2008

A framework for integrating DNA sequenced data

Prabin Dutta

ROCHESTER INSTITUTE OF TECHNOLOGY

Department of Computer Science

A Framework for Integrating DNA Sequenced Data

by

Prabin Dutta

March 24, 2008

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

Rajendra K. Raj, Advisor

Paul Tymann, Reader

Warren R. Carithers, Observer

Signature Block

1) R. K. Raj Prof. Rajendra K. Raj

2) Paul Tymann Prof. Paul T. Tymann

3) Warren R. Carithers Prof. Warren R. Carithers

Permission to Reproduce Thesis

A Framework for Integrating DNA Sequenced Data

I, Prabin Dutta, hereby grant permission to the Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date 03/26/08 Student's Signature Prabin Dutta

Acknowledgement

During this work many people have offered their support, advice and wisdom. I am truly grateful to each one of them and cognizant that I could not have done it alone.

1. Prof. Rajendra K. Raj for his guidance, support and inspiration through out the development of this thesis.
2. Prof. Paul T. Tymann for helping me to stimulate ideas and to find this research topic and agreed to be the reader of this thesis.
3. Prof. Warren R. Carithers who agreed to be the observer of this thesis.
4. Sangeeta Dutta, my wife, for her support and proof reading the thesis.

TABLE OF CONTENTS

A Framework for Integrating DNA Sequenced Data

Abstract

The Human Genome Project generated vast amounts of DNA sequenced data scattered in disparate data sources in a variety of formats. Integrating biological data and extracting information held in DNA sequences are major ongoing tasks for biologists and software professionals. This thesis explored issues of finding, extracting, merging and synthesizing information from multiple disparate data sources containing DNA sequenced data, which is composed of 3 billion chemical building blocks of bases. We proposed a biological data integration framework based on typical usage patterns to simplify these issues for biologists. The framework uses a relational database management system at the backend, and provides techniques to extract, store, and manage the data. This framework was implemented, evaluated, and compared with existing biological data integration solutions.

1. Introduction

This chapter introduces challenges in integrating biological data that normally biological scientists encounter during extracting, retrieving and synthesizing scientific data. The chapter is organized as follows. It begins with a discussion on background of biological data evolution. We will discuss ever enduring challenges of integrating biological data in section 1.2. To understand the problem domain and our research landscape more in depth, we cover characteristics of biological data and different biological data formats in this section too. Section 1.3 covers related works in this problem domain. Section 1.4 states our hypothesis.

1.1 Background on Domain

The mysteries of human cell biology have been captivating human endeavor since the days of Gregor Mendel (1822 – 1884). In 1865 Mendel, the father of genetics published his analysis of heredity. Later on Nobel laureate Barbara McClintock provided the microscopic evidence of chromosome breakage and reunion during recombination [12].

The discovery of double helical structure of DNA is another milestone of genetics. In 1953 Watson and Crick showed that the structure of DNA is a double helix [12]. However prior to the Watson and Crick's study, in 1940 Beadle, Tatum, Ephrussi proved that the function of genes was established as a repository for the information to synthesize a protein.

The ultimate triumph of Genetics is the Human Genome Project [14]. In May 1990 Department of Energy (DOE) declared the beginning of Human Genome Project. This publicly funded project costs $200 million dollar per year and anticipated to be lasting for 15 years. The objectives of this project are

- Mapping and sequencing the human genome and the genomes of model organisms;
- Data collection and distribution;

- Ethical, legal, and social considerations;
- Research training, technology development and
- Technology transfer.

The Human Genome Project is successfully completed in April 13, 2003. The project generated a plethora of information on human genome especially on DNA sequenced data. DNA is simply a long string of four molecules – Adenine (A), Thymine (T), Guanine (G), Cytosine(C) arrayed in sequence and it carries heredity information. However searching, extracting, merging and synthesizing this information held in the molecule remain a complex and daunting task.

1.2 Problem Definition and Motivation

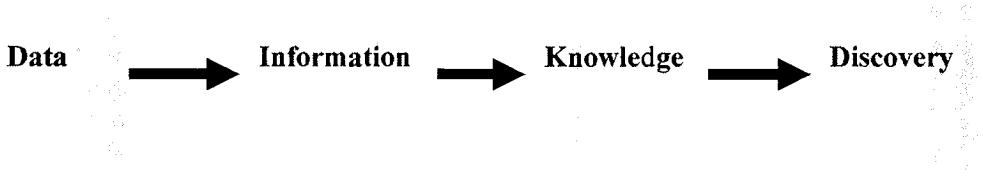
1.2.1 Biological Data Integration Common Problems

While DOE completed the Human Genome Project, other scientific, commercial and educational organizations started storing the genome data and extracting the stored data. The data generated by Human Genome Project is crucial for biological scientists for scientific analysis. The individual organizations publish data in autonomous data sources and these data sources vary considerably in contents, access methods, capacity, query processing and services.

Biological data integration is a complex task for both bioinformaticians and biologists and significant efforts have been made to overcome these complexities during last twenty years. Researchers need integrated access to data from multiple sources. Concept identification and resolution is a complex task because data contained in two data sources may refer to the same object and hence confusion increases. Again different data sources may contain data in different formats and reconciling them to a single repository is another daunting task. We spent a major part of this research only on data transformation. This is because raw data needs to be extracted in order to make them suitable for storing in scientific repository. Concepts overloading (whether two abstract concepts have really same meaning or not) also plays a confusing role in biological data integration.

Therefore, biological data integration is a complex task and much of effort has to be made by scientists to overcome these complexities.

Biologists use computers and computer tools to collect and analyze results from largely automated instruments used in biological sciences. The initial computational analysis is mandatory to answer many biological questions. The web is another source of large amount of information available for scientists. Therefore the biological science discovery process is nothing but converting data into information, information into knowledge and finally knowledge into discovery.



However information integration in discovery process involves lots of enduring challenges. Information integration in biological science greatly differs from traditional information integration in that the nature of biological data (nature of biological data will be discussed in section 1.2.3).

The recent trend in biological science research is moving from wet lab to computer science lab, which is known as information driven discovery. In the recent past, reductionist molecular biology gave biologist the tool to identify and characterized molecules and cells, the fundamental building block of living system. However, a system level approach in biology is necessary to understand and analyze molecules and ultimately cells, function in tissues, organs, organisms and population. The objective of the life science discovery process is to convert data into meaningful discovery. The diagram below depicts the Life Science Discovery Process.

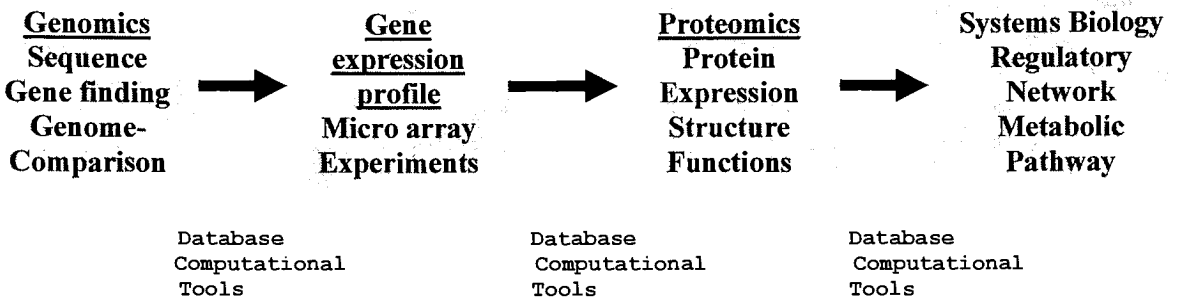


Fig 1.1: Biological Information Discovery Process

Research discovery and synthesis are now driven by the complex information arising intrinsically from biology itself and from the diversity, heterogeneity of experimental observation. The database and computing activities will need to be integrated to produce more meaningful and effective results from biological data. The new emerging techniques, for example Gene finding algorithms and clustering, make the whole process of biological discovery a robust one.

In order to successfully complete the discovery process, the biologist must be able to find, extract, merge and synthesize information from multiple, disparate data sources. Biologists also need to access and retrieve the most up-to-date biological data. To find the most relevant information, biologists need to be able to perform complex queries across multiple heterogeneous databases. The robustness of integration information infrastructure involves complex database queries, computational algorithms, and application software. The most efficient analytical tools and algorithms are required to extract meaningful information from massive volume and diverse biological data.

1.2.2 Diversity and Variability

Another challenge of integrating biological data is its diversity and variability. The nature of biological data is intrinsically complex and is organized in loose hierarchies that reflect our understanding of the complex living systems. Different individuals and species vary tremendously, so naturally biological data does also. On top of diverse and variable nature, biological data are dynamic too. Biological data sources are so diverse that

finding right data source itself is a challenge. Biological data sources represent a loose collection of autonomous web sites, each with its own governing body and infrastructure. Heterogeneous data formats ranges from simple files to fully structured database system. Biological data sources are subject to ongoing changes including data content and data schema. Analysis of biological data requires both database query activities and proper usage of computational analysis tools.

From the above discussion we saw that DNA sequence data sets are intrinsically complex in nature. The data structure spans many orders of magnitudes in time, space and poses challenges in informatics, modeling equivalent or beyond any other colossal scientific challenges. Biological data ranges from plain text of laboratory record and literature publications, nucleic acid and protein sequences, three-dimensional structures of molecules and biomedical images with different levels of resolutions. They also encompasses experimental output from technology and diverse as micro-array chips, gels, light and electronic microscopy, nuclear magnetic resonance and mass spectrometry.

Though high throughput and automated technologies generate a large volume of data, the striking feature of life science data is not volume but its diversity and variability.

1.2.3 Characteristics of Biological Data (DNA Sequence Data)

1.2.3.1 Structure of a nucleotide

The primary objective of this thesis is to propose a solution to store and manage information about DNA sequenced data. DNA is simply a long string of four molecules – Adenine (A), Thymine (T), Guanine (G), Cytosine(C) arrayed in sequence. Understanding the structure of DNA helps us understanding problem domain our research landscape more in depth.

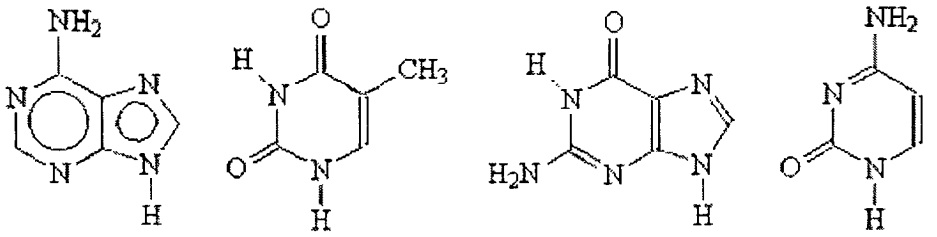


Fig 1.2: The diagram depicts the chemical composition of each nucleotide

```
AAAAGAAAAGGTTAGAAAGATGAGAGATGATAAAGGGTCCATTTGAGGTTAGGTAATATGGTTTG
GTATCCCTGTAGTTAAAAGTTTTTGTCTTATTTTAGAATACTGTGACTATTTCTTTAGTATTAAT
TTTTCCTTCTGTTTTCCTCATCTAGGGAACCCCAAGAGCATCCAATAGAAGCTGTGCAATTATGT
AAAATTTTCAACTGTCTTCCTCAAAATAAAGAAGTATGGTAATCTTTACCTGTATACAGTGCAGA
GCCTTCTCAGAAGCACAGAATATTTTTATATTTCCTTTATGTGAATTTTTAAGCTGCAAATCTGA
TGGCCTTAATTTCCTTTTTGACACTGAAAGTTTTGTAAAAGAAATCATGTCCATACACTTTGTTG
CAAGATGTGAATTATTGACACTGAACTTAATAACTGTGTACTGTTCGGAAGGGGTTCCTCAAATT
TTTTGACTTTTTTTGTATGTGTGTTTTTTCTTTTTTTTTTAAGTTCTTATGAGGAGGGAGGGTAAA
TAAACCACTGTGCGTCTTGGTGTAATTTGAAGATTGCCCCATCTAGACTAGCAATCTCTTCATTA
TTCTCTGCTATATATAAAACGGTGCTGTGAGGGAGGGGAAAAGCATTTTTCAATATATTGAACTT
TTGTACTGAATTTTTTTGTAATAAGCAATCAAGGTTATAATTTTTTTTAAAATAGAAATTTTGTA
AGAAGGCAATATTAACCTAATCACCATGTAAGCACTCTGGATGATGGATTCCACAAAACTTGGTT
TTATGGTTACTTCTTCTCTTAGATTCTTAATTCATGAGGAGGGTGGGGGAGGGAGGTGGAGGGAG
GGAAGGGTTTCTCTATTAAAATGCATTCGTTGTGTTTTTTAAGATAGTGTAACTTGCTAAATTTC
TTATGTGACATTAACAAATAAAAAAGCTCTTTTAATATTAGATAA
```

Fig 1.3: Example of a simple DNA sequenced data

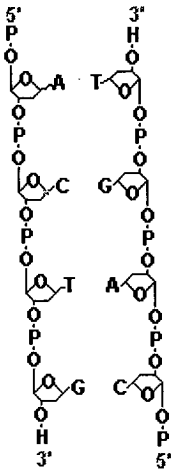


Fig 1.4: Double Helical Structure of DNA

1.2.3.2 Different DNA Sequence formats

One of the major challenges in integrating DNA sequence data is its data format. Currently DNA sequenced data are stored in several different formats. Some of these formats are discussed below.

1.2.3.3 Plain Sequence format

Plain Sequence format only contain one sequence in IUPAC characters including spaces. The example below shows a Plain Sequence Format.

```
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCC
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC
CTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGG
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCC
CTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAG
TTTAATTACAGACCTGAA
```

1.2.3.4 EMBL Format

In EMBL format several sequences could be stored in one file. The example below shows an EMBL sequence format.

```
ID   AB000263 standard; RNA; PRI; 368 BP.
XX
AC   AB000263;
XX
DE   Homo sapiens mRNA for prepro cortistatin like peptide, complete cds.
XX
SQ   Sequence 368 BP;
     acaagatgcc attgtccccc ggcctcctgc tgctgctgct ctccggggcc acggccaccg        60
     ctgccctgcc cctggagggt ggccccaccg gccgagacag cgagcatatg caggaagcgg       120
     caggaataag gaaaagcagc ctcctgactt tcctcgcttg gtggtttgag tggacctccc       180
     aggccagtgc cgggcccctc ataggagagg aagctcggga ggtggccagg cggcaggaag       240
     gcgcaccccc ccagcaatcc gcgcgccggg acagaatgcc ctgcaggaac ttcttctgga       300
     agaccttctc ctcctgcaaa taaaacctca cccatgaatg ctcacgcaag tttaattaca       360
     gacctgaa                                                               368
//
```

The start of the sequence is indicated by "SQ" and the end of the sequence is indicated by "//"

1.2.3.5 FASTA format (University of Virginia)

The sequence file in FASTA format can contain several sequences. An example is shown below.

```
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA for prepro cortistatin like
peptide, complete cds.|len=368
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCC
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC
CTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGG
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCC
CTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAG
TTTAATTACAGACCTGAA
```

1.2.3.6 GCG format (Genetic Computer Group, University of Wisconsin)

In this format the sequence file contains exactly one sequence.

```
ID   AB000263 standard; RNA; PRI; 368 BP.
```

```
XX
AC   AB000263;
XX
DE   Homo sapiens mRNA for prepro cortistatin like peptide, complete cds.
XX
SQ   Sequence 368 BP;
AB000263  Length: 368  Check: 4514  ..
       1  acaagatgcc attgtccccc ggcctcctgc tgctgctgct ctccggggcc acggccaccg
      61  ctgccctgcc cctggagggt ggccccaccg gccgagacag cgagcatatg caggaagcgg
     121  caggaataag gaaaagcagc ctcctgactt tcctcgcttg gtggtttgag tggacctccc
     181  aggccagtgc cgggcccctc ataggagagg aagctcggga ggtggccagg cggcaggaag
     241  gcgcacccca ccagcaatcc gcgcgccggg acagaatgcc ctgcaggaac ttcttctgga
     301  agaccttctc ctcctgcaaa taaaacctca cccatgaatg ctcacgcaag tttaattaca
     361  gacctgaa
```

1.2.3.7 GenBank format (NCBI's DNA Sequence Database)

GenBank format sequence file can contain several sequences. The example is shown below.

```
LOCUS       AB000263                368 bp    mRNA    linear   PRI 05-FEB-1999
DEFINITION  Homo sapiens mRNA for prepro cortistatin like peptide, complete
            cds.
ACCESSION   AB000263
ORIGIN
        1 acaagatgcc attgtccccc ggcctcctgc tgctgctgct ctccggggcc acggccaccg
       61 ctgccctgcc cctggagggt ggccccaccg gccgagacag cgagcatatg caggaagcgg
      121 caggaataag gaaaagcagc ctcctgactt tcctcgcttg gtggtttgag tggacctccc
      181 aggccagtgc cgggcccctc ataggagagg aagctcggga ggtggccagg cggcaggaag
      241 gcgcacccca ccagcaatcc gcgcgccggg acagaatgcc ctgcaggaac ttcttctgga
      301 agaccttctc ctcctgcaaa taaaacctca cccatgaatg ctcacgcaag tttaattaca
      361 gacctgaa
//
```

1.2.3.8 IG format

The sequence file in IG format can contain several sequences. However comment lines are acceptable in this format. The following example shows IG format sequence data.

```
; comment
```

```
AB000263
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCC
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC
CTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGG
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCC
CTGCAGGAACTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAG
TTTAATTACAGACCTGAA1
```

1.3 Related Works

There has been much effort made to overcome biological data integration challenges. The first generation bioinformatics solution employs a series of non-interoperable and non-scalable quick fixes to translate data from one format to another [11]. The second generation data integration solutions provide a more structured environment for code reuse, flexibility and thereby give robust integration [11]. However, all solutions must be able to access and retrieve relevant data from a broad range of disparate data sources, transform retrieved data into designated data model, provide a rich common data model for extracting retrieved data and presenting integrated data objects to the end user applications. The approaches are the Data Warehouse approach, the Federation approach and the Mediator approach. Another solution is Ontology. Ontology is defined as a description of concepts and relationships that exist among concepts for a particular domain of knowledge.

