DEVOR: A DATA WAREHOUSE FOR MICROBES

By

071528 Tarun Pal
071531 Anant Chaturvedi

Under-

Dipankar Sengupta

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

juit

विद्या तत्व ज्योतिसमः

MAY-2011

Submitted in partial fulfillment for the Degree of

**BACHELOR OF TECHNOLOGY
IN
BIOINFORMATICS**

**DEPARTMENT OF
BIOTECHNOLOGY AND BIOINFORMATICS
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,
WAKNAGHAT**

TABLE OF CONTENTS

TOPICS	Page No.
CERTIFICATE	III
ACKNOWLEDGEMENT	IV
SUMMARY	V
LIST OF FIGURES	VII
LIST OF SYMBOLS AND ACRONYMS	VIII
LIST OF TABLES	IX
CHAPTER-1	
INTRODUCTION	**1 - 8**
1.1 REVIEW OF LITERATURE	2
1.2 DATA WAREHOUSE	3
1.3 MICROBES	5
1.4 CURRENT PROBLEM SCENARIO	5
1.5 HISTORY OF DATA WAREHOUSE	6
1.6 GOALS OF DATA WAREHOUSE	7
CHAPTER-2	
TOOLS AND TECHNOLOGIES	**8-20**
2.1 PENTAHO	8
2.2 OLAP: ON-LINE ANALYTICAL PROCESSING	9
2.3 MySQL	10
2.4 HTML	11
2.5 PHP	12
2.6 PERL	13

2.7 WEB SERVER 14

2.8 MICROSOFT IIS SERVER 17

CHAPTER-3

CONSTRUCTION OF DATA WAREHOUSE 21-39

3.1 CONSTRUCTION OF DEVOR 24

3.2 DESIGNING TRANSFORMATIONS USING SPOON 30

3.3 INTERFACE 35

CONCLUSION & FUTURE WORK 40

REFERENCES 41

APPENDICES

A: SCRIPTS USED IN PROJECT 42

B: MYSQL QUERIES USED 49

BRIEF BIO-DATA OF STUDENTS 54

CERTIFICATE

This is to certify that the work titled "**DATA WAREHOUSE FOR MICROBES**" submitted by "**TARUN PAL & ANANT CHATURVEDI**" in partial fulfillment for the award of degree of B. Tech of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor ....

Name of Supervisor DIPANKAR SENGUPTA

Designation ASSOCIATE LECTURER

Date 20|05|2011

ACKNOWLEDGEMENT

"To speak gratitude is courteous and pleasant, to enact gratitude is generous and noble, but to live gratitude is to touch Heaven"

As we conclude our project, we have many people to thank; for all the help, guidance and support they lent us, throughout the course of our endeavor.

First and foremost, we are highly indebted to our esteemed supervisor, ***Dipankar Sengupta****, who has guided us through thick and thin. We deem it a privilege to be a student doing research under Dipankar Sengupta who has endeared himself to his students and scholars.*

Secondly, we pay our most sincere thanks to **Dr. R.S. Chauhan***, Head of Department, Department of Biotechnology and Bioinformatics, for providing us with an opportunity and facilities to carry out the project.*

We also thank **Ms. Somlata Sharma** *(Bioinformatics Laboratory In-charge) for her assistantance and valuable contribution.*

We are indebted to all those who provided reviews and suggestions for improving the results and topics covered in our project, and extend our apologies to any one whom we have failed to recognize in our efforts.

Signature of Student

TARUN PAL (071528)

Date 20/5/11

Signature of Student

ANANT CHATURVEDI (071531)

Date 20/5/11

SUMMARY

DEVOR aims to describe the data warehousing process for efficient storage and retrieval of unique and non redundant data concerned with microbes. It stands out from other approaches by providing up-to-date integrated knowledge, platform and database independence as well as high usability and customization. Through DEVOR we present a biological data warehouse which uses different data source and locally centralized stores, integrates various microbial information such as there genome, proteome and various other useful information under a single resource.

Data incorporated from different sources are loaded and mapped from input table (here .csv files) to output table in order to complete the transformation. Right from data staging, presentation to access is a typical process in order to obtain consistent and normalized data. Once the transformation is successful then the data has to be saved at a centralized location for retrieval through any data access tools. The data (here microbial) can be used to decipher different pattern and for further annotations. It provides a common data model for all data of interest regardless of the data's source. This makes it easier to report and analyze information than it would be if multiple data models were used to retrieve information. Devor is an manual operational data warehouse and handles input for bacteria, fungi and virus from different sources. Interface of Devor is coded in PHP which is embedded in HTML itself.

In a single sentence this process can be summed up as data warehousing includes business intelligence tools, tools to extract, transform and load data into the repository, and tools to manage and retrieve metadata (data around the data). Data is stored on an internal backend server which is connected to main juit sever for proper security checking, in addition incremental backups are taken periodically.

The data warehouse has been hosted via a web server called **DEVOR** and it can be accessed at www.juit.ac.in/attachments/Devor/Index.html .

Signature of Student

TARUN PAL (071528)

Date 20/5/11

Signature of Student

ANANT CHATURVEDI (071531)

Date 20/5/11

Signature of Supervisor

Name Mr. Dipankar Sengupta

Date 20/05/2011

LIST OF FIGURES

Figure 2.1: The working of servers
Figure 2.2: A typical running IIS server interface
Figure 3.1 : Staging schema
Figure 3.2: Dimension_proteome
Figure 3.3 : Fact_table_genome
Figure 3.4 : Fact_table_proteome
Figure 3.5 :Dimension_table_genome
Figure 3.6 : Database connection in Spoon : a part of Kettle ETL
Figure 3.7 : Kettle Transformations: Integrating Genome and Information Genome in the dimension table.
Figure 3.8 Using MySQL queries in Kettle to process data
Figure 3.9:: Interface Homepage
Figure 3.10: Organism selection
Figure 3.11: Bacteria selection
Figure 3.12: Virus selection
Figure 3.13: Fungi selection
Figure 3.14 : Sample Output for "Podospora anserine" , a fungi
Figure 3.15: All the organisms' selection in the Proteome
Figure 3.16: All the proteins in the entered organism
Figure 3.17: Protein information

LIST OF SYMBOLS AND ACRONYMS

SQL- Structured Query Language

PHP-Hypertext Pre Processer

PERL- Program Execution Report Language

HTML- Hyper Text Markup Language

IIS- Internet Information Services

OLAP- OnLine Analytical Processing

PDB-Protein Data Bank

NCBI- National Centre for Biotechnology Information

LIST OF TABLES

Table 2.1 :Work Plan

CHAPTER 1
INTRODUCTION

Rapid advances in biology and biotechnology have resulted in a huge large in the amount of information for genome & proteome being deposited whether in NCBI, Uniprot/Swissprot or various other data sources. In the last decades large amounts on biological data were accumulated and the high-throughput methods in the fields of genomics and proteomics create an ever-increasing rate of high dimensional data on sequence, structure, and function of biological systems. By providing the essential methods to integrate, manage, and analyze these data, the integrative bioinformatics allows to gain new insights and a deeper understanding of complex biological systems. To fully explore an organism its data has to be first stored and storage of large volume of data at a single common consistent place is not easy.

This large chunk of data has to be integrated and stored in a unique easy retrieval format. The scientific focus is moving away from the single-data-domain and problem-oriented approach towards work crossing the borders of data domains. Bioinformatics tools help to analyze data at a large scale. Often, extensive data sets gained from biological experiments cannot be handled individually, especially in the area of genomic data Because integration of widely distributed data is prerequisite to their analysis, several approaches for data integration have been investigated: linked, indexed data connect flat file databases using the World Wide Web (WWW) like SRS or Entrez or federated database systems which integrate heterogeneous database systems by a central query interface (examples of the latter are OPM*QS and the Genome Database, GDB). Thus, the concept of data warehousing helps to overcome two major limitations of distributed database systems: inconsistency of data and time consuming or incomplete queries caused by server restrictions.

Bioinformatics applications can work on data from different domains or on a single domain respectively. The required data is available as MS Access databases, MS Excel and HTML files (phenotype data), Oracle databases and flat files (marker data, sequence data and passport data).

Devor has accepted the challenge to integrate a large volume of microbial data with minimal resources from large operational sources itself. The tools used Pentaho is open source software used in the field of data warehousing and business intelligence. The main source of the data is cleaned, transformed, catalogued and made available for use by researchers and other professionals for data mining, online analytical processing and decision support In order to facilitate for further analyses the integration of publicly available data, in-house data, and information extracted from literature has proven to be a powerful approach of integrative bioinformatics. The goal of Devor is to provide data for any kind of biological research and development.

1.1 REVIEW OF LITERATURE

Before starting the work a review of literature was done and it was found that no consistent data warehouse has been there for microbes till now. Infact the Data warehouse for biological data were also very few in number.

- **Atlas** - A data warehouse for integrative bioinformatics was based on relational data models developed for each of the source data types. Data stored within these relational models are managed through Structured Query Language (SQL) calls that are implemented in a set of Application Programming Interfaces (APIs). Atlas achieves integration of diverse data sets at two levels. First, Atlas stores data of similar types using common data models, enforcing the relationships between data types. Second, integration is achieved through a combination of APIs, ontology, and tools. The Atlas biological data warehouse serves as data infrastructure for bioinformatics research and development. It forms the backbone of the research activities in there laboratory and facilitates the integration of disparate, heterogeneous biological sources of data enabling new scientific inferences.

- **DWARF** – a data warehouse system for analyzing protein families. DWARF integrates data on sequence, structure, and functional annotation for protein fold families. The underlying relational data model consists of

Devor has accepted the challenge to integrate a large volume of microbial data with minimal resources from large operational sources itself. The tools used Pentaho is open source software used in the field of data warehousing and business intelligence. The main source of the data is cleaned, transformed, catalogued and made available for use by researchers and other professionals for data mining, online analytical processing and decision support In order to facilitate for further analyses the integration of publicly available data, in-house data, and information extracted from literature has proven to be a powerful approach of integrative bioinformatics. The goal of Devor is to provide data for any kind of biological research and development.

1.1 REVIEW OF LITERATURE

Before starting the work a review of literature was done and it was found that no consistent data warehouse has been there for microbes till now. Infact the Data warehouse for biological data were also very few in number.

- **Atlas** - A data warehouse for integrative bioinformatics was based on relational data models developed for each of the source data types. Data stored within these relational models are managed through Structured Query Language (SQL) calls that are implemented in a set of Application Programming Interfaces (APIs). Atlas achieves integration of diverse data sets at two levels. First, Atlas stores data of similar types using common data models, enforcing the relationships between data types. Second, integration is achieved through a combination of APIs, ontology, and tools. The Atlas biological data warehouse serves as data infrastructure for bioinformatics research and development. It forms the backbone of the research activities in there laboratory and facilitates the integration of disparate, heterogeneous biological sources of data enabling new scientific inferences.

- **DWARF** – a data warehouse system for analyzing protein families. DWARF integrates data on sequence, structure, and functional annotation for protein fold families. The underlying relational data model consists of three major sections representing entities related to the protein (biochemical function, source organism, classification to homologous families and superfamilies), the protein sequence (position-specific annotation, mutant information), and the protein structure (secondary structure information, superimposed tertiary structure). Tools for extracting, transforming and loading data from public available resources (ExPDB, GenBank, DSSP) are provided to populate the database. The data can be accessed by an interface for searching and browsing, and by analysis tools that operate on annotation, sequence, or structure

- . **Ligand depot-** Ligand Depot is an integrated data resource for finding information about small molecules bound to proteins and nucleic acids. The initial release (version 1.0, November,2003) focuses on providing chemical and structural information for small molecules found as part of the structures deposited in the Protein Data Bank. Ligand Depot accepts keyword-based queries and also provides a graphical interface for performing chemical substructure searches. A wide variety of web resources that contain information on small molecules may also be accessed through Ligand Depot.
- **LCB Data Warehouse-**The Linnaeus Centre for Bioinformatics Data Warehouse (LCB-DWH) is a web-based infrastructure for reliable and secure microarray gene expression data management and analysis that provides an online service for the scientific community. The LCB-DWH is an effort towards a complete system for storage (using the BASE system), analysis and publication of microarray data. Important features of the system include: access to established methods within R/ Bioconductor for data analysis, built-in connection to the Gene Ontology database and a scripting facility for automatic recording and re-play of all the steps of the analysis

1.2 DATA WAREHOUSE

A data warehouse is an isolated database which is used across an enterprise to combine data from different data stores and serve all business task-supporting systems with a unified view of business data. A data warehouse maintains its functions in three layers: staging, integration, and access.

They were developed to meet a growing demand for management information and analysis that could not be met by operational systems. Components of a Data Warehouse include Operational Source Systems, Data Staging Area, Data Presentation Area, and Data Access Tools. Data warehouse is characterized by a strict separation of operational and decisions-making data and systems. It is a place where data is stored for archival, analysis and security purposes. Usually a data warehouse is either a single computer or many computers (servers) tied together to create one giant computer system.

Effective data warehouse had to be integrated, subject oriented, non-volatile, and time variant in nature A data warehouse is sometimes said to be a major role player in a decision support system (DSS). DSS is a technique used by organizations to come up with facts, trends or relationships that can help them make effective decisions or create effective strategies to accomplish their organizational goals.

Data integration allows us to assemble targeted data reagents for bioinformatics analyses, and to discover scientific relationships between data. Integrating these disparate sources of data enables researchers to discover new associations between the data, or validate existing hypotheses. Data can consist of raw or formatted form. Data is most valuable of all these and good data should fulfill at least these:-

(1) Data has to be accessible,

(2) Data has to be current,

(3) Data has to be flexible, and

(4) Data has to be understandable

Some of the benefits that a data warehouse provides are as follows:

- A data warehouse provides a common data model for all data of interest regardless of the data's source. This makes it easier to report and analyze information than it would be if multiple data models were used to retrieve information such as sales invoices, order receipts, general ledger charges, etc.

- Prior to loading data into the data warehouse, inconsistencies are identified and resolved. This greatly simplifies reporting and analysis.
- Information in the data warehouse is under the control of data warehouse users so that, even if the source system data are purged over time, the information in the warehouse can be stored safely for extended periods of time.
- Because they are separate from operational systems, data warehouses provide retrieval of data without slowing down operational systems.
- Data warehouses can work in conjunction with and, hence, enhance the value of operational business applications, notably customer relationship management (CRM) systems.
- Data warehouses facilitate decision support system applications such as trend reports (e.g., the items with the most sales in a particular area within the last two years), exception reports, and reports that show actual performance versus goals.
- Data warehouses can record historical information for data source tables that are not set up to save an update history.

1.3 MICROBES

A microorganism or microbe is an organism that is unicellular or lives in a colony of cellular organisms. Microorganisms are very diverse; they include bacteria, fungi, archaea, and protists; microscopic plants (green algae); and animals such as plankton and the planarian. Some microbiologists also include viruses, but others consider these as non-living.[1][2] Most microorganisms are unicellular (single-celled), but this is not universal, since some multicellular organisms are microscopic, while some unicellular protists and bacteria, like *Thiomargarita namibiensis*, are macroscopic and visible to the naked eye. Microbes are also exploited by people in biotechnology, both in traditional food and beverage preparation, and in modern technologies based on genetic engineering. However, pathogenic microbes are harmful, since they invade and grow within other organisms, causing diseases that kill people, other animals and plants. The study of microorganisms is called microbiology, a subject that began with Anton van Leeuwenhoek's discovery of microorganisms in 1675, using microscope of his own design.

1.4 CURRENT PROBLEM SCENARIO

Problem begins when large bulk of data is to be efficiently stored in a structured manner for easy and quick retrieval. The growth in biological data exceeded Moore's Law, the well-known observation that the number of transistors on a chip doubles every 18 months was stated by George Lake in 2001.

