Pakistan J. Zool., vol. 47(6), pp. 1783-1795, 2015.

A Systematic Approach to Generate Query Model for DNA

Mehwish Nadeem,[1] **Muhammad Usman Ghani Khan,**[2] **Abad Ali Shah**[2] **and Abdul Nasir**[2]
[1]*Embedded Systems and Enterprise Software Solutions Lab, KICS, UET, Lahore Pakistan.*
[2]*University of Engineering and Technology, Lahore, Pakistan*

Abstract.- The Deoxyribonucleic acid (DNA) molecular structure contains lot of important information which is unique in every human being. Storage, processing and manipulation of DNA's structural information in a computer system are still in its infancy. For perfect handling of such type of data, we need a Database Management System. Our research work described in this paper arose from the observation that existing data models and query languages (and the database systems realizing them) do not offer sufficient support for the modelling of DNA structure. This is an attempt to find a good representation for DNA structure and solution to the problem representation of DNA is given in form of an object oriented data model by using the idea of bar code technology. It is shown that the chemical structure of DNA can be encoded in bar code which makes storage of DNA structure a lot simpler than existing approaches. In the end we have proposed a query language (DNA-QL) to store, retrieve and manipulate the biological data. To achieve these objectives, we intend to propose a data model to model DNA structures in a uniform fashion. Development of this type of model and query language enables us the development of DBMS for storing biological structural information of DNA.

Keywords: Growing database, Query Language, Indexed, DNA, Data model, DNA-QL, Constraints, SQL, biological data, data types, Sub Sequence, Information.

INTRODUCTION

DNA is an essential part of all living organisms and biologists are researching on determining functions of DNA. There is enormous amount of complex DNA structure data that needs to be stored efficiently. In current times, new species are being invented at a very rapid phase (Mukhtar, 2015). This species invention have brought an explosion in the amount of molecular biological data which is available for research community.

The existing data modelling techniques are incapable to model these complex structures. New data modelling techniques are required for modelling the DNA structures. Also, exponential growth of new DNA data from the wet laboratories is contributing difficulties and complexity to the data management (such as data modelling, storage, retrieval and manipulation) and the software development methods for bioinformatics. To overcome these issues, some generic object-oriented data models and DBMS have been developed, such as Orion (Kim *et al*., 1990), O2 (Deux *et al*., 1990) and Iris (Wilkinson *et al*., 1990).

The DMBS are unsuitable to use and manage the non-standard data such as DNA and protein data. There are two types of non-standard data inside a DNA structure and they are given below:

Sequence data

Core data supports a range of standard data types including string, date and number. Sometimes, however, we need an attribute's value to be a type that is not supported directly. The existing general-purpose OODBs do not have any standard (built-in) data types and biological domain-specific functional operations for biological research (Wang, 2007). In biological data DNA sequence is non-standard data type and its corresponding data. It is a common and current practice to store metadata of each sequence and its data in a relational DBMS. This practice has a serious problem that the relational DBMS do not support approximate and partial sequence matching queries. There is another approach in practice in which the sequence data is stored in a flat file and external indices are created which are processed by

* Corresponding author: nasirbhutta1@gmail.com
0030-9923/2015/0006-1783 $ 8.00/0

Authors' Contributions: AAS conceived the project and supervised the work; MN, performed the experiments, developed software and tested Codes, AN helped in data collection and analysis; MUGK wrote the article and helped in mathematical formulations.

a special purpose query language (Jagadish *et al*., 2003). This approach is the classical file system approach which lacks in supporting the features of DBMS technology such as structured query language,

Shape data

The shapes of DNA 3D structure and protein-DNA complexes are important objects in the study of the structural biology. These type of data have been supported in some DBMS, but they are not considered as a mainstream research topic in the field of database. There is no query language available for 3D rigid shape-matching in the existing DBMS (Jagadish *et al*., 2003).

So there is an urgent need of a new object-oriented data model that can model and capture different and non-standard characteristics of the DNA structures and data. Although a few object-oriented data models have been proposed for different types of biological data *e.g.*, AceDB (Durbin and Thierry, 1994), MapBase (Lamb and Landis, 1991), VODAK (Klas *et al*., 1994), P/FDM (Gray *et al*., 1990), PDBLib (Chang *et al*., 1994) but there is no data model available specifically for DNA structures. Many databanks (flat file systems) have been developed for DNA, but most of them are developed as flat file systems and some as relational databases. Flat files usually manage data by using strings and some other simple tools that can be easily mastered, but leave users difficulty to manipulate data. In addition, it does not support complex data types, which makes it not be able to meet the requirements of the management of complicated biological data. Relational databases are mature and are successfully applied in many areas, and one of the major reasons is that the relational data model is much simpler than others. However this advantage becomes a big issue in the life science database applications because of the lack of support for complex data types.

To overcome the above mentioned shortcomings of the existing data models and databanks and non-availability of data model for the DNA structures, we propose an object-oriented data model with built-in data types and built-in biological domain-specific functional operations and then propose a Query language to manipulate the complex DNA structure data. In this proposed data model, we use an encoding methodology for coding the chemical structures. The concept of this encoding methodology is derived from the commercially available barcode technology. This paper describes an integrated approach which takes advantage of both automation technologies such as commercial bar coding tools as well as existing molecular structure representation formats for the description of DNA structure. We demonstrate that barcodes are one of the most practical methods for inputting molecular structures into computer systems in a fully automated and less error-prone fashion.