One of the pioneers University of Pennsylvania developed a data integration system called KLEISLI [11]. KLEISLI is a mediator system that encompasses a complex data objects model, high-level query language and a powerful query optimizer. However since it is mediator system KLEISLI does not contain an integrated database management system. Later on University of Pennsylvania developed a new system K2 [11]. K2 is a successor of KLEISLI. Though K2 is a successor of KLEISLI, it has lots of new features including the concept of dictionaries.

SRS [11] (Sequence Retrieval System) is another data integration system developed by European Bioinformatics Institute (EBI). SRS is a read only Data Warehouse and it uses

raw data from EMBL, Swiss-Prot, Medline, HTML and XML files. While SRS shields users from underlying data sources, TAMBIS (Transparent Access to Multiple Bioinformatics Information Sources) uses ontology of molecular biology and bioinformatics to manage presentation and use of source. When TAMBIS receives a query, it automatically identifies the source that can be used to answer the query and construct a valid and efficient plan.

Another mediator approach is P/FDM [11]. P/FDM determines which databases are relevant for answering user's queries. Accordingly it breaks user query into multiple sub queries and sends it to different external data sources.

The result of NCBI's [13] (National Council for Bioinformatics Information) endeavor to integrate biological data is GenExpress [11]. The goal of GenExpress data management system is to integrate expression data with sample and Gene annotations to explore, interpret and analyze expression data. Another approach is DiscoveryLink [11] and it enables integration of diverse data from diverse sources into a single, virtual database, with the goal of making it easier for scientists to find the information they need to prevent and cure diseases.

On top of the scientific, commercial and educational organizations, the individuals or group of researchers are also trying to find a solution to data integration problems of biological data. Bruce Blackwell and Siva Ravada [3] made an effort to integrate biological data by using Oracle9i. Oracle9i is an Object Relational Database Management System. Jake Yue Chen, John V. Carlis [5] proposed Similar-join, a DBMS package to extract data in a more efficient way. L. Krishnamurthy, J. Nadeau, G. Ozsoyoglu, M. Ozsoyoglu, G. Chaeffer, M. Tasan, W. XU [6]'s effort is to find out the DNA sequence by using Pathway Database.

It is time for IT professionals and biologists to work together to overcome ever enduring challenges (finding, extracting, merging and synthesizing) of biological data. In the forthcoming decades the biologists will require a robust, comprehensive information

integration infrastructure underlining all aspects of data. Although tremendous progress has been made in this regards, biological information discovery process, at semantic level is still a major challenge.

1.4 Hypothesis

After going through a details discussion with my committee members, it is concluded that vast amount of biological data, its nature and characteristics, diversity and heterogeneity are the major causes of this ongoing, enduring problem (searching, extracting, merging and synthesizing biological data).

Our need is an integrated system that enables a biological scientist to searching, extracting, merging and synthesizing biological data from disparate data sources efficiently.

We propose a system that uses a Relational Database Management System as backend data repository for scientific data analysis. The System captures DNA sequence data from variety of physical data sources, transforms diverse data and stores and manages it in such a way that data retrieval and analysis become easier for Scientists.

Over the past eleven years I have been consistently working on Relational Database Management Systems. I have developed several systems while working for Global Integrated Software Solutions Inc. Also during the tenure of my teaching career at Seneca College of Applied Arts and Technology at York University, Toronto, Canada, (http://cs.senecac.on.ca/~prabin.dutta/), I encountered lots of challenges with Relational Database Management System. My industrial experience, experimental observations and intuition compel me to believe that Relational Database Management System is one of the best approaches to integrate DNA Sequence data scattered in disparate sources.

1.5 Publicly Available Data Sources

As the molecular biology research explodes from individual researchers to big research organizations, the number and size of publicly available data sources are also growing exponentially. There are hundreds of publicly available data sources currently available now. These databases are divided primarily into two types.

1.5.1 Primary Databases

Primary databases maintain original submission by experimentalists and scientists. The submitter has full control over those databases. Some examples of primary databases are GenBank, DDBJ, EMBL, SNP, and GEO.

1.5.2 Derivative Databases

Derivative databases are built from primary databases. The content of the derivative databases are controlled by third party organizations who has the soul intent to create them. Some examples of derivative databases are Refseq, TPA, RefSNP, UniGene, and NCBI Protein.

The proposed solution RelationalDNA has a derivative database attached to it.

For this research, we will be using only three primary databases.

1. DDBJ
2. GenBank
3. EMBL

These three databases make International Nucleotide Sequence Database Collaboration. These three databases are synchronized every 24 hours and hence they maintain same piece of information. But the format in which data being stored in each database is different. We will take a closer look at GenBank as this data source will be used heavily for this research.

1.6 GenBank [13]

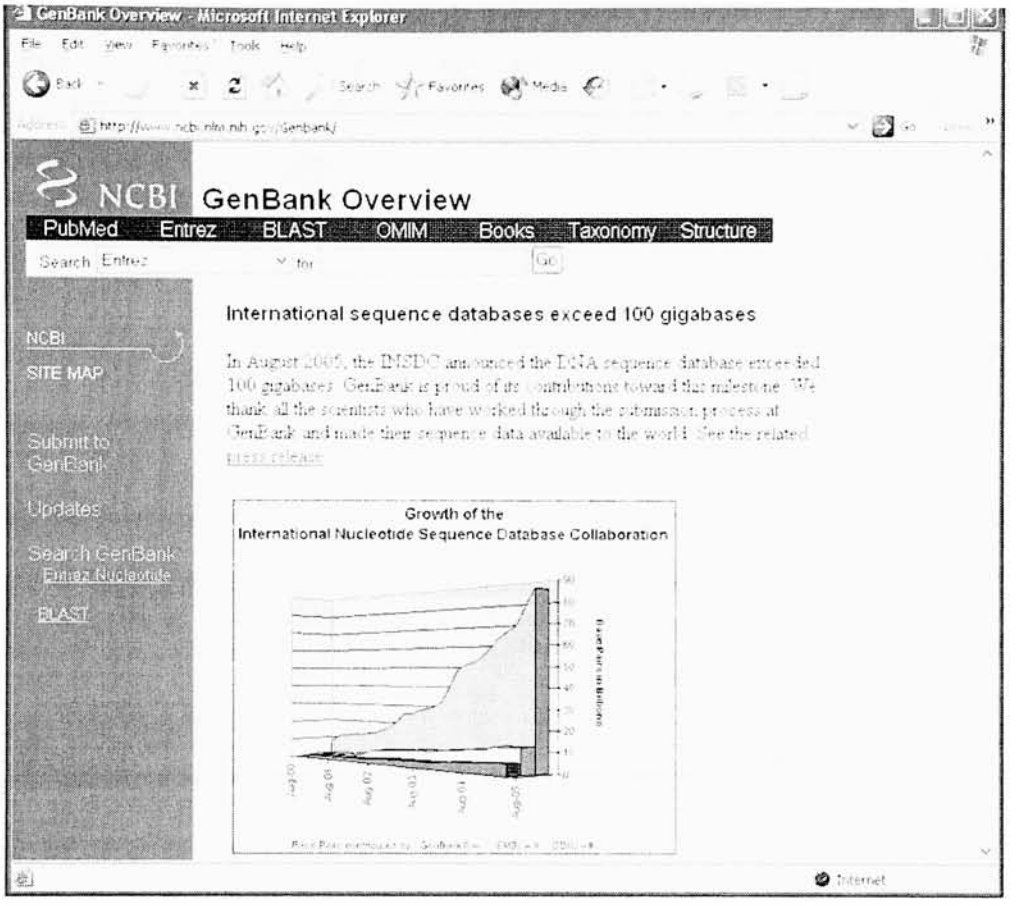


Fig 1.5: GenBank web site

GenBank is the NIH genetic sequence database. This is an annotated collection of all publicly available DNA sequences. The size of GenBank database is 100 GB. At this time there are approximately 65,369,091,950 bases in 61,132,599 sequence records in the traditional GenBank division and each record in GenBank is in GenBank flat file format GenBank allows scientists to submit and update data.

There are several methods used to search GenBank database. Primary method for searching GenBank is its built-in tool Entrez [14]. Entrez is a client-server system for retrieval of information related to molecular biology.

A snapshot of Entrez is shown below.

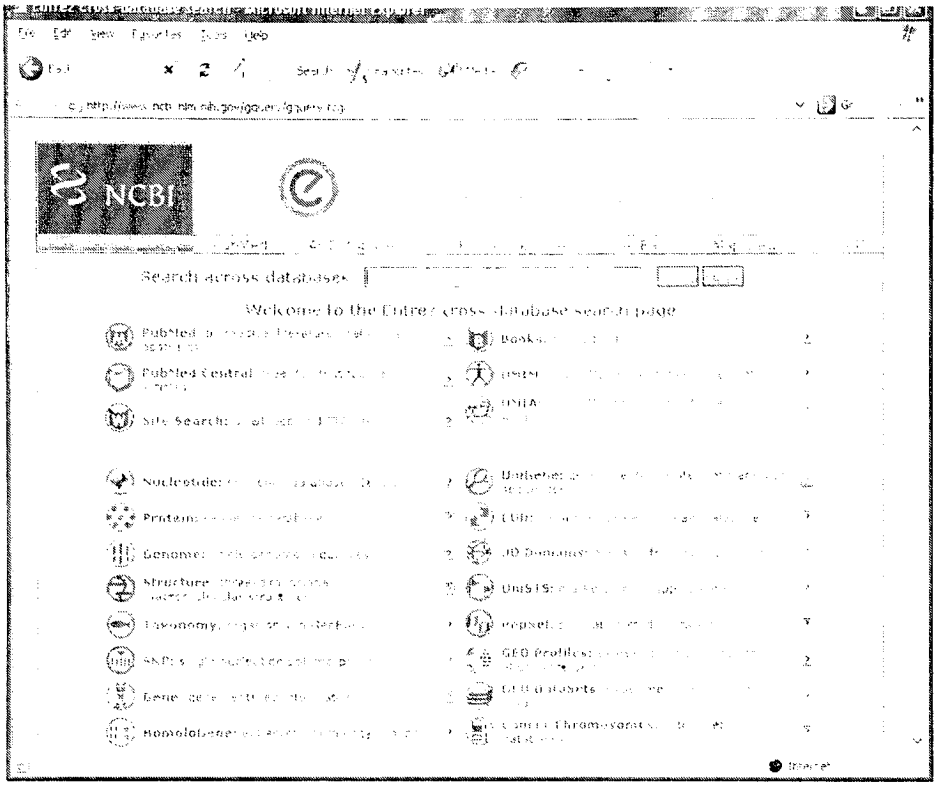


Fig 1.6: Entrez to retrieve DNA sequenced data

1.7 Roadmap

The rest of the report is organized as follows.

We will design of a backend data repository for the proposed data integration solution, RelationalDNA, in chapter 2. DNA sequenced data are collected from GenBank for scientific analysis. After complete analysis of raw data (DNA sequenced data) we will create user views and will undertake a complete normalization (NF1, NF2 and NF3) of these user views. Chapter 2 completes with creation of a fully normalized database.

In chapter 3, we will design and implement all other components of our proposed system (RelationalDNA). The proposed system has three major components. Firstly, there is a back end relational database for data repository. Secondly, we have a seamless data entry

mechanism to load extracted data. In order to create a suitable file to load data into our database, we will design and implement a parser (written in Java) which convert raw DNA sequenced data file to a tab separated file. Then we will design and implement loading mechanism to load data into our back end data repository. Thirdly, we will design and implement a data retrieval mechanism. To accomplish this goal, we will use mostly Oracle built in techniques. We will create a client tool (written in Java) to be used by a biological scientist for data retrieval.

In chapter 4, we will do an analysis to find the efficacy of the proposed system with existing systems.

Chapter 5 is the conclusions. In this chapter firstly, we will define current status of this work. Secondly, we will have a detailed discussion about future work on the same topic and cover lesson learned from this work.

2. Overview of RelationalDNA

Complex data centric applications necessitate a stable and reliable back end data repository. This chapter focuses various facets of designing a data warehouse. Inevitable components of data warehouse such as fact table, dimension table and star schema are discussed at the first section of this chapter. An intricate part of the database design process is database normalization and we have carried out a complete normalization in second section of this chapter. Since it is a data warehouse system, much emphasis is made on Normalization Form 1, Normalization Form 2 and Normalization Form 3. The design process eventually leads us to create a star schema based on normalized tables in section three. This chapter concludes with a thorough discussion of merits and demerits of Relational Database Management System.

2.1 Proposed Relational Database Management System

The design process of data warehouse significantly differs from online transaction processing systems in that the raw data stored in data warehouse is captured, extracted and transformed from disparate data sources. Each external data source uses its own domain specific data model and schema. Due to both syntactic and semantic differences of various data sources, creating a unified schema is always a challenge in data warehouse and has received substantial attention in research.

2.1.1 Introduction to Data Warehouse

The proposed system, RelationalDNA, is a collection of data marts representing DNA sequenced data from different data sources. It is an integrated decision support database whose contents are derived from various operational databases and contains historical, summarized information. The data structure is optimized to perform well aggregate queries running on a large amounts data. Unlike Online Transaction Processing Systems, the structure of data warehouse is easier for end users (biological scientists) to navigate, understand and query against it. In a data warehouse system, significant effort is required

to transform data to a form that is suitable for transporting to destination system. In RelationalDNA we will design and implement custom parsers to transform data (Custom parsers will be discussed in Chapter 3).

The following list depicts objects on which Data warehouse is built.

2.1.2 Fact Table

Fact tables are also known as detailed tables in data warehouse. They contain detailed facts and transactions within warehouse. In a regular schedule fact tables are populated with transformed data from other systems. Though there is less number of fact tables in a data warehouse, their size is really big. They include compound keys, which organize the transactions into dimensions. An example of fact table in RelationalDNA is the **GB_ SEQUENCE** table.

