

# Organization of Variation-Based Personal Genetic Data with Document-Based No-Sql Database

*Araştırma Makalesi/Research Article*

 Onur ÇAKIRGÖZ<sup>1</sup>,  Süleyman SEVİNÇ<sup>2</sup>

<sup>1</sup>Computer Engineering Department, Bartın University, Bartın, Turkey

<sup>2</sup>Labenko Bilişim A. Ş., İzmir, Turkey

[onurcakirgoz@bartin.edu.tr](mailto:onurcakirgoz@bartin.edu.tr), [suleysevinc@gmail.com](mailto:suleysevinc@gmail.com)

(Geliş/Received:07.04.2021; Kabul/Accepted:07.09.2021)

DOI: 10.17671/gazibtd.910465

**Abstract**— Variation-based personal genetic data are at the center of many clinical practices and many studies in bioinformatics. Unfortunately, almost all existing methods developed to organize personal genetic data are not variation-based and these methods have not been tested with a large amount of real data. In applications requiring variation-based data, an intense data conversion problem arises when these existing methods are used. On the other hand, the few solutions available that are variation-based are not entirely structural, and they do not meet the needs of daily practice. In this study, a document-based No-SQL database and related designs are proposed for the organization of variation-based personal genetic data. Our structural solution contains many classes, collections and indexes, and it supports all types of variations (both structural and non-structural). In this database, the variation data of 2504 people published by the 1000 Genomes Project were stored smoothly and efficiently. The spaces occupied by personal genetic data in primary memory and hard disk were analyzed. In addition, some queries that might be frequently used by clinical applications were run and the response times of the database was calculated. The results of the analyzes show that the proposed method provides very important gains.

**Keywords**— no-sql database, personal genome database, personal genetic data, human genome variations, 1000 genomes project

## Varyasyon-Bazlı Kişisel Genetik Verilerin Doküman-Tabanlı No-Sql Veri Tabanı ile Organizasyonu

**Özet**— Varyasyon-bazlı kişisel genetik veriler çoğu klinik uygulamanın ve biyoinformatikteki çoğu çalışmanın merkezinde bulunmaktadır. Ne yazık ki, kişisel genetik verileri organize etmek için geliştirilen mevcut yöntemlerin neredeyse tamamı varyasyon-bazlı değildir ve bu yöntemler büyük miktardaki gerçek verilerle test edilmemiştir. Varyasyon-bazlı verilere ihtiyaç duyan uygulamalarda, bu mevcut yöntemler kullanıldığında, yoğun bir veri dönüştürme problemi ortaya çıkmaktadır. Öte yandan, az sayıdaki mevcut varyasyon-bazlı çözümler tamamıyla yapısal değildir ve günlük pratiğin gereksinimlerini karşılamamaktadır. Bu çalışmada, varyasyon-bazlı kişisel genetik verilerin organizasyonu için doküman-tabanlı No-SQL veri tabanı ve ilgili tasarımlar önerilmektedir. Yapısal çözümümüz çok sayıda sınıf, koleksiyon ve indeks içermektedir ve tüm varyasyon tiplerini (yapısal ve yapısal olmayan) desteklemektedir. Bu veri tabanında, 1000 Genom Projesi tarafından yayınlanan 2504 kişinin varyasyon verileri sorunsuz ve verimli bir şekilde depolanmıştır. Kişisel genetik verilerin ana bellek ve sabit diskte kapladığı alanlar incelenmiştir. Ayrıca, klinik uygulamaların sıklıkla kullanılabileceği bazı sorgular çalıştırılmış ve veri tabanının yanıt süreleri hesaplanmıştır. Analizlerin sonuçları, önerilen yöntemin çok önemli kazanımlar sağladığını göstermektedir.

**Anahtar Kelimeler**— no-sql veritabanı, kişisel genom veritabanı, kişisel genetik veriler, insan genomu varyasyonları, 1000 genom projesi

## 1. INTRODUCTION

The development of faster and cheaper sequencing technologies has made possible the personalized human genomics. Thanks to these technological advances, many personal diploid human genome sequences were created. At the same time, our knowledge of individual genetic variations has increased significantly. On the other hand, personal genome sequencing has now become a common part of medical practice [1]. Both the whole-genome and exome sequencing reveal the genetic causes of diseases and the treatment methods are determined in the light of this information [2].

Very large-scale sequencing projects have been implemented by utilizing next generation sequencing technologies. For example, the 1000 Genome Project [3] [4] was launched in 2008 and is a very large international cooperation project. At the end of the project, a large catalog of human genomic variations was created. Maps of genetic variations have been published using the genetic data of 2504 individuals selected from 26 different populations [5, 6]. The variation data were published by 1000 Genomes Project as VCF [7] and BCF [8] files. The total storage space used for the project is more than 500 TB. Approximately 2 years after the 1000 Genome Project, the UK10K project [9] [10] was launched in July 2010 by the Wellcome Trust Sanger Institute, the largest contributor to the Human Genome Project [11]. The main objective of the UK10K project is to reveal more clearly the association between rare genetic variations and diseases. From this perspective, it can be said that the UK10K project contributed more than the 1000 Genome Project, since the output of the UK10K project contains both genotype and phenotype knowledge. Another project initiated after the UK10K project is the Encyclopedia of DNA elements (ENCODE) project [12]. Within the scope of this project, ChIP-seq and RNA-seq assays were carried out in human and mouse and many experimental data sets were made available. Finally, the large-scale sequencing project that must be mentioned is the Cancer Genome Atlas (TCGA) project [13, 14, 15]. The Cancer Genome Atlas (TCGA) project was carried out to explore genomic changes in human cancer. Through the data portal [16], it provides access to clinical information as well as the tumor genome of many cases of 33 different types of cancer.

Next-generation high-throughput sequencing platforms are generating massive amounts of genetic data. In parallel, a number of tools have been developed to work with these genetic data generated by next-generation sequencing devices. These tools can be actually divided into two, basically. In the first section, read based aligners exist like MAQ [17], BWA [18], and SOAP [19]. In the second section, single nucleotide polymorphism and structural variation detection tools exist like BreakDancer [20], VarScan [21], and MAQ. Shortly after these tools, the SAM file specification [22] emerged as the platform-independent standard format for storing next-generation sequencing data. After the development of SAM, [23] proposed the Genome Analysis Toolkit (GATK), a

structured programming framework, which takes advantage of this common input format to simplify the up-front coding costs for end users. GATK uses the functional programming philosophy of MapReduce [24] and eases the development of analysis tools for next-generation DNA sequencers. On the other hand, a variety of genotyping and variation formats were proposed, including common emerging SNP formats like GLF and VCF [25].