Rapid production of biological data means that outdated data management systems may be patched or adapted to deal with unforeseen quantities of data, since data migration to a new schema in a new DBMS is an arduous process that may be deemed too time-consuming or expensive. Up till now no unique single working integrative platform has been introduced in form of reliable Data Warehouse in bioinformatics although some attempts have been made through Atlas and other data warehouse to bring them at one level. Through this data warehouse we have tried to place a unique and efficient data which is free online and is in direct usable mode through the current web address: www.juit.ac.in/attachments/Devor/Index.html .

1.5 HISTORY OF DATA WAREHOUSE

During late 1980s and 1990s concept of data warehouse was raised. In 1988, IBM researchers Barry Devlin and Paul Murphy coined the term "information warehouse". In 1991, W.H. "Bill" Inmon made data warehouses practical when he published a how-to guide, Building the Data Warehouse (John Wiley & Sons).

Chronology in development of Data Warehouse:

1)1951: The Univac uses magnetic tape as well as punched cards for data storage.

2)1956: IBM introduces first magnetic hard disk drive in its Model 305

RAMAC.

3)1961: Charles Bachman at GE develops the first database management system, IDS.

4)1968: IBM offers the IMS hierarchical database for System/360 mainframes.

5)1969: Edgar F. "Ted" Codd invents the relational database.

6)1973: Cullinane, led by John J. Cullinane, ships IDMS, a network-model database for IBM mainframes.

7)1976: Honeywell ships Multics Relational Data Store, the first commercial relational database.

8) 1979: Oracle introduces the first commercial SQL relational database management system.

9)1983: IBM introduces DB2.

10)1985: The first business intelligence system is designed for Procter & Gamble.

11)1991: W.H. "Bill" Inmon publishes Building the Data Warehouse.

12)1995 — The Data Warehousing Institute, a for-profit organization that promotes data warehousing, is founded.

13)1995 — Daniel Linstedt adds SEI/CMMI and Six Sigma to the *Data Vault Methodology* to manage projects in data warehousing.

14)2000 — Daniel Linstedt releases the *Data Vault*, enabling real time auditable Data Warehouses warehouse.

1.6 GOALS OF DATA WAREHOUSE

1) To maintain clean, non redundant, transformed, cataloged data in order for easy access.

2) Collect Data-Scrub, Integrate & Make it accessible, either for mining or improved decision making from it.

3) Start Managing Knowledge in a much faster and easier manner for third person.

4) Complete platform for Proteome and Genome of all microbes available at free of cost and in one unique format.

CHAPTER 2

DEVOR: TOOLS AND TECHNOLOGIES USED

2.1 PENTAHO

It is an open source Business Intelligence software with integrated reporting, dashboard, data mining, workflow,Data Warehousing and ETLcapabilities. Its headquarter is in Orlando, USA. It offers several products such as Pentaho Data Integration, Pentaho Analysis Services, Pentaho Reporting, Pentaho Data Mining, Pentaho DashBoard, Pentaho for Apache Hadoop.

- Pentaho Data Integration- Data Integration in pentaho is done by kettle. It consists of a core data integration engine, and GUI applications that allow the user to define data integration jobs and transformation.

- Pentaho Analysis Services -Mondrian OLAP server is an open source OLAP (online analytical processing) server, written in Java.It supports the MDX (multidimensional expressions) query language and the XML for Analysis and olap4j interface specifications. It reads from SQL and other data sources and aggregates data in a memory cache.

- Pentaho Reporting- It consists of a core reporting engine, capable of programmatic generating reports based on an XML definition file. Many tools have been developed surrounding the reporting engine, including GUI designers and ad-hoc wizards that guide the user through a step-by-step process of creating a report, using solely graphical tools without the need to write any code.

- Pentaho Data Mining- it uses Weka as comprehensive set of tool for machine learning and data mining. It performs classification, regression, association rules, and clustering algorithms which can be used to analyze and understand the business better and to improve future performance through predictive analytics.

- Pentaho DashBoard- it is an integrated platform to provide insight into your data, where you can view all kind of Interactive reports, charts and cubes created using Pentaho tools such as Pentaho Report Designer. It is a dashboard-style interface which provides a centralized view over your Business Data Movements, letting you monitor them and make decisions.

2.2 OLAP: ON-LINE ANALYTICAL PROCESSING

On-Line Analytical Processing (OLAP) is a category of software technology that enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the user.

OLAP functionality is characterized by dynamic multi-dimensional analysis of consolidated enterprise data supporting end user analytical and navigational activities including:

- calculations and modeling applied across dimensions, through hierarchies and/or across members
- trend analysis over sequential time periods
- slicing_subsets for on-screen viewing
- drill-down to deeper levels of consolidation
- reach-through to underlying detail data
- rotation to new dimensional comparisons in the viewing area

OLAP is implemented in a multi-user client/server mode and offers consistently rapid response to queries, regardless of database size and complexity. OLAP helps the user synthesize enterprise information through comparative, personalized viewing, as well as through analysis of historical and projected data in various "what-if" data model scenarios. This is achieved through use of an OLAP Server.

2.3 MySQL

It is the world's most popular open source database and is a relational database management system(RDBMS). It is named after developer Michael Widenius' daughter, My. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. The SQL phrase stands for Structured Query Language. Many Free-software- for projects use a full-featured database management system MySQL. Written in C and C++. It includes following features

- A broad subset of ANSI SQL 99, as well as extensions
- Cross-platform support
- Stored procedures
- Triggers
- Cursors
- Updatable Views
- True Varchar support
- Information schema
- Strict mode
- X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using Oracle's InnoDB engine
- Independent storage engines (MyISAM for read speed, InnoDB for transactions and referential integrity, MySQL Archive for storing historical data in little space)
- SSL support
- Query caching
- Sub-SELECTs (i.e. nested SELECTs)
- Replication support (i.e. Master-Master Replication & Master-Slave Replication) with one master per slave, many slaves per master, no automatic support for multiple masters per slave.

- Full-text indexing and searching using MyISAM engine
- Embedded database library
- Partial Unicode support (UTF-8 and UCS-2 encoded strings are limited to the BMP)
- Partititoned tables with pruning of partitions in optimizer
- Shared-nothing clustering through MySQL Cluster
- Hot backup (via mysqlhotcopy) under certain conditions

2.4 HTML

Hypertext Markup Language (HTML) enables you to mark up text so that it can function as hypertext on the Web.Invented by Tim Berners-Lee, who was then working as a computer and networking specialist at a Swiss research institute. HTML is written in the form of HTML elements consisting of *tags*, enclosed in angle brackets (like <>), within the web page content. The first tag in a pair is the *start tag*; the second tag is the *end tag*. In between these tags web designers can add text, tables, images, etc. It can embed scripts in languages such as JavaScript which affect the behavior of HTML webpages. HTML allows images and objects to be embedded and can be used to createinteractive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items.

Data types: - HTML defines several datatypes for element content, such as script data and stylesheet data, and a plethora of types for attribute values, including IDs, names, URIs, numbers, units of length, languages, media descriptors, colors, character encodings, dates and times, and so on. HTML versions include:

1) HTML 1.0- HTML 1.0 was the original specification Mosaic 1.0 used, and it supported few elements.

2) HTML 2.0- HTML 2.0 was a huge improvement over HTML 1.0. Background colors and images could be set. Forms became available with a limited set of fields.

3)**HTML 3.2-** HTML 3.2 expanded the number of attributes that enabled designers to customize the look of a page HTML 3.2 didn't include support for frames, but the browser makers implemented them anyway.

4) **HTML 4.0-** HTML 4.0 now clearly deprecates any uses of HTML that relate to forcing a browser to format an element a certain way. All formatting has been moved into the style sheets.

5) **HTML 4.01-**HTML 4.01 is a minor revision of the HTML 4.0 standard. In addition to fixing errors identified since the inception of 4.0, HTML 4.01 also provides the basis for meanings of XHTML elements and attributes, reducing the size of the XHTML 1.0 specification.

6) **XHTML 1.0-**Extensible HyperText Markup Language (XHTML) is the first specification for the HTML and XML cross-breed. XHTML was created to be the next generation of markup languages, infusing the standard of HTML with the extensibility of XML.

2.5 PHP

Personal home page/ PHP: Hypertext Preprocessor is a scripting language to produce dynamic web pages for web development . PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. It also has evolved to include a command-line interface capability and can be used in standalone graphical applications. It was eight years ago, when Rasmus Lerdorf first started developing PHP/FI. It is free software released under the PHP License; it is incompatible with the GNU General Public License (GPL) due to restrictions on the usage of the term *PHP*. PHP is commonly used as the *P* in this bundle alongside Linux, Apache andMySQL, although the *P* may also refer to Python or Perl or some combination of the three. WAMP packages (Windows/ Apache/ MySQL / PHP) and MAMP packages (Macintosh / Apache / MySQL / PHP) are also available. As of April 2007, over 20 million Internet domains had web services hosted on servers with PHP installed and mod_php was recorded as the most popular Apache HTTP Servermodule. PHP is used as the server-side programming language on 75% of all web servers. Major versions of PHP include:

1)PHP/FI- One of the basic features was a Perl-like language for handling form submissions, but it lacked many common useful language features, such as for loops.

2) **PHP/FI 2-**A rewrite came with PHP/FI 2 in 1997, but at that time the development was almost solely handled by Rasmus. One of the most interesting aspects included the way while loops were implemented. At the end of the loop, the file pointer sought back to the saved position, and the whole loop was reread and re-executed.

3) **PHP 3-**Zeev and Andi decided to completely rewrite the scripting language. They then teamed up with Rasmus to release PHP 3. PHP 3 sparked the beginning of PHP's real breakthrough, and was the first version to have an installed base of more than one million domains

4) **PHP 4-**In late 1998, Zeev and Andi looked back at their work in PHP 3 and felt they could have written the scripting language even better, so they started yet another rewrite. PHP 4 was officially released on May 22, 2002, and today its installed base has surpassed 15 million domains.

5) **PHP 5-**Soon after, the demand for more common object-oriented features increased immensely, and Andi came up with the idea of rewriting the objected-oriented part.

2.6 PERL

Perl is a general-purpose, interpreted, dynamic programming language. The language provides powerful text processing facilities without the arbitrary data length limits of many contemporary UNIX tools, facilitating easy manipulation of text. Born from a combination of C & shell scripting for system administration. Perl was originally named "Pearl", after the Parable of the Pearl from the Matthew Larry Wall's background in linguistics led to Perl borrowing ideas from natural language. It is oldest scripting language and there is no separate compilation step needed. Perl gained widespread popularity in the late 1990s as a CGI scripting language, in part due to its parsing abilities. The language provides powerful text processing facilities without the arbitrary data length limits of many contemporary Unix tools, facilitating easy manipulation of text. The Perl Foundation owns an alternative symbol, an onion, which it licenses to its subsidiaries, Perl Mongers, Perl Monks, Perl.org, and others. The symbol is a visual pun on pearl onion.

Perl's strong points:

1) **Ease of Programming:** Perl is simpler and easy to adopt.Perl has certain features that simplify several common bioinformatics tasks. It can deal with information in ASCII text files or flat files, which are exactly the kinds of files in which much important biological data appears, in the GenBank and PDB databases, among others. Any one having knowledge of C (basic programming language) can work on perl too. In Bioinformatics though perl is used atmost but other languages can also be used such as Java and C depending on problem and skills of programmer. [**Beginning Perl for Bioinformatics** James Tisdall].

2) **Rapid Prototyping:** The speed with which a programmer can write a typical Perl program is referred to as rapid prototyping. This has been important to its success in research. In a research environment there are frequent needs for programs that do something new, that are needed only once or occasionally, or that need to be frequently modified.

3) **Portability, Speed, and Program Maintenance:** *Portability* means how many types of computer systems the language can run on. Perl has no problems there, as it's available for virtually all modern computers found in biology labs. *Speed* means the speed with which the program runs. Here Perl is pretty good but not the best. For speed of execution, the usual language of choice is C. *Program maintenance* is the general activity of keeping everything working: such IT-SC 20 activities as adding features to a program, extending it to handle more types of input, porting it to run on other computer systems, fixing bugs, and so forth.

2.7 WEB SERVER

A Web server is a program that, using the client/server model and the World Wide Web's Hypertext Transfer Protocol (HTTP), serves the files that form Web pages to Web users (whose computers contain HTTP clients that forward their requests). Every computer on the Internet that contains a Web site must have a Web server program. Two leading Web servers are Apache , the most widely-installed Web server, and Microsoft's Internet Information Server (IIS). Other Web servers include Novell's Web Server for users of its NetWare operating system and IBM's family of Lotus Domino servers, primarily for IBM'sOS/390 and AS/400 customers.

Web servers often come as part of a larger package of Internet- and intranet-related programs for serving e-mail, downloading requests for File Transfer Protocol (FTP) files, and building and publishing Web pages. Considerations in choosing a Web server include how well it works with the operating system and other servers, its ability to handle server-side programming, security characteristics, and publishing, search engine, and site building tools that may come with it. It's the combination of computer and the program installed on it. Web server interacts with the client through a web browser. It delivers the web pages to the client and to an application by using the web browser and the HTTP protocols respectively. We can also define the web server as the package of large number of programs installed on a computer connected to Internet or intranet for downloading the requested files using File Transfer Protocol, serving e-mail and building and publishing web pages. A web server works on a client server model. A computer connected to the Internet or intranet must have a server program. While talking about Java language then a web server is a server that is used to support the web component like the Servlet and JSP. Note that the web server does not support to EJB (business logic component) component.

The most common use of Web servers is to host Web sites but there are other uses like data storage or for running applications The primary function of a web server is to deliver web pages on the request to clients. This means delivery of HTML documents and any additional content that may be included by a document, such as images, style sheets and JavaScripts.A client, commonly a web browser or web crawler, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so. The resource is typically a real file on the server's secondary memory, but this is not necessarily the case and depends on how the web server is implemented.

While the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting web forms, including uploading of files.

History of web servers

In 1989 Tim Berners-Lee proposed to his employer CERN (European Organization for Nuclear Research) a new project, which had the goal of easing the exchange of information between scientists by using a hypertext system. As a result of the implementation of this project, in 1990 Berners-Lee wrote two programs:

- a browser called WorldWideWeb;
- the world's first web server, later known as CERN httpd, which ran on NeXTSTEP.

Between 1991 and 1994 the simplicity and effectiveness of early technologies used to surf and exchange data through the World Wide Web helped to port them to many different operating systems and spread their use among lots of different social groups of people, first in scientific organizations, then in universities and finally in industry.

In 1994 Tim Berners-Lee decided to constitute the World Wide Web Consortium (W3C) to regulate the further development of the many technologies involved (HTTP, HTML, etc.) through a standardization process.