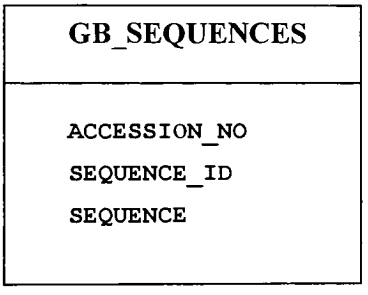


Fig 2.1 the fact table of RelationalDNA

2.1.3 Dimension Table

Dimension tables describe categories of information contained within the fact table. Normally dimension tables are created to perform transactions by product, time, and location or by organizational units. Dimension tables are very small in size and they rarely change. Sometime they are also called lookup or reference table. In RelationalDNA, the following tables are considered as Dimension tables.

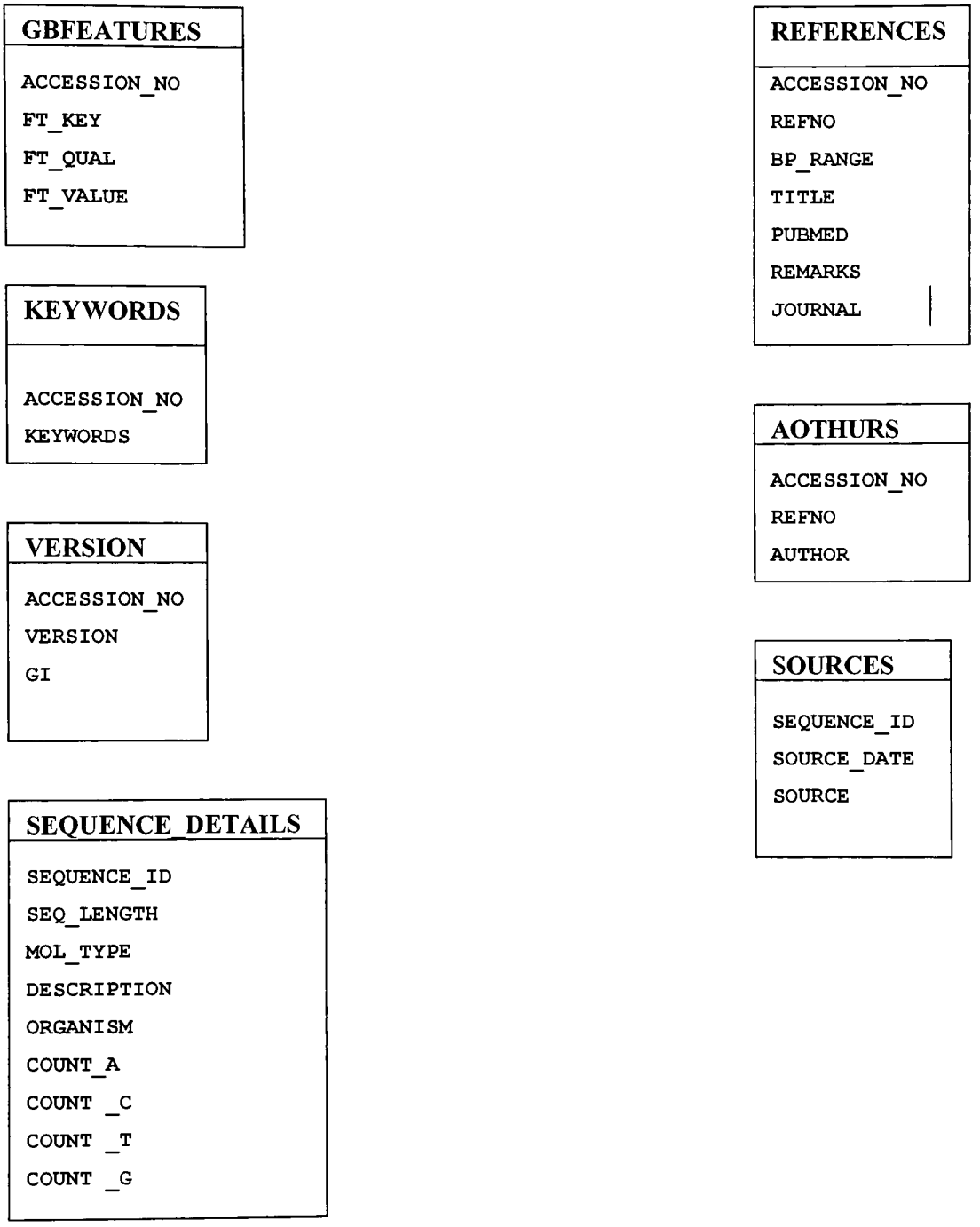


Fig 2.2 the dimension tables of RelationalDNA

2.1.4 Star Schema

Star Schema defines a series of dimension tables around the fact table. The following diagram depicts the star schema of RelationalDNA.

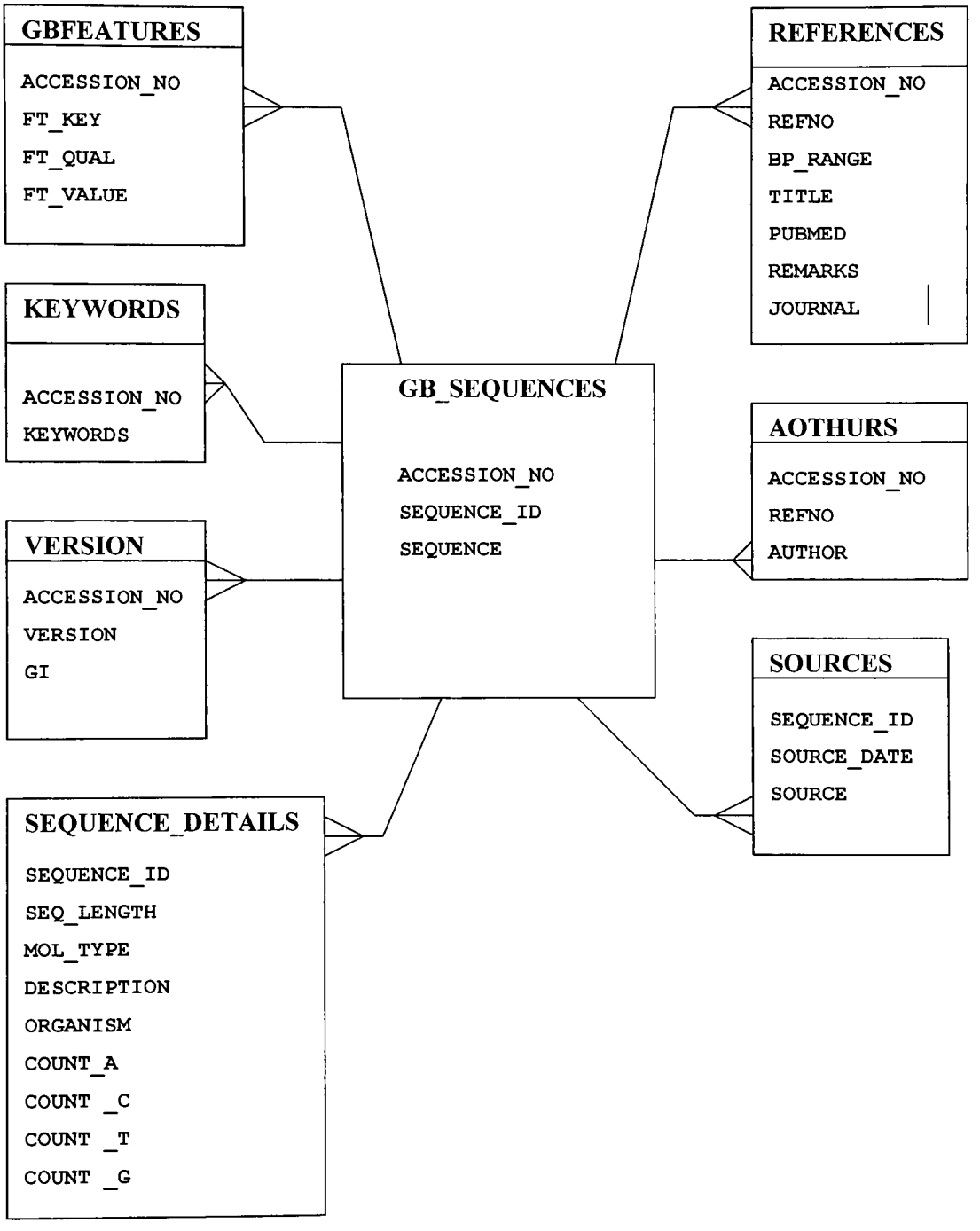


Fig 2.3 the star schema of RelationalDNA

The above diagram depicts the final draft of proposed data warehouse system design. However to achieve this goal we have undertaken a through and long process of database design and normalization. In next a few sections we will be discussing database design and normalization process in details.

2.2 Traditional systems development life cycles

The traditional system development life cycle has seven phases. These phases are briefly described below.

Phase 1: Project identification and selection

Purpose: To develop a preliminary understanding of the business situation that has caused a request for a new or an enhanced information system

Deliverables: A formal request to conduct a project to design and develop an information system solution to business problems and opportunities.

Phase 2: Project initiation and planning

Purpose: To state business situations and how information systems might help solve a problem or make an opportunity possible.

Deliverables: A written request to study the possible changes to an existing system or the development of the new system.

Phase 3: Analysis

Purpose: To analyze the business situation thoroughly to determine requirements, to structure those requirements and to select among competitive system features.

Deliverables: The functional specification for the system that meets user requirements and is feasible to develop and implement.

Phase 4: logical design

Purpose: To elicit and structure all information requirements.

Deliverables: Detailed functional specifications of all data, forms, reports, displays and processing rules.

Phase 5: Physical design

Purpose: To develop all technology and organizational specifications.

Deliverables: Program and database structure, technology purchases, physical site plans and organizational redesign.

Phase 6: Implementation

Purpose: To write programs, build data files, test and install the new system, train users and finalize documentation.

Deliverables: Programs that work accurately and to specifications, documentations and training materials.

Phase 7: Maintenance

Purpose: To monitor the operations and usefulness of a system and repair and enhance the system

Deliverables: Periodic audits to the system to demonstrate whether the system is accurate and still meets our need.

We have been working on this project directly and indirectly from January 2004. Therefore phase 1 through phase 3 (Project identification and selection, Project Initiation

and Planning and Analysis) have already been completed under the supervision of Professor R K Raj.

Other phases are going to be discussed in the remaining sections of this chapter.

2.3 Database Design

As we have mentioned in the previous section that the purpose of database design phase is to elicit and structure all information requirements. A detailed functional specification of all data, forms, reports, displays and processing rules are presented at the end of database design phase. The minimal data rule is applied.

"**All that is needed is there, and all that is there is needed**"

Our database design goals were

1. to determine end-user views, outputs and transaction processing requirement
2. to define entities, attributes and their relationship
3. to identify main processes: seamless data (DNA sequenced data) entry to the system, record update by original author, and seamless data retrieval from the system
4. to define the location of tables and access requirement.

During the initial study phase of this project we tried to determine the entities involved and to create a separate table for each type of entity. The following is partial set of data we have collected from GenBank. The original file had 53 references and 2558bp and most of the references and part of the sequence are removed.

```
LOCUS       NM_002639              2558 bp    mRNA    linear   PRI 23-APR-2006
DEFINITION  Homo sapiens serpin peptidase inhibitor, clade B (ovalbumin),
            member 5 (SERPINB5), mRNA.
ACCESSION   NM_002639
VERSION     NM_002639.2  GI:52851464
KEYWORDS    .
SOURCE      Homo sapiens (human)
ORGANISM    Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini;
            Catarrhini; Hominidae; Homo.
```

```
REFERENCE   1  (bases 1 to 2558)
AUTHORS     Yin,S., Li,X., Meng,Y., Finley,R.L. Jr., Sakr,W., Yang,H., Reddy,N.
            and Sheng,S.
TITLE       Tumor-suppressive maspin regulates cell response to oxidative
            stress by direct interaction with glutathione S-transferase
JOURNAL     J. Biol. Chem. 280 (41), 34985-34996 (2005)
PUBMED      16049007
REMARK      GeneRIF: maspin, through its direct interaction with glutathione
            S-transferase, may inhibit oxidative stress-induced ROS generation
            and vascular endothelial growth factor A induction
REFERENCE   2  (bases 1 to 2558)
AUTHORS     Chim,S.S., Tong,Y.K., Chiu,R.W., Lau,T.K., Leung,T.N., Chan,L.Y.,
            Oudejans,C.B., Ding,C. and Lo,Y.M.
TITLE       Detection of the placental epigenetic signature of the maspin gene
            in maternal plasma
JOURNAL     Proc. Natl. Acad. Sci. U.S.A. 102 (41), 14753-14758 (2005)
PUBMED      16203989
REMARK      GeneRIF: The maspin gene promoter was found to be hypomethylated in
            placental tissues and densely methylated in maternal blood cells
            thus serving as a marker for noninvasive prenatal assessment.
FEATURES             Location/Qualifiers
     source          1..2558
                     /organism="Homo sapiens"
                     /mol_type="mRNA"
                     /db_xref="taxon:9606"
                     /chromosome="18"
                     /map="18q21.3"
     gene            1..2558
                     /gene="SERPINB5"
                     /note="synonyms: PI5, maspin"
                     /db_xref="GeneID:5268"
                     /db_xref="HGNC:8949"
                     /db_xref="HPRD:HPRD_01111"
                     /db_xref="MIM:154790"
     CDS             68..1195
                     /gene="SERPINB5"
                     /go_function="serine-type endopeptidase inhibitor
                     activity"
                     /go_process="cell motility [pmid 8290962]"
                     /note="protease inhibitor 5 (maspin)"
                     /codon_start=1
                     /product="serine (or cysteine) proteinase inhibitor, clade
                     B (ovalbumin), member 5"
                     /protein_id="NP_002630.1"
                     /db_xref="GI:4505789"
                     /db_xref="GeneID:5268"
                     /db_xref="HGNC:8949"
                     /db_xref="HPRD:HPRD_01111"
                     /db_xref="MIM:154790"
                     /translation="MDALQLANSAFAVDLFKQLCEKEPLGNVLFSPICLSTSLSLAQV
                     GAKGDTANEIGQVLHFENVKDIPFGFQTVTSDVNKLSSFYSLKLIKRLYVDKSLNLST
                     EFISSTKRPYAKELETVDFKDKLEETKGQINNSIKDLTDGHFENILADNSVNDQTKIL
                     VVNAAYFVGKWMKKFPESETKECPFRLNKTDTKPVQMMNMEATFCMGNIDSINCKIIE
                     LPFQNKHLSMFILLPKDVEDESTGLEKIEKQLNSESLSQWTNPSTMANAKVKLSIPKF
                     KVEKMIDPKACLENLGLKHIFSEDTSDFSGMSETKGVALSNVIHKVCLEITEDGGDSI
                     EVPGARILQHKDELNADHPFIYIIRHNKTRNIIFFGKFCSP"
     STS             88..222
                     /gene="SERPINB5"
                     /standard_name="GDB:435245"
                     /db_xref="UniSTS:157216"
     STS             1199..1357
                     /gene="SERPINB5"
                     /standard_name="GDB:435240"
                     /db_xref="UniSTS:157215"
     STS             1588..1920
                     /gene="SERPINB5"
                     /standard_name="SHGC-12569"
                     /db_xref="UniSTS:11720"
     STS             2101..2286
                     /gene="SERPINB5"
                     /standard_name="RH17976"
```

```
                     /db_xref="UniSTS:58530"
     polyA_signal    2537..2542
                     /gene="SERPINB5"
ORIGIN
        1 ttgtgctcct cgcttgcctg ttccttttcc acgcattttc caggataact gtgactccag
       61 gcccgcaatg gatgccctgc aactagcaaa ttcggctttt gccgttgatc tgttcaaaca
      121 actatgtgaa aaggagccac tgggcaatgt cctcttctct ccaatctgtc tctccacctc
      181 tctgtcactt gctcaagtgg gtgctaaagg tgacactgca aatgaaattg gacaggttct
      241 tcattttgaa aatgtcaaag atataccctt tggatttcaa acagtaacat cggatgtaaa
      301 caaacttagt tccttttact cactgaaact aatcaagcgg ctctacgtag acaaatctct
      361 gaatctttct acagagttca tcagctctac gaagagaccc tatgcaaagg aattggaaac
      421 tgttgacttc aaagataaat tggaagaaac gaaaggtcag atcaacaact caattaagga
      481 tctcacagat ggccactttg agaacatttt agctgacaac agtgtgaacg accagaccaa
      541 aatccttgtg gttaatgctg cctactttgt tggcaagtgg atgaagaaat ttcctgaatc
      601 agaaacaaaa gaatgtcctt tcagactcaa caagacagac accaaaccag tgcagatgat
      661 gaacatggag gccacgttct gtatgggaaa cattgacagt atcaattgta agatcataga
      721 gcttcctttt caaaataagc atctcagcat gttcatccta ctacccaagg atgtggagga
      781 tgagtccaca ggcttggaga agattgaaaa acaactcaac tcagagtcac tgtcacagtg
      841 gactaatccc agcaccatgg ccaatgccaa ggtcaaactc tccattccaa aatttaaggt
      901 ggaaaagatg attgatccca aggcttgtct ggaaaatcta gggctgaaac atatcttcag
      961 tgaagacaca tctgatttct ctggaatgtc agagaccaag ggagtggccc tatcaaatgt
     1021 tatccacaaa gtgtgcttag aaataactga agatggtggg gattccatag aggtgccagg
     1081 agcacggatc ctgcagcaca aggatgaatt gaatgctgac catcccttta tttacatcat
     1141 caggcacaac aaaactcgaa acatcatttt ctttggcaaa ttctgttctc cttaagtggc
     1201 atagcccatg ttaagtcctc cctgactttt ctgtggatgc cgatttctgt aaactctgca
     1261 tccagagatt cattttctag atacaataaa ttgctaatgt tgctggatca ggaagccgcc
     1321 agtacttgtc atatgtagcc ttcacacaga tagacctttt ttttttcca attctatctt
     1381 ttgtttcctt ttttcccata agacaatgac atacgctttt aatgaaaagg aatcacgtta
     1441 gaggaaaaat atttattcat tatttgtcaa attgtccggg gtagttggca gaaatacagt
     1501 cttccacaaa gaaaattcct ataaggaaga tttggaagct cttcttccca gcactatgct
     1561 ttccttcttt gggatagaga atgttccaga cattctcgct tccctgaaag actgaagaaa
     1621 gtgtagtgca tgggacccac gaaactgccc tggctccagt gaaacttggg cacatgctca
     1681 ggctactata ggtccagaag tccttatgtt aagccctggc aggcaggtgt ttattaaaat
     1741 tctgaatttt ggggattttc aaaagataat attttacata cactgtatgt tatagaactt
     1801 catggatcag atctggggca gcaacctata aatcaacacc ttaatatgct gcaacaaaat
     1861 gtagaatatt cagacaaaat ggatacataa agactaagta gcccataagg ggtcaaaatt
     1921 tgctgccaaa tgcgtatgcc accaacttac aaaaacactt cgttcgcaga gcttttcaga
     1981 ttgtggaatg ttggataagg aattatagac ctctagtagc tgaaatgcaa gaccccaaga
     2041 ggaagttcag atcttaatat aaattcactt tcatttttga tagctgtccc atctggtcat
     2101 gtggttggca ctagactggt ggcaggggct tctagctgac tcgcacaggg attctcacaa
     2161 tagccgatat cagaatttgt gttgaaggaa cttgtctctt catctaatat gatagcggga
     2221 aaaggagagg aaactactgc ctttagaaaa tataagtaaa gtgattaaag tgctcacgtt
     2281 accttgacac atagtttttc agtctatggg tttagttact ttagatggca agcatgtaac
     2341 ttatattaat agtaatttgt aaagttgggt ggataagcta tccctgttgc cggttcatgg
     2401 attacttctc tataaaaaat atatatttac caaaaaattt tgtgacattc cttctcccat
     2461 ctcttccttg acatgcattg taaataggtt cttcttgttc tgagattcaa tattgaattt
     2521 ctcctatgct attgacaata aaatattatt gaactacc

//
```