Numerous compression algorithms have also been proposed to reduce the size of genetic data. BioCompress [26] is the first compression algorithm developed for the compression of DNA sequences. In the BioCompress algorithm, each nucleotide base is represented and encoded by 2 bits. Besides, BioCompress detects reverse complement repeats. The two compression algorithms developed after BioCompress, Cfact [27] and Off-line [28], are actually variations of BioCompress. Both algorithms work in rounds, greedily replacing duplicate text with shorter codes. Then, GenCompress [29] showed that by considering approximate repeats, the results could be improved. Along with the GenCompress algorithm, most DNA compression algorithms are built on various variations of approximate repeat detection. The biggest drawback of many of these early DNA compression algorithms is that they can only compress small files. Now, next-generation sequencing devices can sequence many individual genomes from the same species. Therefore, recent algorithms have addressed the problem of compressing DNA data from the same species. E-mail attachment [30] compresses variation data from human genomes and encodes the small variations with respect to the human reference sequence and known variations recorded in a SNP database. This method is a four-step method and uses multiple compression techniques together. They applied the method to James Watson's (JW) genome [31], and managed to compress it to 4 MB. As with other compression methods, the disadvantage of this method is that it does not contain index. Accordingly, it does not support random access. Other successful and well-known compression algorithms are RLZ [32], XM [33], and RLCSA [34]. One of the best single sequence DNA compression algorithms is XM, but it takes hours for a single chromosome sequence to compress. RLZ specializes in the compression of large, related DNA datasets. The total dataset of three human genome sequences can be represented in 755 MB with RLZ and 3835 MB with RLCSA.

Hadoop [35] and MapReduce [36] frameworks have also been adopted by a few specific bioinformatics tools [37, 38, 39, 40]. In particular, SparkSeq [39] has been proposed as general-purpose tool for processing of DNA and RNA sequence data using the Apache Spark engine [41]. Genome Query Language (GQL) [42, 43] is another well-known structural approach for processing genomic data. GQL is based on genome query algebra and uses an SQL extension. Later, GenoMetric Query Language (GMQL) [40], which is based on Hadoop framework and Apache Pig platform, has been proposed to query and compare multiple and heterogeneous genomic datasets for biomedical

knowledge discovery. GMQL is very similar to GQL, but GQL start from the reads (raw data) of NGS machines, whereas GMQL starts from processed datasets.

The recent study [44] that uses relational database to store variation-based personal genetic data is the closest to our study. In this study, various external data formats were designed to hold variations and genotypes, and all genotypes of a chromosome were (collectively) recorded in the database in the Varbinary binary format and in a compressed form. That is, the relational database alone was not sufficient to represent and store personal genetic data. Naturally, this method does not allow complex SQL queries, most transactions are handled at the application layer.

In this study, a solution was developed for the systematic storage and querying of variation-based personal genetic data needed by most clinical practice and most studies in bioinformatics. For this purpose, a no-sql database was used and many classes, collections and indexes were designed. Our solution supports all variation types. The sections of the article are as follows: In the second section, firstly the data set used in the study was introduced, then the designed collections, classes and indexes were explained. In the third chapter, the analyzes and analysis results were mentioned, the results were discussed, and at the end of the chapter, our method is compared with a recent and successful method. Final remarks are given in the last section.

## 2. MATERIALS AND METHODS

In this section, we described the materials and methods required to create and examine the database mentioned in this study. We used a document-based No-SQL database (Mongo) and created 5 collections, 15 classes and many indices for the organization of variation-based personal genetic data. In this database, the variation data of 2504 people published by the 1000 Genomes Project were stored smoothly and efficiently. Then, the spaces occupied by personal genetic data in primary memory and hard disk were analyzed. In addition, some queries that might be frequently used by clinical applications were run and the response times was calculated.

### 2.1. Dataset

In this study, we used the large catalogue of human genomic variations, which was created by the 1000 Genomes Project and published as BCF files [8]. This large catalogue includes variation-based personal genetic data of 2504 individuals selected from 26 different populations.

### 2.2. Database Schema

The No-SQL database schema, designed to hold variation-based personal genetic data in Mongo database, appears in Figure-1. As shown in Figure-1, the database includes 5 collections. Each of these collections will be discussed separately in the following sections

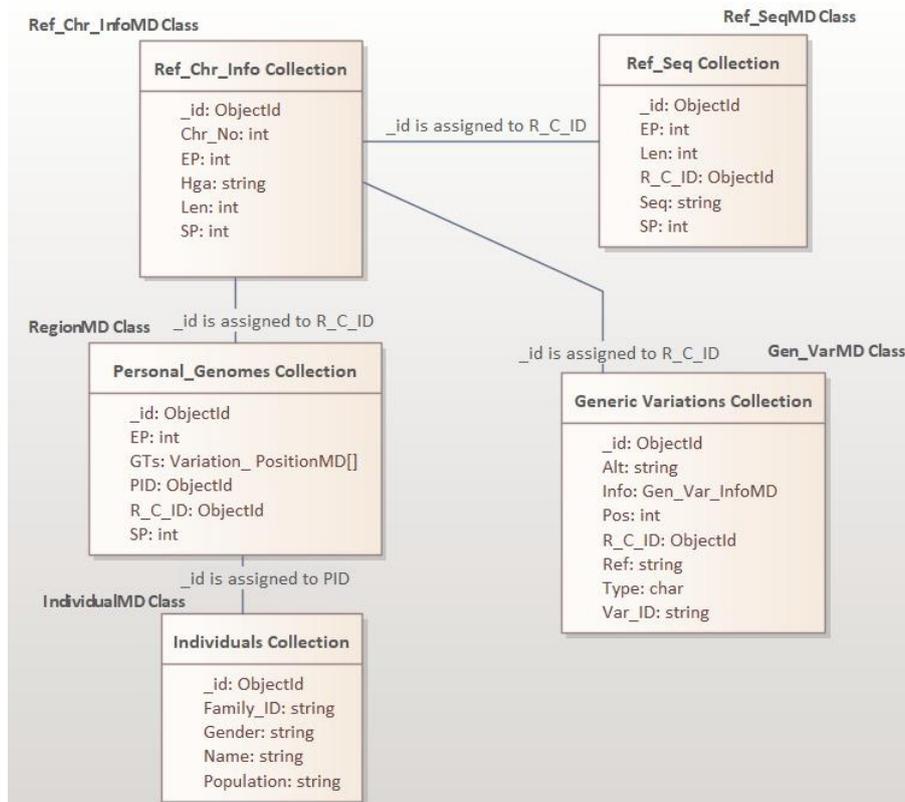


Figure 1. No-SQL database schema

### 2.3. Collections of Genomes Database

#### Individuals Collection

The "Individuals" collection is utilized to store individuals' information and the document type of this collection is "IndividualMD" class. The fields of the "IndividualMD" class and the descriptions of these fields are indicated in detail in Table-1. The fields of the "IndividualMD" class were determined in a way to be compatible with the data of 1000 Genomes Project. Therefore, other fields can be added to this class according to some characteristics (e.g. coverage, size, purpose, etc.) of the applications. The value of the "\_id" field, which uniquely identifies the documents (individuals) within the "Individuals" collection, is assigned to the "PID" field of the "RegionMD" class. In this way, the link between "Individuals" and "Personal\_Genomes" collections is established.