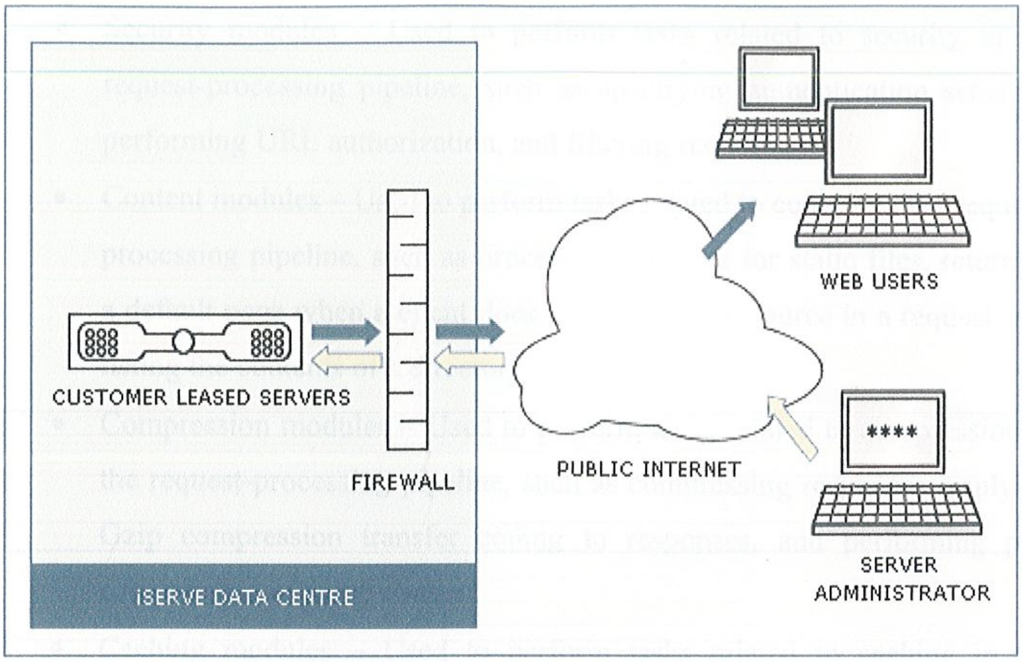

Figure 2.1: The working of servers

2.8 MICROSOFT IIS SERVER:-

Internet Information Services (IIS) – formerly called **Internet Information Server** – is a web server application and set of feature extension modules created by Microsoft for use with Microsoft Windows. It is the most used web server after Apache HTTP Server: As of January 2011, it served 21.00% of all websites on the Internet and 16.22% of the one million busiest websites on the Internet. It is an integral part of Windows Server family of products, as well as all editions of Windows Vista and Windows 7, although some features are not supported on client versions of Windows. IIS is not turned on by default when Windows is installed.

The architecture of IIS is modular. Modules, also called extensions, can be added or removed individually so that only modules required for specific functionality have to be installed. IIS includes native modules as part of the full installation. These modules are individual features that the server uses to process requests and include the following:

- HTTP modules – Used to perform tasks specific to HTTP in the request-processing pipeline, such as responding to information and inquiries sent in client headers, returning HTTP errors, and redirecting requests.

- Security modules – Used to perform tasks related to security in the request-processing pipeline, such as specifying authentication schemes, performing URL authorization, and filtering requests.
- Content modules – Used to perform tasks related to content in the request-processing pipeline, such as processing requests for static files, returning a default page when a client does not specify a resource in a request, and listing the contents of a directory.
- Compression modules – Used to perform tasks related to compression in the request-processing pipeline, such as compressing responses, applying Gzip compression transfer coding to responses, and performing pre-compression of static content.
- Caching modules – Used to perform tasks related to caching in the request-processing pipeline, such as storing processed information in memory on the server and using cached content in subsequent requests for the same resource.
- Logging and Diagnostics modules – Used to perform tasks related to logging and diagnostics in the request-processing pipeline, such as passing information and processing status to HTTP.sys for logging, reporting events, and tracking requests currently executing in worker processes.

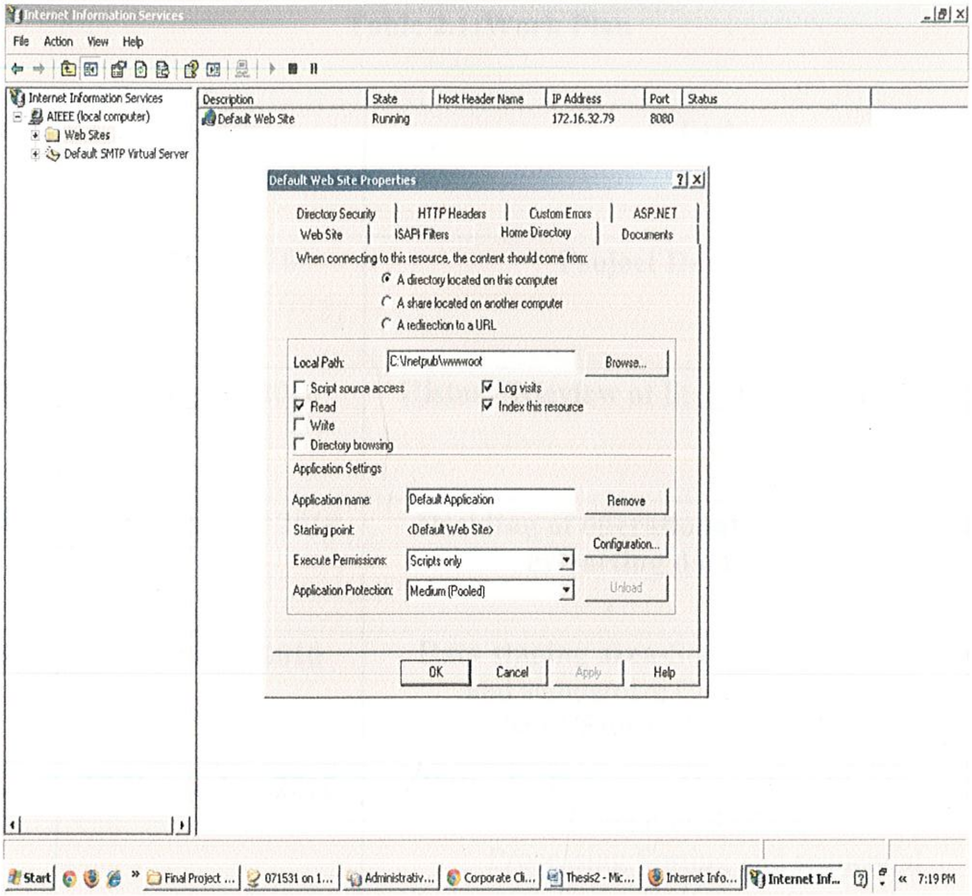

Figure 2.2: A typical running IIS server interface.

Table 2.1:Work Plan

SR	Month	Task
1	18 July 2010	**Project Devor Alloted**
2	18 August 2010	**History/Rieview of literature/Goals Of Devor**
3	18 September 2010	**Deciding of operational source system and extracting data from them**
4	18 October 2010	**Data staging area-**Services:Clean, combine, and standardize Conform dimensions NO USER QUERYSERVICES
5	18 November 2010	**Data staging area-** Data Store:Flat files and relational tables Processing:Sorting and sequential processing
6	22 January 2011	**Data Presentation Area-** Data Mart DIMENSIONAL Atomic and summary data Based on a single process
7	22 February 2011	**Data Presentation Area** -Conformed facts & dimensions
8	22 March 2011	**Data Acess tools-** Query Tools
9	22 April 2011	**Uploading online application of devor on juit Web server**
10	10 May 2011	**Thesis Writing and Compilation**

CHAPTER 3

CONSTRUCTION OF DATA WAREHOUSE (DEVOR)

Construction of Devor involves various and tedious steps in order to manage large volumes of data. The construction of data warehouses, which involves data cleaning and data integration, can be viewed as an important preprocessing step for data mining. Moreover, data warehouses provide on-line analytical processing (OLAP) tools for the interactive analysis of multidimensional data of varied granularities, which facilitates effective data mining. Furthermore, many other data mining functions such as classification, prediction, association, and clustering, can be integrated with OLAP operations to enhance interactive mining of knowledge at multiple levels of abstraction. Hence, data warehouse has become an increasingly important platform for data analysis and online analytical processing and will provide an effective platform for data mining.

The construction of a data warehouse requires data integration, data cleaning, and data consolidation. The utilization of a data warehouse often necessitates a collection of decision support technologies. This allows \knowledge workers" (e.g., managers, analysts, and executives) to use the warehouse to quickly and conveniently obtain an overview of the data, and to make sound decisions based on information in the warehouse. Some authors use the term \data warehousing" to refer only to the process of data warehouse construction, while the term warehouse DBMS is used to refer to the management and utilization of data warehouses.

Data warehousing is also very useful from the point of view of heterogeneous database integration. Many organizations typically collect diverse kinds of data and maintain large databases from multiple, heterogeneous, autonomous, and distributed information sources. To integrate such data, and provide easy and efficient access to it is highly desirable, yet challenging. Much effort has been spent in the database industry and research community towards achieving this goal.

In this step, we'll populate a data warehouse with data from the OLTP system. This phase of the process is known as ETL, which stands for Extract, Transform, Load. This is exactly what needs to be done. Extract the data needed for the fact and dimension tables from all different data sources, transform it to fit our needs and load it into the data warehouse so it can be queried.

Some important terms before we go with construction of Devor:

- Metadata- It is referred as "data about data". Metadata is all the information in the data warehouse environment that is not the actual data itself. Metadata is a loose term for any form of auxiliary data that is maintained by an application. Metadata is also kept by the aggregate navigator and by front-end query tools. The data warehouse team should carefully document all forms of metadata. Ideally, front-end tools should provide for tools for metadata administration. Most of the extraction steps should be handled on the legacy system. This will allow for the biggest reduction in data volumes. is structured data which describes the characteristics of a resource. It shares many similar characteristics to the cataloguing that takes place in libraries, museums and archives. The term "meta" derives from the Greek word denoting a nature of a higher order or more fundamental kind. A metadata record consists of a number of pre-defined elements representing specific attributes of a resource, and each element can have one or more values.

  Each metadata schema will usually have the following characteristics:

  - a limited number of elements
  - the name of each element
  - the meaning of each element

- Data mart- A data mart (DM) is the access layer of the data warehouse (DW) environment that is used to get data out to the users. The DM is a subset of the DW, usually oriented to a specific business line or team. A data mart is a data repository that may or may not derive from a data warehouse and that emphasizes ease of access and usability for a particular designed purpose. There can be multiple data marts inside a single corporation; each one relevant to one or more business units for which it was designed. DMs may or may not be dependent or related to other data marts in a single corporation. If the data marts are designed using conformed facts and dimensions, then they will be related. In some deployments, each department or business unit is considered the *owner* of its data mart including all the *hardware*, *software* and *data*. A database, or collection of databases, designed to help managers make strategic decisions about their business. Whereas a data warehouse combines databases across an entire enterprise, data marts are usually smaller and focus on a particular subject or department. Some data marts, called *dependent data marts,* are subsets of larger data warehouses.

- Data Normalization- Data Normalization means to bring down data into same level so that some conclusion can be formed from it. In the design of a relational database management system (RDBMS), the process of organizing data to minimize redundancy is called normalization. The goal of database normalization is to decompose relations with anomalies in order to produce smaller, well-structured relations. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships. Edgar F. Codd, the inventor of the relational model, introduced the concept of normalization and what we now know as the First Normal Form (1NF), Second Normal Form (2NF) and Third Normal Form (3NF) in 1971,[2] and Codd and Raymond F. Boyce defined the Boyce-Codd Normal Form (BCNF) in 1974. The higher the normal form applicable to a table, the less vulnerable it is to inconsistencies and anomalies. Each table has a "highest normal form" (HNF) by definition, a table always meets the requirements of its HNF and of all normal forms lower than its HNF; also by definition, a table fails to meet the requirements of any normal form higher than its HNF.

- Data cleaning- Data cleaning is getting data into consistent form. Data cleansing or data scrubbing is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database. Used mainly in databases, the term refers to identifying incomplete, incorrect, inaccurate, irrelevant etc. parts of the data and then replacing, modifying or deleting this *dirty data*.After cleansing, a data set will be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by different data dictionary definitions of similar entities in different stores, may have been caused by user entry errors, or may have been corrupted in transmission or storage. The actual process of data cleansing may involve removing typographical errors or validating and correcting values against a known list of entities. A two step process including *DETECTION* and then *CORRECTION* of errors in a data set.

3.1 CONSTRUCTION OF DEVOR

When we move the data into data warehouse normally, it will have two kinds of schemas

1) Staging schema - Data from all the operational data sources here Uniprot (all microbial protein data) and NCBI (all microbial genome data) has been used as original source data. All the data in the staging schema has been dumped from these two data sources.

2) Working schema-From staging to walking schema we will clean up the data (we can share walking schema). Data is not lost by cleaning or filtering but it is made available in some other usable form.

We can't use the data as such in raw form.