Contents of this file are explained below.

- File Header
    - File info line:
        - File name
        - Full database name ('GenBank')
        - Brief description of the file
    - Date: the date of current release in the form `day month year'
    - Release number: the current release number
        - Major release number
        - Version
    - Title: title of the file

- Size number:
  - Number of entries
  - Number of bases
  - Number of sequence

Following elements or fields are related to GenBank entries

- LOCUS field
  - Locus name
  - Sequence Length
  - Molecule Type: The type of molecule that was sequenced
  - GenBank Division: One of the 17 sequence divisions a record belongs to
  - Modification Date: The date of last modification
- DEFINITION field
  - Scientific organism, gene/protein name,
  - Brief description of the sequence's function if the sequence is non-coding or completeness qualifier, such as "complete cds" and its description if the sequence has a coding region (CDS)
- ACCESSION: The unique identifier for a sequence record
- VERSION: A nucleotide sequence identification number that represents a single, specific sequence in the GenBank database
  - GI: "GenInfo Identifier" sequence identification number
- KEYWORDS: Words or phrases describing the sequence.
- SOURCE: Free-format information including an abbreviated form of the organism name, sometimes followed by a molecule type
- REFERENCE field
  - REFERENCE ID: Sequential number
  - AUTHORS: List of authors
  - TITLE: Title of the published work or tentative title of an unpublished work, or direct submission substitution
  - JOURNAL: MEDLINE abbreviation of the journal name
  - MEDLINE: MEDLINE unique identifier (UID)
  - Direct Submission : Contact information of the submitter
- FEATURE: Location of each feature

- Source: Mandatory feature in each record that summarizes the length of the sequence, scientific name of the source organism, and Taxon ID number. Can also include other information such as map location, strain, clone, tissue type, etc.,
  - Organism name
  - Taxon: A stable unique identification number for the taxon of the source organism
  - Chromosome type
  - Map type

Followings are two example features, a complete list of features can be found from GenBank documentation and release note.

- CDS: Coding sequence; region of nucleotides that corresponds with the sequence of amino acids in a protein (location includes start and stop codons).
  - Gene type
  - note
  - codon start position
  - product
  - protein_id: A protein sequence identification number in the accession.version format
  - GI
  - translation: The amino acid translation corresponding to the nucleotide coding sequence (CDS).

- Gene: A region of biological interest identified as a gene and for which a name has been assigned
  - gene type

- BASE COUNT: The number of A, C, G, and T bases in a sequence.

- ORIGIN: Experimentally determined restriction cleavage site or the genetic locus in FASTA format representation. The ORIGIN may be left blank, may appear as "Unreported," or may give a local pointer to the sequence start.

A thorough raw data analysis leads us to the following user views.

Header → LocusNo, version, sequence length, molecule type, date, source, description, organism, keywords, reference, reference number, base range, title, journal, pubmed, remark, authors

Feature→Feature key, feature value,

Sequence→ sequence

It is observed that the header itself has too many attributes and is difficult to normalize.

The next step of logical design is normalization.

2.4 Database Normalization

Normalization is pivotal for success of logical database design and has always been drawing substantial attention from researchers as well as from database designers. Normalization rules are basically processes or steps taken to allow efficient creation of relational databases. Following these rules lead an administrator to create a set of tables that

1) Allow data to be organized in an efficient manner

2) Reduce disk space requirements for data storage

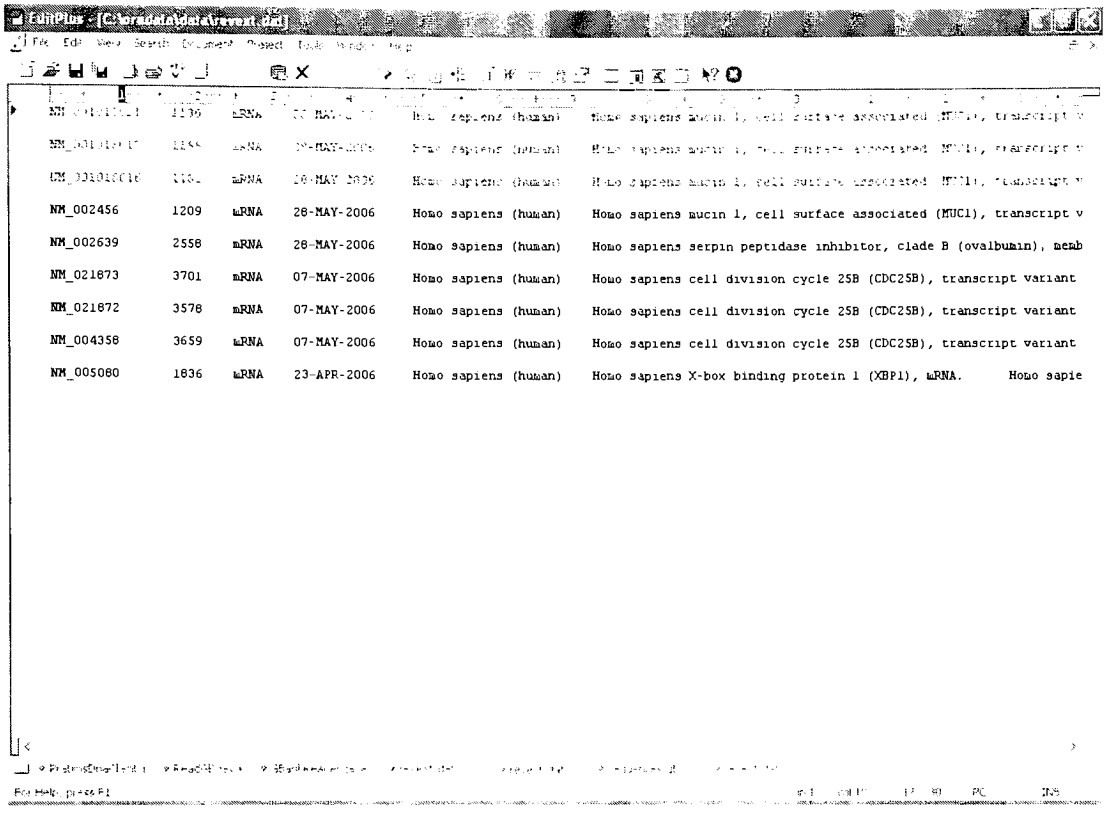
3) Minimize or eliminate redundant data

4) Greatly reduce the chance of data entry errors

5) Provide programmers with a stable data platform from which to create efficient applications.

A poor database design can cripple an application, producing problems with redundancy, inaccuracy, inconsistency, and concurrency of data. Normalization serves to reduce, if not eliminate, the problems with data stated above. The custom parser (custom parsers are going to be designed and implemented in chapter 3) breaks down the GenBank flat file into a tab delimited file. Our goal is to insert the contents of this tab delimited file into relations created through the process normalization.

The process of normalization assigns attributes to entities. It reduces data redundancies and works through a series of stages called normalization forms. The first three stages are described as first normalization form (1NF), and second normalization form (2NF) and third normalization form (3NF). The sample output file created by the parser is shown below



NM_002456	1209	mRNA	28-MAY-2006	Homo sapiens (human)	Homo sapiens mucin 1, cell surface associated (MUC1), transcript v
NM_002639	2558	mRNA	28-MAY-2006	Homo sapiens (human)	Homo sapiens serpin peptidase inhibitor, clade B (ovalbumin), memb
NM_021873	3701	mRNA	07-MAY-2006	Homo sapiens (human)	Homo sapiens cell division cycle 25B (CDC25B), transcript variant
NM_021872	3578	mRNA	07-MAY-2006	Homo sapiens (human)	Homo sapiens cell division cycle 25B (CDC25B), transcript variant
NM_004358	3659	mRNA	07-MAY-2006	Homo sapiens (human)	Homo sapiens cell division cycle 25B (CDC25B), transcript variant
NM_005080	1836	mRNA	23-APR-2006	Homo sapiens (human)	Homo sapiens X-box binding protein 1 (XBP1), mRNA. Homo sapie

Fig 2.4: The parsed GenBank flat file

The following figure depicts dependencies of the input file:

ACCESSION_NO → SEQ_LENGTH, MOL_TYPE, SEQUENCE, DESCRIPTION, ORGANISM, COUNT_A, COUNT_C, COUNT_T, COUNT_G, VERSION, GI, REFNO, AUTHOR, FT_KEY, FT_QUAL, FT_VALUE, REFNO, BP_RANGE, TITLE, PUBMED, REMARKS, JOURNAL, Source DATE, SOURCE

The term first normalization form (1NF) describes the tabular format in which

- All the key attributes are defined
- There are no repeating groups in the table, In other words each row/column intersection can contain one and only one value, not a set of values
- All attributes are dependent on the primary key

A table is in Second Normal form (2NF) if

- It is in 1NF and
- It includes no partial dependencies, that is no attribute is dependent on only a portion of the primary key

A table is in third normal form (3NF) if

- It is in 2NF and
- It contains no transitive dependencies.

We convert our data to 3rd normalization form as follows:

The table structure is shown in the following format

```
TABLE NAME (PRIMARY_KEY_ATTRIBUTES, DEPENDENT ATTRIBUTES)
```

```
GB_SEQUENCES (ACCESSION NO, SEQUENCE ID → SEQUENCE)

SEQUENCE_DETAILS (SEQUENCE ID → SEQ_LENGTH, MOL_TYPE, DESCRIPTION,
ORGANISM, COUNT_A, COUNT _C, COUNT _T, COUNT _G)

VERSION (ACCESION NO) → VERSION, GI)

AUTHORS (ACCESSION NO, REFNO → AUTHOR)

GBFEATURES (ACCESSION NO → FT_KEY, FT_QUAL, FT_VALUE)

REFERENCES (ACCESSION NO → REFNO, BP_RANGE, TITLE, PUBMED,
REMARKS, JOURNAL)

SOURCE (SEQUENCE ID → SOURCE DATE, SOURCE)

KEYWORDS (ACCESSION NO → ACCESSION_NO, KEYWORDS)
```

Fig 2.5: Normalized Schema

2.5 Merits of the System

2.5.1 Program data independence

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods. Also Application

programs and terminal activities remain logically unimpaired when information preserving changes of any kind that theoretically permit unimpairment are made to the base tables.

2.5.2 Minimal data redundancy

Normalized relations would contain only data necessary for a particular relation and dependency of attributes are controlled by enforcing primary key. In order to produce complex information intrinsically generated from star schema, multiple relations are queried. Therefore it is ascertained that the system does not contain redundant data. For example, information about an author is going to be stored in author's relation. However a reference to an author is cited from multiple relations.

2.5.3 Improved data consistency

An improperly normalized database is vulnerable to data anomalies because it stores data redundantly. In the event of a piece of data being stored in two locations and a transaction updates only one of the locations, then the data remains in inconsistent state. Because RelationalDNA is designed with minimal data redundancy principle, updates need to be made only in one place and the operation should propagate to all other references. In case the version number of a particular DNA sequence is altered, this update is made only on version relation. All other relation that cites version number would refer to version relation for version number. The ability to maintain improved data consistency makes the RelationalDNA data repository more robust and useable.

2.5.4 Improved data sharing

RelationalDNA has the ability to support a production or research environment with large number of users. This capability comes from the operating system features as well as from the application itself. Unlike a flat file, data manipulation actions could be

performed from multiple places and is still maintaining data on a central location. RelatinalDNA database is going to be stored in a central server and relations are accessed by multiple scientists at the same time. Data sharing capability improves system scalability and storage performance. Enterprises favor managing centralized data compared to managing distributed data.

2.5.5 Increased productivity of application development

Now application developers have a consistent set of data in a central location. This feature enables application developers to concentrate only on application development and not worrying about data itself. Hence application developers productivity increases drastically. By the same token, when a developer needs to update a piece of information, the developer updates it in one place and hence improves application development productivity.

2.5.6 Improve data accessibility and responsiveness

The process of information extraction from RelationalDNA is much faster than from that of a normal flat file and therefore data accessibility and responsiveness is improved. Advanced data accessibility options like indexes, clusters, index organized tables, materialized views are implemented in RelationalDNA's Relational Database Management System. Improving data accessibility and responsiveness is a continuous process in the industry and the process begins from the day of inception of the system.

