Querying Genome Databases

Extended Abstract

Giansalvatore Mecca[1]

[1]*Dipartimento di Matematica, Informatica ed Economia, Universitá della Basilicata*

Abstract

Genome Databases contain genetic information about human beings and other species of living organisms. Since 1986, year in which the *Human Genome Project* started, they have grown exponentially in size every year. Due to the unusual structure of the stored information, traditional DBMS and query languages have provided only little support so far. In this paper we discuss some of the issues related to genomic information management.

1. Introduction

The *Human Genome Project* is a research effort started back in 1986 as an initiative of two U.S. institutions, the *Department of Energy (DOE)* and the *National Institute of Health (NIH)*. The focus of the project is the study of genetic information, which represents the base for the life of every living organism. The purpose is to gain a better understanding of the nature of genetic and cellular mechanisms, that will provide an unvaluable source of information for medicine and health care in the next years.

Computer science and especially database technology has been by the side of genetics and molecular biology from the start. In fact, the huge amount of experimental data produced and the great complexity of experimental protocols require sophisticated computing applications to manage and interpret data. The main feature of these applications is represented on one side by the sequential structure of data to be stored, and on the other side by the complex nature of the transformations required to study these data. It is rather apparent the fact that commercial DBMS based on relational technology are not completely suited to this purpose. In fact, commercial products have been rather unsuccessful in this field so far.

This paper tries to sketch some of the issues related to sequential data management in the context of Genome databases, pointing out where the main weaknesses of current technology are. The paper mainly focuses on aspects related to query languages.

Due to the interdisciplinary nature of the matter, the paper is organized as follows. In Section 2 we introduce a few basic notions of molecular biology, necessary to gain a better understanding of the domain of interest and of examples throughout the paper. For a detailed introduction to the subject, we refer the reader to [14, 7]. Then, the Human Genome Project is briefly presented in Section 3. Technological issues related to Genome databases are discussed in Section 4, whereas Section 5 specifically focuses on aspects related to querying genomic data.

SEBD 2022: The 30th Italian Symposium on Advanced Database Systems, June 19-22, 2022, Tirrenia (PI), Italy

giansalvatore.mecca@unibas.it (G. Mecca)

2. Notions of Molecular Biology

The term *genome* is used to refer to the whole genetic information of an organism. This information, stored in *nucleic acids*, the *deoxiribonucleic acid (DNA)*, and the *ribonucleic acid (RNA)*, contains the master blueprint for all cellular structures and activities.

Nucleic acids are complex molecules. Their constituents, called *nucleotides* or *bases*, are simpler molecules that we can divide in two groups: *(i) purines*, that is, *adenine*, usually denoted a, and *citosine* (c); *(ii) pyrimidines*, that is, *guanine* (g) and *timine* (t). Nucleotides bind to form long strands called *polynucleotides*, which are essentially polymers [1] in which nucleotides are linked to each other. A DNA molecule is made of two strands wrapped around each other to form a *double helix*. The double helix spatial configuration is due to weak hydrogenous bonds that are established between nucleotides on the two strands. This bonds, often referred to as *base pairings*, respond to a precise *complementarity* relationship according to which a only pairs with t and c only pairs with g. The bond is directional, that is, the DNA molecule has a head (usually called the $5'$ end) and a tail (called the $3'$ end), and one of the strands is called the *direct strand* and the other the *complementary strand*.

Example 1. The term *DNA sequence* denotes the arrangement of nucleotides in each of the two strands of a DNA molecule. Such sequences are naturally represented as strings over the alphabet $\{\mathtt{a}, \mathtt{c}, \mathtt{g}, \mathtt{t}\}$.

For example, the following is a representation of a DNA fragment.

```
... a c t t g c c a a ...
 5' | | | | | | | | | 3'
... t g a a c g g t t ...
```

The high grade of redundancy, which makes possible to produce high-fidelity copies of DNA molecules, is the very base of the replication process occurring in living cells.

