Project Number: **MY12**

Issues in Integrating Scientific Databases

By

Amira Tokatli-Apollon

Mohamad El-Rifai

An Interactive Qualifying Project submitted to the faculty of Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science Degree of Bachelor of Science

Submitted By:

Mohamad El-Rifai	**Amira Tokatli-Apollon**

October 11, 2012

APPROVED:

Advisor: Prof. Mohamed Y. Eltabakh

1- Abstract

Scientific databases have emerged over the past two decades to store, analyze, and query science data such as in chemistry, biology, medicine, and earth sciences, among others. One of the major issues in scientific applications is that data are typically generated from the collaboration among many scientific labs and groups, which usually results in many challenging issues in integrating such data and managing their conflicts. These challenges represent a major hurdle to scientists that slowdown the progress in these fields. In this project, we study several of these challenges and highlight them through examples. We take biological databases as a test case in our project. We then focus on one major issue, which is "*managing the conflicting data*" and propose recommendations and potential solutions to address this issue.

Table of Contents:

List of Figures

2- INTRODUCTION

2.1 Scientific Databases

Scientists in many fields, e.g., biology, chemistry, physics, medicine and healthcare, astronomy, and earth sciences, have been, for many years, collecting and analyzing data outside the database management systems (DBMSs). This is mostly because scientists used old-fashion methods such as file systems for storing their data, and on top of that most analysis tools have been designed to operate on files. For example, biologists used to store their data in flat files and process those using Perl scripts and other domain-specific tools such as BLAST (Basic Local Alignment Search Tool: http://blast.ncbi.nlm.nih.gov/Blast.cgi).

However, with the scientific discoveries and the rate of collecting and generation data, the storage and analysis of scientific data becomes a major hurdle for scientists? They realized that they are of a great need to more complex systems for managing their data than flat files. That is where the database management systems (DBMSs) come into place (Refer to Figure 2.1). DBMSs offer many unique features such as structured schemas, data consistency, high-level query language, recovery and concurrency mechanisms, indexing techniques, and others. Therefore, over the past two decades, DBMSs have provided eminent support for various scientific applications, and since then ***scientific databases*** have become commonly used by the scientific community.

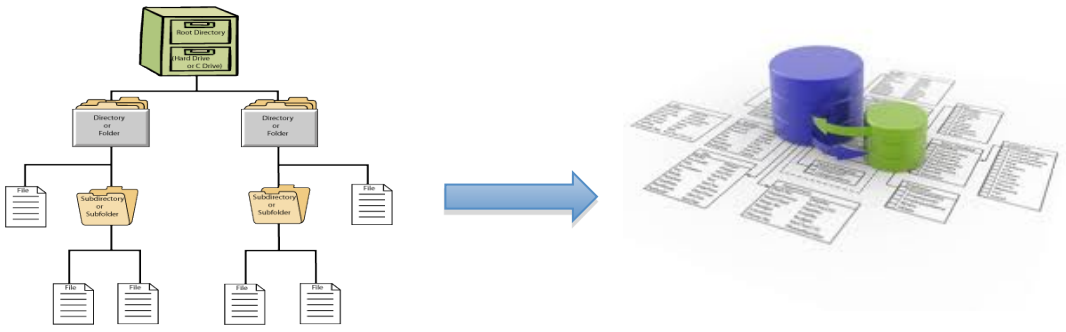


Figure 2.1: Shifting from file system to DBMSs for storing and querying the data.

2.2 Example of Scientific Databases: Biological Databases

In this project, we focus on one type of scientific data, which is biological data. In that domain, biologists need to store and analyze massive amounts of biological data generated from computer stimulations, instruments, and lab in database systems in order to perform their work. That is why scientific databases play significant role in computational biology. There are several hundred biological databases and it is increasing at a staggering rate, which makes it very difficult to keep an accurate track of all of the data collected throughout the years.

Even though many biologists have pen-and-paper method, they found it very hard to keep up using a regular notebook fitting all of the collected data and comparing it with other data. Moreover, with these data accumulating at a high rate, they started to have bigger issues than just collecting the data. Therefore, biology applications were among the first scientific applications to use databases and share the data across the world. Examples of these databases are ***MetaBase***[1], ***DBcat***[2]***,*** and ***GenBank***[3] that are among the biggest biological databases created through collaboration of many research labs. ***DBcat***, for example, is a comprehensive catalog for 500 biological databases, while ***GenBank*** is one of the major databases which contains over four million nucleotide sequences and is considered one of the largest biological databases available for public and for biologist where they can save and share their results.

One of the problems biologists are facing is that the size and growth of biological data collected in these databases become a serious problem. Biologists need various analysis tools to fix data duplication and repetition, missing and wrong values, non-standardized data formats, and conflicts that appear when data from multiple sources are shared and exchanged. Another problem biologists are facing is the lack of knowledge in areas such as computer software, information systems, computer hardware, data management, programming and the differences between query language, and the differences between database schemas.

Typically, the big repositories of databases are used as reference sources and for verifying local, under-developed data. Therefore, Biologists do concentrate in creating smaller more focused database within their research labs. Nevertheless, the success of these databases

[1] MetaBase: http://metadatabase.org/wiki/Help:About
[2] DBcat: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC102454/
[3] GenBank: http://www.ncbi.nlm.nih.gov/genbank/

and the increasing demands of wider international coverage of data collected to be shared create a corresponding pressure for these small and local databases to be integrated and published into the larger repositories, allowing a single query interface too many databases at once. It is very important to integrate and process the data in a shared collaborative environment for better utilization of the data, more accurate results, and faster progress in the scientific domain. This issue is widely known in the information management community as "***Data Integration***". Data integration is basically the process of integrating and sharing data from many sources or sites. Data integration involves many challenges that need to be addressed such as mapping and alignment of data, unifying the structure of the data, resolving mismatches and conflicts, among many others. Scientists always face these issues while sharing and integrating their data. Our work in this project will focus on studying the issues and challenges involved in the data integration of scientific databases.

2.3 Focus and Goals of the Project

In the following, we summarize the goals of our project:

- Study different methods currently available for storing, collecting and sharing data.
- Study several databases that biologist and biomedical researchers currently use.
- Analyzing the collected data and highlighting conflicts found between different datasets.
- Suggesting a better solution to resolve and avoid conflicts.

2.4 Target Audience

Biomedical researchers and biologists are our target in this project. One significant problem they are facing is the conflicting or mismatched data, due to conflicting biological data and unknown problem among researchers that perform similar experiments around the world. It is very important to connect and have the same information correctly processed and shared for better results saving time and money. We also target audience interested in database systems and interested in learning how these systems can serve scientific domains.

2.5 Roadmap

The rest of this report is as follows. In Chapter 3, we give a background discussion on scientific databases and biological concepts. In Chapter 4, we discuss issues related to data integration. In Chapters 5 and 6, we discuss in more detail the datasets we worked on and how it is analyzed. Chapter 7 contains our preliminary ideas on how to address the issue of conflicting data using taxonomy models and database features. In Chapter 8, we conclude our project and findings.

3- Background

3.1 Scientific Data Management Background

A Scientific Data Management System (SDMS) is a specialized information system for electronic record management for most types of analytical data and documentation, which ensures long-term data preservation, accessibility, and recovery. Scientists normally refer to data management as the mere physical data storage and access layer. *Scientific Data Management* is the platform technology where all analytical and related data, including the printed reports generated by instrument scientist use to view the actual contact of the report. SDMS provides the ability for a quicker and easier method to locate and view data with out the software used to create these data for us.

Database is the collection of data that related to each other that organized and useful information may be pulled and used for future studies. The fact that the data are shared regularly prevents the duplication of data collection. Science data center provide the access to both data and the applications that analyze the data these two areas works with each other as specific scientific domain creating massive dataset, giving the user ability to understand and indeed in constantly adding improving the dataset performance.

Scientific data management system has important physical components that have to be present for the data to function properly some of these components are:

- Metadata schema which is the most important component
- The organization of the collected metadata
- Accessing the physical data.
- User data access interface.
- The storage and management for Metadata.
- Workflow description and management.
- Ownership and data lifetime definition.
- Data quality evaluation process.

3.1.1 Challenges Involved in Scientific Databases

Scientific databases involve many technical challenges. One of these challenges results from many different opinions of the same phenomena that are inconsistent, resulting from different methods, measurements, or experts' interpretations. Furthermore, many of the errors found that has already been published are not cleaned or revised. For many scientists once analysis has been published, there is no need to be recreated, but focusing on finding the relation within the given database and link it to others. Another challenge in scientific databases is related to dealing with the diversity, and the complexity of the verity data creates the need of different areas where the scientific data can be managed. Even though there is a wide science agreement using a specific type of techniques and measurements there is still a little to not arrangement on the ways data collected and stored triggering major conflict and no system available linking over all information.

Usually scientists concentrate on creating smaller and more focused database that require a lot of attention and maintenance. They may also store their data in flat files outside the database, which results in many problems including:

- Restricted file and directory naming schemes
- Project data origins are basically big flat directories.
- Researchers retrieve files by discovering and finding out the connection between them without any specific program
- Somewhat the agreement on following a specific techniques and measurement scientist use, but not on the way which data stored neither a specific database system nor application.
- Another important point of the various scientific data is its availability to discover it by browsing that if the data is well described and the flexibility accessing the data, which would allow us as users to establish a relation between data already discovered.

In the table below, we summarize several research areas in scientific database.

Creation of logical collections:	The main purpose of a Data Management system is to abstract the physical data into logical collections.
Physical data handling:	This covers the maps between the physical to the logical data views, which means data reproduction, backup, caching, etc.
Interoperability support:	Normally the data does not exist in in the same place, or different data collection should be put together in the same logical assembly.
Security support:	Data access authorization allowing only the scientists to change his/her data which will provide to trust the actual data and no one else can change it.
Data ownership:	The responsibility for the data quality and explanation.
Metadata collection, management and access:	Are the data talks and explain the current data
Persistence	The lifetime protection for the data against any sudden changes in technology.
Knowledge and information discovery	Capability to pinpoint useful relationships and information within the data collection.
Data dissemination and publication	Method provides parties who are interested in knowing the latest information and any changes in the data collection.