Table 1. The fields of "IndividualMD" class

Field Name	Data Type	Explanation
_id	ObjectId	Document id (primary key)
Name	string	Name of the individual
Family_ID	string	Family_ID of the individual
Population	string	Population of the individual
Gender	string	Gender of the individual

The two indexes created for the "Individuals" collection are shown in Table-2. The first index ("\_id\_") is the default index created by Mongo database. This index indexes documents (individuals) in increasing order depending on the "\_id" field. The second index is created by us. This index indexes individuals in increasing order according to their names. In this way, the data regarding any individual can be obtained from the "Individuals" collection in a very short time.

Table 2. The indexes of "Individuals" collection

Index Name	Keys and Order
_id_	{ "_id" : increasing } (Default index)
Name_1	{ "Name" : increasing }

#### Ref\_Chrr\_Info Collection (Reference Chromosome Information)

The "Ref\_Chrr\_Info" collection holds basic information about reference chromosomes and the document type of this collection is the "Ref\_Chrr\_InfoMD" class. The fields of the "Ref\_Chrr\_InfoMD" class and the explanations of these fields are specified in detail in Table-3. The "Chr\_No" field takes the values of 23 and 24 for the X and Y chromosomes, respectively. On the other hand, the "Hga" field shows the assembly of the human reference genome. These two fields are the basic fields that separate any "Ref\_Chrr\_InfoMD" document from other documents of the "Ref\_Chrr\_Info" collection. When referring to a

personal genotype, we need to specify which reference chromosome (which assembly) was used for the alignment process. Accordingly, the value of the "\_id" field of the "Ref\_Chrr\_InfoMD" document is assigned to the "R\_C\_ID" field of the "RegionMD" class (This class will be described later.). In this way, a link between the "Ref\_Chrr\_Info" collection and the "Personal\_Genomes" collection is established.

Table 3. The fields of "Ref\_Chrr\_InfoMD" class

Field Name	Data Type	Explanation
_id	ObjectId	Document id (primary key)
Chr_No	int	Chromosome number
Hga	string	Human genome assembly
SP	int	Start position of the chromosome
EP	int	End position of the chromosome
Len	int	Length of the chromosome

The two indexes created for the "Ref\_Chrr\_Info" collection appear in the Table-4. The first of these indices is created by default by Mongo database. The second index indexes the chromosomes according to the "Hga" and "Chr\_No" fields (first by "Hga" values, and then, within each "Hga", by "Chr\_No" values).

Table 4. The indexes of "Ref\_Chrr\_Info" collection

Index Name	Keys and Order
_id_	{ "_id" : increasing } (Default index)
Hga_1_Chrr_No_1	{ "Hga" : increasing, "Chr_No" : increasing }

#### Ref\_Seq Collection (Reference Sequences)

"RefSeq" is a collection which is created to hold sequences (bases) of reference chromosomes and the class "Ref\_SeqMD" is the document type of this collection. Here, the chromosome sequences are separated into regions that do not exceed the maximum document size specified by the Mongo database and each region is saved as a document into the collection. All fields of the "Ref\_SeqMD" class and the properties of these fields are shown in Table-5.

Table 5. The fields of "Ref\_SeqMD" class

Field Name	Data Type	Explanation
_id	ObjectId	Document id
R_C_ID	ObjectId	The id of the document (including the ref. chr.) in the "Ref_Chrr_Info" collection
SP	int	Start position of the region
EP	int	End position of the region
Len	int	Length of the region
Seq	string	The sequence of the region defined by "SP" and "EP".

The "Seq" field of the Ref\_SeqMD class stores the sequence. Here, the maximum sequence length that a document can hold was determined to be 1000000, by considering the maximum document size and query performance criterion. On the other hand, the information of which reference chromosome (chromosome number and assembly) the sequences belong to must also be kept somewhere and the "R\_C\_ID" field is used for this purpose. Accordingly, the value of the "\_id" field of the "Ref\_ChromInfoMD" document is assigned to the "R\_C\_ID" field of the "Ref\_SeqMD" document. In this way, the connection between the "Ref\_ChromInfo" collection and the "Ref\_Seq" collection is established.

The two indexes created for the "Ref\_Seq" collection are shown in Table-6. The first of these indices is the index, which is created by default. The second index indexes the documents (chromosome regions) according to the "R\_C\_ID" and "SP" fields (first by "R\_C\_ID" values, and then, within each "R\_C\_ID", by "SP" values). In this way, any desired region(s) of any chromosome can be fetched quickly from the database.

Table 6. The indexes of "Ref\_Seq" collection

Index Name	Keys and Order
_id_	{ "_id" : increasing } (Default index)
R_C_ID_1_SP_1	{ "R_C_ID" : increasing, "SP" : increasing }

*Generic Variations Collection*

In a database developed for the storage of variation-based personal genetic data and for querying related data in clinical applications, a collection that will hold generic variations is one of the indispensable components. Naturally, this data needs to be systematically taken from the database along with personal genetic data. In line with the requirements, the "Generic\_Variations" collection was developed. The document type of this collection is the "Gen\_VarMD" class and the fields of this class and the explanations of these fields are detailed in Table-7.

The second field of Table-7, "R\_C\_ID", is utilized for the purpose of holding the information of which reference chromosome (chromosome number and assembly) the respective variation belongs to. Accordingly, the value of the "\_id" field of the "Ref\_ChromInfoMD" document is assigned to the "R\_C\_ID" field of the "Gen\_VarMD" document. The next five fields are the basic information that defines the variation. In fact, these are all indispensable information; but when a novel variation is detected, the field "Var\_ID" may be null until this variation is given a unique id. Other information about the variation is stored in the "Info" field, and the data type of this field is the "Gen\_Var\_InfoMD" class, as can be seen from Table-7. "Gen\_Var\_InfoMD" class and "Gen\_Var\_InfoExtMD" class, derived from this class, will be explained later.

Table 7. The fields of "Gen\_VarMD" class

Field Name	Data Type	Explanation
_id	ObjectId	Document id
R_C_ID	ObjectId	The id of the document (including the ref. chr.) in the Ref_ChromInfo collection
Var_ID	string	The ID of the variation
Type	char	The type of the variation ('N' for Non-structural, 'S' for structural variations.)
Pos	int	The start position of the variation on the chromosome
Ref	string	Reference allele
Alt	string	Alternate alleles
Info	Gen_Var_InfoMD	Extra information field

The indexes created for the "Generic\_Variations" collection appear in Table-8. In some clinical operations, variation's id, the chromosome to which it belongs, and the position information are generally available in our hands. Moreover, searches also center upon these fields. Moving from here, the second and third indices were designed in this direction. Notice that the third index indexes the variations according to their IDs ("Var\_ID" field), and the type of this index is hash.

Table 8. The indexes of "Generic\_Variations" collection

Index Name	Keys and Order
_id_	{ "_id" : increasing } (Default index)
R_C_ID_1_Pos_1	{ "R_C_ID" : increasing, "Pos" : increasing }
Var_ID_H	{ "Var_ID" : Hashed }

*Personal Genomes Collection (Variation-Based Personal Genotypes)*

The "Personal\_Genomes" collection is the most basic and most significant collection of the "Genomes" database. This collection is utilized to hold variation-based genetic data of individuals (genotypes/haplotypes concerning the variations detected on the genome of the person). To avoid exceeding the maximum document size of 1000 kB, and not to reduce query performance, the chromosomes are divided into regions. The document type of the "Personal\_Genomes" collection is the "RegionMD" class. As is known, there is no restriction on the number of elements of an array that any Mongo database document has (Provided that the maximum document size is not exceeded). In this collection, each document (region) contains a maximum of 2500 genotypes. The 2500 genotypes restriction mentioned here is a value determined by us, considering the maximum document size and search performance. The fields of the "RegionMD" class are given in Table-9.