In fact, the genetic information of an organism is stored in one or more long DNA molecules, called *chromosomes*. For example, in human beings, each cell contains in its nucleus 23 pairs of chromosomes. As long as the organism grows, that is, its cells duplicates, this information is passed on to new cells. In particular, whenever a cell duplicates, each of the *daughter cells* receives a copy of the chromosome set of the original cell. In order to do this, the cell first replicates its whole genome. During such replication, hydrogen bonds in DNA molecules break and strands separate; then, free nucleotides in the nucleus are used to build a new complementary strand for each of the original strands, thus originating two identical copies of the cell genome. In this way, each cell in the living organism has in its nucleus the whole genomic information, and can use it to perform its functions.

The means by which chromosomes affect the structure and the functions of living cells are *proteins*, that can be considered a primary component of living things. From the structural point of view, proteins are polymers themselves, made of basic constituents called *amino acids*. There are 20 different amino acids in nature. Amino acids form peptide bonds to create long chains called *polypeptides*; the sequence of amino acids that form a protein is called its *primary structure*, usually expressed as a string over a conventional 20-symbol alphabet $\{A, C, K, M, N, \ldots\}$.

[1] A *polymer* is a long sequence of similar bonded elements.

Indeed, the primary role of nucleic acids is to carry the *encoding* of the structure of specific proteins. We call a *gene* each portion of chromosome containing information about the primary structure of a protein. It is worth noting that only about 10% of the whole human genome is known to contain protein-coding regions. The remaining part consists of noncoding regions – such as *control regions* – and other completely meaningless regions.

Roughly speaking, we can say that there is one gene for each specific protein in a living organism. There are, in fact, at least 100.000 different genes in humans. A simple and powerful code is used for storing such information. In particular, to each non-overlapping triplet of nucleotides in a gene, called a *codon*, corresponds a particular amino acid. Four distinguished codons are used as *stop codons*, that is, to signal the end of the coding sequence. It is easy to note that also in this case the code is redundant, since there are 4^3 different codons and only 20 amino acids (plus the *stop* signal).

When a protein is syntesyzed in a living cell, the DNA fragment representing the corresponding gene undergoes several transformations.

- the DNA sequence is first *transcribed* into an RNA sequence; RNA is a nucleic acid whose structure is very similar to the one of DNA; the only difference is that a new nucleotide, called *uracil* (u), takes the place of timine (t); RNA molecules are single-stranded, so that each RNA fragment can be easily represented as a string over the alphabet $\{\mathtt{a}, \mathtt{c}, \mathtt{g}, \mathtt{u}\}$; when the transcription of a DNA fragment occurs, the two strands separates and one of them is used as a template to build a RNA strand, according to a slightly different complementarity relationship (the only difference is that each a is transcribed to a u);
- then, the RNA molecule travels out of the nucleus, and undergoes a *splicing* process; in fact, genes are not necessarily contiguous inside chromosomes; if we consider the chromosome region containing a gene, there will be some gene regions that actually code for proteins, called *exons*, and some interleaving regions, called *introns*, which do not contain useful information; when DNA is expressed, introns need to be spliced out, so that exons can be connected together and the whole coding region can be processed; such splicing happens in specialized apparates called *spliceosomes*, which take a transcripted RNA sequence and splice out regions corresponding to introns, connecting exons together; the resulting molecule is called *messenger RNA*,
- the resulting RNA sequence is ready to be expressed; to do this, other apparates, called *transfer RNAs*, *translate* the sequence by simply reading the sequence codon by codon – that is, three symbols at a time – and, for each codon, appending the corresponding amino acid to the protein. For example, whenever the codon aug is encountered, the corresponding amino acid, *methionine*, is appended to the amino acid sequence. The four stop codons, namely gct, gcc, gca and gcg, signal that this translation process has to stop.