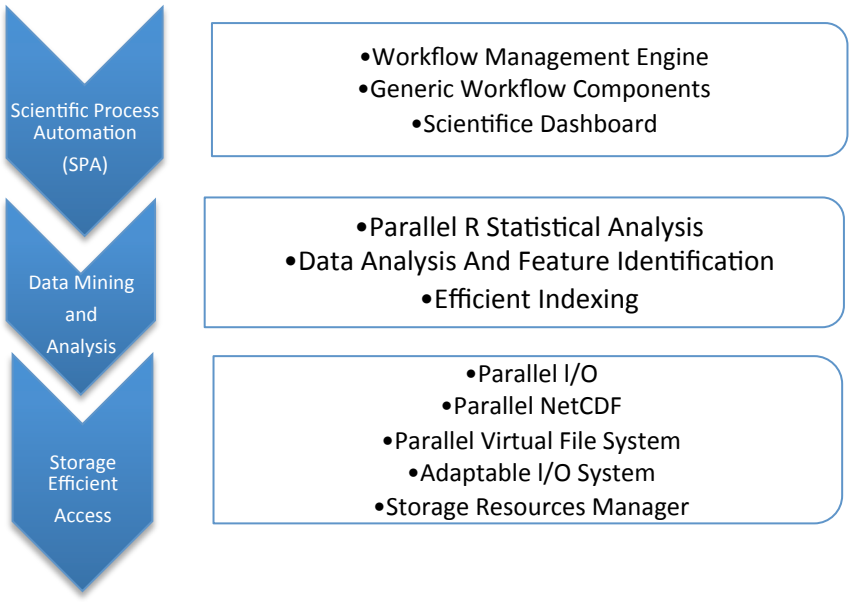
3.1.2 Centers for Storing Scientific Data

In this section, we overview some of the centers created for storing and managing scientific data.

- **Scientific Data Management Center (SciDAC)**

SciDAC is an advanced data management technology most needed in the scientific community helping scientist to better understand the importance of data management and data analytical challenges for their current and future science need because of the increasing volume and the complexity of the collecting data that requires managing, generating and analyzing data starting from initial stage of receiving data to the final analysis of the data. SciDAC improved three most needed areas:

1. Storage system needed where large amount of data can be stored and accessed in timing manner
2. Ability to perform complex data analysis and searches for a large data sets, storing as well as ability to track and locate a specific datasets
3. Generating, collecting and storing data and their results before and after processing.



- **GenBank**

Is the most important and most leading database for research in almost all biological fields where individual laboratories and large centers able to submit directly into the database. GenBank is a sequence database; large collection of digital nucleic acid sequences, protein sequences and other sequences stored on a computer, these sequences could be from only one organism or organisms with a DNA have been sequenced. GenBank is available online with no cost with additional usage rights.

This database is produced and maintained by the *National Center For Biotechnology Information* as part of the *International Nucleotide Sequence Database Collaboration* as well as the *National Institute of Health in the United States*. Direct submission made using web based form called BankIt, upon receiving the submission GenBank staff examine the originality of the data giving them a unique access number performing quality check then releasing the data to the public after verifying the information provided that can be accessed suing the global query cross-database search system or web portal, or could be downloadable by FTP which is a standard network built on the individual server used to transfer files from one host to another using internet, where the data can be controlled.

Bacterial Genome Submission Examples:

Sample FASTA-formatted sequence

- >HTE831 [organism=Oceanobacillus iheyensis] [strain=HTE831]
- actttcaaaaaaatcagcgtaaaaaacatactaatttgggcaaattcccacctgttttta
- gggacatttttctttgaattagagcctcagcagctcgtcattgctgaattttcttgaagt

Sequin table format

- >Feature HTE831
- 1830 2966 gene
  - gene dnaN
  - locus_tag OBB_0002
- 1830 2966 CDS
  - product DNA-directed DNA polymerase III beta chain
  - EC_number 2.7.7.7
  - protein_id gnl|ncbi|OBB_0002
- 3219 3440 gene
  - locus_tag OBB_0003
- 3219 3440 CDS
  - product hypothetical protein
  - protein_id gnl|ncbi|OBB_0003
- 3443 4552 gene
  - gene recF
  - locus_tag OBB_0004
- 3443 4552 CDS
  - product RecF
  - function DNA repair and genetic recombination
  - protein_id gnl|ncbi|OBB_0004
- 5109 7034 gene
  - gene gyrB
  - locus_tag OBB_0006
- 5109 7034 CDS
  - product DNA gyrase subunit B
  - EC_number 5.99.1.3
  - protein_id gnl|ncbi|OBB_0006

GenBank flat file

LOCUS OB_HTE831 3630528 bp DNA circular BCT 11-DEC-2002

DEFINITION Oceanobacillus iheyensis HTE831, complete genome.

ACCESSION

VERSION

KEYWORDS .

SOURCE Oceanobacillus iheyensis HTE831

ORGANISM Oceanobacillus iheyensis HTE831

Bacteria; Firmicutes; Bacillales; Oceanobacillus.

REFERENCE 1 (bases 1 to 3630528)

AUTHORS Takami,H., Takaki,Y. and Uchiyama,I.

TITLE Genome sequence of Oceanobacillus iheyensis isolated from the Iheya Ridge and its unexpected adaptive capabilities to extreme environments

JOURNAL Nucleic Acids Res. 30 (18), 3927-3935 (2002)

PUBMED 12235376

REFERENCE 2 (bases 1 to 3630528)

AUTHORS Takami,H., Takaki,Y. and Chee,G.

TITLE Direct Submission

JOURNAL Submitted (26-DEC-2001) Hideto Takami, Japan Marine Science and Technology Center, Deep-sea Microorganisms Research Group; 2-15 Natsushima-cho, Yokosuka, Kanagawa 237-0061, Japan

FEATURES Location/Qualifiers

source 1..3630528

/organism="Oceanobacillus iheyensis HTE831"

/strain="HTE831"

/db_xref="taxon:221109"

gene 1830..2966

/gene="dnaN"

/locus_tag="OBB_0002"

CDS 1830..2966

/gene="dnaN"

/locus_tag="OBB_0002"

/EC_number="2.7.7.7"

/codon_start=1

/transl_table=11

/product="DNA-directed DNA polymerase III beta chain"

/translation="MRFTIQRDKLINGVSNVMKAISARTVIPILTGMKIEVKNHGVTL

TGSDSDISIEYYIPIEEDGIVHVENIEEGTIILQAKYFPDIVRKLPESTVDIVVDDQL

NVRITSGKAEFNLNGQSAEEYPQLPKVQTENSFELPIDLLKSMIKQTVFAVSTMETRP

ILTGVNLKLVDNSLSFTATDSHRLARREIPVSNAPIEISQIVVPGKSLNELNKILGDS

EETVEISVTNNQILFRTKHLNFLSRLLDGNYPETSRLIPEQSKTKIQLKTKELLGTID

RASLLAKEERNNVVKFNAPGNSMIEISSNSPEVGNVVEEITADQMEGEDVKISFSSKY

MIDALKAIEYDEVQIEFTGAMRPFIIRPVGDDSILQLILPVRTY"

operon 91493..96462

/operon="rrnA"

gene 91493..93058

/gene="rrsA"

/locus_tag="OBB_0089"

/operon="rrnA"

rRNA 91493..93058

/gene="rrsA"

/locus_tag="OBB_0089"

/operon="rrnA"

/product="16S ribosomal RNA"

gene 93292..96213

/gene="rrlA"

/locus_tag="OBB_0090"

/operon="rrnA"

rRNA 93292..96213

/gene="rrlA"

/locus_tag="OBB_0090"

/operon="rrn"

- **PortEco**

Portal for E-coli research type K-12 strains and their phage and mobile elements, launches 14 different web resources for E. coli information and organizing the result that targets to:

- Facilitate access to E. coli information that is distributed over the web
- Make E. coli genomics data easy to access, search and analyze.
- Enable the community to add information to the knowledgebase
- Provide community features such as a calendar, colleague search, blog entries mentioning E. coli; and educational materials

Protein Data Bank:

Protein Data Bank is the single widespread storehouse for the processing and distribution of 3D biological macromolecular structure data of large biological molecules such as proteins and nucleic acid. The data, usually achieved by X-ray crystallography or NMR spectroscopy and submitted by biologists and biochemists from around the world, are easily available on the Internet via the websites of its member associations as well as these database is updated weekly.

As of April third of 2012 the breakdown of the current holding is:

Experimental Method	**Proteins**	**Nucleic Acids**	**Protein/Nucleic Acid complexes**	**Other**	**Total**
X-ray diffraction	65950	1346	3261	2	70559
NMR	8185	979	186	7	9357
Electron microscopy	284	22	116	0	422
Hybrid	44	3	2	1	50
Other	140	4	5	13	162
Total:	74603	2354	3570	23	**80550**

Protein Data Bank file describing a protein consists of hundreds to thousands of lines

```
HEADER    EXTRACELLULAR MATRIX                  22-JAN-98   1A3I
TITLE     X-RAY CRYSTALLOGRAPHIC DETERMINATION OF A COLLAGEN-LIKE
TITLE    2 PEPTIDE WITH THE REPEATING SEQUENCE (PRO-PRO-GLY)
...
EXPDTA    X-RAY DIFFRACTION
AUTHOR    R.Z.KRAMER, L.VITAGLIANO, J.BELLA, R.BERISIO, L.MAZZARELLA,
AUTHOR   2 B.BRODSKY,A.ZAGARI,H.M.BERMAN
...
REMARK 350 BIOMOLECULE: 1
REMARK 350 APPLY THE FOLLOWING TO CHAINS: A, B, C
REMARK 350   BIOMT1   1  1.000000  0.000000  0.000000        0.00000
REMARK 350   BIOMT2   1  0.000000  1.000000  0.000000        0.00000
...
SEQRES   1 A    9  PRO PRO GLY PRO PRO GLY PRO PRO GLY
SEQRES   1 B    6  PRO PRO GLY PRO PRO GLY
SEQRES   1 C    6  PRO PRO GLY PRO PRO GLY
...
ATOM      1  N   PRO A   1       8.316  21.206  21.530  1.00 17.44           N
ATOM      2  CA  PRO A   1       7.608  20.729  20.336  1.00 17.44           C
ATOM      3  C   PRO A   1       8.487  20.707  19.092  1.00 17.44           C
ATOM      4  O   PRO A   1       9.466  21.457  19.005  1.00 17.44           O
ATOM      5  CB  PRO A   1       6.460  21.723  20.211  1.00 22.26           C
...
HETATM  130  C   ACY   401       3.682  22.541  11.236  1.00 21.19           C
HETATM  131  O   ACY   401       2.807  23.097  10.553  1.00 21.19           O
HETATM  132  OXT ACY   401       4.306  23.101  12.291  1.00 21.19           O
...
```

HEADER, TITLE and AUTHOR records	Provide information about the researchers who defined the structure
REMARK records	Contain free-form annotation, but they also accommodate standardized information
SEQRES records	Give the sequences of the three-peptide chains (named A, B and C), which are very short in this example but usually span multiple lines.
ATOM records	Describe the coordinates of the atoms that are part of the protein. For example, the first ATOM line above describes the alpha-N atom of the first residue of peptide chain A, which is a proline residue; the first three floating point numbers are its x, y and z coordinates and are in units of Ångströms.[1] The next three columns are the occupancy, temperature factor, and the element name, respectively.
HETATM records	Describe coordinates of hetero-atoms, that is those atoms which are not part of the protein molecule.