MATERIALS AND METHODS

DNA is a nucleic acid that contains the genetic instructions used for the development and functionality of all known living organisms and some viruses. The main role of DNA molecule is the long-term storage of information. It consists of two long polymers of simple units called nucleotides, with backbones made of sugars and phosphate groups joined by ester (hydrogen) bonds. These two strands run in opposite directions to each other and are therefore anti-parallel (Tseng and Yang, 2013). Attached to each sugar is one of four types of molecules called bases. Fig. 1 shows the flat molecular structure of DNA having four different bases which are commonly found in DNA: adenine (A), guanine (G), cytosine (C), and thymine (T). In their common structural configurations, A and T form two hydrogen bonds while C and G form three hydrogen bonds. Because of the specificity of base pairing, the two strands of DNA are said to be complementary. So it forms the sequence of these four bases along the backbone that encodes information. This information is read using the genetic code, which specifies the sequence of the amino acids within proteins.

The code is read by copying stretches of DNA into the related nucleic acid RNA, in a process called transcription (Berg *et al.*, 2002). Within cells, DNA is organized into structures called chromosomes. These chromosomes are duplicated before cells divide, in a process called DNA replication (Lodish *et al.*, 2000). Eukaryotic

organisms (animals, plants, fungi etc.) store their DNA inside the cell nucleus, while in prokaryotes (bacteria, virus) it is found in the cell's cytoplasm. Within the chromosomes, chromatin proteins such as histones compact and organize DNA. These compact structures guide the interactions between DNA and other proteins, helping control which parts of the DNA are transcribed (Butler, 2001). The structure of DNA is shown in Figure 1.

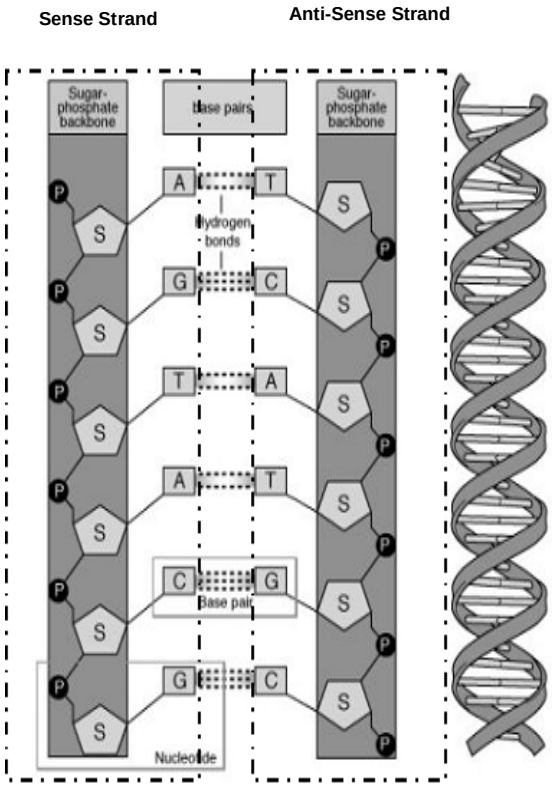

Fig. 1. Flat molecular structure of DNA (www.cnx.org)

Relational vs object oriented databases

Traditional data models such as relational database lack of support for complex data types and built-in biological domain-specific functional operations which is a big issue for DNA data application. Hence many scientists switch to the object-oriented databases since object-oriented nature of life science data perfectly matches the architecture of object oriented databases. Table I summarizes some of the main differences between relational and object oriented databases on the basis of certain criteria. From the above comparison of relational and object oriented data models given in Table 1, we can conclude that OODM allows a structurally adequate representation of the data and can capture the operational semantics of applications (Beynon-davies, 2004). In bio-molecular databases for each fundamental class, dedicated data types are required, that cannot be easily supported for example in relational database systems. The numerous, heterogeneous bio-molecular databases make database integration techniques necessary, for which object–oriented database systems are well suited. Standardization efforts in the field of bio-molecular databases would be extremely valuable, as the number of heterogeneous databases is rapidly growing in the area. An adequate approach to define such a standard would be based on the object–oriented data model (Aberer, 1995).

Object oriented data models for bio-molecular applications

Data modeling in the bio-molecular application domain requires flexible and expressive data models, because of the many complex concepts that are interconnected in various ways. The complex structures, semantic constraints, different data types and derived data of bio-molecular applications make it necessary to develop an object oriented data model for these applications. Many object oriented data models have been made for bio molecular databases. Some of the object oriented data models for bio-molecular applications are described below.

AceDB

AceDB (Durbin and Thierry, 1994) is a special purpose object–oriented database system which was originally designed to meet the needs of the C. elegans mapping and sequencing project. The data model supports classes and methods, but does not support inheritance. The objects can be grouped in a hierarchical way. As key features the data model allows to dynamically change the database schema by adding new attributes, the possibility to attach freely searchable textual annotations everywhere. A number of graphical interface tools support the access to the database system. One of

Table I.- A Comparison of database management systems.

Criteria	RDBMS	ODBMS
Defining standard	SQL2	SQL3/4 (in process) and ODMG-V2.0
Syntax Complexity	User friendly syntax; easier to learn	complex and difficult to learn due to the object oriented technology
Efficiency	Inefficient when querying or processing large amount of data, for example, video stream collections, image collections	Highly efficient for processing large amount to data including multimedia object as well
Languages	SQL (Structured Query Language)	OQL (Object Query Language) as an object-oriented extension of SQL
Technicality	GUI interface are available that makes the technology available to people for querying data who are not highly technical	Technical programmer or developer needed for querying data
object-oriented programming support	Poor Inefficient to querying; programmers spend 1/4 of time to code and map the program object instances to database	Object database mostly handle complex data types and support is direct &extensive.
Complex data relationships	It makes the data independent from application, good for querying data with simple& easy relationships	Objects are a natural way to model; can maintain and manipulate a wide variety of data types and relationships
Simplicity of use	In relational model the structure of table is easy to understand & many end-user tools available	Best for programmers; some SQL access for end users
Extensibility & content	These supports limited set of data types *e.g.* integer, String, date, double etc.	Allow users or programmers to define and use new object data types.
Language maturity	Very mature	Comparatively mature, but difficult to understand and use

the tools supports set–oriented, navigational access to the database. AceDB is also used as a front–end for the Integrated Genome Database (IGD) (Durbin and Thierry, 1994) which integrates existing heterogeneous genome databases and is implemented on top of a relational DBMS.

