling semantic queries across federated bioinformatics databases. *Database*, 2019:pii: baz106, 2019.

[13] Tarcisio Farias, Ana Roxin, and Christophe Nicolle. Fowla, a federated architecture for ontologies. *Rule Technologies: Foundations, Tools, and Applications - RuleML*, LNCS 9202:97–111, 2015.

[14] A. Hasnain, Q. Mehmood, S. Zainab, et al. BioFed: federated query processing over life sciences linked open data. *J Biomed Semant*, 8(1):13, 2017.

[15] M. Djokic-Petrovic, V. Cvjetkovic, J. Yang, et al. PIBAS FedSPARQL: a web-based platform for integration and exploration of bioinformatics datasets. *J Biomed Semant*, 8(1):42, 2017.

[16] Dariusz Mrozek, Jacek Kwiendacz, and Bożena Małysiak-Mrozek. Protein construction-based data partitioning scheme for alignment of protein macromolecular structures through distributed querying in federated databases. *IEEE Transactions on NanoBioscience*, 19(1):102–116, 2020.

[17] Muhammad Saleem, Shanmukha S Padmanabhuni, Axel-Cyrille Ngonga Ngomo, et al. TopFed: TCGA tailored federated query processing and linking to LOD. *Journal of Biomedical Semantics*, 5:47, 2013.

[18] Anna Bernasconi, Arif Canakoglu, Andrea Colombo, et al. Ontology-driven metadata enrichment for genomic datasets. In *11th Int. Conf. Semantic Web Applications and Tools for HealthCare and Life Science, CEUR Workshop Proceedings*, pages 1–10, 2018.

[19] Abdulrahman Kaitoua, Pietro Pinoli, Michele Bertoni, et al. Framework for supporting genomic operations. *IEEE Transactions on Computers*, 66(3):443–457, 2017.

[20] Andrea Gulino, Abdulrahman Kaitoua, and Stefano Ceri. Optimal binning for genomics. *IEEE Transactions on Computers*, 68(1):125–138, 2019.

[21] Eric E Schadt, Michael D Linderman, Jon Sorenson, et al. Computational solutions to large-scale data management and analysis. *Nature Reviews Genetics*, 11(9):647–657, 2010.

[22] Bertil Schmidt and Andreas Hildebrandt. Next-generation sequencing: big data meets high performance computing. *Drug Discovery Today*, 22(4):712–717, 2017.

[23] Damian Smedley, Syed Haider, Steffen Durinck, et al. The BioMart community portal: an innovative alternative to large, centralized data repositories. *Nucleic Acids Research*, 43(W1):W589–W598, 2015.

[24] Junjun Zhang, Syed Haider, Joachim Baran, et al. BioMart: a data federation framework for large collaborative projects. *Database*, 2011:bar038, 2011.

[25] Abdulrahman Kaitoua, Andrea Gulino, Marco Masseroli, et al. Scalable genomic data management system on the cloud. In *Proceedings of the IEEE International Conference on High Performance Computing & Simulation (HPCS 2017)*, pages 58–63. IEEE Computer Society, 2017.

[26] Dariusz Mrozek, Bożena Malysiak-Mrozek, and Artur Klapcinski. Cloud4Psi: cloud computing for 3D protein structure similarity searching. *Bioinformatics*, 30(19):2822–2825, 2014.

[27] Beacon. Online at: `https://beacon-network.org/`, 2015. Accessed: 2-4-2020.

[28] Melissa S Cline, Rachel G Liao, Michael T Parsons, et al. BRCA Challenge: BRCA Exchange as a global resource for variants in BRCA1 and BRCA2. *PLoS Genetics*, 14(12):e1007752, 2018.

[29] CanDIG. Online at: `https://candig.github.io/`, 2019. Accessed: 2-4-2020.

[30] Peipei Xing, Zhichao Liao, Zhiwu Ren, et al. Roles of low-density lipoprotein receptor-related protein 1 in tumors. *Chinese Journal of Cancer*, 35:6, 2016.

[31] Somalee Datta, Keith Bettinger, and Michael Snyder. Secure cloud computing for genomic data. volume 34, page 588. Nature Publishing Group, 2016.

[32] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3:160018, 2016.

[33] Luca Nanni, Pietro Pinoli, Arif Canakoglu, et al. Exploring genomic datasets: From batch to interactive and back. In *Proceedings of the 5th International Workshop on Exploratory Search in Databases and the Web*, ExploreDB 2018, pages 3:1–3:6, New York, NY, USA, 2018. ACM.

[34] Mete Akgün, A. Osman Bayrak, Bugra Ozer, et al. Privacy preserving processing of genomic data: A survey. *Journal of Biomedical Informatics*, 56:103–111, 2015.

[35] Docker. Online at: `https://www.docker.com/`, 2019. Accessed: 2-4-2020.