3.1.3 Scientific Data Formats

In this section, we present some of the common formats for storing scientific data.

- ***CDF—Common Data Format***
  - Self-describing data format for the storage of scalar and multidimensional data in a platform- and discipline-independent way
  - Scientific data management package (CDF Library) allows application developers to manage these data arrays
  - Transparent access to data and metadata through Application Programming Interfaces (APIs)
  - Built-in support for data compression and automatic data un-compression
  - Large file support (> 2GBytes)
  - CDF library includes a suite of tools that allow users to manipulate CDF files

- ***NetCDF—Network Common Data Format (.nc)***
  - NetCDF data file format.
  - Self-describing format for exchanging scientific data.
  - Used in atmospheric research, GIS, and related fields.
  - NetCDF is an acronym derived from network Common Data Form.
  - Binary file format.
  - Conceptually based on NASA's Common Data Format, but incompatible with this format.
  - Developed by the Unidata center at the University Corporation for Atmospheric Research (UCAR).
  -
- ***NASACDF***
  - CDF data file format.
  - General-purpose, self-describing format for storing multidimensional datasets.
  - Used for storage, management, and exchange of scientific data and images.
  - CDF is an acronym for Common Data Format.
  - Developed since 1985 by the National Space Science Data Center at NASA.
  - Binary file format.
  - Related to, but incompatible with, netCDF.
- ***GenBank (.gb, .gbk)***
  - MIME type: chemical/seq-na-genbank
  - GenBank molecular biology format.
  - Native format of the U.S. National Center for Biotechnology Information (NCBI) database.
  - Standard format for storing and exchanging annotated DNA sequences.
  - Plain text format.
  - Developed in 1982 as part of the NIH GenBank project.

- ***HDF5 (.h5)***
  - HDF data format Version 5.
  - General purpose format for representing multidimensional datasets and images.
  - Used for storage, management, and exchange of scientific data.
  - HDF is an acronym for Hierarchical Data Format.
  - Developed by the U.S. National Center for Supercomputing Applications (NCSA).
  - Binary file format.
  - Incompatible with HDF Version 4 and earlier.

3.1.4 Metadata

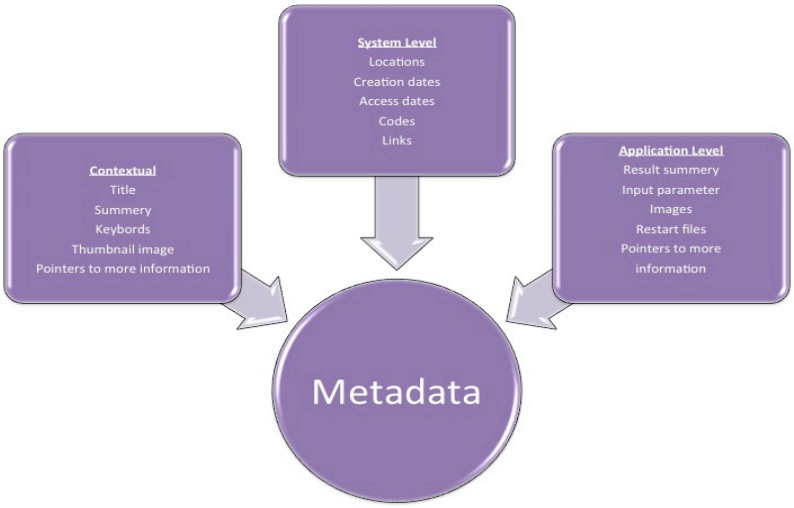
Metadata is structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Metadata is often called data about data or information about information. The term *metadata* is used differently in different communities. Some use it to refer to machine understandable information, while others use it only for records that describe electronic resources. In the library environment, metadata is commonly used for any formal scheme of resource description, applying to any type of object, digital or non-digital. Metadata can describe resources at any level of aggregation. Metadata can help in many tasks for better understanding the data. Examples of these tasks are:

1- ***Resource Discovery:*** Permitting resources to be found by related reasons. As the number of Web-based resources grows rapidly, comprehensive sites or portals are progressively useful in organizing links to resources based on audience or topic.

2- ***Interoperability:*** Describing a resource with metadata allows it to be understood by both humans and machines in ways that promote interoperability. Using defined metadata schemes, shared transfer protocols, and crosswalks between schemes, resources across the network can be searched more flawlessly.

3- ***Digital Identification:*** Most metadata schemes include elements such as standard numbers to individually identify the work or object to which the metadata refers. The actual elements that point to the object, the metadata can be combined to act as a set of identifying data, differentiating one object from another for validation purposes.

4- ***Archiving and Preservation:*** Metadata efforts center on the discovery of recently created resources. However, there is a growing concern that digital resources will not outlast in usable form into the future. Digital information is fragile; it can be corrupted or altered, intentionally or unintentionally. Metadata is the key to confirming that resources will survive and continue to be accessible into the future.



3.2 Biological Science Background

Since our focus in the remaining of this report will be on biological data, we will give in this section and brief background on biological science and define the main terms in biology.

- **Chromosomes Mechanism**

Every chromosome contains a single molecule of DNA, snakelike carrier of hereditary information. If stretched out to its full length, the DNA molecule in a human chromosome would be between 1.7 and 8.5 centimeters long, depending on the chromosome. But it would be vanishingly thin, less than a millionth of a centimeter across. A chromosome quite a complex structure, with the DNA molecule wound around protein spools and fastened into loops, coils, and fibers by other proteins. In a chromosome, protein is the packaging and DNA is the contents of the package. In its most tightly packaged or "condensed" form, a chromosome, which contains several centimeters of DNA, is only a few ten-thousandths of a centimeter long. Generally,

however, chromosomes are fully condensed only in preparation for cell division. The rest of the time, some of the loops and coils are unfastened so that the DNA can do its work, communicating hereditary instructions to the rest of the cell.

- **Genes**

A gene is a small piece of the genome. It's the genetic equivalent of the atom: As an atom is the fundamental unit of matter, a gene is the fundamental unit of heredity. Genes are found on chromosomes and are made of DNA. Different genes determine the different characteristics, or traits, of an organism. In the simplest terms one gene might determine the color of a bird's feathers, while another gene would determine the shape of its beak. The number of genes in the genome varies from species to species. More complex organisms tend to have more genes. Bacteria have several hundred to several thousand genes. Estimates of the number of human genes, by contrast, range from 25,000 to 30,000.

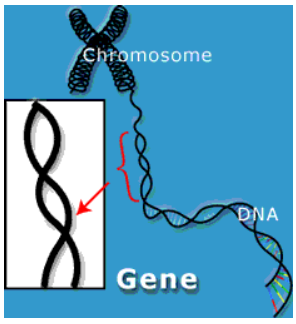


Figure 3.1: Gene Structure

What Do Genes Do?

Genes tell a cell how to make proteins. Roughly speaking, each gene is a set of instructions for making one specific protein. Proteins are a diverse group of large, complex molecules that are crucial to every aspect of the body's structure and function. Collagen, which forms the structural scaffolding of skin and many other tissues, is a protein. Insulin, a hormone that regulates blood sugar, is a protein. Trypsin, an enzyme involved in digestion, is a protein. So is the pigment melanin, which gives hair and skin its color. Still other proteins regulate the body's production of proteins.

What Do Genes Look Like?

A gene has several parts. In most genes, the protein-making instructions are broken up into relatively short sections called exons. These are interspersed with introns, longer sections of "extra" or "nonsense" DNA. Genes also contain regulatory sequences, which help determine where, when, and in what amount proteins are made.

These sequences are crucial to how your body works. They help determine which genes are "turned on," or transmitting their protein-making instructions to the rest of the cell, in different cells throughout the body.

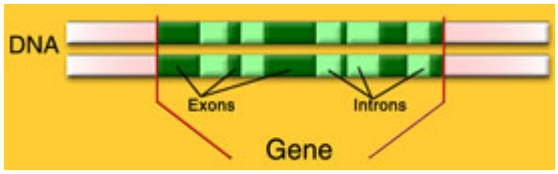


Figure 3.2: Gene Sequence

All your cells contain the same genes, but cells don't make all the proteins they have genes for. A schematic drawing of a gene including these features looks like a ribbon divided into segments. But like many schematic drawings, this one is quite different from the physical reality of a gene. Actually, a gene is rather nondescript from a physical point of view. It basically looks like any other piece of DNA.

- **DNA**

DNA is the molecule that is the hereditary material in all living cells. Genes are made of DNA, and so is the genome itself. A gene consists of enough DNA to code for one protein, and a genome is simply the sum total of an organism's DNA. DNA is long and skinny, capable of contorting like a circus performer when it winds into chromosomes. DNA contains information necessary to build a living organism.