It can be seen from the table that there are two relations (between "Personal\_Genomes" and "Ref\_Chrr\_Info" and, between "Personal\_Genomes" and "Individuals" collections). Thanks to both the fields that make up these relationships and the other fields, the "RegionMD" document holds genotypes/haplotypes regarding the variations that have been identified in any region of a person's any chromosome. The "GTs" field of this document stores genotypes and the data type of this field is "Variation\_PositionMD" array. "Variation\_PositionMD" class and other related classes will be explained in the following section.

Table 9. The fields of "RegionMD" class

Field Name	Data Type	Explanation
_id	ObjectId	Document id
R_C_ID	ObjectId	The id of the document (including the ref. chr.) in the "Ref_Chrr_Info" collection
PID	ObjectId	The id of the document (including individual) in the "Individuals" collection
SP	int	Start position of the region
EP	int	End position of the region
GTs	Variation_PositionMD[]	The array storing the genotypes/haplotypes of the individual

In clinical applications that process/query variation-based personal genetic data, the largest number of queries will be called for this collection and this is the collection that occupies the most space. Therefore, the indices designed for the "Personal\_Genomes" collection are important. In the light of this information, three indexes created for the "Personal\_Genomes" collection are shown in Table-10.

Table 10. The indices of "Personal\_Genomes" collection

Index Name	Keys and Order
_id_	{ "_id" : increasing } (Default index)
PID_1_R_C_ID_1_ SP_1_EP_1	{ "PID" : increasing, "R_C_ID" : increasing, "SP" : increasing, "EP" : increasing }
PID_1_R_C_ID_1_ SP_1_EP_1_GTs.Pos_1	{ "PID" : increasing, "R_C_ID" : increasing, "SP" : increasing, "EP" : increasing, "GTs.Pos" : increasing }

The first index ("\_id\_") is the default index created by Mongo database. The second and the third indices were designed by us. The purpose of designing these two indices is to find the genotype/genotypes in any region of any chromosome of a person very quickly and to fetch this data from the database. In fact, although these two indices are quite similar to each other, there is one difference between them. In addition to the second index, the third index indexes the data according to the "Pos" property of the

"GTs" field at the last stage. At this point, one might ask: "If the third index fully covers the second index, why is the second index needed?". The answer to this question will be given in the next section when comparing the query performances and the space requirements of the indices.

2.4. Other Classes

In the previous chapter, the "Genomes" database designed to hold variation-based personal genetic data, the collections that make up this database and the document types of these collections were described. It was also stated that the "Personal\_Genomes" collection is the most important collection among these collections, and it holds personal genetic data. But, the "Variation\_PositionMD" class, which is the data type of the "GTs" field that holds personal genotypes and, other related classes were not mentioned. Accordingly, the classes "Variation\_PositionMD", "Gen\_Var\_InfoMD" and "Gen\_Var\_InfoExtMD" will be explained in detail in this section.

Class Design for Personal Variations (Haplotypes)

A total of five classes were designed to hold a variation (personal haplotype) detected at any position of any chromosome of a person. The designed classes, and inheritance and polymorphism relations between these classes are shown in Figure-2. As can be seen from the figure, "VariationMD" is the class (super class) at the top of this hierarchy. The "VariationMD" class has a single field, and that field holds the type of the variation. This field takes a total of 11 different values in a way to support all variation types. To that end, three characters are utilized for the non-structural variations, whereas eight characters are utilized to represent the structural variations.

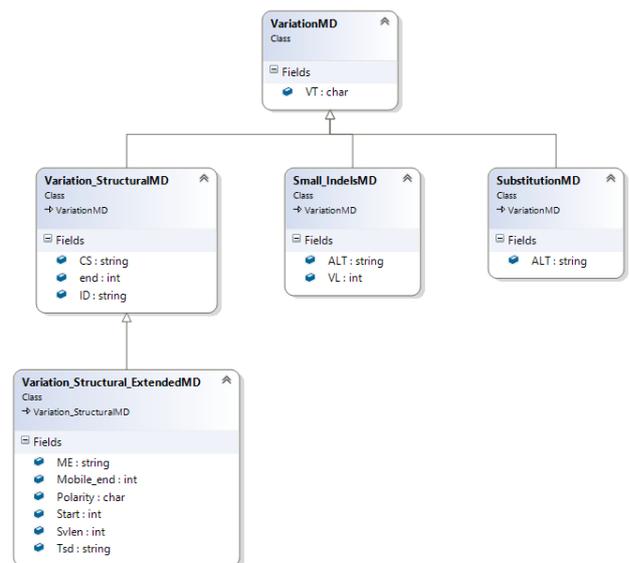


Figure 2. Class design for variations

The class, which is designed to hold a variation (haplotype) of type small insertion or deletion, is "Small\_IndelsMD". This class is derived from the "VariationMD" class and it

has two fields, except for the "VT" field of the "VariationMD" class. The first field ("VL") is of type integer and holds the length of the variation (haplotype). What we mean here by the term "variation length" is the difference between the length of the "ALT" field and the length of the reference allele. In the variations of type short insertion, since the length of the alternate allele is greater than the length of the reference allele, the value of the "VL" field is always positive. In the variations of type short deletion, the situation is exactly the opposite, that is, the value of the "VL" field is always negative. The "VL" field is actually one of the key fields and is also used for a significant operation, such as obtaining the raw sequence data using genotypes/haplotypes of an individual. Thanks to this field, the total change in a particular region can be easily found. Also, raw sequence data can be obtained, by creating an array with length (length of the particular region + total change). On the other hand, the second field ("ALT") is used to hold the haplotype and it is naturally of type string.

The class, which is designed to hold a variation (haplotype) of type substitution, is "SubstitutionMD". As in the "Small\_IndelsMD" class, this class is also derived from the "VariationMD" class. The "SubstitutionMD" class has only one field, except for the "VT" field of the "VariationMD" class. The name of this single field is "ALT" and it is used for the same purpose as the "ALT" field of the "Small\_IndelsMD" class. In the operation of obtaining raw sequence data by using personal genotypes/haplotypes, the "SubstitutionMD" documents are not considered when calculating the total length change.

Two classes "Variation\_StructuralMD" and "Variation\_Structural\_ExtendedMD" were designed to store the structural variations. As can be seen from Figure-2, the class "Variation\_structuralMD" is derived from the class "VariationMD", whereas the class "Variation\_Structural\_ExtendedMD" is derived from the class "Variation\_StructuralMD". The class "Variation\_StructuralMD" was designed to store five different structural variations: CNV, DEL, DUP, INS:MT and INV. On the other hand, except for the "VT" field of the "VariationMD" class, "Variation\_StructuralMD" has three attributes and these are "ID", "CS" and "end". These attributes are used to store id of the structural variation, source call set, and end coordinate of the variation, respectively.