MapBase

MapBase is a system to support the experimental workflow in a laboratory for constructing genome maps. MapBase has been built using an object oriented database management system ObjectStore (Lamb and Landis, 1991), which uses C++ as data definition and manipulation language. Some problems were faced during development of the project using ObjectStore *e.g.*, lack of well- defined data model, lack of schema evolution, chances of memory leak, insufficient query facilities etc. So MapBase was built as an intermediate layer between ObjectStore and the bio-molecular applications. There are some C++ classes that are stored in the database of MapBase with 40 additional classes that are used only during execution of the MapBase server. Some classes represent specialized scalar types while others represent more complex objects. Due to powerful language of C++ MapBase is able to define DNA sequences of arbitrary length efficiently (Goodman *et al*., 1994).

Docking-D

Docking-D is an object oriented integrated

Table II.- A comparison of biological database management systems.

Criteria	AceDB	MapDB	Docking-D	P/FDM	PDBLib
Data Model	AceDB	C++	VML	FDM	C++
Query Language	Navigational Graphical	Navigational	VQL	Functional	MMQL
Query Optimization	No	No	Yes	Yes	No
Support for schema evolution	Yes	Yes	No	No	NA

database for storing, retrieving and updating protein data. The data is taken form flat file and relational databases like SWISS-PROT (Bairoch and Boeckmann, 1994) and PDB (Berman *et al*., 2000) and etc. Docking-D is implemented using an object oriented DBMS VODAK (Klas *et al.*, 1994). VODAK provides new application specific data modeling features along with the standard features of object oriented data models. Docking-D integrates databases in two basic phases: a syntactic transformation phase and a semantic integration phase. In the syntactic transformation phase, heterogeneous data models are mapped to a uniform data model. The object oriented data model of VODAK, is used as the canonical data model into which the external schemas are mapped. In semantic integration phase, several export schemas are combined on the basis of uniform data model.

The integrated schema is generated by generalization *i.e.* classes are constructed that are containers for the union of the instances of different classes carrying information about the same real world aspect (Aberer, 1995).

P/FDM

P/FDM is a functional object oriented data model for storage and retrieval of protein data derived from PDB. In the functional object oriented data model classes can be accessed only by means of functions. The attributes of objects also correspond to stored functions. P/FDM is a network of objects which represent protein at primary, secondary and tertiary levels. P/FDM is accessed either by using the logic programming language PROLOG or by functional query language DAPLEX (Gray *et al.*, 1990).

PDBLib

PDBLib is implemented in the object–oriented programming language C++ and provides memory resident data structures that can be derived from PDB database entries. In this way it provides by means of a C++ class library an abstract interface to PDB. The classes of the library are divided into four different groups (Shindyalov *et al.*, 1994). So-called intrinsic classes describe the macromolecular structures of protein chains, residues and atoms. Extensible classes provide a layer that separates the implementation details of intrinsic classes from the other parts of the library and the user. Iterative classes model sets of molecular objects and allow iterating and filtering over them. I/O classes are used to load molecular structures from files (Chang *et al*., 1994).

In the above, Table II summarizes some properties of the existing Data models for bio-molecular application. We can examine that there is no data model available specifically for DNA structures. They do not have any built-in data types for biological research and built-in biological domain-specific functional operations for handling DNA Structure Data.

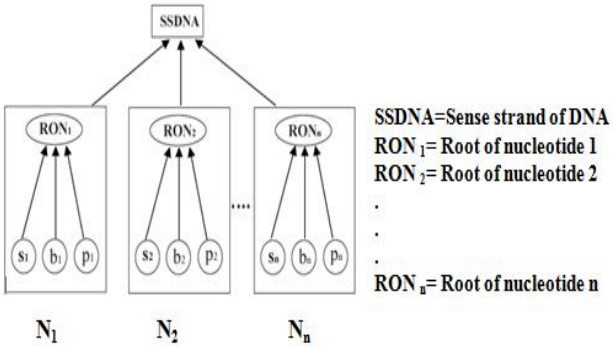

Fig. 2. Schema of DNA sense strand.

Object oriented data model for DNA structure

In this section we have proposed an Object Oriented Data Model for DNA Structure. As mentioned earlier, DNA is a collection of four

different types of nucleotides. Figure 2 shows the general schema of DNA sense strand, where SSDNA denotes root of sense strand of DNA system, with n nucleotides *i.e.,* $N_1, N_2, N_3 \ldots N_n$ as its children.

Dnao object

In the proposed object oriented Data model, object is referred to as *Dnao*, and is defined by a 3-tuple as follows:

Dnao (DS, DO, Constraints) ……..…..…... (1)

The definition of Dnao as described in Expression (1), where

1. The first part of Dnao is DS that denotes a DNA structure. DS is defined by a set of instance-variables/attributes of a Dnao (Sha *et al.,* 2008).
2. The second part of a Dnao is DO that denotes DNA Operation, and it is defined by a set of operations/methods/ functions that operate on data values assigned to the instance-variables as defined in DS (Sha *et al.,* 2008).
3. The third part is Constraints which is defined by a set of constraints on the DS.

Here we are again referring to Figure 1 which describes the detailed structure of DNA. It is clear from the figure that the only difference between the 'sense' and 'anti-sense' strand is the direction and sequence of bases. Same Sugar and Phosphate groups are present on both strands. It is assumed that if we have sense strand base sequence then we can extract DNA anti-sense strand base sequence, RNA strand base sequence and protein amino-acids sequence by applying the operations present in Dnao object in Expression (1) which can be elaborated mathematically in Set theory notation as follows.