2.6 Demerits of the System

2.6.1 New specialized personnel

Designing, implementing and maintaining a system like RelationalDNA needs highly skilled and specialized personnel. The recent trend of Information Technology shows that finding highly skilled and specialized personnel is a challenging task. However in flat

file and unstructured data modeling environment, anybody with minimal computer skills can maintain data. Also hiring very highly skilled personnel may cost too much money for an organization. While it is acceptable for commercial organizations, it might be a hindrance for research organizations.

2.6.2 Installation and management cost complexity

Installation and management cost complexity is also another important point to be considered. The application software is expensive. Especially vendors like Oracle, Microsoft licenses are going to cost enormous amount to an organization. The high end applications need specialized hardware too, which are also expensive. If the trade off between cost and performance is not optimum, it is definitely not worth to use proposed solution. Once the system is in place, it has to be maintained in a daily basis. Therefore the proposed solution is not universally applicable. A trade off analysis among cost, performance and usability is always required before making any decision.

2.6.3 Conversion Cost

Converting existing system to a new system always cost money, time and resource, and it is also inconvenient for users. In some instances a seamless conversion is not even possible without interrupting normal day to day operations. Again by the same token as above, conversion process needs a special team of professionals who will be working on the project for long time. The cost incurred for all these are very high and it needs to be considered while making a decision about migrating to the new system.

2.6.4 Need of explicit backup and recovery

Now that we have data stored in a central repository, we need explicit backup and recovery strategy in place. A well documented backup and recovery strategy will make sure that in case of disaster we could bring our data back. There are some built-in backup and recovery tools always come with most of the Database Management Systems.

However there are lots of advanced backup and recovery tools (for example Veritas, Trivoli etc) available in industry. To operate these highly advanced tools we need highly skilled professionals and again in this dynamic Information Technology age it is very difficult to find them.

3. System Design and Implementation

The process of building a new data warehouse system relevant to DNA sequenced data involves transforming, integrating, cleansing multiple data sources as well as adding new data and annotations. This chapter covers complete implementation of the proposed system RelationalDNA. The experimental project is composed of five subsystems – a custom parser, a data loader, a data retrieval mechanism, a data validation mechanism and a logging system. The chapter begins with an overview of the proposed system. Parsing biological data (DNA sequenced data) is an ongoing challenge in bioinformatics and much research and development have been undertaken in this context. In Section 3.3 we introduce our proposed custom parser and its implementation. Next we discuss a custom data loader that enters data seamlessly into the system. Main focus will be made on Oracle built-in tool SQL*Loader and external tables. In order to retrieve data efficiently from data repository, we design and implement a data retrieval mechanism and it uses Oracle built-in techniques like SQL, PL/SQL, Materialized views and indexes. And finally we design and implement a data validation mechanism and logging system in this chapter.

3.1 System Overview

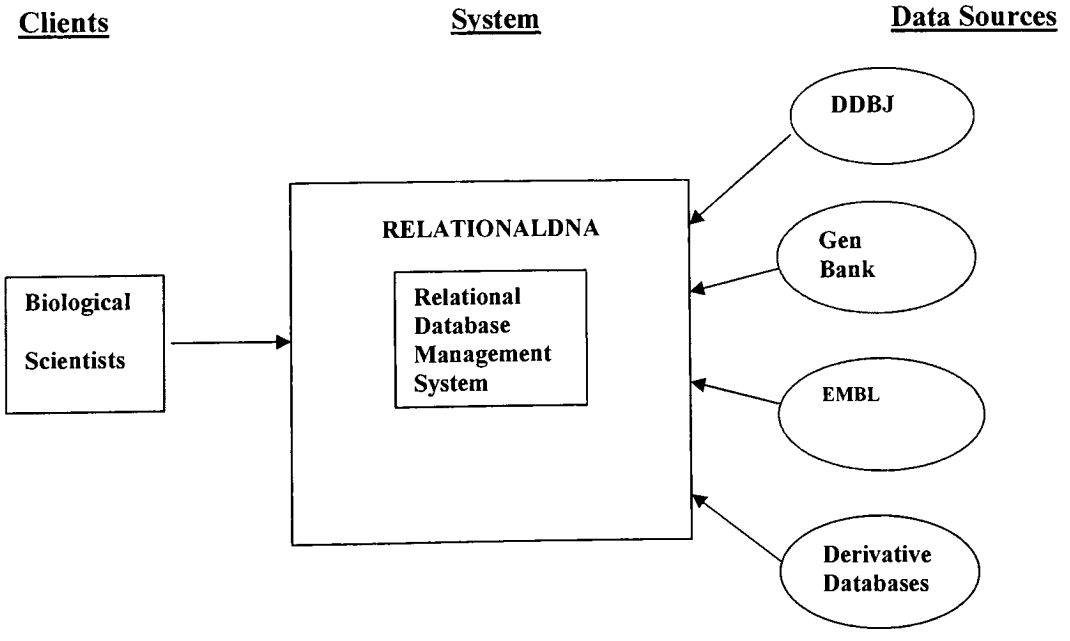


Fig 3.1: Integrating different data sources

In the process of building the body of the work which has led to this thesis and through the conversation with my mentor Professor Rajendra K Raj and Professor Paul Tymann, a set of desiderata have been articulated for the design of a DNA sequence data integration system. The following diagram depicts the system.

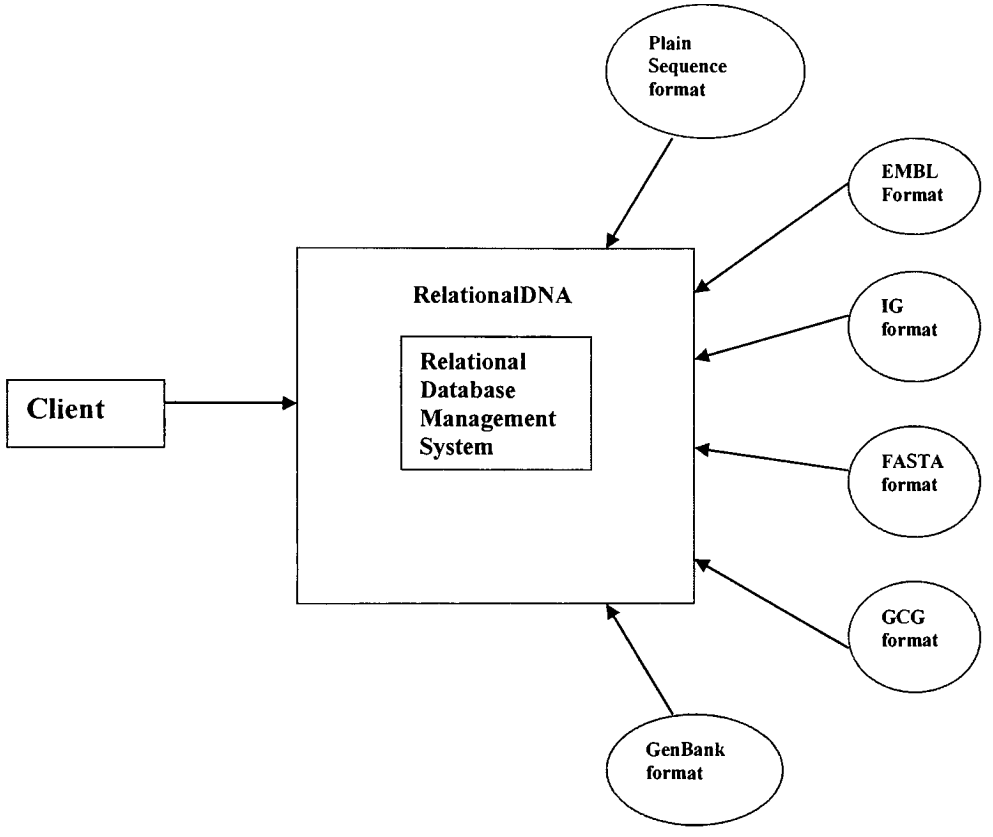


Fig 3.2: DNA sequence data is available in different format.

DNA sequence data is publicly available in many different sources and in different formats (different formats were discussed in details in section 1.2.2). This thesis proposes a solution to warehouse disparate data into one single repository.

3.2 Design and Implementation of Data Repository

In chapter 2 we have gone through the process of database design and normalization process. The database design and normalization process led us to achieve the star schema created in chapter 2.

The first step before we create this schema is to create a dedicated tablespace. In the proposed implementation we have created a tablespace called RelationalDNA. The initial size of the tablespace is kept minimal so that later on we could add more datafiles to increase its size.

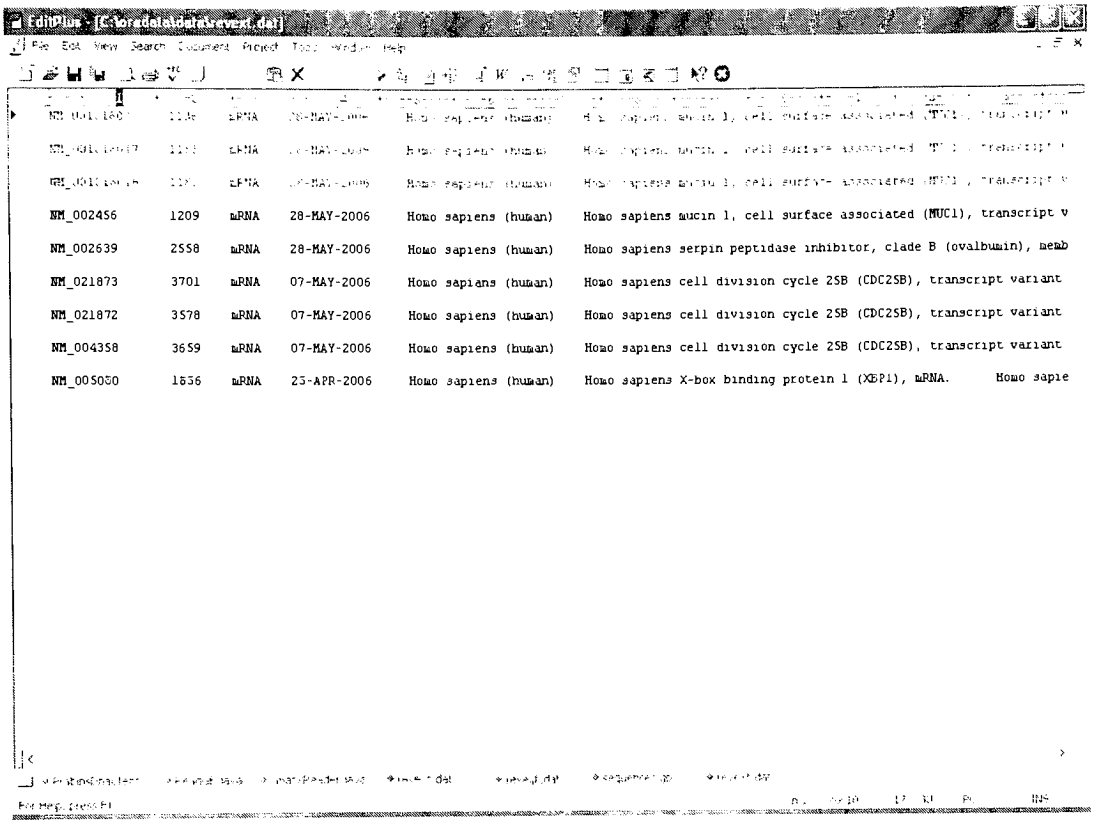
3.3 Design and implementation of Custom Parser

3.3.1 Introduction to Parsers

Most of the DNA sequenced data sources have their own data model and one of the daunting tasks of this experimental project is to parse DNA sequenced data to a suitable format so that it could be imported to the data repository. Our experimental system collects raw data from GenBank, and GenBank data model is flat file system. Because our primary objective is to load and retrieve data to and from the database, parsing and importing data is crucial to this experimental project. The data to be loaded to data repository is broken down into a parser for each entry. The resulting format of the parsed data is dependent on type of vendor specific data source and type of the data loader being used. We have decomposed the GenBank flat file into small tab delimited text file so that it is compatible with Oracle's built-in data loader and external table.

3.3.2 RelationalDNA Parser Implementation

The design goal of the custom parser is to break down GenBank flat file into small tab delimited file. In chapter 2, we have seen how data is stored in a GenBank flat file. Fig 3.4 shows how a GenBank flat file is converted to a tab delimited file. This tab delimited file is created by the custom parser. We used APIs from biojava to construct objects for our custom parser.



NM_002456	1209	mRNA	28-MAY-2006	Homo sapiens (human)	Homo sapiens mucin 1, cell surface associated (MUC1), transcript v
NM_002639	2558	mRNA	28-MAY-2006	Homo sapiens (human)	Homo sapiens serpin peptidase inhibitor, clade B (ovalbumin), memb
NM_021873	3701	mRNA	07-MAY-2006	Homo sapiens (human)	Homo sapiens cell division cycle 25B (CDC25B), transcript variant
NM_021872	3578	mRNA	07-MAY-2006	Homo sapiens (human)	Homo sapiens cell division cycle 25B (CDC25B), transcript variant
NM_004358	3659	mRNA	07-MAY-2006	Homo sapiens (human)	Homo sapiens cell division cycle 25B (CDC25B), transcript variant
NM_005080	1656	mRNA	23-APR-2006	Homo sapiens (human)	Homo sapiens X-box binding protein 1 (XBP1), mRNA. Homo sapie

Fig 3.3: A parser converted GenBank flat

3.4 Design and Implementation of Front End

A front end User Interface is designed and developed by using JAVA. In this design we are trying to make the front end as transparent as possible to the end users. A snap-shot of the front end user interface is attached herewith. The complete implementation code is also available in appendix – A.

Fig 3.4: The figure above displays DNA sequence for DNA whose Accession number is NM_002456.

Considering the client is a biological scientist, the user interface has several options to extract meaningful information from the database. The snap-shot shown above reveals some of these options that the client tool has. The biological scientist may like to extract the sequence, author, and journal information for a particular locus number. The client tool has these alternatives to select an option and result is going to be displayed in detailed pane. The client tools is also capable of displaying the number of A's, C's, T's and G's in a particular sequence.

This is a generic and basic client tool that could be used by anybody. However every biological research would be different and so is biological scientist's need. Once

scientist's requirement is learnt and understood, the base client tool can be easily extended to meet specific requirement.

3.5 Design and Implementation of Custom Data Loader

Our implementation of custom parser is shown in section 3.3. The custom parser extracts raw DNA sequenced data from disparate data sources and transforms it into a tab delimited flat file. The custom loader inserts tab delimited flat file into RelationalDNA's back end data repository. This process involves lots of technical challenges. Therefore most of the RDBMS vendors pay special attention to this issue and have built-in data transfer and data transformation tools on it. Since we are using Oracle 9i as back end data repository, Oracle 9i's built-in external table feature and JDBC are used in our project to load data. External table is a central repository of DNA sequenced data and data specific to a dimension or fact table is further extracted from the external table and is inserted to dimension or fact table.

3.6 Design and Implementation of Data Retrieval Mechanism

Another objective of our work is to retrieve sequence data as and when needed by a life scientist from the designed system. Seamless data retrieval mechanism retrieves DNA sequenced data from underlying database and returns it to the user. The amount of data that is going to be stored in the database is enormous. While retrieving data, all techniques to optimize query is being used. For example, usages of appropriate index, writing efficient SQL statement, using materialized views etc are some existing techniques. In this section first we will discuss how Structured Query Language (SQL) is used and followed by its successor Procedural Structured Query Language (PL/SQL). We will then discuss indexes, proper implementation of which will improve query performance drastically by avoiding full table scan. Finally we will discuss materialized views to improve I/O performance for complex queries that joins multiple tables by rewriting the query.

3.6.1 Structured Query Language (SQL)

Structured Query Language (SQL) is always incorporated with any Relational Database Management System implementation from its inception. In this experimental project, RelationalDNA, we are using JDBC to connect to the backend data repository to retrieve data. Result set is collected from multiple tables and complex joins are created. SQL code is being tuned time and again so that I/O operation is minimized as much as possible. We are enclosing herewith some SQL statements being used by the system in chapter 4.