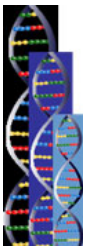


Figure 3.3: DNA

What is DNA made of?

DNA is a very large molecule, made up of smaller units called nucleotides that are strung together in a row, making a DNA molecule thousands of times longer than it is wide. Each nucleotide has three parts: a sugar molecule, a phosphate molecule, and a structure called a nitrogenous base. The nitrogenous base is the part of the nucleotide that carries genetic information, so the words "nucleotide" and "base" are often used interchangeably. The bases found in DNA come in four varieties: adenine, cytosine, guanine, and thymine—often abbreviated as A, C, G, and T, the letters of the genetic alphabet.

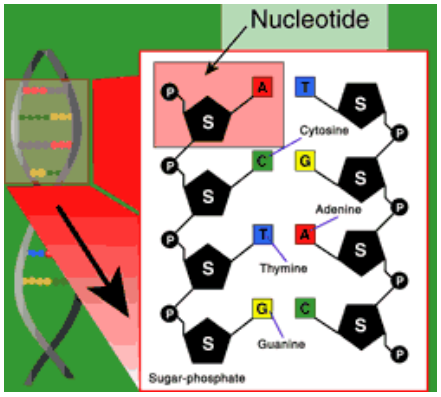
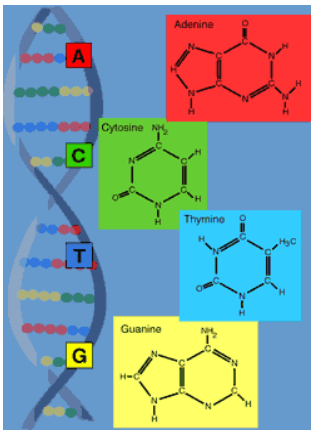


Figure 3.4: DNA Molecule Structure

What Does DNA Look Like?



A DNA molecule is a double helix, a structure that looks much like a ladder twisted into a spiral. The sides of the ladder are made of alternating sugar and phosphate molecules, the sugar of one nucleotide linked to the phosphate of the next. DNA is often said to have a sugar and phosphate "backbone."

In other words, the order of bases on one DNA strand, or side of the ladder, determines the bases on the other side of the ladder. Thus, DNA sequences are often written as if DNA were only single-stranded: AGTCTGGAT…. Scientists need sequence only one DNA strand in order to know the sequence of both strands.

AGTCCGCGAATACAGGCTCGGT

Figure 3.5: Example of DNA Sequence

How Are New DNA Molecules Made?

New DNA molecules are made by copying, using old DNA molecules as a template.

When a cell needs to copy a DNA molecule, it "unzips" part of the double helix, breaking the rungs of the ladder in half so that the molecule separates down the middle. New nucleotides, floating free in the cell, can then hook up with complementary nucleotides along each strand. Gradually the unzipping proceeds and the new strands continue to grow until one DNA molecule becomes two identical DNA molecules. A cell copies its entire DNA in this fashion each time it divides. In the cells of complex organisms such as humans, this process takes an average of 8 hours. In other words, each human cell can read and reproduce the entire genome sequence in one working day.

How Does DNA Tell A Cell About Making Proteins?

DNA tells a cell how to make proteins through the genetic code. Both DNA and proteins are long molecules made from strings of shorter building blocks. While DNA is made of nucleotides, proteins are made of amino acids, a group of 20 different chemicals with names like alanine, arginine, and serine. The genetic code enables a cell to translate the nucleotide language

of DNA into the amino acid language of proteins. In the genetic code, each group of three nucleotides—known as a "triplet" or "codon"—stands for a specific amino acid. For example, GCA stands for alanine, AGA stands for arginine, and AGC stands for serine. There are 64 possible codons, but only 20 amino acids, so more than one codon may code for a single amino acid. For example, GCA, GCC, and GCG all mean alanine. For the most part, the genetic code is the same across every form of life, from bacteria to sea stars to German shepherds to humans.

How Much DNA Is In A Gene? How Much Is In A Genome?

Both genes and genomes come in a variety of sizes. About 1,000 base pairs would be enough DNA to encode most proteins. But introns—"extra" or "nonsense" sequences inside genes—make many genes longer than that. Human genes are commonly around 27,000 base pairs long, and some are up to 2 million base pairs. Very simple organisms tend to have relatively small genomes. The smallest genomes, belonging to primitive, single-celled organisms, contain just over half a million base pairs of DNA. But among multicellular species, the size of the genome does not correlate well with the complexity of the organism. The human genome contains 3 billion base pairs of DNA, about the same amount as frogs and sharks. But other genomes are much larger. A newt genome has about 15 billion base pairs of DNA, and a lily genome has almost 100 billion.

- **Genome Sequencing**

Genome sequencing is figuring out the order of DNA nucleotides, or bases, in a genome—the order of As, Cs, Gs, and Ts that make up an organism's DNA. The human genome is made up of over 3 billion of these genetic letters. A DNA sequence that has been translated from life's chemical alphabet into our alphabet of written letters might look like this:

That is, in this particular piece of DNA, an adenine (A) is followed by a guanine (G), which is followed by a thymine (T), which in turn is followed by a cytosine (C), another cytosine (C), and so on.

What is Genome Sequencing?

Genome sequencing is often compared to "decoding," but a sequence is still very much in code. In a sense, a genome sequence is simply a very long string of letters in a mysterious language.

Why Is Genome Sequencing Important?

Sequencing the genome is an important step towards understanding it. At the very least, the genome sequence will represent a valuable shortcut, helping scientists find genes much more easily and quickly. A genome sequence does contain some clues about where genes are. Genes account for less than 25 percent of the DNA in the genome, and so knowing the entire genome sequence will help scientists study the parts of the genome outside the genes.

What Makes Sequencing The Human Genome Different From Sequencing Other Genomes?

The human genome is a lot bigger than other genomes that have been sequenced in the past. Most genomes that have been sequenced to date belong to viruses, bacteria, or other simple forms of life with relatively small genomes. The human genome is about a thousand times larger than an average bacterial genome. Even the fruit fly genome, the largest genome sequenced prior to the human genome, is just 165 million base pairs—less than a tenth the size of the human genome.

In addition, the human genome is about 25 to 50 percent repetitive DNA, but bacterial and viral genomes contain very little of this exasperating stuff. In repetitive DNA, the same short sequence is repeated over and over again. Repetitive DNA may also be more difficult to sequence than other DNA. Sometimes the procedures used to copy DNA and prepare it for sequencing do not work on repetitive DNA, and a sequencing machine may have a hard time reading the same string of letters over and over.

Genome Variations:

Genome variations are differences in the sequence of DNA from one person to the next. Just as you can look at two people and tell that they are different, you could, with the proper chemicals and laboratory equipment, look at the genomes of two people and tell that they are different, too. In fact, people are unique in large part because their genomes are unique.

How Different Is One Human Genome From Another?

The more closely related two people are, the more similar their genomes. Scientists estimate that the genomes of non-related people—any two people plucked at random off the street—differ at about 1 in every 1,200 to 1,500 DNA bases, or "letters." Whether that's a little or a lot of variation depends on your perspective. There are more than three million differences between your genome and anyone else's. On the other hand, we are all 99.9 percent the same, DNA-wise. Most genome variations are relatively small and simple, involving only a few bases—an A substituted for a T here, a G left out there, a short sequence such as CT added somewhere else, for example. Your genome probably doesn't contain long stretches of DNA that someone else's lacks.

Why Is Every Human Genome Different?

Every human genome is different because of mutations—"mistakes" that occur occasionally in a DNA sequence. When a cell divides in two, it makes a copy of its genome, and then parcels out one copy to each of the two new cells. When a mutation occurs in a sex cell—a sperm or an egg—it can be passed along to the next generation of people. Your genome contains about 100 "new" mutations—changes that occurred as your parents' bodies made the egg and sperm cells that became you. Other variations in our genome arose many generations ago and have been passed down from parent to child over the years, until they ended up in you. We are probably share each one of these older variations with many other people all over the world, but still, no one else has the exact same combination of variations that you have.

Where Are Genome Variations Found?

Variations are found all throughout the genome, on every one of the 46 human chromosomes. But this variation is by no means distributed evenly: It's not as if there is one difference every 1,000 bases as regular as rain. Instead, some parts of the genome are "hot spots" of variability, with hundreds of possible variations of a sequence. Other parts of the genome, meanwhile, don't vary much at all between individuals—in scientific parlance, they are said to be "stable.". The majority of variations are found outside of genes, in the "extra" or "junk" DNA that does not affect a person's characteristics. Mutations in these parts of the genome are never harmful, so variations can accumulate without causing any problems. Genes, by contrast, tend to be stable because mutations that occur in genes are often harmful to an individual, and thus less likely to be passed on.

What Kinds Of Genome Variations Are There?

Genome variations include mutations and polymorphisms. Technically, a polymorphism (a term that comes from the Greek words "poly," or "many," and "morphe," or "form") is a DNA variation in which each possible sequence is present in at least 1 percent of people. For example, a place in the genome where 93 percent of people have a T and the remaining 7 percent have an A is a polymorphism. If one of the possible sequences is present in less than 1 percent of people, then the variation is called a mutation.

Informally, the term mutation is often used to refer to a harmful genome variation that is associated with a specific human disease, while the word polymorphism implies a variation that is neither harmful nor beneficial. However, scientists are now learning that many polymorphisms actually do affect a person's characteristics, though in more complex and sometimes unexpected ways.

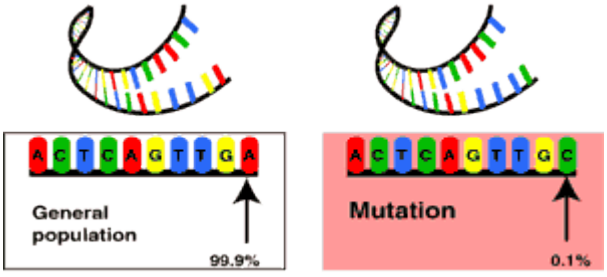


Figure 3.6 : Genome Mutation

About 90 percent of human genome variation comes in the form of single nucleotide polymorphisms, or SNPs (pronounced "snips"). Any one of the four DNA bases may be substituted for any other—an A instead of a T, a T instead of a C, a G instead of an A, and so on. Theoretically, a SNP could have four possible forms, or alleles, since there are four types of bases in DNA. But in reality, most SNPs have only two alleles.