Mathematical modeling

For mathematical modeling we again refer to sense strand base sequence presents in Figure 3. The set Ns contains the sequence of nucleotide presents on sense strand.

$$Ns= \{Ns1, Ns2, Ns3\ldots. Nsn\}\ldots (2)$$

The set NAs contains the sequence of nucleotide presents on Anti-sense strand.

$$NAs= \{NAs1, NAs2, NAs3 ,\ldots. NAsn\}\ldots (3)$$

If we have only sense strand nucleotide sequence, then we can extract anti-sense strand nucleotide sequence. Equation (2) nucleotides Ns1, Ns2, Ns3 ,…. Nsn can further represented in the form of sugar, base and phosphate molecules. We knew that every nucleotide is composed of sugar, base and phosphate molecules.

$$b = \{A, T, C, \text{and } G\}\ldots (4)$$

The set Ns may also be defined as:

$$Ns= \{(S \cup b_1 \cup P), (S \cup b_2 \cup P), (S \cup b_3 \cup P)\ldots (S \cup b_n \cup P)\} (5)$$

The only difference among the elements of Ns is base molecules attached with sugar and phosphate molecules, that can belong to the set **b** in equation 4. For simplification in mathematical modeling we simplify our set of sense strand nucleotides by taking S and P common and only considering bases present in the set. Such that the simplified form of Equation 5 becomes:

$$Ns = S \cup P \cup \{b_1, b_2 , b_3\ldots. b_n \}\ldots..(6)$$
$$Ns = S \cup P \cup B$$

For simplicity we are only considering set B of bases from equation 6

$$B= \{b1, b_2, b_3\ldots. b_n \}\ldots(7)$$

By applying complement operation on set B in equation 7 we obtain set of complement bases B´ that are present in anti-sense strand in set NAs ,which is given below in simplified form.

$$B'= \{b1', b2', b3'\ldots. bn'\}\ldots(8)$$

Where b1´, b2´, b3´…. bn´ are complement bases which also belong to the set **b**= {A, T, C, G} of bases described in equation 4. The complement of the bases A, T, C, and G in sense strand are given in Table III.

Table III.- Bases and their complements

Sense-Strand Base	Anti-sense strand complement base
A	T
T	A
C	G
G	C

Finally the set of anti-sense strand may be defined as:

$$N_{As} = \{ (S \cup b_1' \cup P), (S \cup b_2' \cup P) , (S \cup b_3' \cup P),... (S \cup b_n' \cup P) \}\ldots(9)$$
$$N_{As} = S \cup P \cup B'$$

For simplicity we are only considering set B ´ of complement bases from equation 9.

$$B' = \{b1', b2', b3'\ldots. bn'\}\ldots.(10)$$

The only difference among the elements of NAs is complement base molecule, which also belong to the set **b** in Equation 4 attached with sugar and phosphate molecules. As mentioned earlier in the previous sections that by applying transcription operation "getDTrancript" (Algorithm given in Appendix section) on antisense strand sequence B´= { b1´, b2´, b3´…. bn´ }…(5) RNA strand sequence is obtained.

$$N_{Rna} = \text{getDTrancript}(B')$$

The base sequence obtained after applying transcription represented in set notation in equation 6 as follows.

$$N_{Rna} = \{b1'', b2'', b3''\ldots. bn'' \}\ldots(6)$$

The base sequence b1´´, b2´´, b3´´…. bn´´ belongs from the set b´={A,U,C,G}. Thus for each C base encountered on DNA anti-sense strand, a G base is inserted in the RNA; for each G, a C; and for each T, an A is inserted. However, each A on the DNA anti-sense strand guides the insertion of the uracil (U base). Where T is not present in RNA strand as given in Table IV.

In this way a DNA strand converts itself into an RNA sequence by going through an intermediary step of transcription.

Table IV.- Bases obtained after transcription.

Anti-Sense Strand Base	RNA strand base after Transcription
A	U
T	A
C	G
G	C

Encoding DNA structure using bar-code technology

Barcode contains a set of black bars in varying width separated by white spaces encoding alphanumeric characters. Traditionally barcodes are printed on products so that they can be identified easily and efficiently. An example barcode is given in Figure 3. We have used the bar-code technology for the encoding the complex structure of DNA for its efficient storage and retrieval

Fig. 3. A typical linear barcode (www.dataid.com)

The National Chemical Laboratory, Pune, employs 2D barcode encoded structures for inhouse repository management, where barcodes can also be used for querying the database for similar or substructures of the query structure (Karthikeyan, 2005. The chemical structures are represented in 2D (PDF417) barcode representation. For the web-based applications, an interface is developed which interprets these barcodes, and export them as optimized 3D chemical structures. Applications of this barcode representation also perform some important functions such as automatic storing and web-based tracking of molecular samples.

We use the same idea in encoding DNA's sequences but doing some necessary modifications. Fig 4 shows the mapping of DNA Flat structure to Deoxyribonucleic acid-code by keeping in consideration the two types of strands present in DNA. In our modified encoding format, we use

numeric digits instead of black and white bars. This modified barcode (Deoxyribonucleic acid-code) format is shown in Figure 4.

089 090 SSDNA 091	092 Hydrogen Bond 093

Fig. 4. Deoxyribonucleic acid-code format, encoding DNA.

In Figure 4, the Deoxyribonucleic acid-code format consists of three parts. Format starts with code digit 089 and ends with code digit 096. We have used 3-digit code-words in our Deoxyribonucleic acid-code ranging from 000 to 999. The code-words from 000 to 088 are reserved for the structural information of a DNA.