3.6.2 Procedural Structured Query Language (PL/SQL)

Some PL/SQL stored procedures are created and called via JDBC to get desired results. The advantage of creating and using PL/SQL stored procedure is to minimize server resources and network utilizations. Complex user-defined cursors could be used to retrieve result set. Instead-of triggers, though we are not using in this experimental project, would be very helpful to update and insert complex biological data. Otherwise it would be impossible to accomplish the desired result. Other PL/SQL enhancements like batch processing, minimizing network load, using structured exception handling are also used in this project.

We are confident that later implementation of this project could exploit other advanced features of PL/SQL.

3.6.3 Indexes

As the DNA sequenced data grows exponentially, size of RelationalDNA's RDBMS is going to be immensely high. Therefore, the system, at any cost would not be able to afford full table scan. Special attention to the details is paid to avoid full table scan by using indexes. The database itself created indexes for all primary keys. On top of these built-in indexes, user defined indexes are also created. For low cardinality columns such

as authors_name, bitmap indexes are created. However for high cardinality columns B-Tree indexes are created.

We have used Oracle built-in EXPLAIN_PLAN utility to find out how data is being accessed from databases. None of the SQL statement suffers from full table scan.

3.6.4 Materialized Views

Since the queries to retrieve data are very complex, query rewrite feature is enabled through Materialized Views. Most queries are joining multiple tables. Retrieving data from multiple tables would be very expensive I/O operation even proper indexes are created. . Therefore we have created several materialized views. Materialized View refresh is also enabled so that users don't get out of date data from Materialized View.

In chapter 4 (Analysis), we will give a detail comparison of performance evaluation with and without Materialized views. Desired results could not have been achieved without using Materialized Views.

3.7 Design and Implementation of Data Validation

The integrity of Biological data is very important to the scientists as the scientific analysis is entirely dependent on raw data being used for analysis. In this experimental project, we are trying to use RDBMS integrity constraints as much as possible to maintain integrity of data at database level. Each table has a primary key and a referential constraint is added to the dimension tables.

User defined exceptions are created either through Java or through PL/SQL to notify end users regarding any violation of data integrity. Primary focus is given to the custom error messages so that error messages are self explanatory to the users. We are trying to avoid Oracle built in error messages so that users are not confused.

PL/SQL custom triggers are very useful to maintain high level of data integrity which could not be accomplished by normal database constraints. We are maintaining a table that contains information about invalid data. A row has been inserted into invalid_data table for each event of invalid data insertion.

Our goal of maintaining reliable data is accomplished by using Oracle built-in integrity constraints, creating custom exceptions and creating and implementing database triggers.

3.8 Design and Implementation of Logging

A trigger will insert a row into each log table when there is a DML operation performed on that table. Log tables are

GB_FEATURE_LOG
REFERENCES LOG
KEYWORDS_LOG
GB_SEQUENCE_LOG
ACCESSION_LOG
AUTHORS_LOG

3.9 Motif Discovery

Biological scientists encounter a major challenge to find important regions in DNA sequence where crucial functionalities are transcribed. The sequence pattern or motifs are normally found in promoter regions or regulatory proteins that denote intron/exton boundaries. Identifying motifs can provide the most meaningful insight into the mechanisms of transcriptional regulations [17]. The identification workflow is determined by proper data mining which is crucial because a single regulatory protein often recognizes a variety of similar sequences.

The most common and simple motif model is consensus sequence where a preferred sequence of nucleotides (ACTG) is determined. Position Weight Matrix (PWM) is

another motif model which is represented by a matrix of nucleotide scores indexed by letter and position. PWM is further investigated by [18] where pairwise dependencies are incorporated into Weight Matrix. Bayesian Networks [19] allows arbitrary dependencies between positions to model motif.

3.9.1 Motif Discovery Algorithms

According to [17], there are four different algorithms for motif discovery in practice – Enumerative method, Alignment method, EM (expectation-maximization) Algorithm and Gibbs sampling. The type of motif discovery algorithm being used is dependent on the type of motif model.

Consensus sequence model uses Enumeration method that involves exhaustive enumeration of words. Dictionary-based algorithm is a variation of Enumeration algorithm where sequences are generated from a dictionary of possible words. Alignment algorithm uses probability matrix to find common motifs in all input sequences. Many variations of alignment algorithms are developed or being developed. EM consists of two steps: in the E-step, the expected likelihood of the observed sequence data is calculated based on the current setting of the parameters, and in the M-step, the parameters are updated to maximize the expected-likelihood function. Gibbs algorithm is another method that involves drawing random samples of hidden variables from a distribution and is suitable for discovering motifs with incomplete information.

3.9.2 Implementing Motif Discovery

Motif is buried inside a background DNA sequence. Motif frequency and motif overrepresentation influences motif discovery significantly. Keeping the input sequence short in order to minimize the amount of uninformative background DNA from which the motif must be distinguished is advised by [17]. Our implementation of consensus sequence motif discovery uses Enumeration method. The implementation determines motif frequency by doing an exhaustive enumeration of words. Once a motif of interest is identified, the information is passed to biologists for further analysis and the topic is beyond the scope of this thesis.

3.10 Automatic File Transfer Module

RelatinalDNA stores a filtered offline copy of GenBank database. However there are thousands of records uploaded to GenBank database everyday and it is felt very important that biological scientists can access the most up to date data. To meet this need we developed an automatic FTP module that searches the GenBank database everyday at a convenient time to any new update. To make the search process convenient to the submitter and the receiver, GenBank GIU (Genbank Incremental Update) stores these updates in the following ftp site (ftp://ftp.ncbi.nih.gov/genbank/daily-nc/). GIU stores these updated files as ncMMDD.flat.gz where 'MM' represents a 2-digit value for the month and 'DD' represents a two-digit value for the year.

Our implementation uses a job that is scheduled to run everyday and the job eventually runs two scripts GenBank.bat and script.ftp. The job has multiple steps and flows as below.

Step 1. Connect to the GenBank Online Service over the Internet using the FTP program:

```
ftp ftp.ncbi.nih.gov
login: anonymous
password: pxd9974@cs.rit.edu
```

Step 2. Change directories to the directory containing the nightly update data:

```
cd genbank/daily-nc
```

Step 3. Set the FTP file transfer mode:

```
binary mode transfer
```

Step 4. Use the GET command to obtain the file containing the cumulative nightly update:

```
get ncMMDD.flat.gz
```

Where MM and DD are variables and are replaced by system month and date.

Step 5. Once the file has been completely transferred, terminate the FTP connection:

```
bye
```

The compressed downloaded files are further uncompressed and our custom parser parses it to tab delimited files for custom loader to load into our database.

4. Analysis

This thesis explored major challenges in finding, extracting, merging and synthesizing biological data and proposed a solution for integrating biological data. The performance and usability of the designed and implemented system RelationalDNA ought to be evaluated with contemporary biological data integration systems. This chapter is dedicated for a comparative analysis of ReationalDNA in regards to performance, security, data autonomy, dependability and maintainability of a unified data model. The chapter begins with different goals achieved from this research. Section 4.1 discusses how query performance is improved, how a higher level of security could be maintained, how data autonomy is enhanced and how our dependability to other factors such as network connectivity, availability of collaborative system etc are going to be reduced.

4.1 What has been accomplished?

Overall the system is capable of delivering more consistent data. Now it reduced the cost of searching and extracting data from disparate data sources by maintaining its own standardize data architecture. Besides, unavailability of collaborative system (DDBJ/EMBL/GenBank) or a temporary disconnected environment would not affect biological scientists any more.

4.1.1 Writing Custom Query

Now that we have a unified proprietary data model, we have more flexibility to retrieve meaningful information. In the contemporary data models (GenBank, DDBJ, EMBL etc), there are a predefined set of criteria to retrieve data. However if biological scientists' need is to retrieve a different piece of information, all we have to do is write a custom query to meet the new requirement. For example we can create new query to find out number of A's, T's, C's and G's present in a sequence and the accession number. The following query will retrieve this

```
SELECT gs.accession_no, sd.count_A, sd.count_C, sd.Count_G,
sd.Count_T
FROM gb_sequences gs, sequence_details sd
WHERE sd.sequence_id = gs.sequence_id
AND accession_no='NM_002356'
/
```

Fig 4.1: Custom query to retrieve number of A's, T's, C's and G's and accession_no in a sequence

As stated in chapter 3, every biological project or research is different and so is the requirement for biological scientists. Therefore more meaningful queries could be written once the need of a scientist is learnt.

Biological data extraction process is hence enhanced.

4.1.2 Query Performance Improved

Performance tuning is a primary concern in any commercial or research data centric systems. We are optimistic that in future this work will be implemented commercially. Therefore, as applicable to other systems, performance tuning is paramount and begins from the day of project's inception.

To begin with, data retrieval is faster than before. We have been using indexes to avoid full table scan. Also materialized views are used to rewrite query so that I/O operations are minimized as much as possible. Every query is first evaluated for best performance by using Explain Plan utility. The optimizer uses costing methods (cost based optimizer - CBO) to determine the most efficient way to produce the result of the query

Following example shows the result of explain plan.

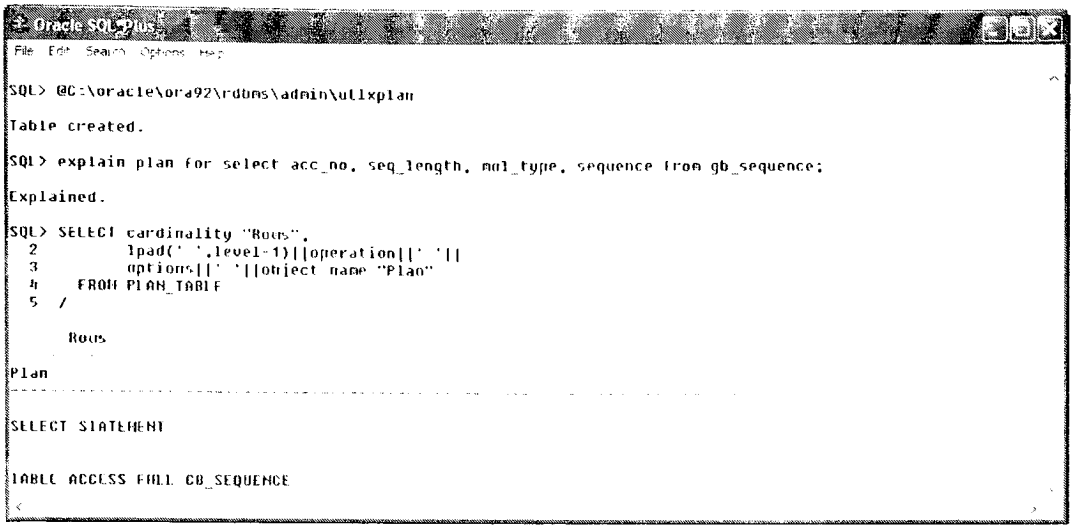


Fig 4.2: A snap-shot of explain plan query performance analysis

The explain plan results are evaluated and we are especially looking for plan that the deriving table has the best filter. The join order in each step means that the fewest number of rows are being returned to the next step. The join method is appropriate for the number of rows being returned. For example, nested loop joins through indexes may not be optimal when many rows are being returned. Views are used efficiently. There is no intentional Cartesian product. Each table is being accessed efficiently.

Secondly, update operations would not create any data anomaly. The database is normalized to third normal form and there is no data redundancy found in the database. Therefore all updates need to be performed only on one place. And hence data consistency is maintained. The scientist, who would be responsible for updating the database, now has full control over updated data.

Thirdly, bulk insert is optimized for best performance by bypassing log operation. We are using SQL Loader to insert data into the system. Since we have the option to run SQL Loader in direct or in conventional mode, we are using SQL Loader in Direct mode when

bulk amount of data is inserted into the system. SQL Loader direct mode will bypass redo log buffer operations and therefore insert operations are faster.

Finally, data removal operations did not cause any problem with data integrity. The RelationalDNA is data warehouse system and is intended to store historical data. Therefore data removal is a rare operation for this system. However occasional data removal would not cause any data integrity problem because referential integrity constraints are created on all relations.

4.1.3 Higher level of Security

RelationalDNA is our own private system and depending on our need we can implement different levels of security on it. We are sure that some of the implementations of RelationalDNA (depending on client who uses it) are going to maintain highly secured nature of data. Some of the security measures we can take are discussed below.

To begin with, in our integrated Relational Database Management System least privilege principle is maintained by using Oracle9i's data control language where only privilege needed is assigned to the user. If the biological scientist desires data to be well protected, we apply security at each row level.

Secondly, RelationalDNA can use 3-tier security by proxy authentication. The task of proxy authentication could be accomplished by use of X.509 digital certificates or distinguished names for credential proxy, support for thick JDBC, connection pooling for users via thick and thin JDBC and OCI, and integration with LDAP.

Thirdly, achieving fine-grained access control where it is ensured that each client can see only their own data and not everyone else's data. Virtual private databases are common in industry now-a-days. Every scientist who uses RelationalDNA would have their own domain of data set.

Fourthly, data in RelationalDNA can be encrypted and this is specially catered for situations where information held in database is sensitive and must be protected from any malicious users.

RelationalDNA's integrated database stores and manages information about DNA sequenced data which is considered to be highly sensitive. On top of that a biological scientist prefers to have their data well protected. In essence, if RelationalDNA is used in highly sensitive zone, it has the capability to protect data from malicious users.

4.1.4 Enhanced Data Autonomy

RelationalDNA has the ability to manage its own data. This includes making administrative decisions about the data and performing any required administrative tasks without the need for approval from another authority. If a biological research group prefers to manage their research specific data, they can do it without updating data from collaboration (DDBJ/GenBank/EMBL). It is imperative that during a particular research project a scientist group prefer to control a set of biological data. As well as during the research project a scientist group may want to modify data to accommodate their need.

Data autonomy does not necessarily imply that data is isolated.

4.1.5 Complete Data Isolation:

Data isolation involves exclusive control over DNA sequenced data by a research group or an organization that owns RelationalDNA. Once data is loaded onto RelationalDNA's integrated data repository, it is completely isolated from collaboration (DDBJ/GenBank/EMBL) database. Data isolation requires service administration too. Now service administration is in the hand of a person who administers RelationalDNA's database.

In essence data isolation ensures that data stored in RelationalDNA's data repository is completely isolated from the data sources from where data came originally.

Though we talked about complete data autonomy and data isolation, it is also important that RelationalDNA maintains most updated data from collaboration (DDBJ/GenBank/EMBL) database. An automated FTP module continually looks for updated data and updates the data repository. Data autonomy and data isolation is only an option required by biological scientists.

4.1.6 Dependability

Now scientists are not dependent on any external or internal circumstances as RelationalDNA itself is an independent system. We are discussing some factors RelationalDNA can overcome.

Network instability is a common problem in industry because it is dependent on physical media, network operating systems, routers and switches etc. For whatever reason if we don't have the network connection established, we don't have access to collaboration (DDBJ/GenBank/EMBL) database. Now that we have RelationalDNA in place, users are not dependent on external sources.

Most biological data sources are publicly available and they are available 24 hours a day and 7 days a week. If these databases are made offline for maintenance or for routine checks, though it is unlikely, users are interrupted. Having RelationalDNA in place, these problems are taken care of.

RelationalDNA is also used as backup of original data sources. Though RelationalDNA maintains data in its own data model, we could always transform data into the original data model. If there is problem with the original data source, the transformed data could be sent back to the source.

4.1.7 Maintain Unified Data Model

Ontology is one way to resolve problems arising from different data formats. Ontology is formal descriptions of the concepts and entities for a domain of interest and the relationships that hold among them. There are some algorithms that define well identified concepts and are represented through computationally tractable data structure. Ontology uses concepts rather than raw data itself. They typically contain synonyms and lexical variants for terms and therefore resolving raw terms to concepts may be achieved through straightforward string matching algorithms. Lexical variant or synonym resolution can be accomplished with a thesaurus and does not require a full fledged ontology. They can be used to resolve the semantic type for a concept. For example, the concepts "Leukemia, B-Cell, Acute" and "Leukemia" can be retrieved from the same query if the underlying ontology records that "Leukemia" is a parent concept of "Leukemia, B-Cell, Acute".