Neil Campbell., Jane Reece., (2008) From Gene to Protein. Powerpoint presentation. Biology Eighth Edition.

4- Issues in Integrating Scientific Databases

Data Integration is the process of integrating data from different sources with the purpose of creating a single over-all view, using the combined information to answer queries. There are two kinds of integrating data; ***Physical***, which means copying data to a warehouse, and ***Virtual*** which means keeping the data at the original sources while retrieving the answer from each source at query time.

4.1 Overview on Data Warehouse

It is a repository gathering data from a variety of data sources and providing integrated information for Decision Support Systems of an enterprise. The data sources provide the original data to the warehouse. A data warehouse may integrate data from multiple autonomous and heterogeneous data sources, which could be either remote or local, and not under the control of the data warehouse users and administrators. A data warehouse has the following characteristics:

- **Subject oriented:** data warehouse only includes the data that will be used for the organization's Decision Support System (DSS) processes.
- **Integrated:** Data warehouses collect data from multiple data sources, which may be distributed, heterogeneous and autonomous.
- **Nonvolatile:** The warehouse data are normally long-term, not updated in real-time and just refreshed periodically.
- **Time Variant:** Information from one past time point to the present may be contained in the data warehouse.

Data warehouse consist of three components:

- **Detailed Data:** it is the lowest level of source information necessary for supporting the DSS processes.
- **Summarized Data:** The summarized data is derived from the detailed data, in order to allow faster processing of specific DSS functionality.
- **Metadata:** A data warehouse not only provides integrated data, but also pro- vides information about the content and context of the data.

4.2 Integrating Heterogeneous Data

Integration of multiple information systems generally aims at combining selected systems so that they form a unified new whole and give users the illusion of interacting with one single information system. There are many reasons for data heterogeneity among the different sources. The following figure summarizes these reasons.

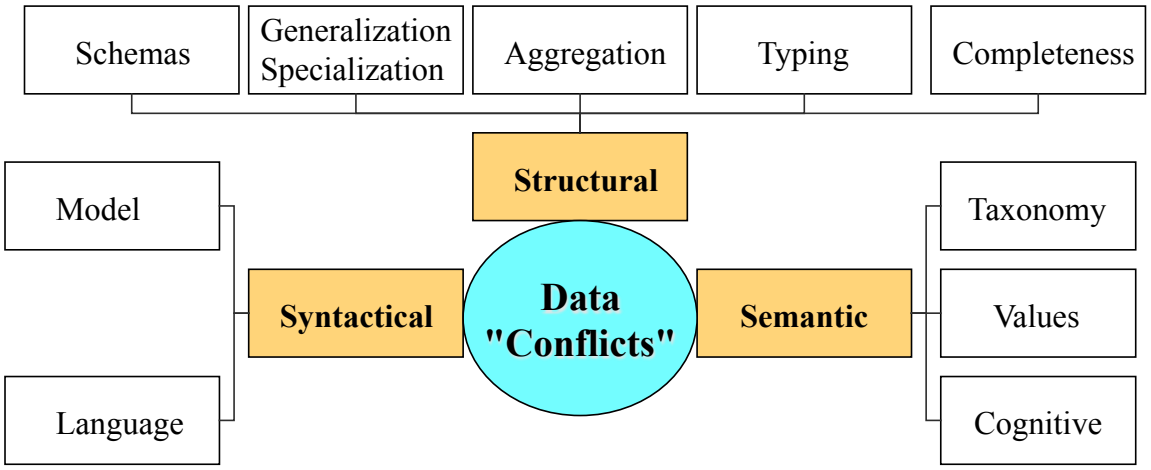
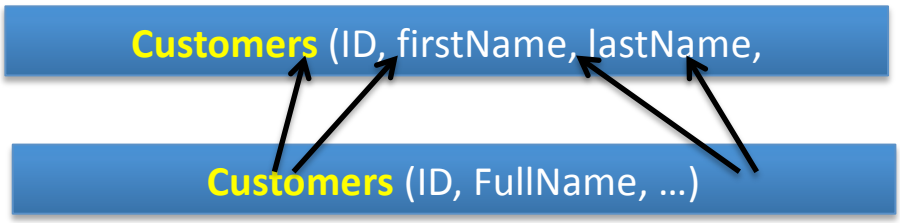


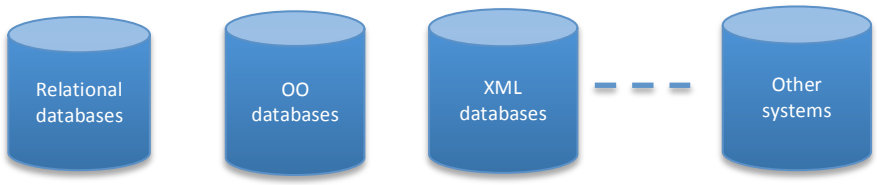
Figure 4.1: Reasons of Data Heterogeneity.

As we notice from the figure, the reasons for data heterogeneity can be classified into three categories: ***Structural, Syntactical***, and ***Semantic***.

- ***Example of Structural Heterogeneity:*** An example of the Structural heterogeneity due to different schemas can be illustrated in the following figure. The structure of the tables storing the data can be different (even if storing the same data). For example, in one table the customer name is divided into two fields; *firstName* and *lastName*, while in the second table, the name is stored in one field, *FullName*.



- ***Example of Syntactical Heterogeneity:*** An example of syntactical heterogeneity is that data can be stored in different systems such as relational databases, object-oriented database, XML databases, or even flat files.



- ***Example of Semantic Heterogeneity:*** An example of semantic heterogeneity is that same logical values stored in different ways such as the following examples:

E.g., 'Prof', 'Prof.', 'Professor'

E.g., 'Right', 'R', '1' …… 'Left', 'L', '-1'

Or same values in different sources can mean different things. For example, Column 'Title' in one database means 'Job Title' while in another database it means 'Person Title'.

4.3 Heterogeneity Problems

As highlighted above heterogeneity results in many problems that can be classified as:

Communication Heterogeneity: It's a problem due to factors such as differences in structures, semantics of data; the constraints supported or query language. Differences in structure occur when two data models provide different primitives such as object-oriented models that support specialization and inheritance and relational models that do not.

Schema Heterogeneity: Dealing with incompatible data types or query syntax. The structure of tables storing the data can be different

Semantic heterogeneity: Data across constituent databases may be related but different. Some values in different sources may mean different things.

Data Type heterogeneity: Is storing the same data using different data types

Value Heterogeneity: Same logical values stored in different ways

4.4 Models of Data Integration

There are three main models for data integration, which we overview in this section. These models are: ***Federated databases***, ***Warehousing***, and ***Mediation***.

Federated Database: is a type of meta-database management system (DBMS), which transparently maps multiple autonomous database systems into a single federated database. It is a simplest architecture where every pair of sources can build their own mapping and transformation, so if source X needs to communicate with source Y they have to build a mapping between X and Y called *Wrapper*. The following figure shows the structure of federated systems.

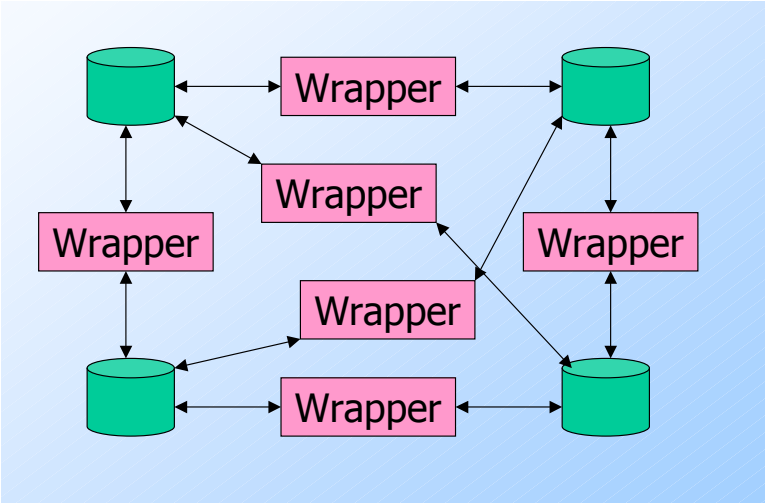


Figure 4.2: Federated Databases Diagram.

Data Warehouse: A data warehouse is a relational database that is designed for query and analysis rather than for transaction processing. It usually contains historical data derived from transaction data, but it can include data copied and stored from other sources. It separates analysis workload from transaction workload and enables an organization to consolidate data from several sources. Users can only query the warehouse database and not insert new records or alter existing ones. The figure below shows the structure of a warehouse system, where the data have to go through an ETL (Extract-Load-Transform) component first in order to be unified before inserting the data into the warehouse.

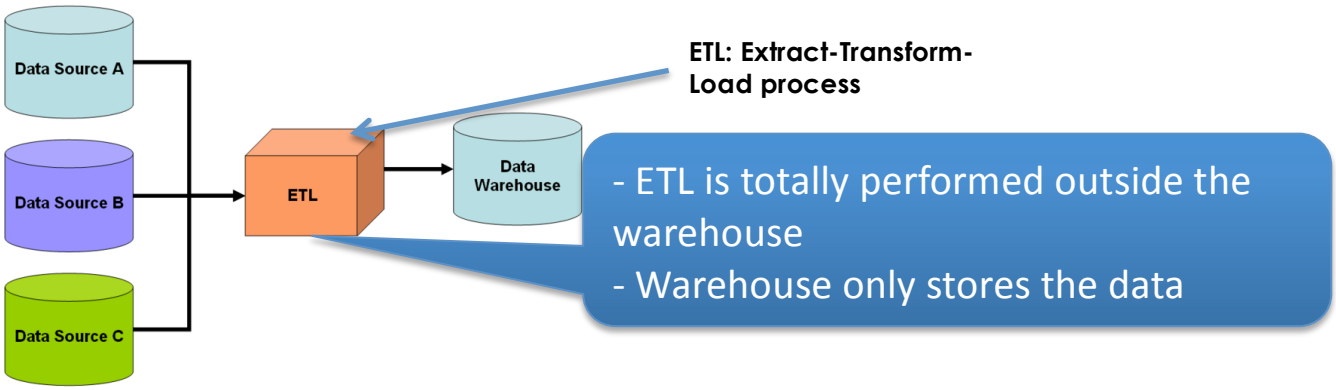


Figure 4.3: Warehouse Database Diagram.