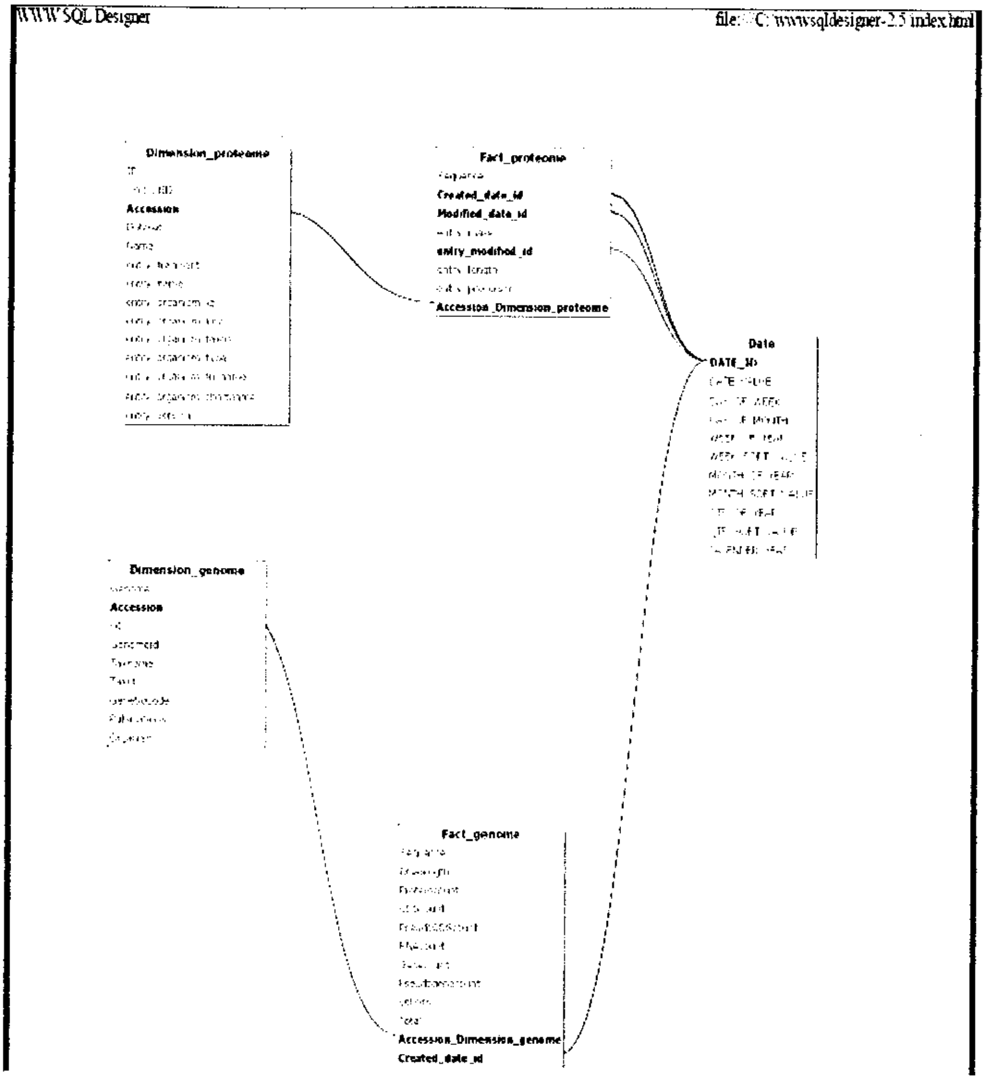

Figure- 3.1 – Working schema

Devor Working Schema:Dimension & Fact Tables

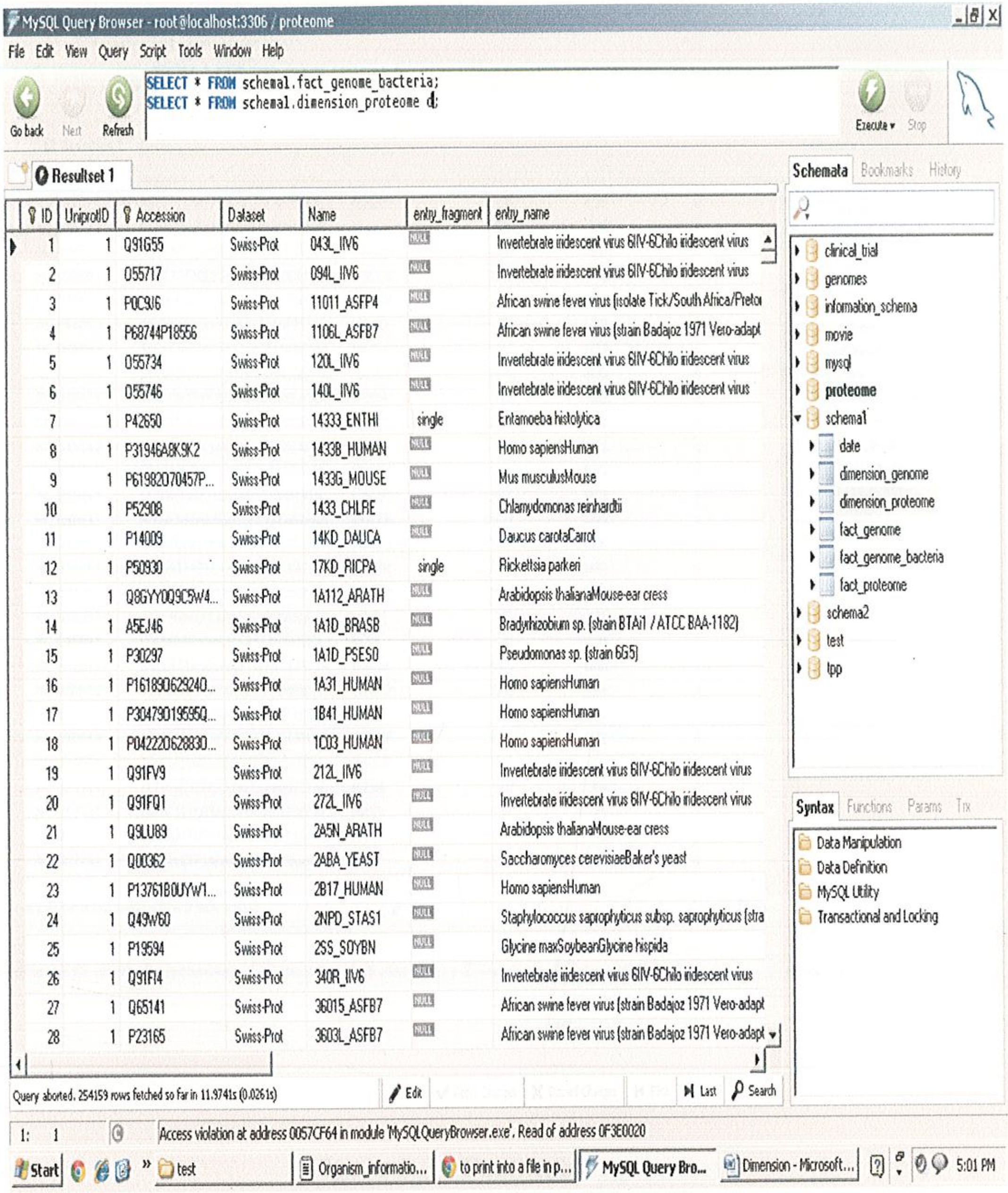

Figure 3.2- Dimension_proteome

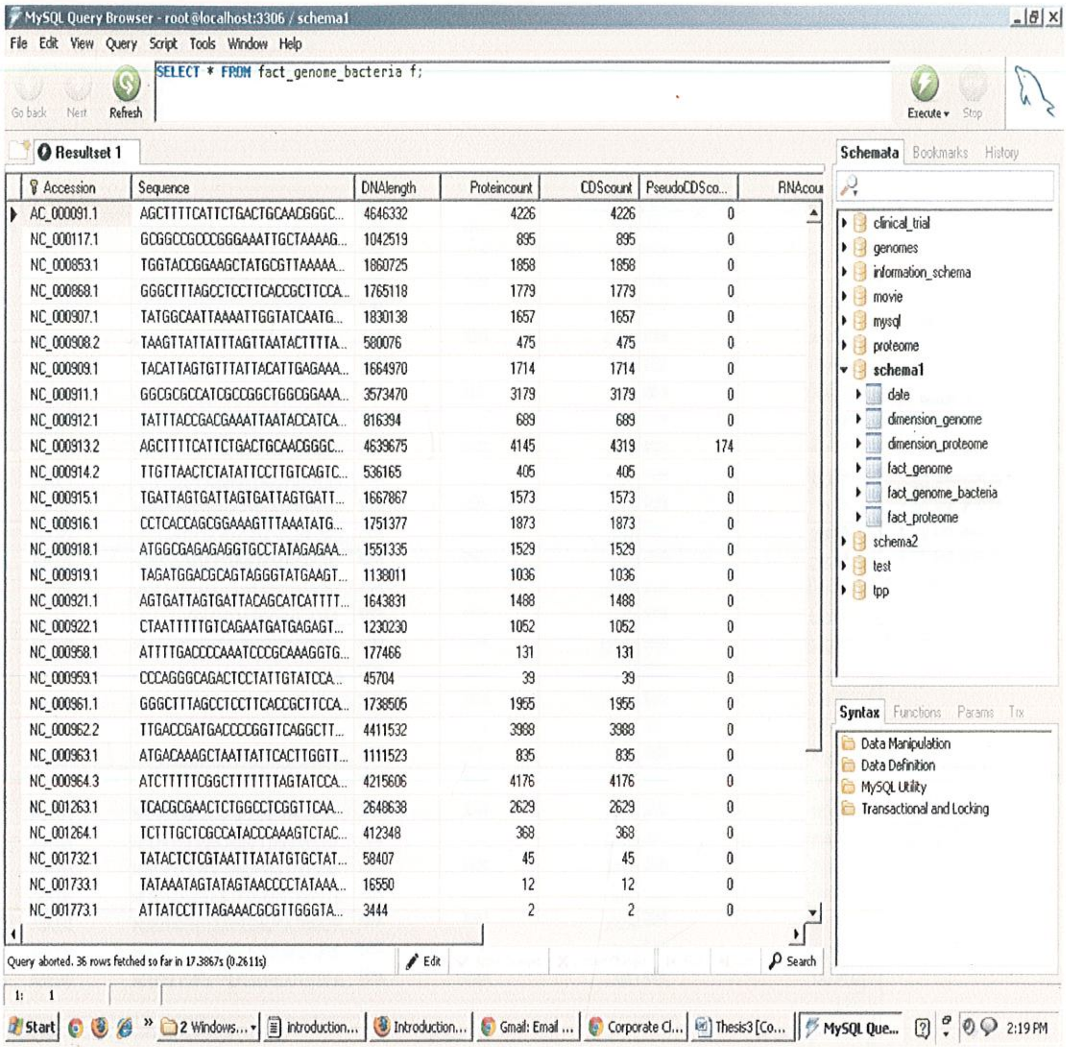

Figure 3.3 - Fact_table_genome

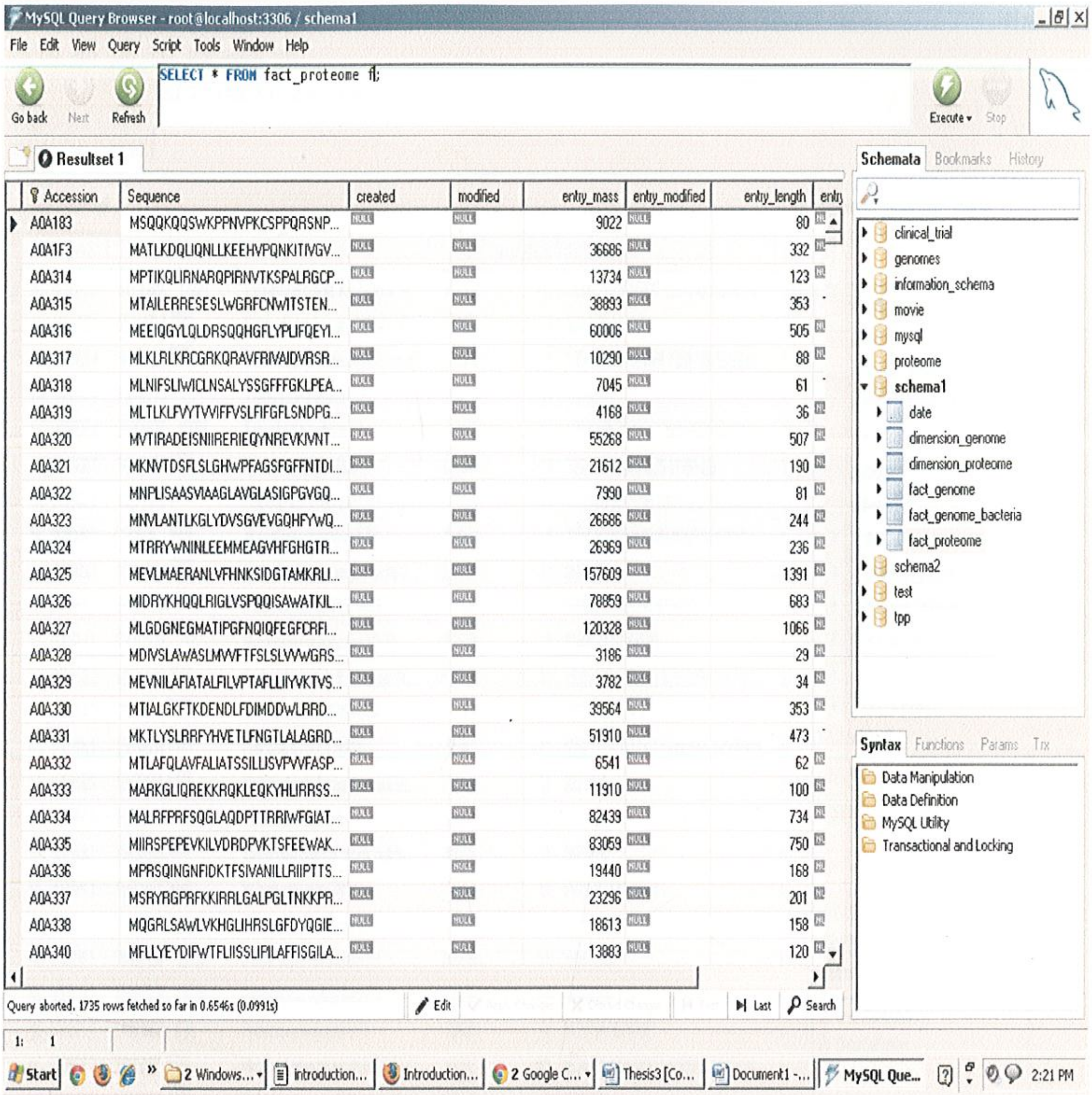

Figure 3.4 - Fact_table_proteome

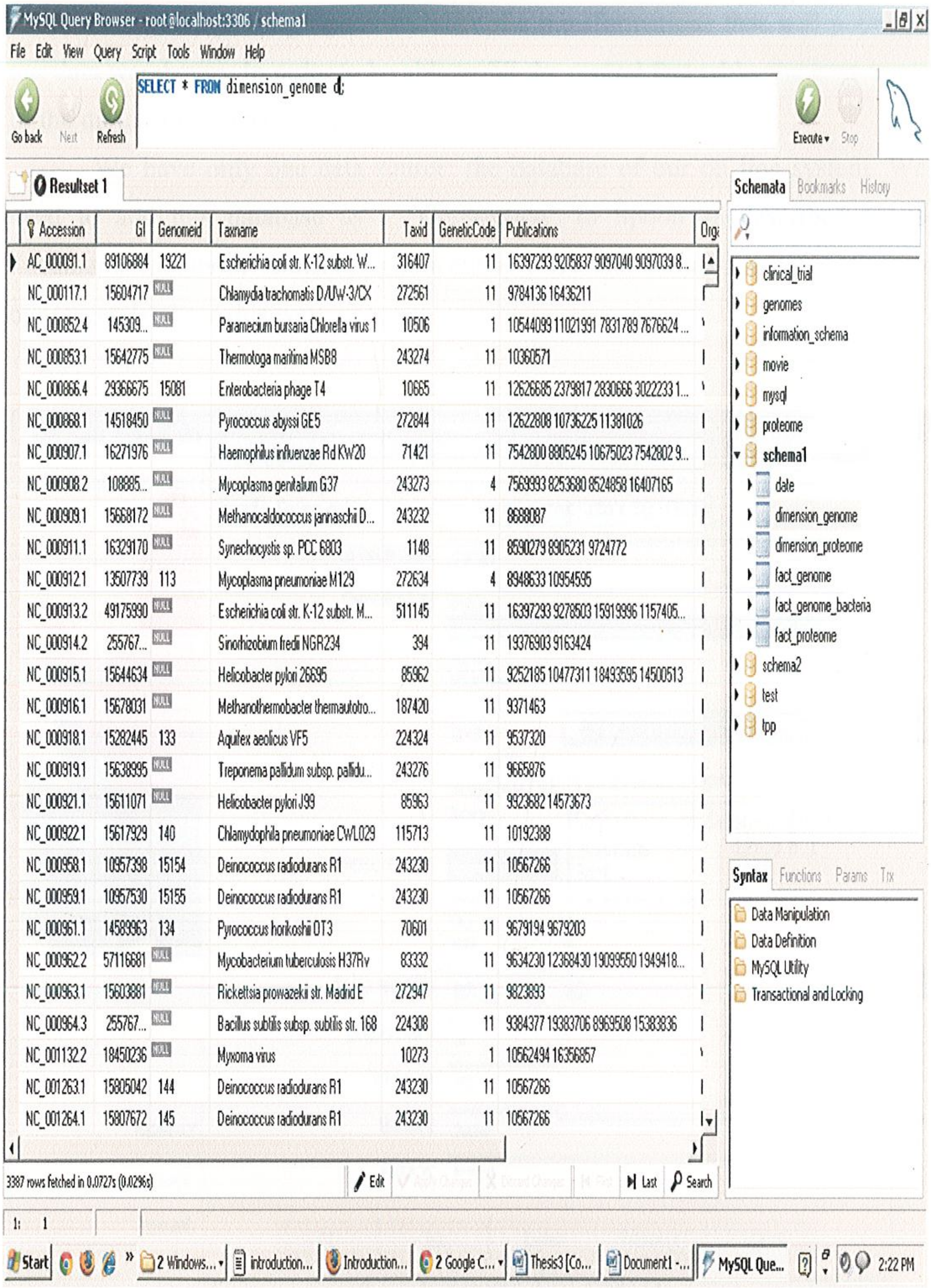

Figure 3.5 -Dimension_table_genome

3.2 DESIGNING TRANSFORMATIONS USING SPOON

We start by using Spoon to make the transformations that will populate our data warehouse. In order to be able to fill the central fact table, the keys to all of the dimensions must be known.

We have only one data source: the database of our on line system. We need to add this database as a "Connection" to Spoon as described in its documentation about "Database connections". We also define the connection to our target database this way.