Example 2. As an example, consider the following DNA fragment representing a gene.

```
   1 2 3 4 5 6 7 8 9 0 1 2 3 4
   t a c g c c a a c t t a c t
5' | | | | | | | | | | | | | | 3'
   a t g c g g t t g a a t g a
```

Suppose also that we know that the exons are from base 01 to base 06 and from base 09 to base 14. When the direct strand undergoes transcription, the following messenger RNA is produced.

```
a u g c g g u u g a a u g a
```

After splicing, the two exons are spliced together, so that the sequence to translate and the resulting protein are the following.

```
a u g c g g g a a u g a

<met> <arg> <glu> <stop>
```

Here, *met, arg* and *val* stand for the *methionine (M), arginine (A)* and *glutamic acid (G)* aminoacids, respectively.

3. The Human Genome Project

The final goal of the Human Genome project is to determine the exact sequence of the whole human genome, i.e. of every gene present in the human race, and the corresponding functions. To give a measure of the effort, it is worth noting that human genomes are about 3 billion base long. If compiled in books, the data would fill an estimated 200 volumes, each the size of a 1000 pages telephone book, and reading it would require about 30 years working around the clock.

The achievement of such an ambitious goal is unfortunately very far away for now, as the current experimental technology does not allow to read sequences longer than a few hundred bases. In fact, time and cost considerations make large-scale sequencing projects totally impractical. Consider that the smallest human chromosome (chromosome Y) is 50 million bases. The best available equipment can sequence only 50.000 to 100.000 bases per year at an approximate cost of $1 to $2 per base. At this rate, an incceptable amount of 30.000 work-years and at least $3 billion would be required for sequencing alone. It is apparent how the study of new sequencing technologies is a major concern in genome research these days.

Such a strong limitation imposes complex experimental protocols to reduce long chromosomes into manageable fragments. On the other side, the study of chromosomes is usually focused on different kinds of information rather than on the actual sequence. The objective is to build *maps* of chromosomes, that is, to find the approximate position of relevant sites on the chromosomes, corresponding to functionally important regions.

Due to the enormous size of human genomes and to the difficulty to perform experiments on human beings, much of the research work is performed on other species, as mice, insects, yeasts and bacteria. The following table gives an idea of the size of different genomes studied so far.

Comparative Sequence Sizes	
Yeast Chromosome 3	350.000 bases
Escherichia Coli (bacterium) genome	4.6 million bases
Entire yeast genome	15 million bases
Smallest human chromosome (Y)	50 million bases
Largest human chromosome (1)	250 million bases
Entire human genome	3 billion bases

4. Genome Databases

The huge amount of data generated by genome research will be used as a primary information source for human biology and medicine into the future. A main issue is therefore the management and distribution of genome data. Experimental information, sequences, maps need to be collected and stored in order to be fully accessible to the research community.

Clearly, a strong database technology is necessary to provide full support for these activities. Unfortunately, traditional DBMS have shown serious shortcomings in terms of genome data management [9].

The first category of limitations is concerned with the data model. In fact, DNA data requires a flexible *sequence data type* to be properly represented. Relational database systems do not provide such a type, so that the only way to implement a DNA sequence type is to store sequences as text strings. This technique, anyway, do not seem to be completely satisfactory, since, on one side a 8-bit per base representation is a clear waste of disk space - DNA sequences can be easily represented in a compress way by using only 2 bits per base; on the other side, manipulation of text fields, when present in commercial systems, is usually associated with serious limitations in terms of access. For example, indexing is usually not provided, and operators such as *substring* and *concatenation* need to be implemented by the user.

Indeed, these limitations have made the role of relational DBMSs a very poor one in genome research. As an example, consider the world most famous sequence database, *GenBank* [3]. *GenBank* is the NIH genetic sequence database, a collection of all known DNA sequences. There are approximately 230 million bases and 238.000 sequences as of December 1994. It represents a fundamental reference for the research community, since often sequenced DNA regions are required to be submitted to GenBank for electronic publication even before they are published in printed form. Information in GenBank is organized in records called *entries*; each *entry* contain a sequence along with some *annotations*, which provide details about the source organism, gene or DNA region, and the internal structure of the reported sequence, i.e. the number and position of exons, introns and relevant control regions.