However Ontology solution is not suitable for situations where relationship between experimental descriptions and expressions values are not explicitly represented. In our RelationalDNA a proper data model has been created to represent required relations. And the data model is represented through an integrated data warehouse. Our database stores finished data. The finished data is populated directly by scientists or from other sources.

4.1.8 Maintain Partial Replica:

The human genome project generated a plethora data in disparate data sources with different data formats. It is daunting task for a scientist to search, extract and synthesize these data from all available data sources. And also for a specific scientific project a scientist does not even need these entire set of data.

Therefore RelationalDNA could be used as a partial replica of data that is needed for a specific research. A scientist can collect and store only required data in RelationalDNA. This will save tremendous amount of time and resources.

4.2 Database Compression

DNA sequence data is enormous in size and managing and maintaining DNA sequence data would be an enduring task for Information Technologists. To resolve this issue different compression techniques and algorithms are being used. To compress, an encoding algorithm is used and to decompress, a decoding algorithm is used. Lossy algorithm, which can only construct an approximation of the original data, is beyond unacceptable to compress DNA sequence data. This is because of the sensitive nature of data. The lossless algorithm, which can construct exact original data, is used to compress DNA sequence data.

One of the lossless compression algorithm Run-length encoding identifies strings of adjacent messages of equal value and replace them with a single occurrence along with a count. After converting the original data, a probabilistic coder can be used to code both the message and the count. A simple DNA string "CCCCCCCAAAAAAAACCCCCCCCCTTTTTGGGGGG" would be encoded to "7C8A9C5T 6G" after run-length compression. Efficiency of run-length encoding is dependent on pattern of sequence and is unpredictable unless experimented.

Another emerging lossless compression algorithm is 2–bit encoding. Instead storing each 4 nucleotide (ACTG) in 16-bit spaces, these can be stored in 2-bit spaces. For example, A is represented as 00, C is represented as 01, T is represented as 10, G is represented as 11. This algorithms use two properties of the DNA sequence, that is, approximate repeat and palindrome. They can compress a sequence in up to 2 bits per base but require lots of time to search the optimal approximate repeat and palindrome for compression. However, in the compression of a DNA sequence, not only the compression rate but also the compression speed is important.

4.2.1 Implementation

We implemented compression in our experimental project by using run-length encoding. Since we used JAVA to implement the entire project, we used JAVA to implement compression as well. The following is a code snippet summarizes our implementation.

```
static String DNACompression(String sequence)
	{
	sequence += "&";
	int inc = 1, i=0;
	while(sequence.charAt(i) != '&') {
		if(sequence.charAt(i) != sequence.charAt(i+1)) {
			if(inc > 1)   {
				String first = sequence.substring(0, i-inc+2) + inc;
				String nextChar = sequence.substring(i+1);
				sequence = first+nextChar;
				i=0;
				}
			inc=1;
			}
		 else {
			inc++;
			}
		i++;
		}
		return sequence.substring(0, sequence.length()-1);
	}
```

4.2.2 Analysis

We performed a comparative performance and space usage analysis with and without compression. We provided we provided many different sequences to produce the desired result.

Space usage was very unpredictable with run-length encoding. Similar length sequences with different shows different results. For example each of the following sub sequences have 47 bases. Even though they have same length bases, there compression ratio is different.

Reading 1:
Original sub-sequence:
AAAAGAAAAGGTTAGAAAGATGAGAGATGATAAAGGGTCCATTTGAG
Compressed String:
A4GA4G2T2AGA3GATGAGAGATGATA3G3TC2AT3GAG
Compression Gained = 23%

Reading 2:
Original sub-sequence:
GTTAGGTAATATGGTTTGGTATCCCTGTAGTTAAAAGTTTTTGTCTTAT
Compressed String:
GT2AG2TA2TATG2T3G2TATC3TGTAGT2A4GT5GTCT2AT
Compressed = 21%

Reading 3:
Original sub-sequence:
TTTAGAATACTGTGACTATTTCTTTAGTATTAATTTTTCCTTCTGTTTTC
Compressed String:
T3AGA2TACTGTGACTAT3CT3AGTAT2A2T5C2T2CTGT4C
Compressed = 21%

Reading 4:
Original sub-sequence:
CTCATCTAGGGAACCCCAAGAGCATCCAATAGAAGCTGTGCAATTATG

Compressed String
:CTCATCTAG3A2C4A2GAGCATC2A2TAGA2GCTGTGCA2T2ATG
Compressed = 12%

We did not observe much CPU (%Processor Time) difference in our experiment. This is because we have resource limitations and could use very large DNA sequences for experimentation. For large databases compression and decompression would degrade the system performance.

5. Conclusions

Many problems in biomedical research require access to many DNA sequenced data sources that are voluminous, heterogeneous, complex and geographically dispersed. If these data sources are successfully integrated into a new database, researchers can decipher relationships among DNA sequences that enable them to make better decisions. This thesis explored major challenges in finding, extracting, merging and synthesizing biological data and proposed an integrated data warehouse solution for DNA sequenced data integration. Further research must continue to address subsets of this problem domain in order to build more intelligent next generation data integration systems. In this chapter we discuss current status of this work, future research landscape and the lesson learned from this research.

5.1 Current Status

The entire work has been divided into three major components.

a. The back end data repository

b. The seamless data entry mechanism

c. The seamless data retrieval mechanism

5.1.1 The back end data repository

The RelationalDNA has a built in Oracle 9i data repository and this database is fully normalized to NF3. The data repository does not suffer from any anomalies (update anomalies, insert anomalies delete anomalies are fully resolved).

We consider RelationalDNA as a prototype and therefore the defined schema contains only DNA sequenced data. However the existing schema could be extended and we discuss that in future work section.

We tested the schema for joins and their performance seems to be excellent. The database triggers are working fine and therefore we have our logging mechanism working as expected.

5.1.2 The seamless data entry mechanism

The major challenge of this work was to design and implement the custom parser. The parser extracts information held in GenBank flat file and stores in tab delimited flat file. Using Oracle 9i's external table and SQL loader the file is loaded into our database. We have written one parser to extract data from GenBank format. However this parser could further be extended to incorporate other DNA sequenced data format.

5.1.3 The seamless data retrieval mechanism

An efficient data retrieval mechanism is the key for biological scientists to extract information held in DNA sequences. In other words, final goal of this research is to ensure that a biological scientist can capture and reveal information from RelationalDNA. We have incorporated multiple queries with our client tool to retrieve data. Again most of the advance data retrieval mechanisms such as indexes, materialized views are implemented. The queries are tuned time and again for better performance. However it is important to stress that different research problems are likely to need different set of data for analysis and queries are going to be created accordingly.

5.2 Future work

An essential goal of this research is to provide foundation for searching, extracting and synthesizing biological data that will allow many valuable directions for future work. This thesis and the experimental project have contributed a vital portion of the groundwork necessary for ongoing challenges of working with biological data. However further research must continue on following areas.

5.2.1 Database Security

Security is paramount for most of the commercially implemented data warehouse as the data warehouse needs to be protected from intruders. Security is especially applicable to RelationalDNA as it stores sensitive DNA sequenced data. Substantial amount of work has to be undertaken if this concept is used in human genome cloning. Even in the experimental project there are lots to implement in regards to the security.

5.2.2 Extending Schema and Data validation

The experimental project encompasses only DNA sequenced data and we could extend the database schema to include more biological data. During this research we had resource limitation. However if there is no resource (especially hard disk space) limitation then more biological data could be stored for analysis. Data validation is another area that can further be improved. The verification needs to make sure that the flat file format can be imported, and possibly to check the sequences. More data integration would make our data more reliable to the biological analysis. To verify the flat file, a program is needed to query the database, and gather the information to reconstruct the database. A regular expression tool, such as grep, can be used to find the differences, and if there is no difference then it was a successful load of the file.

5.2.3 Additional User Interface

A more robust user interface can be created for someone trying to access RelationalDNA with external links to other databases. Another user interface would be an update interface. Update interface would also have a trigger to notify a user of updates of items that fit their profile. A graphical interface to navigate related items would prove interesting to this work.

5.2.4 System Performance

Another area where this thesis could be extended to is system performance. RelationalDNA's integrated database would be queried by many scientists at the same time. Therefore the server contains the database needs high performance hardware. On top of hardware, there are so many places in the database the performance could be improved. Index organized tables, clusters, more sophisticated materialized views, proper placement of data files and redo log files, optimizing queries are some techniques we can add on to this experimental project to improve performance of our system.

5.3 Lessons learned

Through the experimental project analysis and the chapters in this dissertation we have argued that data warehousing is a better approach to search, extract and synthesize biological data. The argument made in this dissertation began with a survey of the research activity within this area of biological data.

- In chapter 1 we had gone through a literature search and work under taken in this field.

By taking the course on "Biological Data Integration Challenges" (http://www.cs.rit.edu/~pxd9974/898/pdf/ and http://www.cs.rit.edu/~pxd9974/898/ppt/) under Professor R. K. Raj, challenges associated with biological data are analyzed. Working with biological data is a complex task for both bioinformaticians and biologists and significant efforts has been made to overcome these complexities during last twenty years. Researchers need integrated access to data from multiple sources. Concept identification and resolution is a complex task because data contained in two data sources may refer to the same object and hence confusion increases. Again different data sources may contain data in different format and reconciling them to single repository is a daunting task

- In Chapter1 we have analyzed challenges associated with biological data.

- We also articulated the need of an integrated biological data system that is capable of meeting a scientist need for efficient data retrieval, data extraction and reliable data repository.

By using experimental project, we sought to offer a comprehensive approach to integrate biological data with Relational Database Management System. As first step to explore merits and demerits of Relational Database Management System, we designed our database schema in chapter 2. During the process of designing schema we have gone through steps of database normalization and denormalization.

- In chapter 2 we designed the database to fulfill the need of scientists.

We began to speculate the need of a unified data model since biological data are scattered in disparate sources and different data formats. We carefully considered how data would be extracted from disparate sources and transformed into desired format

- We have designed and implemented a custom tool to extract, transform and load data into RDBMS data repository in chapter 3.

We designed a User Interface for biological scientists to retrieve data from the system. This is a form based tool that asks for input values. We tested the tool to ensure that a biological scientist obtain consistent and meaningful information.

- In chapter 3 a client user interface is designed and created to interact with the database.

As an integrated biological data system, RelationalDNA will help a scientist to conduct activities like data retrieval, data analysis etc. However the system efficacy has to be evaluated with some existing systems for comparative analysis.

- In chapter 4 we have analyzed the RelationalDNA and listed the goals we accomplished.

This is not the only work on this ever challenging task nor is this research a complete work. This is going to be continuous effort in future.

- In chapter 5 we describe future research directions in biological data integration areas.

Appendix – A (Front end implementation code)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.SQLException;
/**
* This class creates a Swing GUI that allows the user to select
*their criteria.It then obtains a ResultSetTableModel for the
*query and uses it to display the results of the query in a
**/JTable component.
public class PrabinsDnaClient extends JFrame implements
MouseListener {
ResultSetTableModelFactory factory;
//A factory to obtain our table data
JTable table; // The table for displaying data
JLabel msgline; // For displaying messages
static JTextField driver, database, usertf,passtf;
static String qs,username, password;
JRadioButton [] radioButtons ;
public PrabinsDnaClient(){}
/**
* This constructor method creates a simple GUI and hooks up an
*event listener that updates the table when the user select a
*query.
**/
public PrabinsDnaClient(ResultSetTableModelFactory f) {
super("A Framework for Integrating DNA Sequenced Data");
// Set window title
// Window closing
addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent e) { System.exit(0); }
});
// Remember the factory object that was passed to us
this.factory = f;
// Create the Swing components we'll be using
table = new JTable(); // Displays the table
msgline = new JLabel(); // Displays messages
final int numButtons = 5;
radioButtons = new JRadioButton[numButtons];
final ButtonGroup group = new ButtonGroup();
radioButtons[0] = new JRadioButton("<html> <font
color=maroon><font size=3>Display 1 : ACGT
radioButtons[0].addMouseListener(this);
radioButtons[1] = new JRadioButton("<html> <font
color=maroon><font size=3> Display 2: Seque
radioButtons[1].addMouseListener(this);
radioButtons[2] = new JRadioButton("<html> <font
color=maroon><font size=3>Display 3: Desc
radioButtons[2].addMouseListener(this);
```

```
radioButtons[3] = new JRadioButton("<html><font
color=maroon><font size=3>Display 4: Sources
radioButtons[3].addMouseListener(this);
radioButtons[4] = new JRadioButton("<html><font
color=maroon><font size=3>Display 5: Sequenc
radioButtons[4].addMouseListener(this);
for (int i = 0; i < numButtons; i++) {
group.add(radioButtons[i]);
}
JPanel top ,row, mid ,bot;
top=new JPanel();
top.setLayout( new GridLayout(5,1) );
JLabel text = new JLabel("DNA SEQUENCED DATA",JLabel.CENTER);
text.setForeground(Color.blue);
text.setFont(new Font("Times-Roman", Font.BOLD, 20));
row=new JPanel();
row.add(new Label("Driver:") );
driver=new JTextField("oracle.jdbc.driver.OracleDriver",30);
row.add(driver);
top.add(row);
row=new JPanel();
row.add(new Label("user:") );
usertf=new JTextField("scott",30);
row.add(usertf);
top.add(row);
row=new JPanel();
row.add(new Label("password:") );
passtf=new JTextField("tiger", 30);
row.add(passtf);
top.add(row);
row=new JPanel();
row.add(new Label("Database:") );
database=new JTextField("jdbc:oracle:thin:@bishal:1521:bio",30);
row.add(database);
row.add(new Button("Connect database"));
top.add(row);
mid=new JPanel();
mid.setLayout( new GridLayout(6,1) );
mid.add(text);
mid.add(radioButtons[0] );
mid.add(radioButtons[1]);
mid.add(radioButtons[2]);
mid.add(radioButtons[3]);
mid.add(radioButtons[4]);
// Place the components within this window
Container contentPane = getContentPane();
contentPane.add(mid, BorderLayout.NORTH);
contentPane.add(new JScrollPane(table), BorderLayout.CENTER);
contentPane.add(top, BorderLayout.SOUTH);
username=usertf.getText();
password=passtf.getText();
// Now hook up the JTextField so that when the user types a query
```

```
// and hits ENTER, the query results get displayed in the JTable
for (int i = 0; i < numButtons; i++){
radioButtons[i].addActionListener(new ActionListener() {
// This method is invoked when the user hits ENTER in the field
public void actionPerformed(ActionEvent e) {
// Get the user's query and pass to displayQueryResults()
// displayQueryResults(query.getText());
if ( e.getSource().equals(radioButtons[0]))
{
// qs = "select * from gb_sequences";
qs ="select gs.sequence_id,sd.accession_no,
sd.seq_length,sd.mol_Type,"
+ "sd.count_a, sd.count_c,sd.count_t, sd.count_g from
sequence_details s
+"gb_sequences gs where sd.accession_no=gs.accession_no";
displayQueryResults(qs);
}
if ( e.getSource().equals(radioButtons[1]))
{
//qs = "select Accession_no from gb_sequences ";
qs="select gs.accession_no, v.version, v.GI, " +
"gs.sequence from gb_sequences gs, version v "+
"where gs.accession_no=v.accession_no";
displayQueryResults(qs);
}
if ( e.getSource().equals(radioButtons[2]))
{
qs= "select s.accession_no, sd.description,"+
"sd.organism from sources s, sequence_details sd where "+
" sd.accession_no=s.accession_no";
displayQueryResults(qs);
}
if ( e.getSource().equals(radioButtons[3]))
{
qs = qs= "select sd.accession_no, s.source_date, s.source "+
" from sources s, sequence_details sd where "+
" sd.accession_no=s.accession_no";
displayQueryResults(qs);
}
if ( e.getSource().equals(radioButtons[4]))
{
qs = qs= "select sequence_id from gb_sequences order by 1";
displayQueryResults(qs);
}}});}
}
/**
* This method uses the supplied SQL query string, and the
* ResultSetTableModelFactory object to create a TableModel that *
* holds the results of the database query. It passes that * *
*TableModel to the JTable component for display.
**/
public void displayQueryResults(final String q) {
```