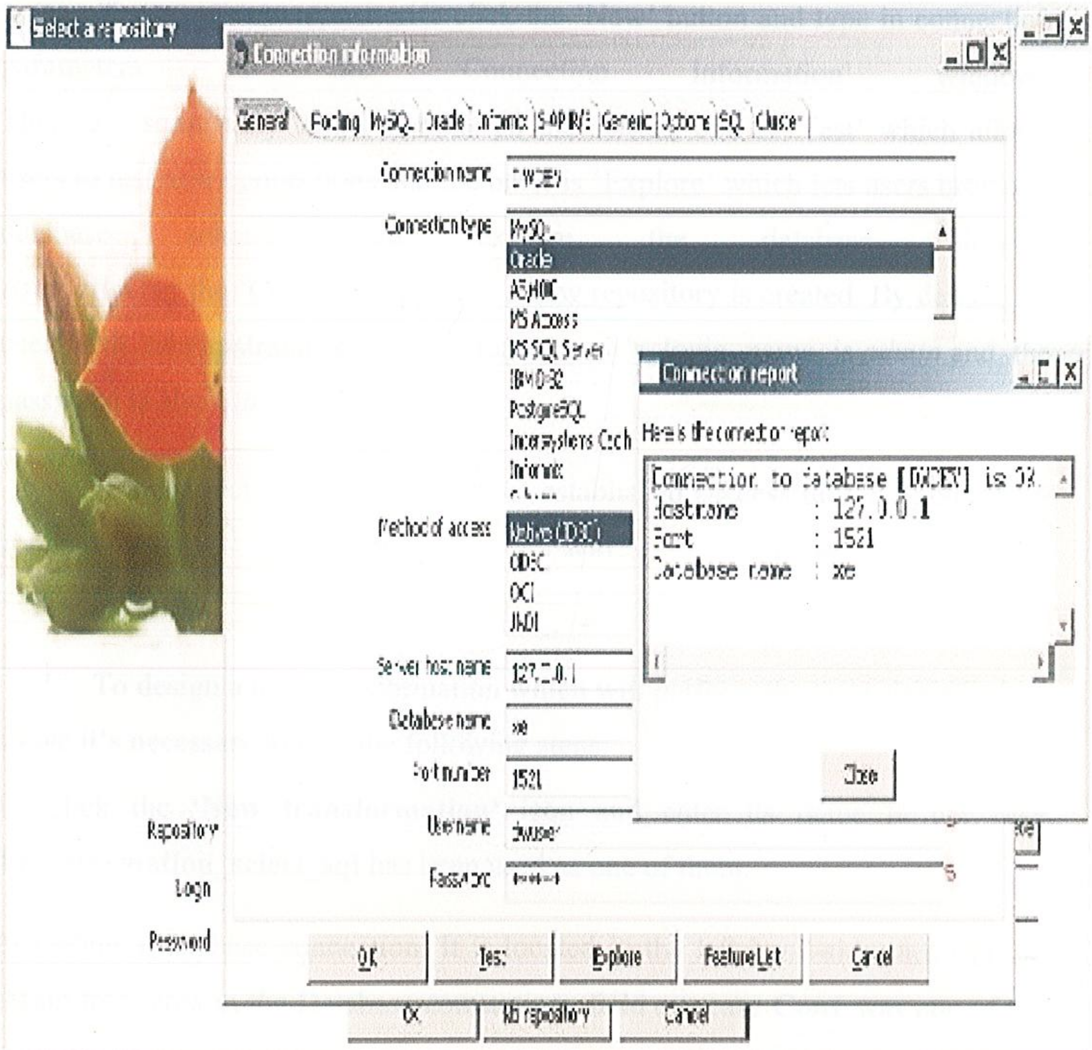

Figure 3.6 - Database connection in Spoon - a part of Kettle ETL

Usually a significant amount of transformation of data occurs at the passage from the operational level to the data warehouse level

The transformation in Devor will read records from a input .CSV files, and then it will filter them out and write output to a separate table. The records which will pass the validation rule will be spooled into a text file and the ones that won't will be redirected to the rejects link which will place them in a different text file.

Assuming that the Spoon application is installed correctly, the first thing to do after running it is to configure a repository. Once the **'Select a repository'** window appears, it's necessary to create or choose one. A repository is a place where all Kettle objects will be stored here MySQl database has been installed .

To create new repositories click the 'New' button and type in connection parameters in the 'Connection information' window. There are some very useful options on the screen, one is 'Test' which allows users to test new connections and the other is 'Explore' which lets users browse a database schema and explore the database objects. After clicking the 'Create or Upgrade' a new repository is created. By default, an user with administrator rights is created – it's login name is *admin* and the password is also *admin*.

If a connection with repository is established successfully, a Spoon main application window will show up.

To design a new transformation which will perform the tasks described above it's necessary to take the following steps:

1) Click the **'New transformation'** icon and enter its name in our case **Transformation_select_sql** has been used as one of them.

2) Define a database connection. It is located in the left hand-side menu in the 'Main tree' area in the Database connections field our case **Con1** was one of the connections.

3) Drag and drop the following elements from the **'Core Objects'** menu to the transformation design area in the center of the screen: **Table Input (menu Output), Join Rows (cartesian Product) and one Field Output table objects** (menu Output).

A mapping is the Kettle solution for transformation re-use. For example if you have a complex calculation that you want to re-use everywhere, you can use a mapping.A mapping is also called a sub-transformation because it is a transformation just like any other with a couple of key differences: Every mapping needs a Mapping Input step to define the fields that are **required** for the mapping to work correctly.Every mapping needs a Mapping Output step to define the fields that are **generated** by the mapping. Because of the static nature of a mapping, Previewing mapping makes no sense.

4) Edit the Table Input – choose a source database and define an SQL query which will return records to the transform flow. The **'Preview'** option is usually very useful here as it shows the preview of the records returned from the database.

5) Next thing to do is to link the objects together. The links between elements are called **Hops** and they indicate which direction the transform flows go. Hops elements can be found, created and edited in the Main Tree section.
The easiest way to create a Hop is to drag and drop a link between two objects with left SHIFT pressed.

6) The last thing to do is to change the text files output configuration(MySQL table here). Enter the names of the files and its extension in the properties window and if needed, adjust other text files specific options here.

7) Save and run the transform (menu -> Transformation -> Run or just press the F9 key). Please find below execution log entries for a correctly configured and run Spoon transform.

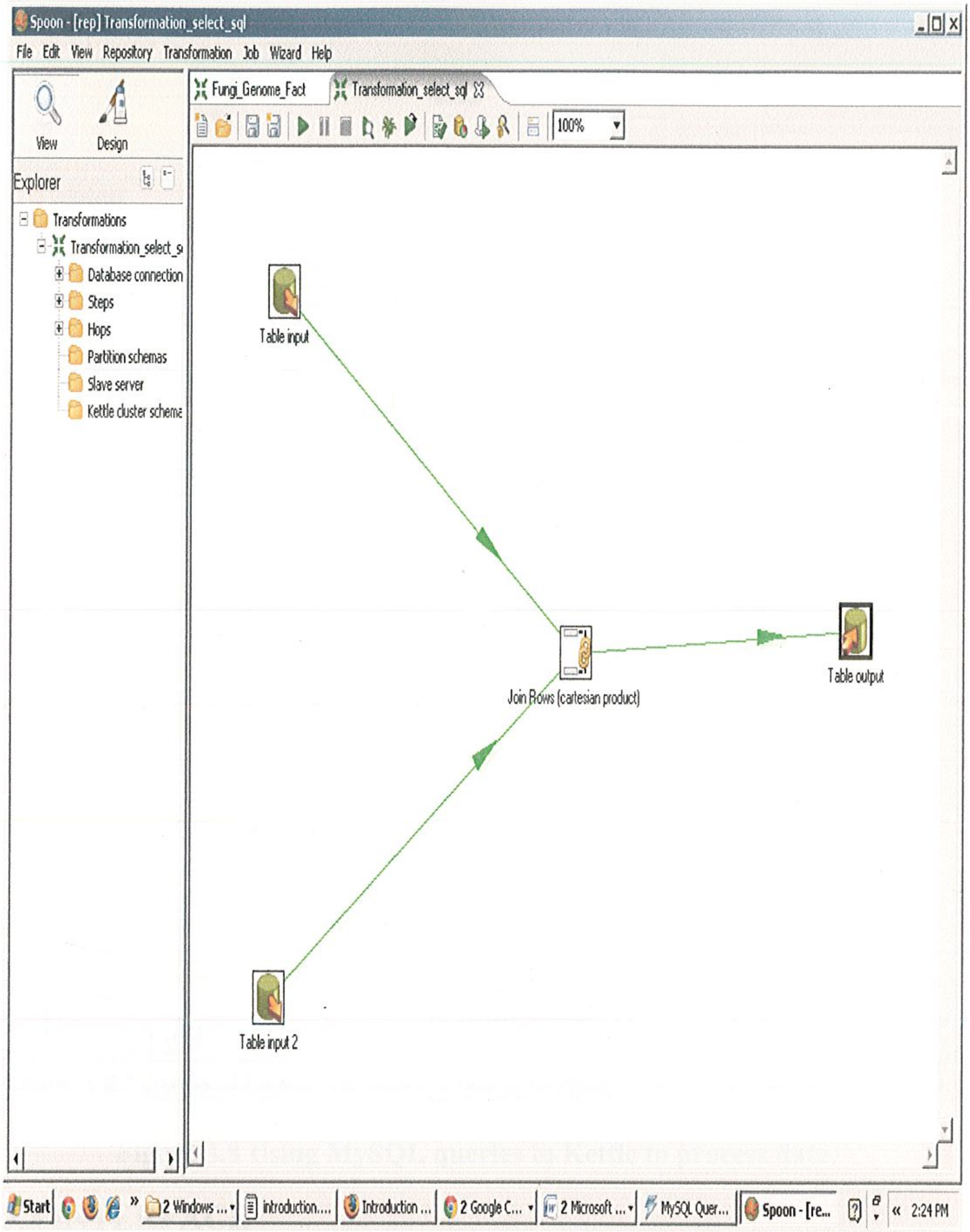

Figure 3.7 - Kettle Transformations: Integrating Genome and Information Genome in the dimension table.

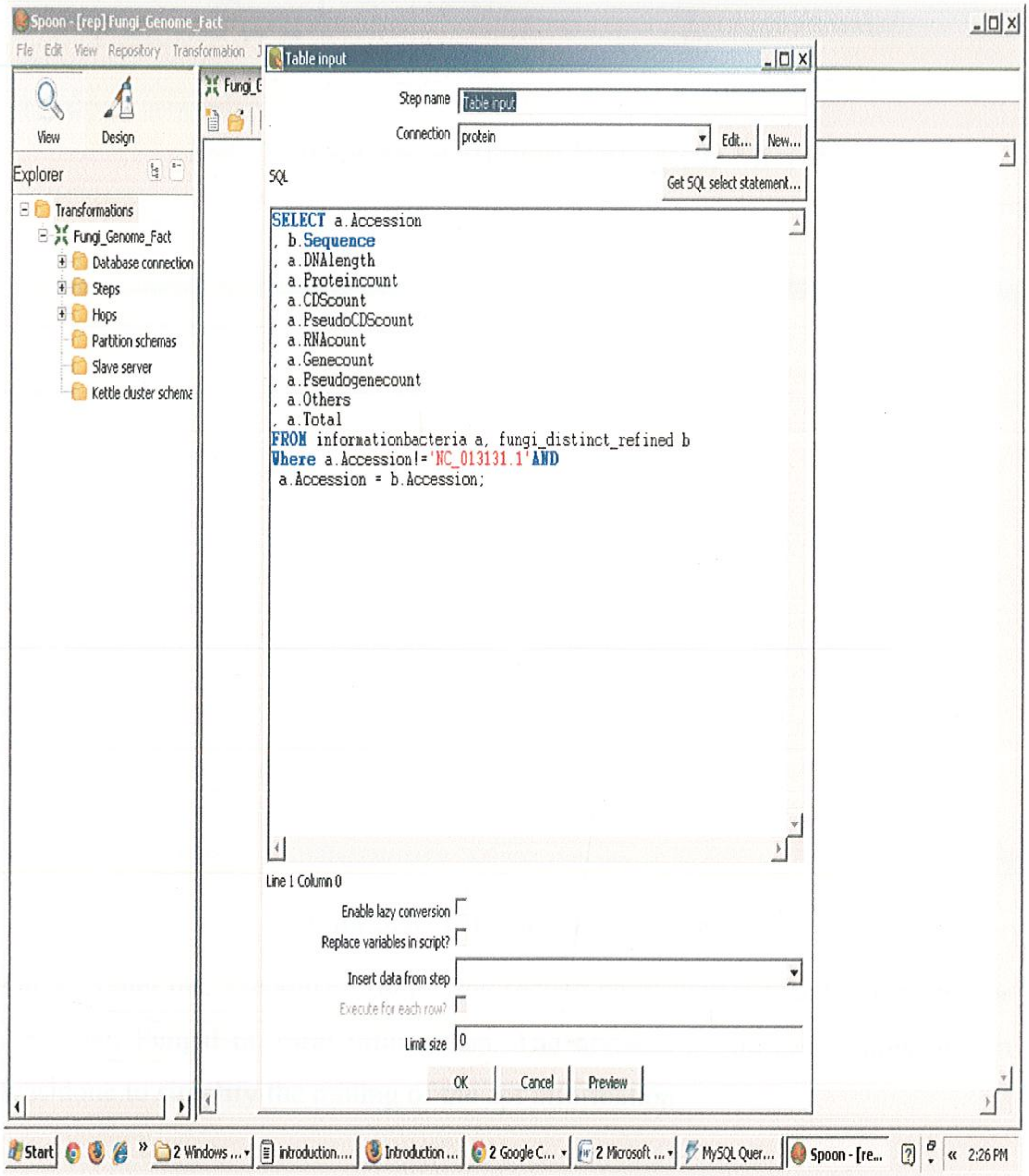

Figure 3.8 Using MySQL queries in Kettle to process data.

3.3 INTERFACE

The interface of the project has been designed in HTML embedded with PHP scripts to make the web pages user interactive. The homepage asks the user to choose whether he or she wants the genomic or proteomic information.