Although GenBank uses a relational DBMS for internal data manipulation, external access to the database is not possible through a relational interface. In fact, the most common way to access the database is to download a flat file, text-based dump of the current version, in which all entries are listed sequentially.

Other very popular databases have also adopted alternative models of representation. For example, the *ACeDB* [19], the integrated *nematode worm* genome database, use a specialized hierarchical file system to represent information, and the user can navigate the database by following links from one record to the other. This model has proven very successful in the genome community, and other researchers are implementing databases for other species based on a similar architecture.

The lack of a unifying model for representing information is probably the first obstacle to be solved on the way of a real integrated access to genome data. In fact, the different databases spread in the world, containing informations about proteins, maps, metabolic pathways, speak different languages in terms of data models and data access facilities, so that today is not possible to ask for biologically relevant queries that involve data contained in different databases.

5. Query languages for genomic data

The other important cathegory of limitations of current database systems is related to query languages. The very rich structure of genomic sequences makes querying a critical activity. We can say that there are two different aspects of queries over sequences that have to be taken into account.

On one side, an expressive query language should provide powerful *pattern matching* capabilities. In fact, testing similarities of newly sequenced sequences against existing databases is a fundamental activity. Right now, a researcher willing to test the degree of omology of a new sequence against a database needs to recur to dedicated packages. In particular, there are some well known query servers, for example *BLAST* [1] or *FASTA* [16] which can be invoked through the Internet to test local alignment of short sequences against GenBank. Unfortunately, such specialized resources do not have a flexible query interface, so that, for example, one cannot specify queries such as *"test the similarities of sequence X against any sequence in the database in which the organism field equals* human*"*, in which additional conditions are specified to narrow the range of interesting sequences and speed up the search.

To solve these problems, in the literature, several *pattern languages* have been proposed (e.g. [12, 18]). These languages usually provide a flexible syntax for defining *patterns*, that is, expressions that correspond to a set of strings over a given alphabet. In principle, one can think to enrich a relational query language, say SQL, by means of an extended *selection operator*, that, given a pattern, allows for extracting the set of strings in the database that match the pattern. Although the resulting language might represent a good solution to many sequence analysis problems, it would still be too weak in terms of *data restructurings*.

In fact, *restructurings* also play a central role. The raw sequence data usually provides little insight about the structure and functions of the corresponding DNA. For example, given a DNA sequence, one would like to be capable of finding all the regions of interest inside the sequence, that is, *(i)* exons; *(ii)* introns; *(iii)* important control regions responding to particular patterns or *consensus sequences*. More important, all these conceptually derived data should be accessible exactly as the base, stored data is, that is, the language should provide a flexible *view definition* mechanism to *restructure* information in the database. As an example, a simple way to represent exons and introns inside a given sequence is to identify them by means of the starting and ending position inside the sequence, but then, a primitive to retrieve the corresponding subsequence is necessary.

Moreover, sequences often need to be transformed. For example, it should be possible to find the reverse complement of a given sequence, or, given the set of exons of a gene inside a DNA sequence, splice exons together and translate them into the corresponding protein primary structure. This means that some primitives for computing sequence mappings are needed, potentially introducing new sequences in the database.

On the basis of the these considerations, we can briefly sketch a set of important requirements for a genome sequence query language as follows:

- the language should be *expressive*, that is, able to capture the complexity of genomic data, both in terms of pattern extraction and sequence restructurings;
- the language has to be declarative and easy to use; it should have a clear semantics and a

nice syntax, and a suitable view definition mechanism, so that it can eventually become an effective tool for biologists;

- last but not least, the language should be *safe* in the sense that queries should always terminate, and the semantics be tractable; note, in fact, that queries on sequences may end up in nonterminating computations even when the base alphabet is finite; in fact, by growing in lenght, sequences can easily become infinite.