```
// It may take a while to get the results, so give the user some
// immediate feedback that their query was accepted.
msgline.setText("Contacting database...");
// In order to allow the feedback message to be displayed, we
// don't run the query directly, but instead place it on the
//event queue to be run after all pending events and redisplays
//are done.
EventQueue.invokeLater(new Runnable() (
public void run() (
try (
// This is the crux of it all. Use the factory object
// to obtain a TableModel object for the query results
// and display that model in the JTable component.
table.setModel(factory.getResultSetTableModel(q));
// We're done, so clear the feedback message
msgline.setText(" ");
}
catch (SQLException ex) (
// If something goes wrong, clear the message line
msgline.setText(" ");
// Then display the error in a dialog box
JOptionPane.showMessageDialog(PrabinsDnaClient.this,
new String[] ( // Display a 2-line message
ex.getClass().getName() + ": ",
ex.getMessage()
});
}
//List all the methods in the mouseListener interface
public void mouseClicked(MouseEvent e)()
public void mousePressed(MouseEvent e) ()
public void mouseExited(MouseEvent e) ()
public void mouseEntered(MouseEvent e) ()
public void mouseReleased(MouseEvent e) ()
/**
* This simple main method tests the class. It expects four
*command-line arguments: the driver classname, the database URL,
*the username, and the password
**/
public static void main(String args[]) throws Exception (
// Create the factory object that holds the database connection
//using the data specified on the command line
ResultSetTableModelFactory factory = new
ResultSetTableModelFactory(args[0], args[1]);
// new ResultSetTableModelFactory(args[0], args[1], args[2],
//args[3]); Create a PrabinsDnaClient component that uses the
//factory object.
PrabinsDnaClient qf = new PrabinsDnaClient(factory);
// Set the size of the PrabinsDnaClient, then pop it up
qf.setSize(500, 600);
qf.setVisible(true);
}
}
```

```
////////////////////////////////////////////////////////

import java.sql.*;
import javax.swing.table.*;
import javax.swing.event.*;
/**
* This class takes a JDBC ResultSet object and implements the
*TableModel interface in terms of it so that a Swing JTable
*component can display the contents of the ResultSet. Note that
*it requires a scrollable JDBC 2.0 ResultSet. Also note that it
*provides read-only access to the results
**/
public class ResultSetTableModel implements TableModel {
ResultSet results; // The ResultSet to interpret
ResultSetMetaData metadata; // Additional information about the
//results
int numcols, numrows; // How many rows and columns in the table
/**
* This constructor creates a TableModel from a ResultSet. It is
*package private because it is only intended to be used by
* ResultSetTableModelFactory, which is what you should use to
*obtain a ResultSetTableModel
**/
ResultSetTableModel(ResultSet results) throws SQLException {
this.results = results; // Save the results
metadata = results.getMetaData(); // Get metadata on them
numcols = metadata.getColumnCount(); // How many columns?
results.last(); // Move to last row
numrows = results.getRow(); // How many rows?
}
/**
* Call this when done with the table model. It closes the
*ResultSet and the Statement object used to create it.
**/
public void close() {
try { results.getStatement().close(); }
catch(SQLException e) {};
}
//** Automatically close when we're garbage collected */
protected void finalize() { close(); }
// These two TableModel methods return the size of the table
public int getColumnCount() { return numcols; }
public int getRowCount() { return numrows; }
// This TableModel method returns columns names from the
//ResultSetMetaData
public String getColumnName(int column) {
try {
return metadata.getColumnLabel(column+1);
} catch (SQLException e) { return e.toString(); }
```

```
}
// This TableModel method specifies the data type for each
column.
// We could map SQL types to Java types, but for this example,
//we'll just convert all the returned data to strings.
public Class getColumnClass(int column) { return String.class; }
/**
* This is the key method of TableModel: it returns the value at
*each cell of the table. We use strings in this case. If anything
*goes wrong, we return the exception as a string, so it will be
*displayed in the table. Note that SQL row and column numbers
*start at 1, but TableModel column numbers start at 0.
**/
public Object getValueAt(int row, int column) {
try {
results.absolute(row+1); // Go to the specified row
Object o = results.getObject(column+1); // Get value of the
//column
if (o == null) return null;
else return o.toString(); // Convert it to a string
} catch (SQLException e) { return e.toString(); }
}
// Our table isn't editable
public boolean isCellEditable(int row, int column) { return
false; }
// Since its not editable, we don't need to implement these
methods
public void setValueAt(Object value, int row, int column) {}
public void addTableModelListener(TableModelListener l) {}
public void removeTableModelListener(TableModelListener l) {}
}

//////////////////////////////////////////////////////////////////

import java.sql.*;
import javax.swing.table.*;
/**
* This class encapsulates a JDBC database connection and when the
user select thir data
// it returns a ResultSetTableModel object suitable for display
* in a JTable Swing component
**/
public class ResultSetTableModelFactory {
Connection connection; // Holds the connection to the database
//** The constructor method uses the arguments to create db
//Connection */
public ResultSetTableModelFactory( String username, String
password)
throws ClassNotFoundException, SQLException
{
// Look up the JDBC driver by class name.
Class driver = Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
// Now use that driver to connect to the database
connection =
DriverManager.getConnection("jdbc:oracle:thin:@bishal:1521:bio",
username, password );
}
/**
* This method takes a SQL query, passes it to the database,
*obtains the results as a ResultSet, and returns a
*ResultSetTableModel object that holds the results in a form that
*the Swing JTable component can use.
**/
public ResultSetTableModel getResultSetTableModel(String query)
throws SQLException
{
// If we've called close(), then we can't call this method
if (connection == null)
throw new IllegalStateException("Connection already closed.");
// Create a Statement object that will be used to excecute the
//query.
Statement statement =
connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
// Run the query, creating a ResultSet
ResultSet r = statement.executeQuery(query);
// Create and return a TableModel for the ResultSet
return new ResultSetTableModel(r);
}
/**
* Call this method when done with the factory to close the DB
*connection
**/
public void close() {
try { connection.close(); } // Try to close the connection
catch (Exception e) {} // Do nothing on error. At least we tried.
connection = null;
}
//** Automatically close the connection when we're garbage
//collected */
protected void finalize() { close(); }
}
```

Appendix – B (Custom Parser implementation code)

```
/**
* This class is for parsing plain text files in the GenBank
format
* @see
* <a
href="http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html">
* GenBank Sample Record</a><br>
*
* We are using biojava library in this class.
* BioJava is a general bioinformatics toolkit. It provides a
framework
* for building a parser. The bioJava code-base is open source.
*
* Author: Prabin Dutta
*
*/
import org.biojava.bio.seq.*;
import org.biojava.bio.seq.io.*;
import java.io.*;
import org.biojava.bio.*;
import java.util.*;
import org.biojava.bio.symbol.*;
public class GBankReader {
public static void main(String[] args) throws Exception{

// Create a filewriter object to write the data

FileWriter f1= new FileWriter("file1.dat");
FileWriter f2= new FileWriter("file2.dat");

// Create an Print Writer and attach it to the file

PrintWriter out1 = new PrintWriter(f1);
PrintWriter out2 = new PrintWriter(f2);

//String variables to hold the Annotation result

String LOCUS = new String();
String ORGANISM= new String();
String VERSION= new String();
String DEFINITION= new String();
String SOURCE= new String();
String MDAT= new String();
String DIVISION= new String();
String ACCESSION= new String();
String TYPE= new String();
String SIZE= new String();
String AUTHORS= new String();
String REFERENCE= new String();
```

```
String GI= new String();
String CDS= new String();
String KEYWORDS= new String();
String ORIGIN= new String();
String CIRCULAR= new String();
String FEATURE= new String();
String Journal= new String();
BufferedReader br = null;
try {

//create a buffered reader to read the sequence file specified by
args[0]

br = new BufferedReader(new FileReader(args[0]));
}
catch (FileNotFoundException ex) {

//can't find the file specified by args[0]

ex.printStackTrace();
System.exit(-1);
}

//read the GenBank File

SequenceIterator sequences = SeqIOTools.readGenbank(br);

//iterate through the sequences

GBankReader.java 4/18/2006 6:25 PM
while(sequences.hasNext()){
try {

//Get a sequence

Sequence seq = sequences.nextSequence();
int c = 0;
int a = 0;
int g = 0;
int t = 0;
for (int pos = 1; pos <= seq.length(); ++pos) {
Symbol sym = seq.symbolAt(pos);
if (sym == DNATools.g()) {++g;}
if (sym == DNATools.a()) {++a;}
if (sym == DNATools.c()) {++c;}
if (sym == DNATools.t()) {++t;}
}

// System.out.println(seq.getName() + " total a : " + a + " total
C : " + c
// + " total g : " + g + " total t : " + t);
```

```
java.util.Iterator theIterator = seq.features();

//make a Filter for "CDS" types

FeatureFilter ff = new FeatureFilter.ByType("CDS");

//get the filtered Features

FeatureHolder fh = seq.filter(ff);

//iterate over the Features in fh

for (Iterator i = fh.features(); i.hasNext(); ) {
Feature f = (Feature)i.next();
FEATURE=f.toString();

//System.out.println(FEATURE);
//}
// Get annotation for the whole sequence

Annotation theAnnot = seq.getAnnotation();

// to get the keys

java.util.Set theKeys = theAnnot.keys();

//to get the value

java.util.Iterator iterator = theKeys.iterator();
while(iterator.hasNext()) {
String key = iterator.next().toString();
String result = theAnnot.getProperty(key).toString();

// obtain value of source field
//and store it in a String object

if (key.compareTo("SOURCE")==0){
SOURCE=result;
}

// obtain value of CDS field
//and store it in a String object

else if (key.compareTo("CIRCULAR")==0){
CIRCULAR=result;

//System.out.println("CIRCULAR= "+CIRCULAR);
}
// obtain value of ORGANISM field
//and store it in a String object

else if(key.compareTo("ORGANISM")==0){
```

```
ORGANISM =result;
}

// obtain value of LOCUS field
//and store it in a String object
else if(key.compareTo("LOCUS")==0){
LOCUS = result;
}

// obtain value of SIZE field
//and store it in a String object

else if(key.compareTo("SIZE")==0) {
SIZE= result;
}

// obtain value of TYPE field
//and store it in a String object

else if(key.compareTo("TYPE")==0) {
TYPE= result;
}

// obtain value of DIVISION field
//and store it in a String object

else if(key.compareTo("DIVISION")==0){
DIVISION =result;
}

// obtain value of MDAT field
//and store it in a String object

else if(key.compareTo("MDAT")==0) {
MDAT =result;
}

// obtain value of DEFINITION field
//and store it in a String object

else if(key.compareTo("DEFINITION")==0) {
DEFINITION=result;
}

// obtain value of AUTHORS field
//and store it in a String object

else if(key.compareTo("AUTHORS")==0){
AUTHORS=result;
}

// obtain value of REFERENCE field
```

```
//and store it in a String object

else if(key.compareTo("REFERENCE")==0){
REFERENCE= result;
}

// obtain value of ACCESSION field
//and store it in a String object

else if(key.compareTo("ACCESSION")==0){
ACCESSION= result;
}

// obtain value of GI field
//and store it in a String object

else if(key.compareTo("GI")==0) {
GI=result;
}

// obtain value of VERSION field
//and store it in a String object

else if(key.compareTo("VERSION")==0) {
VERSION=result;
}

// obtain value of KEYWORDS field
//and store it in a String object
else if(key.compareTo("KEYWORDS")==0) {
KEYWORDS=result;
} else if(key.compareTo("ORIGIN")==0) {
ORIGIN=result;

//System.out.println( ORIGIN);

}
char buf[]= new char[AUTHORS.length()];
AUTHORS.getChars(0,AUTHORS.length(),buf,0);
CharArrayReader in = new CharArrayReader(buf);
PushbackReader f= new PushbackReader(in);
int ca;
if ((ca=f.read())!='\n')
{

// System.out.println( References.getAuthors());

}
}

//Write the output in the outputfile
```

```
outl.println(seq.getName() +" "+seq.length()+" "+a +" "+ t +"
"+g+" "
GBankReader.java 4/18/2006 6:25 PM
+c+" "+TYPE+ " "+MDAT+" "+SOURCE +" "
+ORGANISM+" " +VERSION+" "+LOCUS+" "+DIVISION+" "+GI+" " +
KEYWORDS+" "+ DEFINITION+"
"+seq.seqString());//((String)System.getProp
}
catch (BioException ex) {

//not in GenBank format

ex.printStackTrace();
}catch (NoSuchElementException ex) {

//request for more sequence when there isn't any

ex.printStackTrace();}
}
}
}
```

Appendix – C (Figures)

REFERENCES:

[1] Prof R.K Raj "CCSCNE 2004 workshop: Bioinformatics Resources for Computer Science Educators" Troy, New York 2004

[2] Arek Kasprzyk "BioMart Query Network". Workshop on Database Issues in Biological Databases (DBiBD), Edinburgh, Scotland, January 2005.

[3] Bruce Blackwell and Siva Ravada. "Oracle's Technology for Bioinformatics and Future Directions" First Asia-Pacific Bioinformatics Conference, Adelaide, Australia 2004.

[4] Jonathan Barker and Janet Thornton "Software engineering challenges in bioinformatics" Proceedings of the 26th International Conference on Software Engineering (ICSE'04)

[5] Jake Yue Chen, John V. Carlis "Similar_Join: Extending DBMS with a Bio-specific operator" Proceedings of the 2003 ACM symposium on Applied Melbourne, Florida

[6] L. Krishnamurthy, J. Nadeau, G. Ozsoyoglu, M. Ozsoyoglu,2, G. Chaeffer , M. Tasan, W. XU 'Pathways Database System: An Integrated set of tools for Biological pathways, Symposium on Applied Computing Proceedings of the 2003 ACM symposium on Applied computing Melbourne, Florida

[7] Tony Travis, Peter Gray. "Database Issues in Nutritional Genomics Workshop on Biological Databases (DBiBD), Edinburgh, Scotland, January 2005

[8] Zoe Lacroix, Omar Boucelma, Mehdi Essid "The Biological Integration System" Proceedings of the fifth ACM international workshop on Web information and data management New Orleans, Louisiana, USA

[9] Jason E. Sew Hoyl Alan F. McCulloch2 John R. McDonald "BRINet: A BioResource Integration Network" ACM International Conference Proceeding Series Proceedings of the second conference on Asia-Pacific bioinformatics Volume 29 Dunedin, New Zealand Pages: 29 - 34 Year of Publication: 2004

[10] Mathew Palakal, Snehasis Mukhopadhyay, Javed Mostafa "Bioinformatics: An intelligent biological information management system" March 2002 Proceedings of the 2002 ACM symposium on Applied computing

[11] Zoe Lacroix, &Terence Critchlow (2003). "*Bioinformatics Managing Scientific*

Data". Morgan Kaufmann

[12] Lodish, Baltimore, Berk, Zipursky, Matsudaira, Darnell (1999). "*Molecular Cell Biology*". W. H. Freeman (4th Edition)

[13] NCBI(1990) – Retrieved August 12, 2006, from NCBI web site : http://www.ncbi.nlm.nih.gov/Genbank/

[14] Human Genome Project (1990) - Retrieved August 12, 2006, from NCBI site http://www.ornl.gov/sci/techresources/Human_Genome/project/timeline.shtml

[15] Arthur M. Lesk (2002). "*Introduction to Bioinformatics*". Oxford University Press

[16] Cynthia Gibas & Per Jambeck(2001). "*Developing Bioinformatics Computer Skills*" O'Reily (First Edition)

[17] Kenzie D MacIsaac and Ernest Fraenkel "*Practical Strategies for Discovering Regulatory DNA Sequence Motifs*" PLoS Computational Biology, 2006

[18] Hong P, Liu XS, Zhou Q, Lu X, Liu JS "*Bosting approach for motif modeling using ChIP-chip data*" Bioinformatics. 2005

[19] Barash, Y.;Elidan, G.;Friedman, N.; Kaplan, T. "*Modeling dependencies in protein-DNA binding sites" RECOMB 2003*