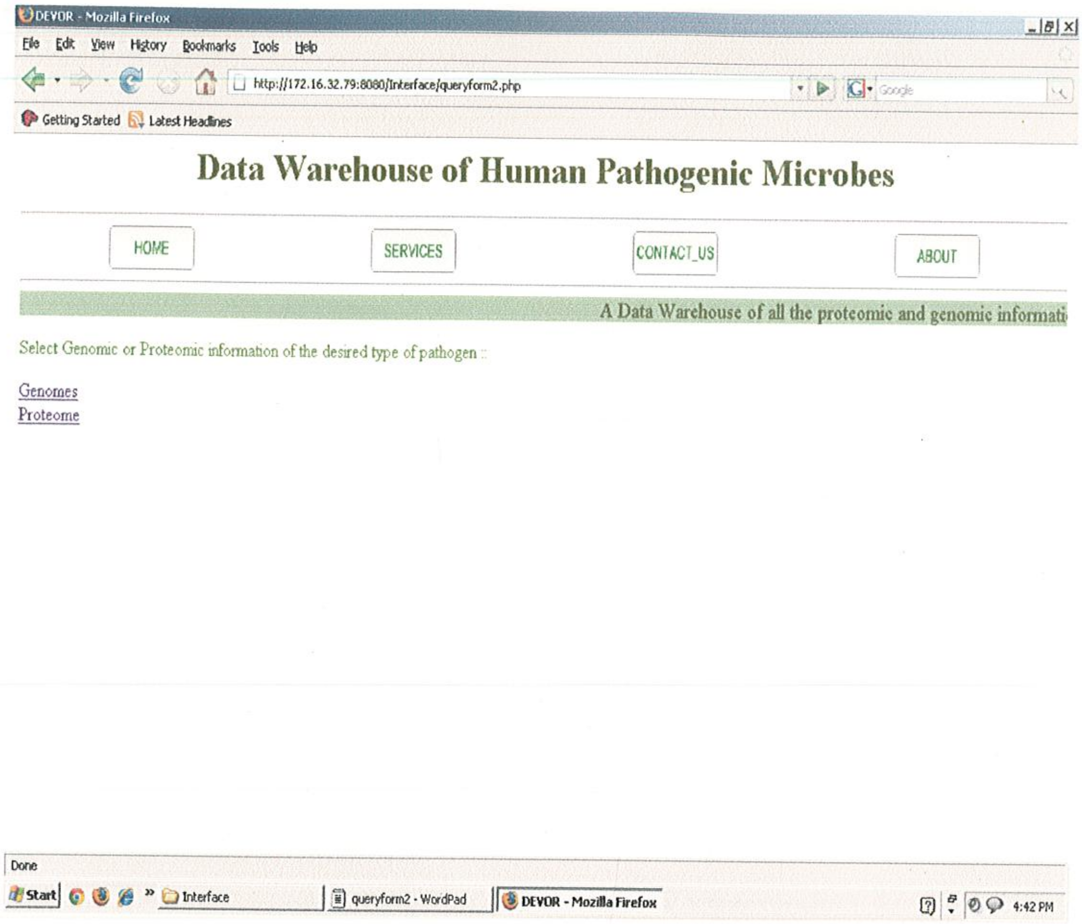

Figure 3.9:- Interface Homepage

On selecting the "Genomes" option the second page queries whether it would be Bacterial, Fungal or Viral information. The division in the data representation was done to simplify the mining of the apt information.

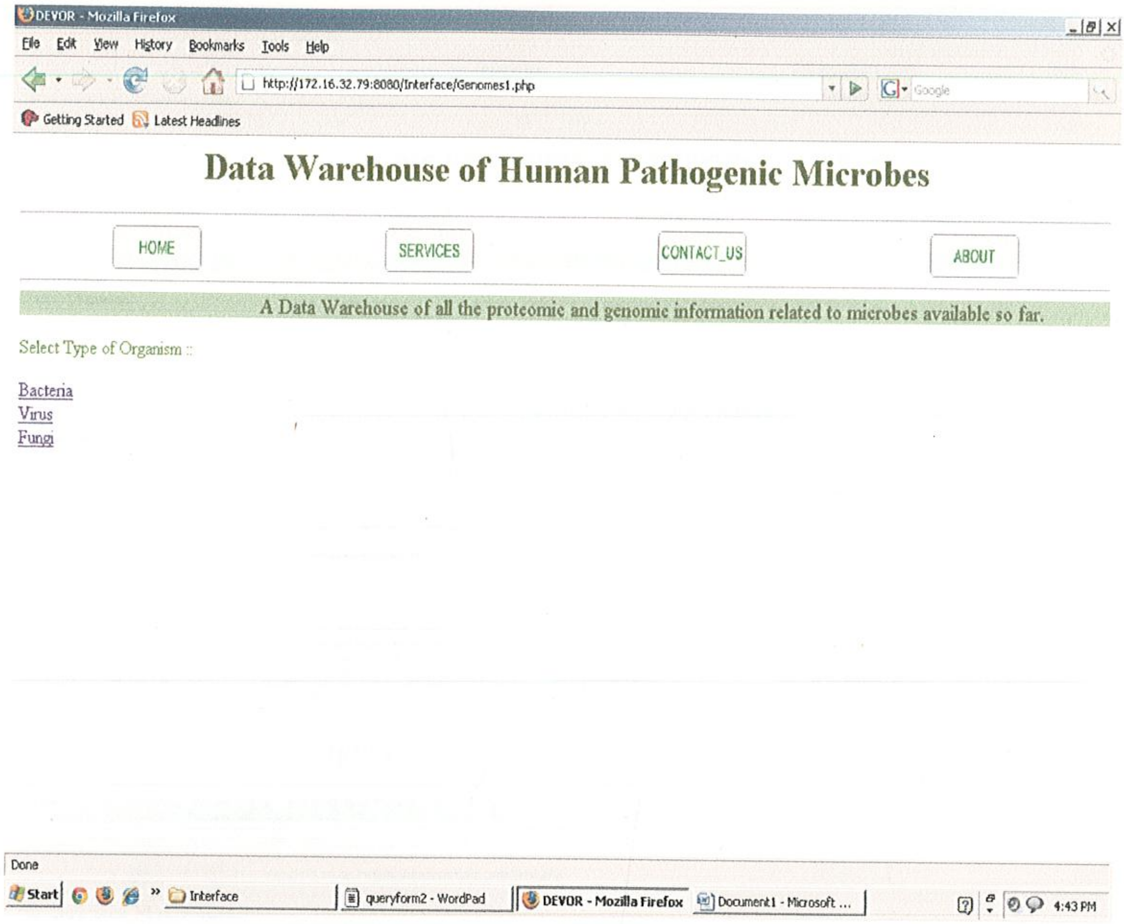

Figure 3.10- Organism selection

The three links present the user options to choose from. These contains the names of all the bacteria , fungi ,viruses collected so far and staged.