1. The first part of the format starts from the code 089 which shows start symbol of Deoxyribonucleic acid-code. Then 090 shows start symbol of Sense Strand of DNA(SSDNA). SSDNA ends with the code 091. Here the notation SSDNA represents the sense strand of DNA which already has been explained.
2. The second part of the format starts from the code word 092 which describes the hydrogen bond which can be either double hydrogen bond or triple hydrogen bond and ends with the code word 093.
3. The third part of the format starts from the code word 094 contains the term ASDNA which represents the anti-sense strand of DNA and ends with the code word 095.At the end 096 represents the End symbol of Deoxyribonucleic acid-code.

The hierarchal representaion of DNA Barcode structure (Fig. 4) is given in Figure 5.

In Table V we have given the code-words and their meanings for encoding DNA Structure in the proposed data model. The encoding scheme that is proposed above can be extended for the future use. We have used only the first 100 code-words (Table V) out of the 1000 code-words. The remaining 900 code-words in Table V are available for encoding the information about other bio-molecular structures in the future. This shows extendibility of the proposed data model.

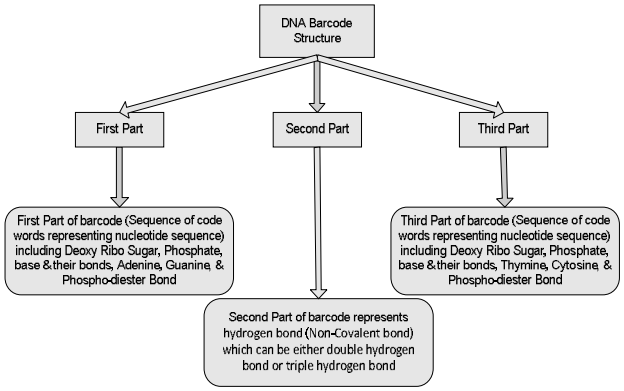

Fig. 5. Hierarchal representation of DNA Barcode Structure.

Table V.- DNA/RNA structure code-words.

Code-words	**Description**
089	Start symbol of Deoxyribonucleic acid-code
096	End symbol of Deoxyribonucleic acid-code
090	Start symbol of Sense strand of DNA (SSDNA)
091	End symbol of Sense strand of DNA (SSDNA)
092	Start Symbol of hydrogen bond
093	End Symbol of hydrogen bond
094	Start symbol of anti-sense strand of DNA (ASDNA)
095	End symbol of anti-sense strand of DNA (ASDNA)
097	Start symbol of RNA strand after Transcription
098	End symbol of RNA strand after Transcription
000	Double Hydrogen Bond
001	Triple hydrogen bond
002	Phospho-diester bond
003	DeoxyRibo Sugar-Base bond
004	DeoxyRibo Sugar-Phosphate bond
005	Thymine
006	Adenine
007	Guanine
008	Cytosine
009	DeoxyRibo Sugar
010	Phosphate
011	Uracil
096-999	Reserved for Future Use

Constraints

A constraint is a description of some condition that must be satisfied by a database state if it is to reflect its real world semantics accurately. In order to come up with the efficient querying to DNA structure data we need a set of constraints in the core of an object Model. Constraints on a data

model are of two types, implicit constraints and explicit constraints. The explicit constraints are defined by the user while the implicit constraints are defined by the data model developer (Durbin and Thierry, 1994). We have defined following (5) implicit integrity constraints.

i. There are several attributes in the DNA database schema that store numerical properties of the data *e.g.* the molecular weight (mol.wt) and length attributes (mol.len) of the class DNA. These attributes are defined as either floats or integers, whereas in fact they have much smaller domains, since none of them can take a negative value. We can use semantic constraints to reduce domains of these attributes as follows:
   a) constrain each d in DNA
      so that mol.wt(d) > 0;
   b) constrain each d in DNA
      so that seq_length(d) > 0;

ii. We can constraint the molecular weight of DNA since we know that the value must be either equal or greater than the sum of the molecular weight (mol.wt) of its constituent chain. As we know that a particular fragment of DNA is directly proportional to its length *i.e.* its molecular weight (Berg *et al.*, 2002), so we can say that:

    DNA αmol.wt of chain
    so
    DNA= k × mol.wt of chain
    Where k $\geq$1
    so
    Constrain each d in DNA to have mol.wt (d) $\geq$
    sum(mol.wt(component-Dna(d) as chain))

iii. The information in DNA is stored as a code made up of four chemical bases: adenine (A), guanine (G), cytosine (C), and thymine (T) so we can enforce this essential biochemical "rules" about DNA structure using integrity constraints.

    constrain each b in DNA Sequence
    so that base(b) = {A, T, C, G};

iv. The number of adenine residues is equal to the number of thymine residues (A=T) and the number of guanine residues is equal to the number of cytosine residues (G=C), so we can enforce this essential biochemical "rules" about DNA structure using integrity constraints. Since we know that b={A,T,C,G}, so firstly we can constraint the possible size of Nitrogen Base length as:

    constraint each b in DNA
    so that base_length(b)=4;
    Constrain each r in residue
    To have num_residue(sense_strand_
    component(r) as chain)
    = num_residue (anti_sense_strand_
    component(r) as chain)

v. We can further constrain the possible number of codons that can be made from the DNA nucleotides since we know that the nucleotides in DNA are grouped in triplets, or 3-letter 'words', known as codon (Kaestle *et al.*, 2006). So we can constraint on codon length as:

    Constrain each c in codon
    So that codon_length(c) = 3

And the number of possible codons is recorded accurately by following constraint:

    Constrain each n in nucleotide
    So that num_codon(n) $\leq$ 64

Class hierarchy of the DNA structure is given in Figure 6.