It is apparent how a good query language is the result of a good compromise between expressive power and effective computability. Various proposals have appeared in the recent literature about languages for querying sequences. Indeed, in most cases the trade-off between *expressiveness*, *finiteness* and effective *computability* is a hard one. In many cases [8], powerful logics for expressing sequence transformations are proposed, but a great part of the expressive power has to be sacrificed to enforce finiteness. In other cases [11], both expressiveness and finiteness are achieved, but at expense of an effective procedure for evaluating of queries, *i.e.*, at the expense of an operational semantics.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers and D. J. Lipman. Basic Local Alignment Search Tool. *Jour. of Mol. Biology*, 215:403–410, 1990.

[2] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier and Z. Zdonik. The object oriented database manifesto. In *1st Intl. Conference on Deductive and Object Oriented Databases (DOOD)*, Kyoto, 1989.

[3] C. Burks, M. Cassidy, M. J. Cinkosky, K. E. Cumella, P. Gilna, J. Hayden, G. M. Keen, T. A. Kelley, M. Kelly, D. Kristofferson and J. Ryals. GenBank. *Nucleic Acids Res.*, 19*(Suppl.)*:2221–2225, 1991.

[4] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Data Bases*. Springer-Verlag, 1989.

[5] W. Clocksin, C. Mellish. *Programming in Prolog*. Springer-Verlag, 1981.

[6] J. Collado Vides. The search for a grammatical theory of gene regulation is formally justified by showing the inadequacy of context–free grammars. *Computer Appications in the Biosciences*, 7(3), 1991.

[7] Department of Energy. *Primer on Molecular Genetics*, 1994.

[8] S. Ginsburg and X. Wang. Pattern matching by RS-Operations: Towards a unified approach to querying sequenced data. In *11th ACM SIGMOD Symp. on Principles of Database Systems (PODS)*, 1992.

[9] N. Goodman, S. Rozen and L. Stein. A glimpse at the DBMS Challenges Posed by the Human Genome Project. Whitehead Institute for Biomedical Research, 1994.

[10] N. Goodman, S. Rozen and L. Stein. Requirements for a Deductive Query Language in the MapBase Genome Mapping Database. In *Workshop on Programming with Logic Databases (following ILPS '93)*, 1993.

[11] G. Grahne, M. Nykanen and E. Ukkonen. Reasoning about strings in databases. In *13th ACM SIGMOD Symp. on Principles of Database Systems (PODS)*, 1994.

[12] C. Hegelsen and P. R. Sibbald. PALM – A pattern language for molecular biology. In *Proc. First Int. Conference on Intelligent Systems for Molecular Biology*, 1993.

[13] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages and computation.* Addison-Wesley, 1979.

[14] L. Hunter. Molecular Biology for computer scientists. In L.Hunter, editor. *Artificial Intelligence and Molecular Biology*, AAAI Press, 1994.

[15] G. Mecca and A. J. Bonner. Sequences, Datalog and Transducers. To appear in *14th ACM SIGMOD Symp. on Principles of Database Systems (PODS)*, 1995.

[16] W. B. Pearson and D. Lipman. Improved tools for biological sequence comparison. In *Proc. Nat. Academy of Science*, 85(8):2444–2448, 1988.

[17] D. B. Searls. Representing genetic information with formal grammars. In *Proc. Nat. Conf. of the American Association for Artificial Intelligence*, 7:386–391, 1988.

[18] D. B. Searls. Investigating the linguistic of DNA with Definite Clause Grammars. In *Proc. North Am. Conference on Logic Programming*, pages 189–208, 1989.

[19] J. Tierry-Mieg and R. Durbin. ACeDB: A database for genomic information. In *Proceedings of the Genome Mapping and Sequencing Workshop*, Cold Spring Harbor Laboratory, New York, 1992.