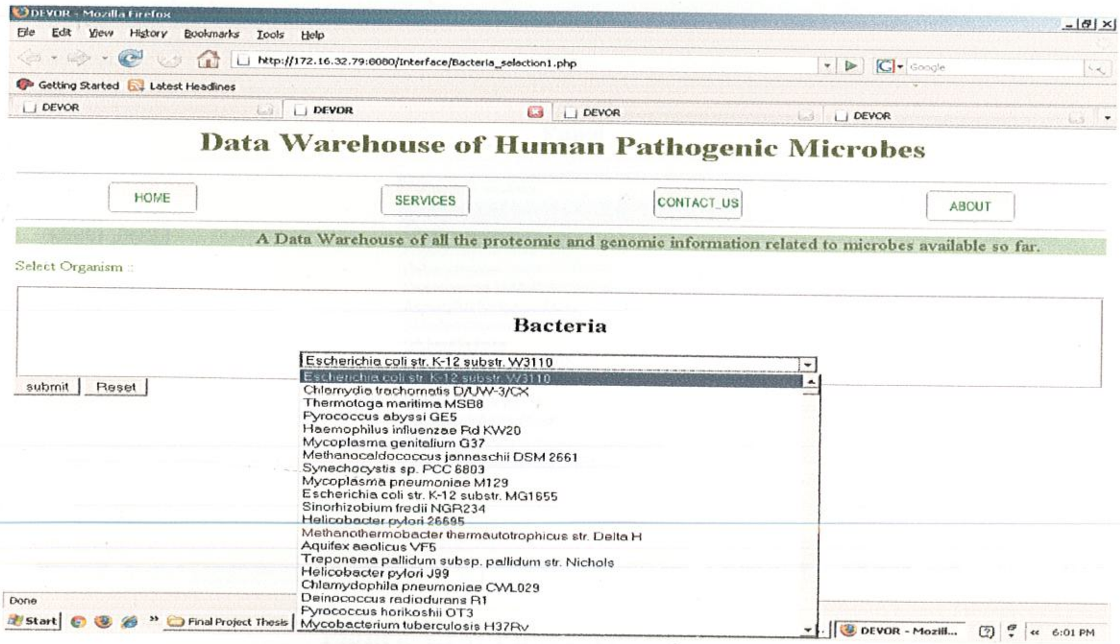

Figure 3.11: Bacteria selection

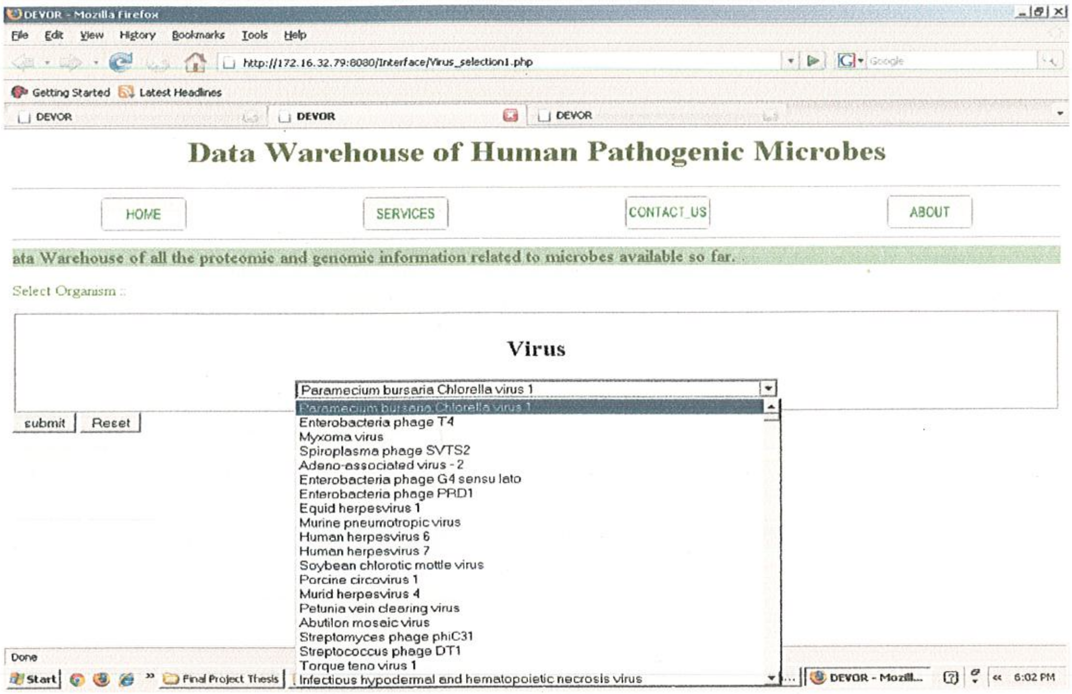

Figure 3.12: Virus selection

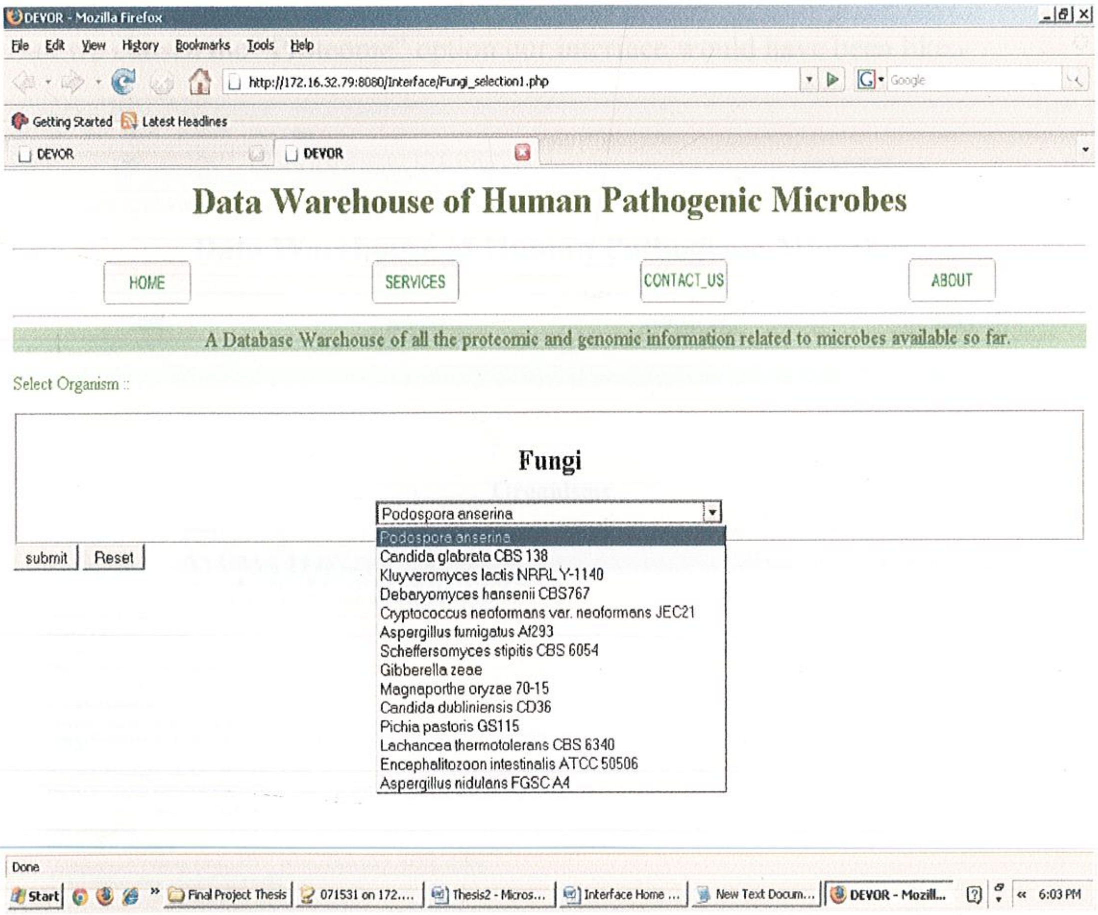

Figure 3.13: Fungi selection

On submitting the option the output is displayed as –

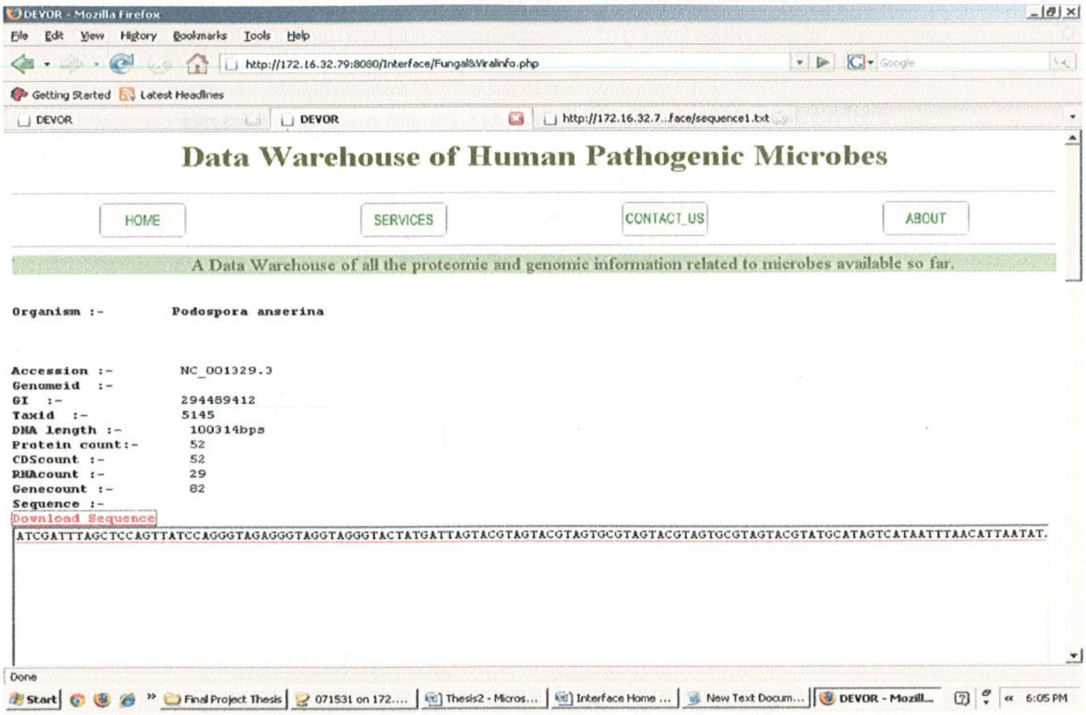

Figure 3.14 : Sample Output for "Podospora anserine" , a fungi

Had we chosen the "Proteome" option our interface would have been like:-

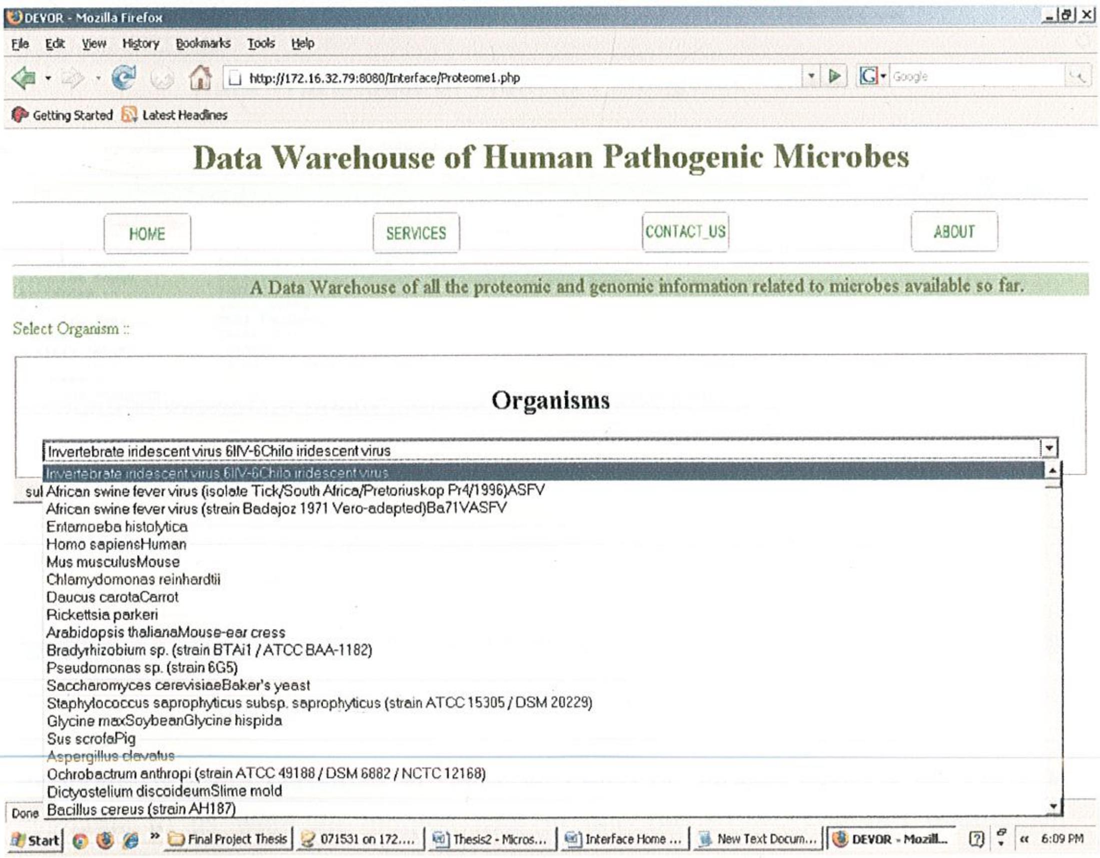

Figure 3.15: All the organisms' selection in the Proteome

On entering the name of the organism:-

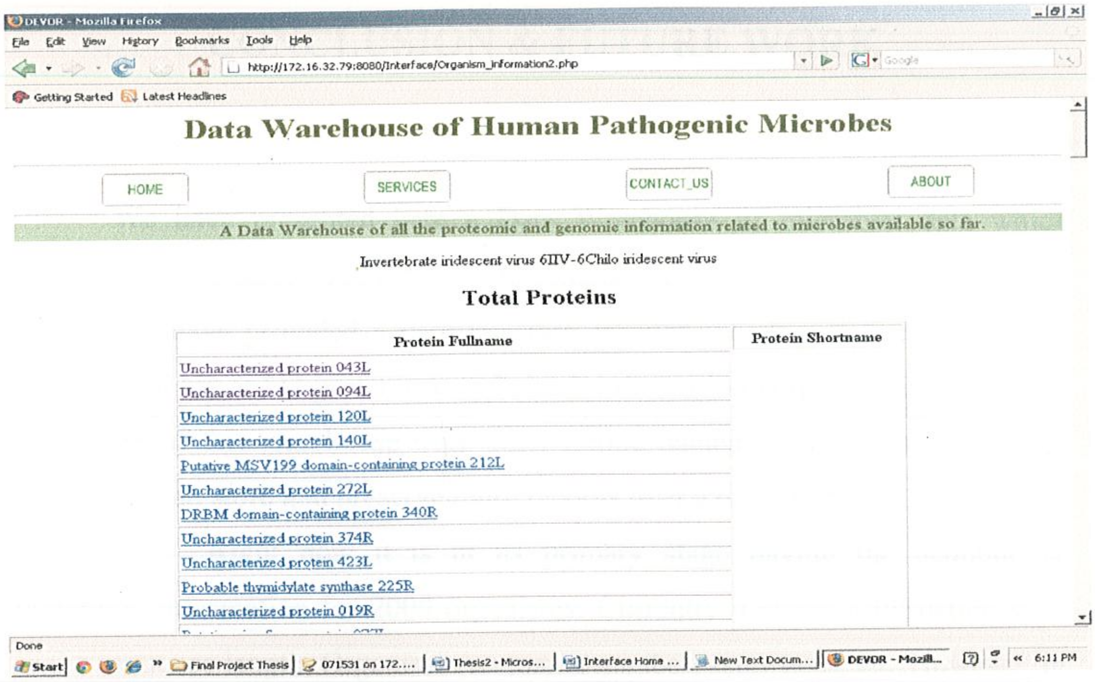

Figure 3.16: All the proteins in the entered organism

On choosing the protein we get information like:-

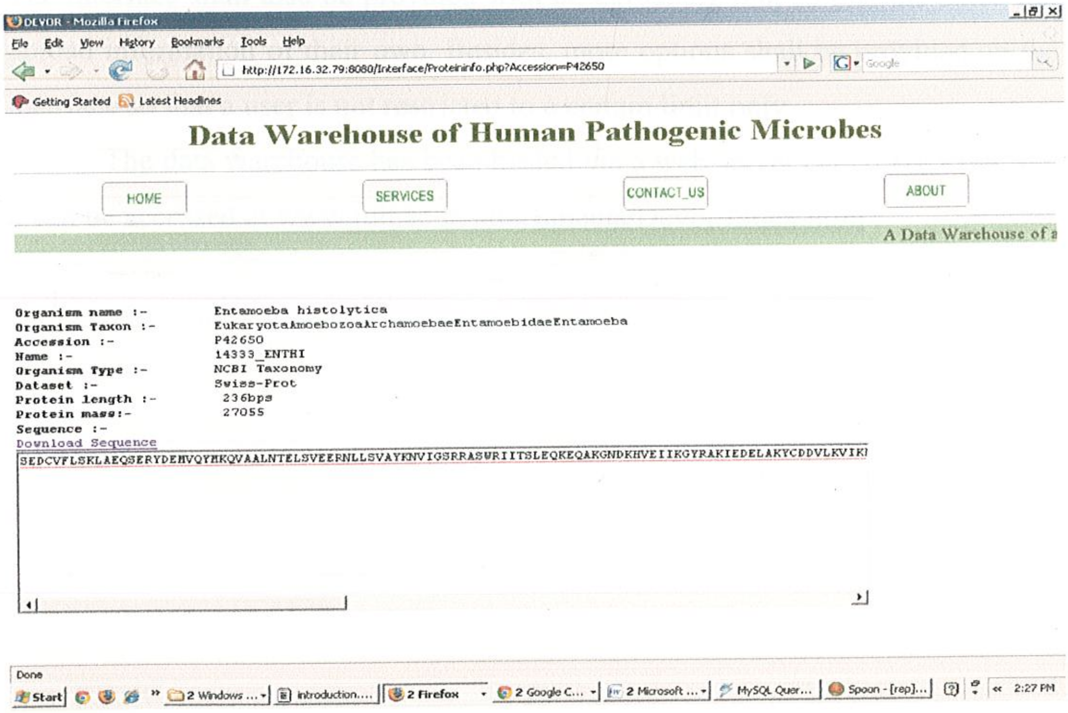

Figure 3.17:- Protein information

Proteome was not classified in Bacteria, Fungi or Viruses because the proteins available to us were all merged data in one file so we couldn't distinguish between them.

CONCLUSION & FUTURE WORK

We have presented here a Data Warehouse of microbial data which includes the Genomic and Proteomic information about the microbes.

We have provided a user interface through which a user can enter the organism of his or her preference and check information corresponding to it. The advantage of this warehouse over other online databases is that it is static in nature and can store and preserve more records over a timeline.

Right now it is in its primary stage having the genomic & proteomic information for 2000 organisms. Currently it stores information such as the sequence length, Organism Accession number, sequences, etc. However, in the future it would be our effort to include more data such as the graphical PDB structures and detailed information about the genomic and proteomic sequences. The interface shall also be provided with an uploading link so that the users can upload information of their own. Besides, more options shall be provided on the interface so that a user is not restricted to a certain limit only.

The data warehouse has been hosted via a web server called **DEVOR** and it can be accessed at www.juit.ac.in/attachments/Devor/Index.html .

REFERENCES

1. Sohrab P Shah, Yong Huang, Tao Xu, Macaire MS Yuen, John Ling and BF Francis Ouellette,"**Atlas – a data warehouse for integrative bioinformatics",** BMC Bioinformatics , **6**:34, 2005.

2. Markus Fischer, Quan K Thai, Melanie Grieb and Jürgen Pleiss," **DWARF – a data warehouse system for analyzing protein families",** BMC Bioinformatics , **7**:495, 2006.

3. Adam Ameur, Vladimir Yankovski, Stefan Enroth, Ola Spjuth and Jan Komorowski "**The LCB Data Warehouse",** Bioinformatics Applications Note, , 1024:1026, 2006.

4. Zukang Feng, Li Chen, Himabindu Maddula, Ozgur Akcan, Rose Oughtred, Helen M. Berman and John Westbrook," **Ligand Depot: a data warehouse for ligands bound to macromolecules**", Bioinformatics Applications Note,2153-2155,2004.

5. Ralph Kimball and Margy Ross "**The Data Warehouse Toolkit – The complete guide to dimensional modeling**", USA, Wiley Publications, 2002.

APPENDIX A

Scripts used in the Project:-

Script1 – Append1.pl – Used to format .txt, .fnn , .faa , etc. to .csv file format the way we want it. It adds a blank space between sequences.

```
#! usr/bin/perl
$my_file="abc1.csv";
print "\n Enter the file name \n";

$new_file=<stdin>;
open(f1,"$new_file");
$i=0;
open(PLOT,">>$my_file")||die ("The file cannot be opened!");
#print PLOT "\n";
while($_=<f1>)
{
	if(/^>/)
	{
	print PLOT "\n";
	chomp $_;
	print PLOT $_;
	}
	else
	{
	chomp $_;
	print PLOT $_;
	}

}
close(PLOT);
```

Script 2:- Fields.pl – Used for adding fields "gene,Field1" to the sequences. It was essential to enter sequences into the database.

```
#!usr/bin/perl

for($i=5;$i<=8;$i++)
{
$file_naml="add$i.txt";
open (f2,">>$file_naml");
print f2 "gene,Field1";

close(f2);
}
```

Script 3 – Text.pl – Used to format Fasta formatted sequences into text format.

```
#!usr/bin/perl
for($i=2;$i<=8;$i++)
{
#$file_name="add$i.txt";
#$file_naml="add1$i.csv";
$file_name="abc.csv";
$file_naml="abc1.csv";
```

```
open (f1,"$file_name");
open (f2,">>$file_naml");
while($s=<f1>)
{
	chomp $s;
	@arr=split('',$s);
	foreach $y(@arr)
	{
		if($y eq '>')
	{
		print f2 "\n";
		print f2 ">";
	}
	}
}
close(f2);
close(f1);
}
```

PHP scripts:-

Script1:- To display Bacterial information:-

Bacterialinfo.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<style>
    div {
      padding-top: 5px;
      padding-bottom: 5px;
      padding-right: 5px;
      padding-left: 30px;
      border: 3px solid;
    }
    h2 {
      text-indent: -25px;
    }
 </style>
<title>DEVOR</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
<script language="JavaScript" type="text/JavaScript">
<!--
function MM_swapImgRestore() { //v3.0
  var i,x,a=document.MM_sr;
for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
}

function MM_preloadImages() { //v3.0
  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
    var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0;
i<a.length; i++)
    if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j+
+].src=a[i];}}
}

function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document;
if((p=n.indexOf("?"))>0&&parent.frames.length) {
```

```
    d=parent.frames[n.substring(p+1)].document;
n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!
x&&i<d.forms.length;i++) x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++)
x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_swapImage() { //v3.0
  var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array;
for(i=0;i<(a.length-2);i+=3)
   if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!
x.oSrc) x.oSrc=x.src; x.src=a[i+2];}
}
//-->
</script>
</head><body
onload="MM_preloadImages('logos/new/HELP.JPG','logos/new/cntct.JP
G','logos/new/2DSTR.JPG','logos/new/PDB.JPG')"><h1
align="center"><font color="#336600" face="Times New Roman,
Times, serif">Data Warehouse of Human Pathogenic
Microbes</font></h1>

<hr>
<table border="0" cellpadding="0" cellspacing="0" width="100%">
<tbody><tr>
<td align="center"><a href="queryform2.php"
onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('Home','','logos/new/Home.png',1)"><img
src="queryform1_files/Home.png" alt="Home" name="Home" border="0"
height="39" width="84"></a></td>
<td align="center"><a href="services.php"
onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('PDB','','logos/new/Services.png',1)"><
img src="queryform1_files/Services.png" alt="Services"
name="Services" border="0" height="39" width="84"></a></td>
<td align="center"><a href="contact.php.html"
onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('contact
us','','logos/new/Contact_us.png',1)"><img
src="queryform1_files/Contact_us.png" alt="Contact administrator"
name="contact us" border="0" height="39" width="84"></a> </td>
<td align="center"><a href="introduction.php.html"
onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('Help','','logos/new/About.png',1)"><im
g src="queryform1_files/About.png" alt="Introduction"
name="About" border="0" height="39" width="84"></a></td>
</tr>
</tbody></table>
<hr>
<table border="0" cellpadding="0" cellspacing="0" width="100%"
align="center">
  <tbody><tr bgcolor="#99cc99">
    <td><marquee behavior="scroll" direction="left"><font
color="#336600" face="Times New Roman, Times, serif" size="+1">A
Data Warehouse of all the proteomic and genomic information
related to microbes available so far.</font></marquee></td>
  </tr>
</tbody></table>
```

```
<p align="left">
<pre>
<font size="3">
<?php
print "<b>Organism :-          ".$_POST['Organisms']."</b>";
print "<br>";
?>
</font>
</pre>
</p>
<p align="left">
<pre>
<font size="3">
<?php
$name=$_POST['Organisms'];
$link=mysql_connect("localhost","root","root")or
die(mysql_error());
mysql_select_db("schema1")or die(mysql_error());
$query="select Accession,GI,Genomeid,Taxid from dimension_genome
where Taxname='$name'";
$result=mysql_query($query,$link)or die(mysql_error());
while($row=mysql_fetch_array($result))
{
	$acc=$row['Accession'];$gi=$row['GI'];
$genomeid=$row['Genomeid'];$taxid=$row['Taxid'];
	print "<b>Accession :-         </b>".$acc."<br>";
	print "<b>Genomeid  :-         </b>".$genomeid."<br>";
	print "<b>GI  :-               </b>".$gi."<br>";
	print "<b>Taxid  :-            </b>".$taxid."<br>";