The "Variation\_Structural\_ExtendedMD" class was designed to hold structural variations of types INS:ME:ALU, INS:ME:LINE1 and INS:ME:SVA. This class has six fields, except for the "VT" field of the "VariationMD" class and the three fields defined in the "Variation\_StructuralMD" class. The names of these fields are "Svlen" (int), "Tsd" (string), "ME" (string), "Start" (int), "Mobile\_end" (int) and "Polarity" (char). According to the order, these fields store the length of the structural variation (difference in length between REF and ALT),

precise target site duplication for bases, mobile element name, start and end positions, and the polarity.

### Class Design for Personal Genotypes

In the previous section, considering only one of the chromosome pair, document types were described that can hold the variation (haplotype) on any position of the chromosome. In this section, document types will be explained that can express the variation on any position, taking both chromosomes of the chromosome pair (both alleles) into consideration. Also, since males have single X and single Y chromosomes, the document types that can store the variations on these chromosomes were also designed. Consequently, a total of 5 classes are used to hold the personal genotype (or haplotype for X and Y chromosomes of males) at any position. The designed classes, and inheritance and polymorphism relations between these classes are shown in Figure-3. As seen from the figure, "Variation\_PositionMD" is the class (super class) at the top of this hierarchy. The class "Variation\_PositionMD" has a single field, which is an integer variable representing the position of the variation on the chromosome. It was mentioned in the "Personal\_Genomes" collection that the "GTs" field of the "RegionMD" class holds the variation-based personal genotypes/haplotypes and that the data type of this field is the "Variation\_PositionMD" array. Since "Variation\_PositionMD" is a super class, objects of all other classes descended from this class can be kept in the "GTs" field.

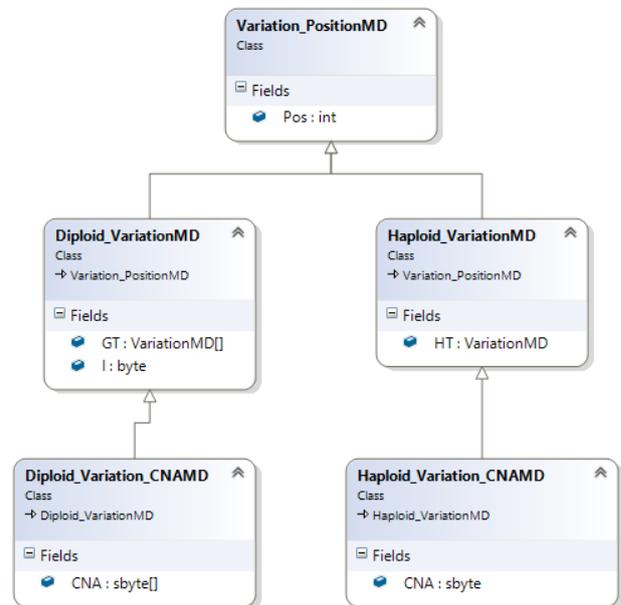


Figure 3. Class design for genotypes

The "Diploid\_VariationMD" is one of the classes derived from the "Variation\_PositionMD" class and, as the name implies, it is used in diploid situations. Here, what is meant by diploid situations are autosomal chromosomes and sex chromosomes of females. Except for the "Pos" field inherited from the "Variation\_PositionMD" class, the

"Diploid\_VariationMD" class has two fields. The field with the name "GT" is an array of type "VariationMD" and thus, it can hold the variations (genotype) on both alleles. Thanks to the fact that the class "VariationMD" is a super class and that the other classes representing the variation types derive from this class, the "Diploid\_VariationMD" class supports all possible cases. The second field ("I") of this class gives us information about the "GT" field. When both alleles are considered, the possibilities that can emerge can be expressed as: The existence of the variation in only first allele, the existence of the variation in only second allele, the existence of the same variation in both alleles, and finally, the existence of distinct variations in both alleles. Apart from these four main cases, in some structural variation types (CNV, DEL, DUP), same variation whose only CNA values are different can be observed in both alleles. Consequently, there are five different cases in total. Accordingly, the "I" field can take five different values to support these five different possible cases. The field "I" takes the values 0, 1, 2, 3 and 4 for the cases above, respectively. There is an important reason for adding the "I" field: the idea of saving space. In the case that the same variation is seen on both alleles, only one of these variations is saved in the "GT" field and "2" is assigned to the "I" field to express this case. Also, the number of the occurrences of the same variation on both alleles is very huge. Therefore, substantial space savings are achieved. In addition, the value of the "I" field directly indicates the genotype status. Therefore, regardless of the "GT" field, information about the alleles (e.g. homozygote/heterozygote) can be obtained.

"Haploid\_VariationMD" is another class derived from the "Variation\_PositionMD" class and it is used to keep the variations (haplotypes) detected on the sex chromosomes of males. Except for the "Pos" field inherited from super class, "Haploid\_VariationMD" class has only one field ("HT"), and this field's data type is "VariationMD". Since male sex chromosomes are not diploid, "HT" field can hold a variation on any position in a single allele. As in "Diploid\_VariationMD", the "Haploid\_VariationMD" class can hold all variation types.

As stated before, in some structural variation types (CNV, DEL, DUP), same variation whose only CNA values are different can be observed in both alleles. The classes "Diploid\_Variation\_CNAMD" and "Haploid\_Variation\_CNAMD" were developed to store such a genotype/haplotype and they are utilized only for the three types of structural variations specified above. In fact, these two classes are equivalent. The "Diploid\_Variation\_CNAMD" class is used in diploid cases, whereas the "Haploid\_Variation\_CNAMD" class is used in haploid cases. Apart from the fields inherited from the upper classes, the class "Diploid\_Variation\_CNAMD" has only one field ("CNA" of type sbyte[]). As might be expected, this field holds the CNA (Copy Number Allele) values. Finally, the "Haploid\_Variation\_CNAMD" class represents the structural variations (CNV, DEL and DUP) detected in male sex chromosomes.

### 3. RESULTS AND DISCUSSION

Testing the database and indexes with real genetic data and evaluating the results of the analysis are just as valuable and important as designing. For this purpose, the variation data published by 1000 Genome Project was used. The various physical properties of the computer, where the database is installed and which was used to test the system, are shown in Table-11. In addition, other parameters/technologies related to the database are as follows: "MongoDB Community Edition 3.4" version of mongo database was used. Besides, "The official MongoDB C#/.NET Driver version 2.3" was selected as the driver.