After purposing the OODM for DNA structure, we have designed a DNA class- hierarchy given in Figure 6. This class hierarchy helps to identify DNA object classes, their internal structure, and the relationships in which they participate.

The above DNA structure class hierarchy describes all structural aspect of DNA structure data. As Figure 6 identifies the classes involved in formation of DNA structure, with relationships that exist between such classes. These relationships appear in class hierarchy with help of different notations given in Table V. *e.g.*, the rectangle shows classes and objects and arrows show relationships between these classes or objects. The used notations

and their representations in this hierarchal structure are given below in the Table V.

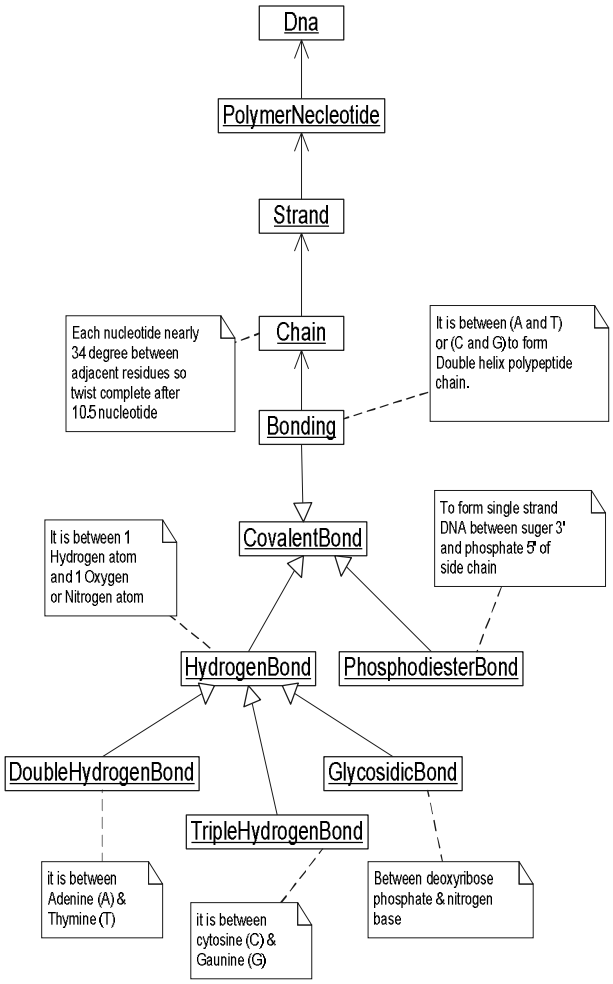

Fig. 6. DNA structure class hierarchy.

Table VI.- Notation & their representation used in DNA structure class hierarchy

Notation used in DNA structure class hierarchy	
	Object Class
	"is a" relationship
	"has a" relationship
	Shows description of particular class or object

This hierarchal representation provides O-O paradigm features and implicitly a constrained description of DNA structure that making it easier to understand the semantics of DNA Structure. The instance variables and methods of DNA object are given in Figure 6.

Object (Dna)
Instance-Variable{
Name;
Classification;
Barcode;
molecular_weight
sequence_length;
Array[] <String>niterogen_base [];
ArrayList<Strand_sequence>dna_strands;
}
Methods {similarity-search, substructure search, location search, numberOfSubsequence, barcodeSearch}
Constraints { No.of base in dna_strand= No.of base in dna_anti_sense_strand,
dna_sequence base[] = {A,T,C,G},
dna_mol_weight≥ sum(mol_weight(each base residue)),
sequence_length&mol_weight>0}

Fig. 7. Class type of DNA structure

RESULTS AND DISCUSSION

Encoding DNA structure-an illustrative example

Now, we demonstrate our encoding scheme though an example of a DNA structure and using our proposed Deoxyribonucleic acid-code format given in fig. 4 which has been given as follows:

The first part of the format in Figure 4 describes the Coding Strand of Deoxyribonucleic acid structure. We are only encoding the sense strand shaded box in the direction of arrow in Figure 7.

In the above encoding pattern of Figure 8 we have encoded in the following pattern such as in the direction of arrow from upward to downward:

"Phosphate, Deoxy Ribo Sugar- Phosphate Bond, Deoxy Ribo Sugar, Deoxy Ribo Sugar-Base Bond, Adenine, Phosphate, Deoxy Ribo Sugar-Phosphate Bond, Deoxy Ribo Sugar, Deoxy Ribo Sugar-Base Bond, Guanine, Phospho-diester Bond".

We simplify our encoding of sense strand nucleotides in Figure 9 by taking some codes common.

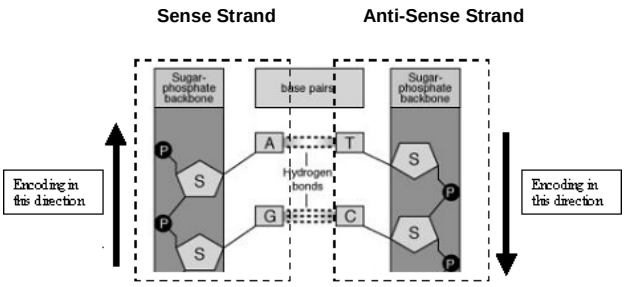

Fig. 7. Two nucleotides used for encoding (www.cnx.org)

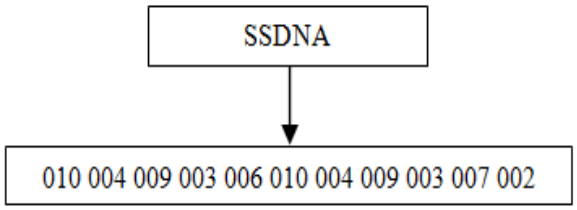

Fig. 8. Encoding of SSDNA.