Mediation: Mediator is a virtual view over the data (it does not store any data). Data is stored only at the source. Mediator has a virtual schema that combines all schemas from the sources. The mapping takes place at query time and this is unlike warehousing where mapping takes place at upload time. The following figure shows the structure of the mediation systems.

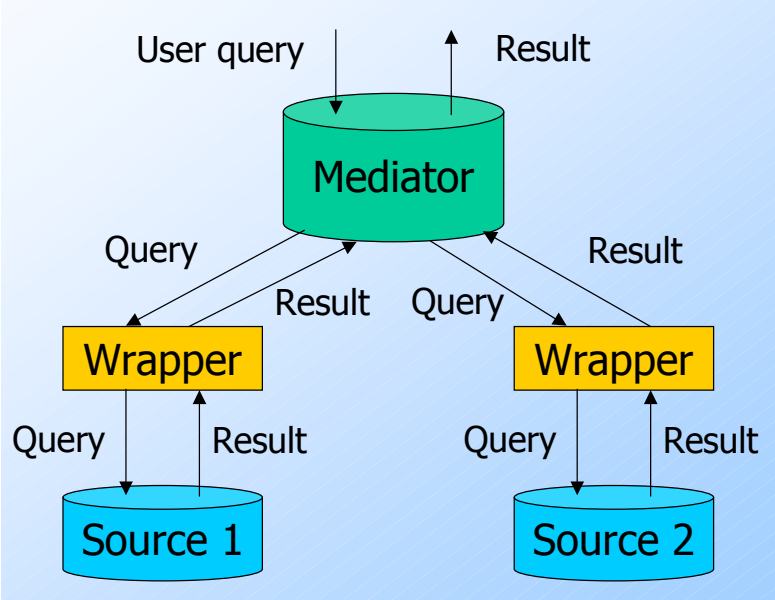


Figure 4.4: Mediation Mechanism Diagram.

In a mediation system, a given user's query is evaluated as follows:

- User Query is mapped to multiple other queries
- Each query (or set of queries) are sent to the sources
- Sources evaluate the queries and return the results
- Results are merged (combined) together and passed to the end-user

5- Data Collection

5.1 Preparation and Collection Process

In this section, we describe the sources from which we collect the data needed for the project. We also present how we massage and transform the data from its original format to fit our local database from which we will analyze the data.

Intuition and Motivation*:* One of the main reasons that lead to data conflicts is that multiple scientific groups or labs are working on the same set of objects. Each group or lab will have their own methods, techniques, and instruments for generating and collecting the data. Hence, as the number of these isolated groups increases, the changes for having significant conflicts among the data values also increases---especially if the datasets has many attributes, some of them are standard and others are not. For example, conflicts may arise because of human errors, missing data, use of different precision instruments or algorithms. Therefore, we extensively searched for scientific datasets that have the properties described above.

Searching Process (Internet and Local Companies): Given the intuition mentioned above, we started our search on the Internet looking for at least two or three good sources of data serving our needs. At the same time, we contacted several local companies working on biological and biomedical data analysis such as Abbott[4] and UMASS Medical School[5] in order to collect potential datasets for our project. We scheduled meetings with their data analysis groups to describe our project and get feedback from them and possible collaboration. The feedback we got from the meetings was very positive as we found that both labs face the issue of integrating data from many sources and they always have to deal with conflicting data. From our discussion with them, it turned out that they mostly resolve these conflicts manually, which is both a time-consuming process and error prune. However, despite the interest they showed in the project, they could not share data with our group due to privacy restrictions and the companies' policies. On the other side, our search on the Internet was successful and we managed to found very good data sources that serve our needs.

[4] Abbott Laboratories health care company: http://www.abbott.com/index.htm
[5] UMASS Medical School: http://www.umassmed.edu/research/index.aspx

5.2 Data Sources

***Data Source I (PortEco)*:** PortEco (the new face of EcoliHub), which is *the next-generation resource for knowledge and data about the biology of Escherichia coli K-12 group strains*. PortEco search mechanism allow you to search results from more that 15 different E.coli data resources, which gave us the opportunity to search through the website and found variety of data resources about genes, proteins, enzymatic reactions, etc. The official website for the PortEco system can be found at: http://www.prfect.org. The PortEco system is supported by NIH and is used for both educational and research purposes. Figure 5.1 shows possible search mechanisms from PortEco website where we can search the raw data of genes or search experimental data and publications.

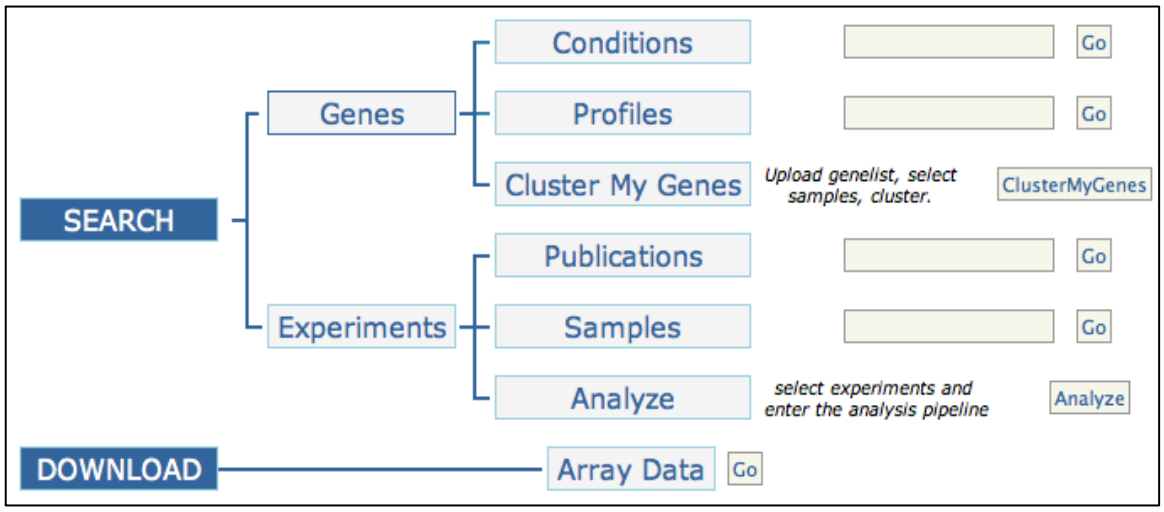


Figure 5.1: Search interface for ProEco system.

Figure 5.2, on the other hand, illustrates the possible datasets that we can search or download. As shown in the figure, the closeness of the origin of the datasets is also maintained in the system.

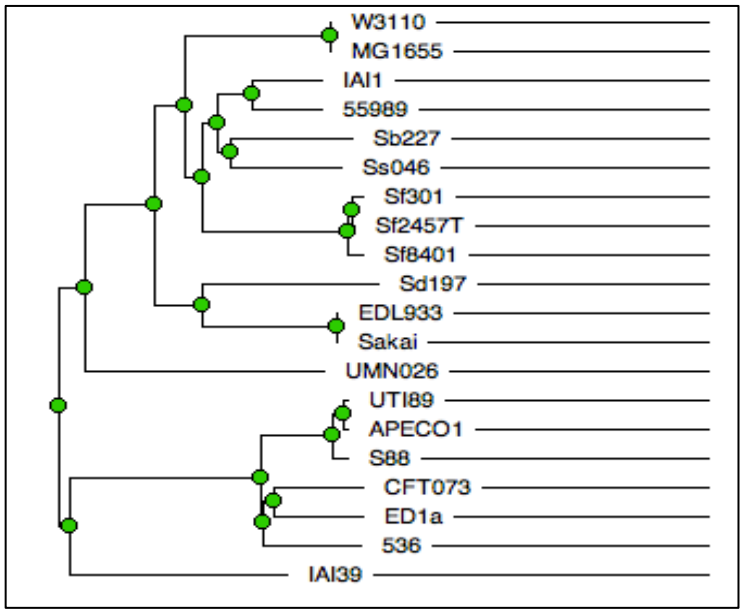


Figure 5.2: Datasets available in PortEco system.

Example of the data that can be retrieved from PortEco system is shown in Figure 5.3. The figure shows four gene information coming from one dataset (with ID = 13).

WID	Name	NucleicAcidWID	SubsequenceWID	Type	GenomeID	CodingRegionStart	CodingRegionEnd	Direction	Interrupted	DataSetWID
5136484	yhcC	5133007	5133008	polypeptide	b3211	3351143	3352072	reverse	NULL	13
5136487	gltF	5133007	5133008	polypeptide	b3214	3359198	3359962	forward	NULL	13
5136495	nanE	5133007	5133008	polypeptide	b3223	3368369	3369058	reverse	NULL	13
5136496	nanT	5133007	5133008	polypeptide	b3224	3369106	3370596	reverse	NULL	13

Figure 5.3: Example data from PortEco system.

Data Source II (GenoBase): GenoBase website which has been created as a major resource of high-throughput data being collected to understand comprehensively the living E.coli K-12 model cell. GenoBase is linked also to PortEco, but the nice thing with this website is the ability to issue SQL queries from the website (as shown in Figure 5.4) and to get back a nice graphical presentation for genes data. This was very helpful to our group as it allowed us to compare results between structured results containing data about the same objects. In order to find the data conflict between these tables. GenoBase Key Search method displays two rows for each gene: one row shows data for the E. coli K-12 MG1655 genome; the other shows data for the E. coli K-12 W3110 genome. The website of the GenoBase system can be found at: http://ecoli.naist.jp/GB8/index.jsp.