Table 11. The properties of the test computer

Property	Value
Computer name-version	Asus K55VJ-SX077D
Operating system	Windows 8.1 Pro 64 bit
Processor type and speed	Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz (3.40 GHz TB)
Processor cache	6 MB Intel® Smart Cache
System memory (RAM)	8 GB DDR3 1600 MHz
Disk capacity and speed	750 GB 7200rpm
Disk interface	2.5" SATA

There are two very significant parameters when evaluating both No-sql databases and other types of databases: The space occupied by the recorded data on the hard disk and the average completion times of various queries. In this study, the database, which contains the genotypes (on the whole-genome) of 2504 individuals and other related data (e.g. reference chromosome sequences, generic variations and etc.) was meticulously analyzed according to both criteria. The size of the space required to store any data in the database is a very important criterion for designers. In particular, given the fact that human DNA is composed of approximately 3.2 billion base pairs, this criterion becomes even more important. On the other hand, besides the space occupied by variation-based personal genetic data, the size of the indices created to speed up the query results should also be taken into account. In the light of this information, Table-12, which shows the results (space requirements of "Personal Genomes" collection and related indices) of the analyses, is shown below.

Table 12. The spaces occupied by the "Personal Genomes" collection and the related indices

Gender	"Personal Genomes" Collection (MB)	Index-1 (KB)	Index-2 (KB)	Index-3 (MB)	Total (MB)	Total without Index-3 (MB)
Female	70.227	18.884	30.651	41.032	111.307	70.275
Male	69.148	18.675	41.65	55.757	124.964	69.207

The values in Table-12 reflect the average of 2504 people. That is, the values in the table relate to one person. In addition, there are other significant points to be considered in this table. The first of these is that the values in the columns “Index-1” and “Index-2” are in kilobytes unlike the other columns. On the other hand, the index-1 values are consistent with the values of the “Personal\_Genomes” collection, whereas the index-2 and index-3 values are not very consistent. The reason of this is related to the way the mongo database stores indices. In fact, the most noticeable thing on the table is the index-3 values. When the index-3 values are compared with index-1 and index-2 values, the gap between them is clearly visible. Such a huge difference is quite surprising, and this issue will be discussed again later. As already mentioned, one of the main purposes of this study is to keep personal genetic data in the database with minimum space requirement. When this case is considered, the most important part of the table is the values in the last two columns. Due to the special case of index-3, two different calculations were made. In the case where index-3 is not included in the total, variation-based genetic data of a person for the whole genome occupies approximately 70 MB in the database. Although the values related to index-3 are specified in Table-12, the use of index-3 was later abandoned. The reason for the abandonment of index-3 will be explained later.

Besides the size of the space needed to store data, there are other important criteria used to evaluate the success of the database: The time to save the data into the database, the size of the spaces where data is stored in primary memory, and the response times of the database to various queries. As already mentioned, personal genetic data are recorded in the Mongo database in the form of regions. The recording times of all regions that make up the genome (in short, genome recording times) are nearly 65 and 64 seconds for female and male, respectively. In fact, all regions that make up a person's genome can be recorded in the database in one go. In this case, the average recording time of the genome will be much less. Besides the space where personal genetic data is stored on the hard disk, the space they occupy in the primary memory is also extremely important. Since the RAM capacity of the computer is limited, the fit of the genetic data of as many people as possible into the RAM, which is of limited size, means that the clinical applications will take less time. In the light of this information, the average RAM spaces occupied by the objects of type "RegionMD" were calculated. Except for the regions of male sex chromosomes, the average size of the RAM space occupied by one region is approximately 0.3 MB. Since male sex chromosomes are haploid, the average RAM space occupied by a region belonging to these chromosomes is approximately 0.25 MB. The total RAM spaces required for the whole-genome of an individual are nearly 539 and 531 MBs for female and male, respectively. Finally, the total number of regions forming these genomes are nearly 1817 and 1799 for female and male, respectively. Please note that the values

specified here are the average of 2504 people (for sex chromosomes, average values were computed based on the male and female counts).

Although the time required to store variation-based personal genetic data into the database is important, more important than this, is the response times of the database to various queries that are frequently performed in clinical applications. The reason is quite obvious: The personal genetic data is recorded once in the database. On the other hand, this data can be frequently questioned in clinical operations. Here, there are a number of factors that affect completion times of the queries. Especially, the elements that are directly related to the design, considerably affect query performance and data size, depending on the quality of the design. In the previous chapter, it was stated that the design of the database, related documents and indices was carried out in line with the needs of various clinical applications. From this point of view, various queries were devised that these applications can frequently perform, and the related analysis were carried out using these queries. The devised queries are shown in Table-13.

Table 13. The queries utilized to test the indices

Query No	Explanation
1)	A genotype on a particular position (Chr No:1, Position: 196696932)
2)	The genotypes on two close positions (Chr No:1, Positions: 196696932, 196659236)
3)	The whole of the region containing the position (Chr No:1, Position: 196696932)
4)	Two distant regions containing the positions (Chr No:1, Positions: 100, 196696932)

Table-14 shows both the response times of the database to the queries given above and the memory space it used while running the queries. Note that the time values in Table-14 are in milliseconds and size values are in megabytes. In the analyses made to calculate these values, the method used is as follows: The computer is restarted before each query is run and the query is run when the hard disk usage is 0%. The same query is executed a second time immediately after (without losing any time) the first query is completed. Namely, the same query is run twice in the same session. Actually, depending on the number of people specified in the query, the same query is run again and again (within a loop) for different people. In short, in a single session, the actual run count of the query is more than 2. This situation is clearly shown in the table. The iteration count in the table indicates this. In addition, these operations were repeated 10 times for each of the different triplets (query, case, index). Therefore, all the values shown in the table are average values. On the other hand, although there are 2504 people in the database, the number of people queried in the analysis is 2500. The reason is that 2500 can be divided by 5, 25, and 100 (three different cases in terms of people count).

Table 14. The effect of indices and number of queried people on time and space \*

		Individual Count & Iteration Count (Total = 2500 Individuals)												
		500 & 5 (Case-1)				100 & 25 (Case-2)				25 & 100 (Case-3)				
		Time	Time	Space	Space	Time	Time	Space	Space	Time	Time	Space	Space	
Queries (Qry) & Indexes	Qry-1	Indexes 2&3	43652	19616	920.4	922.5	45671	19396	939.6	921.7	47127	19711	917	917.5
		Index 2	43557	19197	905.7	905.4	45566	19212	918.7	920	46954	19700	915	915.4
		Index 3	254457	155299	2965	3195	257771	159582	3004	3075	263831	166525	2939	3177
	Qry-2	Indexes 2&3	248408	204996	3102	3114.3	265887	211644	3027.6	3084.5	282381	207522	3116.3	3142
		Index 2	250324	205160	3014.7	3144.2	260415	209428	3017.4	3098	265353	209142	3012.4	3129.7
		Index 3	1669826	1636897	3164	3485	1677501	1661483	3146	3468	1687321	1662153	3148	3470
	Qry-3	Indexes 2&3	292056	278209	915.7	915.2	319761	285249	914.9	915	323700	295155	905.4	904
		Index 2	293012	279306	899.4	900.2	317891	285998	899.5	900.3	321647	292626	899.5	900.7
		Index 3	505795	398305	2996	3165	509475	405510	3009	3200	513296	411233	3008	3174
	Qry-4	Indexes 2&3	777042	751526	2918.3	2994.5	800127	751800	3030.8	3160.5	800499	756581	3112.1	3200.1
		Index 2	775562	746223	2980	2983	798224	747788	2957.7	3120.2	799002	756106	2999	3114
		Index 3	1847894	1815398	3211	3375	1853521	1822988	3177	3441	1873800	1823588	3219	3477

\* Time values are in milliseconds and size values are in megabytes.