Referring to Table V we can see that Code 006 represents "Adenine" and Code 007 represents "Guanine". By applying complement operation on Figure 9 referring to Table III we get.

In the above encoding pattern of Figure 10 we have encoded in the following pattern such as in the direction of arrow from upward to downward:

" Phosphate, Deoxy Ribo Sugar- Phosphate Bond, Deoxy Ribo Sugar, Deoxy Ribo Sugar-Base Bond, Thymine, Phosphate, Deoxy Ribo Sugar-Phosphate Bond, Deoxy Ribo Sugar, Deoxy Ribo Sugar-Base Bond, Cytosine, Phosphodiester Bond ".

By applying transcription operation "getDTrancript" on antisense strand sequence in Figure 10 RNA strand sequence is obtained. The algorithm of operation "getDTrancript" has been given in Appendix. RNA sequence is obtained after applying transcription algorithm on ASDNA sequence by (Lamb and Landis, 1991). Thus for each C base encountered on the DNA anti-sense strand, a G base is inserted in the RNA; for each G, a C; and for each T, an A is inserted. However, each A on the DNA anti-sense strand guides the insertion of the Uracil (U base). There is no T present in RNA strand given in Table IV.

Fig. 9. Extracting common codes.

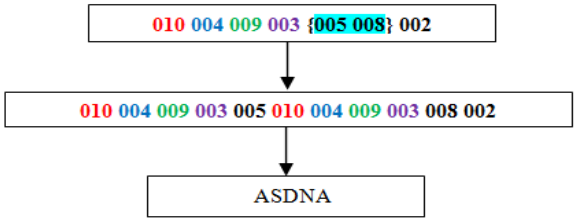

Fig. 10. Anti-sense encoding obtained after complement.

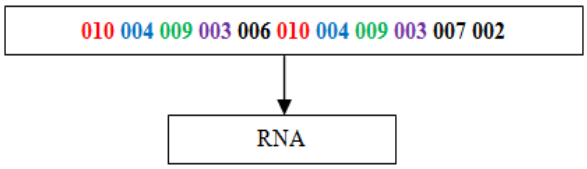

Fig. 11. RNA sequence obtained after transcription.

So we have examined that how DNA strand converts itself into a RNA sequence by going through an intermediary step of transcription.

After proposing an object oriented data model for DNA structure, we have proposed a Query language to store, retrieve and manipulate the DNA structure data. In order to send request and get results from database user has to make use of query language to fetch the information to and from database. Our proposed query language is a domain specific query language that is designed for DNA domain.

Syntax of our proposed DNA-QL

Backus-Naur notation (BNF) is a way to describe the syntax of the programming languages. Any sentence which is derived using the rules specified in BNF notation is said to be syntactically correct. The rules are called production rules. These rules are defined with the help of some specific symbols. The BNF symbols used for our proposed DNA-QL are given in Table VII.

Table VII.- BNF Symbols used for DNA-QL.

Symbol	Description
::=	Means "is defined as"
<>	(<>) correspond to terminal or non-terminals, describes portion of the language but not part of the actual syntax.
[ ]	To enclose optional elements, square brackets are used.
|	Represent the logical OR
{ }	The constructs within braces are grouped together

DNA Select_Query ::=SELECT<asterisk>|
<attribute_list>|<query_operator>[{
<comma><attribute_list> }...]
FROM <table_name>
[WHERE <conditions>]
[ARCHIVE <YES/NO>]
[BARCOMP <barcode>]

The interpretation of additional clauses is as follows:

1. *The SELECT and FROM Clause:* This is similar to the SELECT command of the SQL and allows filtering out only the relevant attributes of those instance variables of an object which fulfill the given criteria. In the SELECT clause object methods or path expressions can be listed, with a comma between them. The objects in the database containing the data are instances of the classes listed in the FROM clause. To each listed class an object variable is associated: it is used to refer to object instances of the related class in the database. According to the object oriented approach, object attributes are referred through methods listed in the clauses, hiding implementation details to users. When a method is specified in a clause, the related code is executed.
2. *The WHERE Clause:* In the WHERE clause the logical conditions which express the constraints that must be satisfied by the objects which will be selected in the database are specified. Complex constraints may be made composing simpler conditions, using the logical connectives AND, OR, NOT. As in SQL this clause supports a set of arithmetic operators {=, ≠, <, >, ≤, ≥} for comparisons. The comparison conditions are also allowed to use the aggregate functions of the SQL such as avg, sum, count, etc. The syntax of the clause will be:
WHERE<object> operator <condition>
{ AND<object> operator <condition>
OR<object> operator <condition> }
3. *The ARCHIVE Clause:* We have proposed an additional clause named as ARCHIVE that introduced the new concept of "GROWING DATABASE" which is so, far a dark area in the field of databases. This clause enables the user to store Query results in the form of tables. With the help of this clause a database history can be maintained in an efficient way. This clause provides two options, (YES or NO). If ARCHIVE clause is set to YES the user will be able to store Query results as a table in the database. These resultant tables can be used for further manipulations in future.
4. *MinMax:* These operators are used for determining percentage (%) similarity of the queried structure *e.g.* if min is 85 and max is 95 then it means that the structure should b at least 85% and at most 95% similar to the structure in query.
5. *The BARCOMP Clause:* This clause is used to find similar DNA structure by simple comparison of stored barcodes of the DNA structures.

Our proposed DNA-QL provides some additional operators based on DNA domain, to which user can conveniently and easily send queries without any extra learning. Algorithms for these operators (*e.g.,* Location, length of strand, sub sequence etc) are given in appendix section.