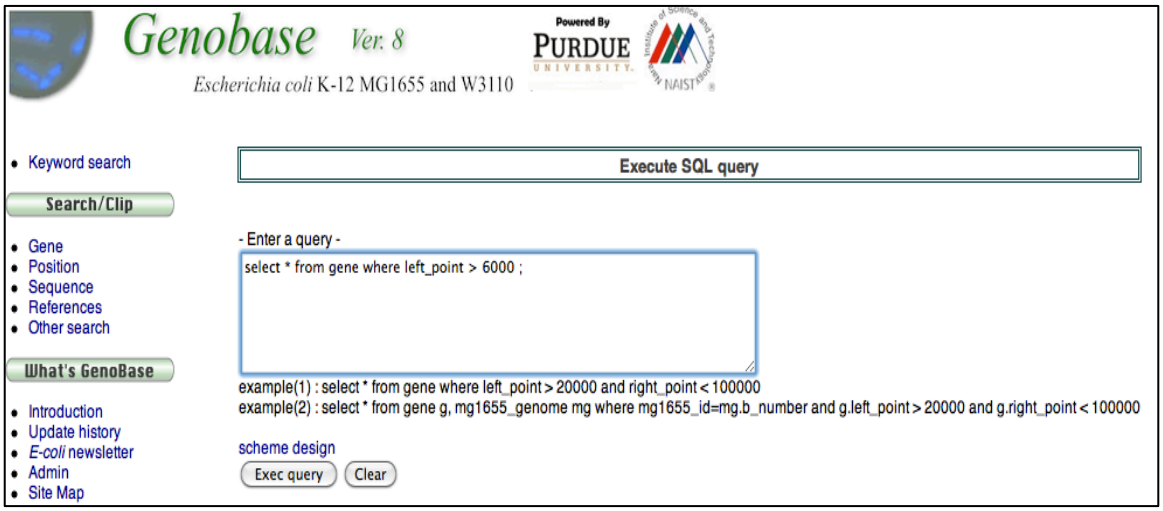


Figure 5.4: SQL search interface for GenoBase.

An example of an SQL query results is depicted in Figure 5.5.

id	b_number	eck_number	jw_number	feature	gene_name	synonym	direction	left_point	right_point	comment
3158	b3211	ECK3201	JW3178	CDS	yhcC		-1	3351143	3352072	(no change)
3161	b3214	ECK3204	JW3181	CDS	gltF	ossB	1	3359198	3359962	(no change)
3170	b3223	ECK3212	JW3192	CDS	nanE	yhcJ	-1	3368369	3369058	(no change)
3171	b3224	ECK3213	JW3193	CDS	nanT		-1	3369106	3370596	start codon change

Figure 5.5: The answer to an SQL query in GenoBase.

5.3 Downloading the Data

We performed the following steps in order to download the needed datasets from the PortEco website:

1- Creating an account on the EcoliHub server. This account enables us to login to their system and retrieves the data.

2- Installing MySQL server version 5.5 on a local Ubuntu virtual machine in order to establish a connection with EcoliHub server.

3- Issuing multiple queries to the EcoliHub server to retrieve all data belonging to certain tables. The retrieved data are downloaded into local files which will be later uploaded into our database.

The following command shows an example of downloading the protein table from PortEco and storing the data into file “protein.txt”.

```
> mysql -u mohamaddb -p -h publichouse.ai.sri.com
ecolihouse -e "select * from Protein;" > Protein.txt
```

Using these commands we installed two main tables, Gene and Protein, from the original EcoliHub database, along with some other tables related to these two tables. After we installed the data from EcoliHub database we end up carrying more than 50,000 rows of data distributed between several files. Then, in order to upload the data from the files into the local database, we wrote a small program using Ruby language. Ruby is a flexible and powerful language that deals with files much simpler and faster than other languages. The program takes as input the file containing the data, reads it line by line, and then transfers each single line into an insert statement that is stored in another file. During the transforming mechanism, the program deals with the NULL values in the data vary carefully and makes sure to transform them to the right representation so that when we insert them in our database, they will be NULL values and not a strings. In the following, we illustrate an example of the transformation process.

Data Processing Scenario:

Example of one line downloaded from the website:

- 5461 ure1_ecoli NULL NULL NULL NULL NULL 3418.94 NULL calc_pi NULL 1001

Example from Ruby program:

- f.write "insert into Protein(WID, Name, AASequence, Length, LengthApproximate, Charge, Fragment, MolecularWeightCalc, MolecularWeightExp, PICalc, PIExp, DataSetWID)

Example of the output:

- insert into Protein(WID, Name, AASequence, Length, LengthApproximate, Charge, Fragment, MolecularWeightCalc, MolecularWeightExp, PICalc, PIExp, DataSetWID) values

5.4 Building the Local Database

In this section, we describe in some details the database that we built locally to store the data. Then we use this database to analyze the data by issuing many queries over the data as we will describe in Section 6.

For our purpose, we created three main tables; ***Genes***, ***Proteins***, and ***Subsequences***. We present below the SQL "Create Table" commands we used to create the tables including the columns in each table. As we will explain later, each of the two tables contains many fields but not all of them are used by our analysis---we focused on some important fields. Some of the fields are not giving us the ability to search for conflicts between data. For example, Primary Keys and Foreign Keys have no actual real-world meaning between the different datasets. Therefore, by comparing ID's between different datasets we will not get the right analysis for Data conflicts. On the other hand, the main fields related to our research are the fields that can define properties of the data. These fields should not be different from one dataset to another. For example, in table Protein, the *Name*, *AASequence* fields should be the same across datasets for the same protein. Similarly, in table Gene, the *Name*, *Type*, and *CodingRegionStart* should be the same. Therefore, comparing data across datasets depending on the uniqueness of the Name or *AASequence* will allow us to extract data conflicts between datasets.

The SQL “Create Table” Statements as presented below:

CREATE TABLE Gene

```
•CREATE TABLE Gene
•( WID  BIGINT  NOT NULL,
•Name  VARCHAR(255),
•NucleicAcidWID  BIGINT,
•SubsequenceWID  BIGINT,
•Type  VARCHAR(100),
•GenomeID  VARCHAR(35),
•CodingRegionStart  INT,
•CodingRegionEnd  INT,
•CodingRegionStartApproximate
 VARCHAR(10),
•CodingRegionEndApproximate
 VARCHAR(10),
•Direction  VARCHAR(25),
•Interrupted  CHAR(1),
•DataSetWID  BIGINT  NOT
 NULL,
```

CREATE TABLE Protein

```
• (WID  BIGINT  NOT NULL,
• Name  TEXT,
• AASequence  LONGTEXT,
• Length  INT,
• LengthApproximate  VARCHAR
  (10),
• Charge  SMALLINT,
• Fragment  CHAR(1),
• MolecularWeightCalc  FLOAT,
• MolecularWeightExp  FLOAT,
• PICalc  VARCHAR(50),
• PIExp  VARCHAR(50),
• DataSetWID  BIGINT  NOT
  NULL,
```

CREATE TABLE Subsequence

```
• ( WID  BIGINT  NOT NULL,
• NucleicAcidWID  BIGINT  NOT NULL,
• FullSequence  CHAR(1),
• Sequence  LONGTEXT,
• Length  INT,
• LengthApproximate  VARCHAR(10),
• PercentGC  FLOAT,
• Version  VARCHAR(30),
```

5.4.1 Description of Important Fields

Gene.Name: is common first name (common shorted form)

Gene.Type: It is a name given to some stretches of DNA and RNA that code for a polypeptide or for an RNA chain that has a function in the organism.

Gene.CodingRegionStart and Gene.CodingRegionEnd: The coding region of a gene, also known as the coding sequence or CDS (from Coding DNA Sequence), is that portion of a gene's DNA or RNA, composed of exons, that codes for protein.

Gene.Direction: It is Transcription in which the process of creating a complementary RNA copy of a sequence of DNA.

Gene.NuclecAcidWID: ID referencing the subsequence table.

Gene.Interrupted: is a gene that contains sections of DNA called exons, which are expressed as RNA and protein, interrupted by sections of DNA called introns, which are not expressed.

Gene.DataSetWID: It is the maximal and minimal values of any gene in the sample.

Protein.AASequence*: Amino Acid Sequence* is the order that amino acids join together to form peptide chains, or polypeptides.

Protein.Charge: The charge on amino acid side chains depends on the pH.

Protein.Fragment: Protein fragmentation and domain swapping are valuable methods for the study of inter- and intra-domain and subdomain interactions in proteins.

Protein.MolecularWeightCalc: used to approximate charge of a peptide or protein sequence.

Protein.MolecularWeightExp: The relationship between subunit molecular weight and heterozygosis.

Protein.DataSetWID: is a repository for the 3-D structural data of large biological molecules, such as proteins and nucleic acids

Subsequence.sequence: Determine the amino acid sequence of a protein, as well as which conformation the protein adopts and the extent to which it is complexes with any non-peptide molecules.

Subsequence.FullSequence: contains the genome sequence.

The relational schema that captures the relationships among the tables is presented in the figure below.

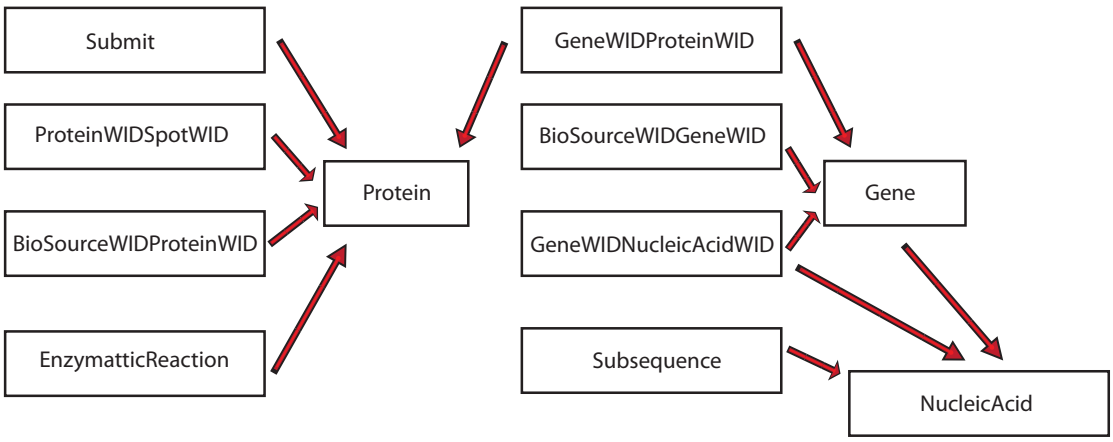


Figure 5.6: Tables relational schema.