	$query2="select
Sequence,DNAlength,Proteincount,CDScount,RNAcount,Genecount from
fact_genome_bacteria where Accession='$acc'";
	$result2=mysql_query($query2,$link)or die(mysql_error());
	while($row1=mysql_fetch_array($result2))
	{
	print "<b>DNA length :-         </b>".
$row1['DNAlength']."bps <br>";
	print "<b>Protein count:-       </b>".
$row1['Proteincount']."<br>";
	print "<b>CDScount :-           </b>".
$row1['CDScount']."<br>";
	print "<b>RNAcount :-           </b>".
$row1['RNAcount']."<br>";
	print "<b>Genecount :-          </b>".
$row1['Genecount']."<br>";
	print "<b>Sequence :-           </b>"."<br>";
	$fh=fopen("Sequence.txt",'w')or die("Can't open file");
	fwrite($fh,">gi|");
	fwrite($fh,$gi);
	fwrite($fh,"|genomeid|");
	fwrite($fh,$genomeid);
	fwrite($fh,"|");
	fwrite($fh,$name);
	fwrite($fh,"|");
	fwrite($fh,$row1['Sequence']);
	fclose($fh);

?>
<a href="Sequence.txt">Download Sequence</a>
<textarea rows="100" cols="120">
```

```
<?php
      print $row1['Sequence'];
?>
</textarea>
<?php
      }

}


?>
</font>
</pre>
</p>


</body></html>
```

Script 2: To display proteome information:
Proteininfo.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>

<title>DEVOR</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
<script language="JavaScript" type="text/JavaScript">
<!--
function MM_swapImgRestore() { //v3.0
  var i,x,a=document.MM_sr;
for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
}

function MM_preloadImages() { //v3.0
  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
    var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0;
i<a.length; i++)
    if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j+
+].src=a[i];}}
}

function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document;
if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document;
n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!
x&&i<d.forms.length;i++) x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++)
x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_swapImage() { //v3.0
  var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array;
for(i=0;i<(a.length-2);i+=3)
   if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!
x.oSrc) x.oSrc=x.src; x.src=a[i+2];}
```

```
}
//-->
</script>
</head><body
onload="MM_preloadImages('logos/new/HELP.JPG','logos/new/cntct.JP
G','logos/new/2DSTR.JPG','logos/new/PDB.JPG')"><h1
align="center"><font color="#336600" face="Times New Roman,
Times, serif">Data Warehouse of Human Pathogenic
Microbes</font></h1>

<hr>
<table border="0" cellpadding="0" cellspacing="0" width="100%">
<tbody><tr>
<td align="center"><a href="queryform2.php"
onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('Home','','logos/new/Home.png',1)"><img
src="queryform1_files/Home.png" alt="Home" name="Home" border="0"
height="39" width="84"></a></td>
<td align="center"><a href="services.php"
onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('PDB','','logos/new/Services.png',1)"><
img src="queryform1_files/Services.png" alt="Services"
name="Services" border="0" height="39" width="84"></a></td>
<td align="center"><a href="contact.php.html"
onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('contact
us','','logos/new/Contact_us.png',1)"><img
src="queryform1_files/Contact_us.png" alt="Contact administrator"
name="contact us" border="0" height="39" width="84"></a> </td>
<td align="center"><a href="introduction.php.html"
onmouseout="MM_swapImgRestore()"
onmouseover="MM_swapImage('Help','','logos/new/About.png',1)"><im
g src="queryform1_files/About.png" alt="Introduction"
name="About" border="0" height="39" width="84"></a></td>
</tr>
</tbody></table>
<hr>
<table border="0" cellpadding="0" cellspacing="0" width="100%"
align="center">
  <tbody><tr bgcolor="#99cc99">
    <td><marquee behavior="scroll" direction="left"><font
color="#336600" face="Times New Roman, Times, serif" size="+1">A
Data Warehouse of all the proteomic and genomic information
related to microbes available so far.</font></marquee></td>
  </tr>
</tbody></table>
<pre>
<font size="3">
<p align="left">
<?php
$link=mysql_connect("localhost","root","root") or
die(mysql_error());
mysql_select_db("schema1")or die(mysql_error());
$query="select * from dimension_proteome where Accession='".
$_GET['Accession']."'";
$result=mysql_query($query,$link)or die(mysql_error());
while($row=mysql_fetch_array($result))
{
      $acc=$row['Accession'];
      print "<b>Organism name :-        </b>".
$row['entry_name']."<br>";
```

```
      print "<b>Organism Taxon :-        </b>".
$row['entry_organism_taxon']."<br>";
      print "<b>Accession :-             </b>".
$row['Accession']."<br>";
      print "<b>Name :-                  </b>".
$row['Name']."<br>";
      print "<b>Organism Type :-         </b>".
$row['entry_organism_type']."<br>";
      print "<b>Dataset :-               </b>".
$row['Dataset']."<br>";
      $query2="select Sequence,entry_mass,entry_length from
fact_proteome where Accession='$acc'";
      $result2=mysql_query($query2,$link)or die(mysql_error());
      while($row1=mysql_fetch_array($result2))
      {
      print "<b>Protein length :-        </b>".
$row1['entry_length']."bps <br>";
      print "<b>Protein mass:-           </b>".
$row1['entry_mass']."<br>";
      print "<b>Sequence :-              </b>"."<br>";
      $fh=fopen("Proteinsequence.txt",'w')or die("Can't open
file");
      fwrite($fh,">Accession|");
      fwrite($fh,$acc);
      fwrite($fh,"|");
      fwrite($fh,$row['Name']);
      fwrite($fh,"|");
      fwrite($fh,$row1['Sequence']);
      fclose($fh);

?>
<a href="Proteinsequence.txt">Download Sequence</a>
<textarea rows="10" cols="100">
<?php
      print $row1['Sequence'];
?>
</textarea>
<?php
      }

}
?>
</font>
</pre>
</p>

</body></html>
```

APPENDIX B

MySQL Queries used:-

To create the working schema :-

```
-- ---
-- Table 'Dimension_proteome'
--
-- ---

DROP TABLE IF EXISTS `Dimension_proteome`;

CREATE TABLE `Dimension_proteome` (
  `ID` INTEGER AUTO_INCREMENT DEFAULT NULL,
  `UniprotID` INTEGER DEFAULT NULL,
  `Accession` BLOB DEFAULT NULL,
  `Dataset` VARCHAR(10) DEFAULT NULL,
  `Name` VARCHAR(11) DEFAULT NULL,
  `entry_fragment` VARCHAR(8) DEFAULT NULL,
  `entry_name` VARCHAR(153) DEFAULT NULL,
  `entry_organism_id` INTEGER DEFAULT NULL,
  `entry_organism_key` INTEGER DEFAULT NULL,
  `entry_organism_taxon` VARCHAR(239) DEFAULT NULL,
  `entry_organism_type` VARCHAR(13) DEFAULT NULL,
  `entry_organism_fullname` VARCHAR(133) DEFAULT NULL,
  `entry_organism_shortname` VARCHAR(117) DEFAULT NULL,
  `entry_version` INTEGER DEFAULT NULL,
  PRIMARY KEY (`Accession`)
);

-- ---
-- Table 'Fact_proteome'
--
-- ---

DROP TABLE IF EXISTS `Fact_proteome`;

CREATE TABLE `Fact_proteome` (
  `Sequence` BLOB DEFAULT NULL,
```

```
 `Created_date_id` INTEGER DEFAULT NULL,
 `Modified_date_id` INTEGER DEFAULT NULL,
 `entry_mass` INTEGER DEFAULT NULL,
 `entry_modified_id` INTEGER DEFAULT NULL,
 `entry_length` INTEGER DEFAULT NULL,
 `entry_precursor` VARCHAR(5) DEFAULT NULL,
 `Accession_Dimension_proteome` BLOB DEFAULT NULL,
 PRIMARY KEY (`Created_date_id`, `Modified_date_id`, `entry_modified_id`,
`Accession_Dimension_proteome`)
);

-- ---
-- Table 'Date'
--
-- ---

DROP TABLE IF EXISTS `Date`;

CREATE TABLE `Date` (
 `DATE_ID` INTEGER AUTO_INCREMENT DEFAULT NULL,
 `DATE_VALUE` INTEGER DEFAULT NULL,
 `DAY_OF_WEEK` VARCHAR(20) DEFAULT NULL,
 `DAY_OF_MONTH` VARCHAR(20) DEFAULT NULL,
 `WEEK_OF_YEAR` VARCHAR(20) DEFAULT NULL,
 `WEEK_SORT_VALUE` INTEGER DEFAULT NULL,
 `MONTH_OF_YEAR` VARCHAR(20) DEFAULT NULL,
 `MONTH_SORT_VALUE` INTEGER DEFAULT NULL,
 `QTR_OF_YEAR` VARCHAR(20) DEFAULT NULL,
 `QTR_SORT_VALUE` INTEGER DEFAULT NULL,
 `CALENDER_YEAR` INTEGER DEFAULT NULL,
 PRIMARY KEY (`DATE_ID`)
);

-- ---
-- Table 'Dimension_genome'
--
-- ---

DROP TABLE IF EXISTS `Dimension_genome`;
```

```
CREATE TABLE `Dimension_genome` (
 `Genome` BLOB DEFAULT NULL,
 `Accession` VARCHAR(50) DEFAULT NULL,
 `GI` INTEGER DEFAULT NULL,
 `Genomeid` VARCHAR(50) DEFAULT NULL,
 `Taxname` VARCHAR(100) DEFAULT NULL,
 `Taxid` INTEGER DEFAULT NULL,
 `GeneticCode` INTEGER DEFAULT NULL,
 `Publications` MEDIUMTEXT DEFAULT NULL,
 `Organism` VARCHAR(50) DEFAULT NULL,
 PRIMARY KEY (`Accession`)
);

-- ---
-- Table 'Fact_genome'
--
-- ---

DROP TABLE IF EXISTS `Fact_genome`;

CREATE TABLE `Fact_genome` (
 `Sequence` BLOB DEFAULT NULL,
 `DNAlength` INTEGER DEFAULT NULL,
 `Proteincount` INTEGER DEFAULT NULL,
 `CDScount` INTEGER DEFAULT NULL,
 `PseudoCDScount` INTEGER DEFAULT NULL,
 `RNAcount` INTEGER DEFAULT NULL,
 `Genecount` INTEGER DEFAULT NULL,
 `Pseudogenecount` INTEGER DEFAULT NULL,
 `Others` INTEGER DEFAULT NULL,
 `Total` INTEGER DEFAULT NULL,
 `Accession_Dimension_genome` VARCHAR(50) DEFAULT NULL,
 `Created_date_id` INTEGER DEFAULT NULL,
 PRIMARY KEY (`Accession_Dimension_genome`, `Created_date_id`)
);

-- ---
-- Foreign Keys
-- ---
```

```
ALTER TABLE `Fact_proteome` ADD FOREIGN KEY (Created_date_id)
REFERENCES `Date` (`DATE_ID`);
ALTER TABLE `Fact_proteome` ADD FOREIGN KEY (Modified_date_id)
REFERENCES `Date` (`DATE_ID`);
ALTER TABLE `Fact_proteome` ADD FOREIGN KEY (entry_modified_id)
REFERENCES `Date` (`DATE_ID`);
ALTER TABLE `Fact_proteome` ADD FOREIGN KEY
(Accession_Dimension_proteome) REFERENCES `Dimension_proteome`
(`Accession`);
ALTER TABLE `Fact_genome` ADD FOREIGN KEY
(Accession_Dimension_genome) REFERENCES `Dimension_genome`
(`Accession`);
ALTER TABLE `Fact_genome` ADD FOREIGN KEY (Created_date_id)
REFERENCES `Date` (`DATE_ID`);

-- ---
-- Table Properties
-- ---

-- ALTER TABLE `Dimension_proteome` ENGINE=InnoDB DEFAULT
CHARSET=utf8 COLLATE=utf8_bin;
-- ALTER TABLE `Fact_proteome` ENGINE=InnoDB DEFAULT
CHARSET=utf8 COLLATE=utf8_bin;
-- ALTER TABLE `Date` ENGINE=InnoDB DEFAULT CHARSET=utf8
COLLATE=utf8_bin;
-- ALTER TABLE `Dimension_genome` ENGINE=InnoDB DEFAULT
CHARSET=utf8 COLLATE=utf8_bin;
-- ALTER TABLE `Fact_genome` ENGINE=InnoDB DEFAULT
CHARSET=utf8 COLLATE=utf8_bin;


-- ---
-- Test Data
-- ---

-- INSERT INTO `Dimension_proteome`
(`ID`,`UniprotID`,`Accession`,`Dataset`,`Name`,`entry_fragment`,`entry_name`,`
entry_organism_id`,`entry_organism_key`,`entry_organism_taxon`,`entry_organi
sm_type`,`entry_organism_fullname`,`entry_organism_shortname`,`entry_versio
n`) VALUES
-- ('','','','','','','','','','','','','','');
```

```
-- INSERT INTO `Fact_proteome`
(`Sequence`,`Created_date_id`,`Modified_date_id`,`entry_mass`,`entry_modified
_id`,`entry_length`,`entry_precursor`,`Accession_Dimension_proteome`)
VALUES
-- ('','','','','','','','');
-- INSERT INTO `Date`
(`DATE_ID`,`DATE_VALUE`,`DAY_OF_WEEK`,`DAY_OF_MONTH`,`WEE
K_OF_YEAR`,`WEEK_SORT_VALUE`,`MONTH_OF_YEAR`,`MONTH_SO
RT_VALUE`,`QTR_OF_YEAR`,`QTR_SORT_VALUE`,`CALENDER_YEAR`
) VALUES
-- ('','','','','','','','','','','');
-- INSERT INTO `Dimension_genome`
(`Genome`,`Accession`,`GI`,`Genomeid`,`Taxname`,`Taxid`,`GeneticCode`,`Pub
lications`,`Organism`) VALUES
-- ('','','','','','','','','');
-- INSERT INTO `Fact_genome`
(`Sequence`,`DNAlength`,`Proteincount`,`CDScount`,`PseudoCDScount`,`RNAc
ount`,`Genecount`,`Pseudogenecount`,`Others`,`Total`,`Accession_Dimension_g
enome`,`Created_date_id`) VALUES
-- ('','','','','','','','','','','','');
```

BRIEF BIO-DATA OF STUDENTS

Anant Chaturvedi is currently commencing his final semester BTech studies in Bioinformatics from Jaypee University of Information Technology from Jaypee University of Information Technology and will be completing his degree in June 2011. His area of interest lies in Data Warehousing and Database Management Systems. He has been selected in Accenture & Wipro in campus placement for the year 2010-11 and will be joining either of the firms in June 2011.

EMAIL-ID: anantchaturvedi071531bi@gmail.com

Tarun Pal is currently commencing his final semester BTech studies in Bioinformatics from Jaypee University of Information Technology from Jaypee University of Information Technology and will be completing his degree in June 2011. His area of interest lies in Data Warehouse and Chemoinformatics. He has also presented a poster on ***"In silico design of Vaccine Candidate against Stomach cancer focusing on Helicobacter pylori"*** in International conference held at Punjab University and was awarded for it. He has been selected in Accenture & Infosys in campus placement for the year 2010-11 and will be joining either of the firms in June 2011.

EMAIL-ID: tarunpal33@gmail.com