CONCLUSION

The exiting practice is to handle the DNA data does not offer adequate and powerful support of the data storage, retrieval and manipulation like a DBMS. Also, the DNA structures are highly complex structures, and the existing database technology is unsuitable to handle them. To

overcome the drawbacks of the existing database and databanks technology in handling DNA structures and data, we have proposed an object-oriented data model for DNA structures. The novelty of this data model is that it captures complex DNA structures in a simple fashion using the concept of barcode technology; and it provides a basis to answer complex queries of biologists. This data model also has provision for its extension for modeling and handling of the other biological data. In this paper, we have also given outlines of the DNA query language (DNA-QL) as an extension of SQL. In DNA-QL, we have introduced some new operators which are needed to query DNA data and structures in meaningful and simple manner.

The supplementary Material `Appendix' is available **http://www.zsp.com.pk/supplementary%20material/1783-1795%20(37)%20APPENDIX.pdf**

REFERENCES

ABERER, K., 1995. *The use of object–oriented datamodels for biomolecular databases*. GMD-IPSI, Dolivostr. 15, 64293 Darmstadt, Germany

ALBERTS, R., JOHNSON, A., LEWIS, J., RAFF, M., ROBERTS, K. AND WALTER, P., 2002. *Molecular biology of the cell*. 4th edition. Garland Science. New York ISBN-10: 0-8153-3218-1ISBN-10: 0-8153-4072-9

BAIROCH, A. AND BOECKMANN, B., 1994. The SWISS-PROT protein sequence data bank: Current Status, *Nucl. Acids Res*., **22**: 3578-3580.

BERG, J., TYMOCZKO, J. AND STRYER, L., 2002 *Biochemistry*. W. H. Freeman and Company ISBN 0-7167-4955-6

BERMAN, H.M., WESTBROOK, J., FENG, Z., GILLILAND, G., BHAT, T.N., WEISSIG, H., SHINDYALOV, I.N. AND BOURNE, P.E., 2000. The protein data bank, *Nucl. Acids Res*., **28**: 235-242.

BUTLER, J.M., 2001. *Forensic DNA typing*. Elsevier. ISBN 978-0-12-147951-0. OCLC 223032110 45406517.pp. 14–15.

BEYNON-DAVIES, P., 2004. *Database systems*. 3rd Edition Palgrave, Basingstoke, UK.ISBN 1-4039-1601-2

CHANG, W., SHINDYALOV, I.N., PU, C. AND BOURNE, P.E., 1994. *Design and application of PDBlib*, A C++ Macromolecular Class Library CABIOS.

DEUX, O., 1990. The story of O2. IEEE *Trans. Know. Data. Engin*., **2**: 91-108.

DURBIN, R. AND THIERRY-MIEG, J., 1994. The ACeDB genome database. In: *Computational methods in genome research*. Springer US, pp. 45-55.

GOODMAN, N., ROZEN, S. AND STEIN, L., 1994. *Building a Laboratory Information System around a C++–Based Object–Oriented DBMS*. Proceedings of the 20th VLDB Conference, Santiago, Chile.

GRAY, P.M.D., PATON, N.W., KEMP, G.J.L. AND FOTHERGILL, J.E., 1990. An object–oriented database for protein structure analysis. *Protein Engin.*, **3**: 235–243.

HIGGS, P. G. AND ATTWOOD, T. K., 2009. *Bioinformatics and molecular evolution*. John Wiley

JAGADISH, H.V., OLKEN, F., RASCHID, L., WOOLEY, J. C., ECKMAN, B., RUSSELL, L., 2003. *Report on Workshop on Data Management Technology for Molecular and Cell Biology*.

KAESTLE, F., KITTLES, R., ROTH, A. AND UNGVARSKY, E., 2006. *Database limitations on the evidence.*

KARTHIKEYAN, M., 2005. *Encoding and Decoding Graphical Chemical Structures as Two-Dimensional (PDF417) Barcodes. J. Chem. Inf. Model*.; **45**:572–580.

KIM, W., GARZA, J.F., BALLOU, N. AND WOELK, D., 1990. Architecture of the orion next-generation database system. *IEEE Trans. Know. Data Engin*., **2**: 109-124.

KLAS, W., FANKHAUSER, P., MUTH, P., RAKOW, T. AND NEUHOLD, E.J., 1994. *Database Integration using the Open Object-Oriented Database System VODAK*.

LAMB, C., LANDIS, G., ORENSTEIN, J. AND WEINREB, D., 1991. The object store database system. *Commun. ACM*, **34**: 32–39.

LODISH, H., BERK, A., ZIPURSKY, SL., MATSUDAIRA, P., BALTIMORE, D. AND DARNELL, J., 2000. *Molecular cell biology* 4th edition. W. H. Freeman, New York.

MUKHTAR, M. K. 2015. Two new species of the genus Cheiracanthium CL Koch (Araneae: Eutichuridae) from Punjab, Pakistan. *Pakistan J. Zool*., **47**: 467-472.

SHINDYALOV, I. N., CHANG, W., PU, C. AND BOURNE, P.E., 1994. Macromolecular query language (MMQL): prototype data model and implementation. *Protein Engin*., **7**: 1311-1322.

SHA, A., KHALID, H., AHSAN, S. AND NASEER, I., 2008. *Query model for object-oriented data model of protein structure.* The International Conference on Bioinformatics and Computational Biology (BIOCOMP'08) July 14-17 Am. Crim. Law Rev., **43**:17-23.

TSENG, C.C. AND YANG, X., 2013. DNA structure: What is DNA? In: *Learning basic genetics with interactive computer programs*. Springer, New York, pp. 19-35.

WILKINSON, K., LYNGBAEK, P. AND HASAN, W., 1990. The Iris Architecture and Implementation. *IEEE Trans. Know. Data Engin*., **2**: 63-75.

WANG, Y., 2007. *Protein structure data management system*. Dissertation, Georgia State University.

(Received 24 April 2014, revised 4 June 2015)