The values in Table-14 allow us to make very important inferences. First of all, let us remember the crucial and intriguing question about indexes asked in the "Personal\_Genomes" section: "If the third index fully covers the second index, why is the second index needed?". Surprisingly, it was observed that index-3 does not provide the expected gain, and even, contrary to what is expected, increases query times so much more. As seen, there is a cliff (in terms of both time and space) between the second and third indices. For these reasons, index-3 was abandoned.

The second inference obtained from the values specified in Table-14 is related to the queries. Consider query-1 and query-2 for case-1. In query-1 section, the first reading value for index-2 is 43557. On the other hand, in query-2 section, the first reading value for index-2 is 250324. That is, the query-2 time is almost six times the query-1 time. Also, the amount of RAM space used by Mongo database is 905.7MB in the first run of query-1, whereas this value is 3014.7MB for query-2. Namely, the space utilized for query-2 is more than 3 times the space utilized for query-1. In fact, these two queries are quite similar. The only difference between them is that query-1 queries a single genotype (at a specific location), whereas query-2 queries two genotypes (at two different locations). Furthermore, the two positions queried by query-2 are relatively close to each other. Therefore, in a certain part of the queried people, these two positions are more likely to fall into the same region. In short, rather than using Query-2, it makes more sense to run query-1 twice for two different positions. A similar situation applies to the relation between query-3 and query-4.

The third inference made from the results of the analysis is related to the number of people queried in each query. The increase in the number of people queried in a single iteration reduces the total time spent for 2500 people. It

would be a logical choice to query (in one go) the genetic data of as many people as possible in the queries. No obvious difference was observed between the cases in terms of space. The last inference made from the values is that the time spent in the second run of the same query is less than the first run. This is true for all queries and all cases, but the proportional difference between the times required for first run and second run varies depending on the query type.

The recent study [44] of Çakıröz & Sevinç, published in 2018, uses relational database to store variation-based personal genetic data. In this study, various external data formats were designed to hold variations and genotypes, and all genotypes of a chromosome were (collectively) recorded in the database in the Varbinary binary format and in a compressed form. Namely, in this way, 23 and 24 rows are used to store variation-based genetic data for the entire genome of females and males, respectively. As in our study, the proposed method by Çakıröz & Sevinç (2018) was also tested with real data of 2504 people, published by 1000 Genome Project. On the other hand, although it has some limitations (For instance, it does not allow complex SQL queries, most transactions are handled at the application layer.), the method proposed by Çakıröz & Sevinç (2018) has given the most successful results in terms of space requirement among recent similar studies. Since it uses the same data set and gives successful results, our method was compared with this method using the same configurations. The comparison results regarding the space requirements are shown in Table-15. Although there is a certain difference between the two methods in terms of hard disk space requirement, both methods resulted in significant space savings. Compared to the space required to store the raw sequence data, the proposed method by Çakıröz & Sevinç (2018) yielded a space gain of 99.74%. This value is 98.86% for our method. On the other hand, in our method, the total RAM spaces required for the whole-genome of an individual are nearly 539 and 531 MBs for female and male, respectively. These values are slightly

lower in the other method, but there is no significant difference.

Table 15. Comparison table for size requirements

Gender	Size Requirements on Harddisk		Size Requirements in RAM	
	Our Method	Method of Çakırgöz & Sevinç (2018)	Our Method	Method of Çakırgöz & Sevinç (2018)
Female	70.275 MB	15.270 MB	538.93 MB	496.630 MB
Male	69.207 MB	15.086 MB	531.34 MB	484.463 MB

In the method of Çakırgöz & Sevinç (2018), during the process of reading the data stored in the database and transforming it into class objects, the data are passed through many stages. First, the data held in the form of compressed byte array is fetched from the database, then, this data is decompressed, and finally, the decompressed data is transformed into class objects. According to the analysis results (using the same computer configuration), these three operations take 340.96 milliseconds for chromosome-1 on the average (for one person). For 2500 people, these three processes take a total of 852400 milliseconds, not including the time it takes to find the region or genotype and return it. So, no matter what query type, this time is inevitable. When the Table-14 showing the times taken by our method for 4 different query types is examined, even query-4 (Index-2, case-1, first run), which requires the most time among query types, was realized below this time. In other query types, the difference is much greater. In short, our method performed much better in terms of query times.

#### 4. CONCLUSION

In this study, a document-based no-sql database (Mongo) was utilized for the organization of variation-based personal genetic data, and the space requirements and query performances of this database was computed. Thanks to both the advantages of no-sql database and our class designs that support all types of variation, various clinical applications and studies using personal variation data will be able to use the data in the database directly without the need for any data conversion.

After the database was created, the personal genetic data of 2504 people were recorded in the database in the form of regions. As a result of the analyses made on the database, it was seen that the proposed method provides very important gains. The various important gains of the study are as follows: Variation-based genetic data of a person for the whole genome occupies approximately 70 MB in the database. By using this proposed method, the hard disk space required to store all the variations in the genome of a person is approximately 1.14% of the space required to store the raw sequence of this person. In terms of hard disk space, this method provides a saving of approximately 98.86%. Except for the regions of male sex chromosomes, the average size of the RAM space occupied by one region (containing up to 2500 genotypes) is approximately 0.3 MB. This value is approximately 0.25 MB for a region of

male sex chromosomes. On the other hand, the RAM spaces required to store the genotypes/haplotypes in the whole-genome of a female and male are approximately 538 MB and 531 MB, respectively. These values are nearly 8.8% of the space required to store the raw sequence data of this person. In terms of RAM space, this method provides a saving of approximately 91.2%.

#### REFERENCES

- [1] N. J. Schork, "Personalized medicine: time for one-person trials", *Nature*, 520(7549), 609-611, 2015.
- [2] C. Gonzaga-Jauregui, J. R. Lupski, R. A. Gibbs, "Human genome sequencing in health and disease", *Annual review of medicine*, 63, 35-61, 2012.
- [3] 1000 Genomes Project Consortium, "A map of human genome variation from population-scale sequencing", *Nature*, 467(7319), 1061, 2010.
- [4] 1000 Genomes Project Consortium, "An integrated map of genetic variation from 1,092 human genomes", *Nature*, 491(7422), 56-65, 2012.
- [5] 1000 Genomes Project Consortium, "A global reference for human genetic variation", *Nature*, 526(7571), 68-74, 2015.
- [6] 1000 Genomes Project Consortium, "An integrated map of structural variation in 2,504 human genomes", *Nature*, 526(7571), 75-81, 2015.
- [7] Internet: 1000 Genomes Project Consortium, /vol1/ftp/release/20130502/ directory, ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/, 05.01.2021.
- [8] Internet: 1000 Genomes Project Consortium, /vol1/ftp/release/20130502/supporting/bcf\_files directory, ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/supporting/bcf\_files, 05.01.2021.
- [9] M. Futema, V. Plagnol, R. A. Whittall, H. A. W. Neil, S. E. Humphries, "Use of targeted exome sequencing as a diagnostic tool for Familial Hypercholesterolaemia", *Journal of medical genetics*, 49(10), 644-649, 2012.
- [10] P. N. Taylor, E. Porcu, S. Chew, P. J. Campbell, M. Traglia, S. J. Brown, Y. Memari, "Whole-genome sequence-based analysis of thyroid function", *Nature communications*, 6(1), 1-11, 2015.
- [11] International Human Genome Sequencing Consortium, "Finishing the euchromatic sequence of the human genome", *Nature*, 431(7011), 931, 2004.
- [12] I. Dunham, E. Birney, B. R. Lajoie, A. Sanyal, X. Dong, M. Greven, J. Dekker, et. al., "An integrated encyclopedia of DNA elements in the human genome", *Nature*, 489, 57-74, 2012.
- [13] Cancer Genome Atlas Research Network, "The cancer genome atlas pan-cancer analysis project", *Nature genetics*, 45(10), 1113, 2013.
- [14] G. F. Gao, J. S. Parker, S. M. Reynolds, et. al., "Before and after: comparison of legacy and harmonized TCGA genomic data commons' data", *Cell systems*, 9(1), 24-34, 2019.

- [15] J. Carrot-Zhang, N. Chambwe, J. S. Damrauer, et. al., “Comprehensive analysis of genetic ancestry and its molecular correlates in cancer”, *Cancer Cell*, 37(5), 639-654, 2020.
- [16] Internet: Cancer Genome Atlas Research, GDC, <https://portal.gdc.cancer.gov/>, 05.01.2021.
- [17] H. Li, J. Ruan, R. Durbin, “Mapping short DNA sequencing reads and calling variants using mapping quality scores”, *Genome research*, 18(11), 1851-1858, 2008.
- [18] H. Li, R. Durbin, “Fast and accurate short read alignment with Burrows–Wheeler transform”, *Bioinformatics*, 25(14), 1754-1760, 2009.
- [19] R. Li, Y. Li, K. Kristiansen, J. Wang, “SOAP: short oligonucleotide alignment program”, *Bioinformatics*, 24(5), 713-714, 2008.
- [20] K. Chen, J. W. Wallis, M. D. McLellan, et. al., “BreakDancer: an algorithm for high-resolution mapping of genomic structural variation”, *Nature methods*, 6(9), 677-681, 2009.
- [21] D. C. Koboldt, K. Chen, T. Wylie, et. al., “VarScan: variant detection in massively parallel sequencing of individual and pooled samples”, *Bioinformatics*, 25(17), 2283-2285, 2009.
- [22] H. Li, B. Handsaker, A. Wysoker, et. al., “The sequence alignment/map format and SAMtools”, *Bioinformatics*, 25(16), 2078-2079, 2009.
- [23] A. McKenna, M. Hanna, E. Banks, et. al., “The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data”, *Genome research*, 20(9), 1297-1303, 2010.
- [24] J. Dean, S. Ghemawat, “MapReduce: simplified data processing on large clusters”, *Communications of the ACM*, 51(1), 107-113, 2008.
- [25] Internet: VCFtools, <https://vcftools.github.io/specs.html>, 05.01.2021.
- [26] S. Grumbach, F. Tahi, “Compression of DNA sequences”, **DCC93: Data Compression Conference**, 340-350, IEEE, 1993.
- [27] E. Rivals, J. P. Delahaye, M. Dauchet, “A guaranteed compression scheme for repetitive DNA sequences”, **Data Compression Conference**, 453-453, IEEE Computer Society, March, 1996.
- [28] A. Apostolico, S. Lonardi, S. “Compression of biological sequences by greedy off-line textual substitution”, **DCC 2000, Data Compression Conference**, 143-152, IEEE, March, 2000.
- [29] X. Chen, S. Kwong, M. Li, “A compression algorithm for DNA sequences and its applications in genome comparison”, *Genome informatics*, 10, 51-61 1999.
- [30] S. Christley, Y. Lu, C. Li, X. Xie, “Human genomes as email attachments”, *Bioinformatics*, 25(2), 274-275, 2009.
- [31] D. A. Wheeler, M. Srinivasan, M. Egholm, et. al., “The complete genome of an individual by massively parallel DNA sequencing”, *Nature*, 452(7189), 872-876, 2008.
- [32] S. Kuruppu, S. J. Puglisi, J. Zobel, “Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval”, **International Symposium on String Processing and Information Retrieval**, Springer, Berlin, Heidelberg, October, 201-206, 2010.
- [33] M. D. Cao, T. I. Dix, L. Allison, C. Mears, “A simple statistical algorithm for biological sequence compression”, **Data Compression Conference (DCC'07)**, 43-52, IEEE, March, 2007.
- [34] V. Mäkinen, G. Navarro, J. Sirén, N. Välimäki, “Storage and retrieval of highly repetitive sequence collections”, *Journal of Computational Biology*, 17(3), 281-308, 2010.
- [35] K. Shvachko, H. Kuang, S. Radia, R. Chansler, “The hadoop distributed file system”, **IEEE 26th symposium on mass storage systems and technologies (MSST)**, 1-10, IEEE, May, 2010.
- [36] J. Dean, S. Ghemawat, “MapReduce: a flexible data processing tool”, *Communications of the ACM*, 53(1), 72-77, 2010.
- [37] H. Nordberg, K. Bhatia, K. Wang, Z. Wang, “BioPig: a Hadoop-based analytic toolkit for large-scale sequence data”, *Bioinformatics*, 29(23), 3014-3019, 2013.
- [38] A. Schumacher, L. Pireddu, M. Niemenmaa, et. al., “SeqPig: simple and scalable scripting for large sequencing data sets in Hadoop”, *Bioinformatics*, 30(1), 119-120, 2014.
- [39] M. S. Wiewiórka, A. Messina, A. Pacholewska, et. al., “SparkSeq: fast, scalable and cloud-ready tool for the interactive genomic data analysis with nucleotide precision”, *Bioinformatics*, 30(18), 2652-2653, 2014.
- [40] M. Masseroli, P. Pinoli, F. Venco, et. al., “GenoMetric Query Language: a novel approach to large-scale genomic data management”, *Bioinformatics*, 31(12), 1881-1888, 2015.
- [41] M. Zaharia, M. Chowdhury, T. Das, et. al., “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing”, **9th {USENIX} Symposium on Networked Systems Design and Implementation**, 15-28, 2012.
- [42] V. Bafna, A. Deutsch, A. Heiberg, et. al., “Abstractions for genomics”, *Communications of the ACM*, 56(1), 83-93, 2013.
- [43] C. Kozanitis, A. Heiberg, G. Varghese, V. Bafna, “Using Genome Query Language to uncover genetic variation”, *Bioinformatics*, 30(1), 1-8, 2014.
- [44] O. Çakırgöz, S. Sevinc, “Organization of Variation Based Personal Genetic Data with Relational Database”, *International Journal of InformaticsTechnologies*, 11(3), 295–307, 2018.