6- Data Analysis

Goal: Our Data analysis focuses on interpreting the two main tables; *Gene* and *Protein* presented in Section 5.4. The analysis is pointed toward collecting statistics to highlight the degree of match/mismatch between the different datasets. These statistics can be used as a measure to assess whether data coming form multiple sources usually conflicts with each other. In order to achieve these goals we wrote a set of queries to process the data inside the database. In the following, we describe each SQL (Structured Query Language) query, its syntax, the output from the query, and finally brief comment on the result.

Query#1

- ***Description:***

  This query wrote to find Datasets differences. In other words, find how many Genes or Proteins in Di and not in Dj

- ***Query Steps:***
  1. Create a separate view for each dataset in tables Gene and Protein containing all data belongs to this dataset.
  2. Join the selected dataset view (Di) with the comparable dataset view (Dj) based on names equality.
  3. Count the number of records which found in Di but not in Dj.

Query SQL Syntax:

```
• Select count(*) from Dj
• Where Dj.Name
• NOT IN
• (Select Di.Name from Di);
```

- ***Results:***

Gene Data Sets:

Di \ Dj	1001	7	11	12	13	14
1001	0	1986	2114	2201	2303	1979
7	2087	0	500	450	552	351
11	2017	302	0	325	427	147
12	2105	253	332	0	0	161
13	2105	253	332	0	0	161
14	2058	329	322	291	393	0

- *Value V[Di,Dj] = # of Genes in Di & not in Dj*

Protein Data Sets:

Di \ Dj	1001	2	7	11	12	13	14
1001	0	0	0	0	0	0	0
2	5154	0	3741	4706	3792	3792	3407
7	5154	5086	0	4365	3273	3273	2890
11	0	0	0	0	0	0	0
12	5154	5298	3364	2758	0	0	2233
13	5154	5298	3364	2785	0	0	2233
14	5154	5307	3368	3976	2430	2430	0

- *Value V[Di,Dj] = # of Proteins in Di & not in Dj*

Query#2

- ***Description:***

  This query wrote to find Datasets similarities. In other words, find how many Genes or Proteins in Di and in Dj.

- ***Query Steps:***

  1. Create a separate view for each dataset containing all data belongs to this dataset.
  2. Join the selected dataset view Di with the comparable dataset view Dj based on names equality.

3. Count the number of records which found in Di and in Dj.

Query SQL Syntax:

- *Select count(*) from Dj*
- *Where Dj.Name*
- *NOT IN*
- *(Select Di.Name from Di);*

- ***Results:***

Gene Data Sets:

Di \ Dj	1001	7	11	12	13	14
1001	4405	2318	2388	2300	2300	2347
7	2318	4304	4002	4051	4051	3975
11	2388	4002	4502	4176	4176	4179
12	2300	4051	4170	4501	4603	4165
13	2300	4051	4170	4501	4603	4165
14	2347	3975	4180	4210	4210	4326

- *Value V[Di,Dj] = # of Genes in Di & in Dj*
- *Value V[Dk,Dk] = The total number of records in Gene dataset Dk*

Protein Data Sets:

Di \ Dj	1001	2	7	11	121	13	14
1001	0	0	0	0	0	0	0
2	0	5685	333	115	147	147	91
7	0	328	4303	200	277	277	141
11	0	96	187	4975	1396	1396	113
12	0	140	275	1568	4146	4146	314
13	0	140	275	1568	4146	4146	314
14	0	92	141	129	327	327	3724

- *Value V[Di,Dj] = # of Proteins in Di & in Dj*
- *Value V[Dk,Dk] = The total number of records in Protein dataset Dk*

Query#3

- ***Description:***

  This query wrote to find the records that have the same name and different in at least one field value between two datasets Di and in Dj in table Protein**.**

- ***Query Steps:***

  1. Create a separate view for each dataset containing all the records that belong to this dataset in table Protein.
  2. Join the selected dataset view (Di) with the comparable dataset view (Dj) based on the equality of all the fields in a record from Di to all the fields in a record from Dj.
  3. Count the number of records resulting from the join.

- ***Results:***

Protein Data Sets:

D_i \ D_j	1001	2	7	11	121	13	14
1001	0/0	0/0	0/0	0/0	0/0	0/0	0/0
2	0/0	5685/5685	333/333	115/93	147/147	147/147	91/91
7	0/0	328/328	4303/0	200/200	277/277	277/277	141/141
11	0/0	96/74	187/187	4975/0	1396/1396	1396/1396	113/107
12	0/0	140/140	275/275	1568/1568	4146/0	4146/1077	314/314
13	0/0	140/0	275/0	1568/0	4146/1077	4146/0	314/314
14	0/0	92/1	141/141	129/123	327/327	327/327	3724/0

- *Value V[Di,Dj] in form of pairs (V1/V2)*
- *V1 = # of Proteins in Di & in Dj*
- *V2 = # of Proteins common between Di & in Dj, but different in at least on field value.*

- ***Comments on Results:***

Using this type of queries we were able to find the number of records which have at least one different field value. By joining the two selected datasets on the equality of every field value

between *Di and Dj* we don't just count the number of the records matching the join, but also what we found is the number of identical records between different protein datasets, because AASequence was part of table Protein, so the equality from comparing AASequence means they are identical, since protein sequence is unique per protein name.

Query#4

- ***Description:***

This query wrote to find the records that have the same name and different in at least one field value between two datasets Di and in Dj in table Gene**.**

- ***Query Steps:***
  1. Create a separate view for each dataset containing the Name, Direction, Type, GenomeID from table Gene and Gene sequence which is a Subsequence from table Subsequence with starting point equal to *CodingRegionStart* and a length equal to *CodingRegionEnd - CodingRegionStart*.
  2. Join the selected dataset view (Di) with the comparable dataset view (Dj) based on the equality of all the fields in a record from Di to all the fields in a record from Dj.
  3. Count the number of records resulting from the join.

Query SQL Syntax:

```
•Select count(Distinct Di.Name)
•From Di, Dj
•Where
•IFNULL(Di.Name,0) = IFNULL
 (Dj.Name,0) and IFNULL
 (Di.Type,0) = IFNULL(Dj.Type,0)
 and
•IFNULL( Di GID,0) = IFNULL
 (Dj.GID,0) and
•IFNULL(Di Sub12,0) = IFNULL
 (Dj.Sub13,0) and IFNULL
 (Di.Dir,0) = IFNULL(Dj.G13Dir,0);
```

Create View Example:

```
•Create view Di as
•Select Name as Di Name,
 Substring
 (Sequence,CodingRegionStart,
 CodingRegionEnd-
 CodingRegionStart) as Sub12,
 Direction as Di Dir, Type as Di
 Type, GenomeID as Di GID
•From Subsequence, Gene
•Where
 Subsequence.NucleicAcidWID=Ge
 ne.NucleicAcidWID and
 Gene.DataSetWID=12;
```

- ***Results:***

Gene Data Sets:

D_i \ D_j	1001	7	11	12	13	14
1001	4405/0	2318/2318	2388/2388	2300/2300	2300/2300	2347/2347
7	2318/2318	4304/4304	4002/4002	4051/4051	4051/4051	3975/3975
11	2388/2388	4002/4002	4502/4502	4176/4176	4176/4176	4179/4179
12	2300/2300	4051/4051	4170/4170	4501/0	4603/334	4165/4165
13	2300/2300	4051/4051	4170/4170	4501/232	4603/0	4165/4165
14	2347/2347	3975/3975	4180/4180	4210/4210	4210/4210	4326/4326

- *Value V[Di,Dj] in form of pairs (V1/V2)*
- *V1 = # of Genes in Di & in Dj*
- *V2 = # of Genes common between Di & in Dj, but different in at least on field value.*

- ***Comments on Results:***

In order to find the records that are different in at least one field value notice we didn't include all Gene data fields in the view. Also you can notice that we have to find the Gene sequence from table Subsequence and include it in the view in order to do the comparison. We only selected the important data fields from table Gene , the ones which can define the uniqueness's of the record, and also to make us sure and accurate about records being identical we have to find the Gene sequences and compare them.

Notice here that the only two datasets that contain the same exact records are 12 and 13 where the rest of them are not, because some of the datasets contain one or more field with NULL values for all of the records where in other datasets those fields has a value not equal to NULL, and this will lead to recognize them different in at least 1 field value.

Query#5

- ***Description:***

  This query wrote to find records common in names, but different in gene sequence between datasets Di and Dj in table Gene.

- ***Query Steps:***

  1. Create a separate view containing the Name and sequence for all the records for each dataset in table Gene.
  2. Join the selected dataset view (Di) with the comparable dataset view (Dj) based on equal names, but different gene sequences.
  3. Count the number of records resulting from the join.

Create View Example:

- *Create view Dj as select Name, substring(Sequence, CodingRegionStart, CodingRegionEnd - CodingRegionStart) as Subseq*
- *From Gene, Subsequence*
- *Where Gene.NucleicAcidWID = Subsequence.NucleicAcidWID and Gene.DataSetWID=7;*

Query SQL Syntax:

- *Select count(Distinct Dj.Name)*
- *From Dj, Di*
- *Where Dj.Name = Di.Name and Dj.Subseq!= Di.Subseq;*

- ***Results:***

Gene Datasets:

D_i \ D_j	1001	7	11	12	13	14
1001	0	0	0	0	0	0
7	0	0	0	0	0	0
11	0	0	0	0	0	0
12	0	0	0	0	25	0
13	0	0	0	25	0	0
14	0	0	0	0	0	0

- *Value V[Di,Dj] = # of Genes common in names, but different in sequences between Di & Dj*

Query#6

- ***Description:***

  This query wrote to find records common in names, but different in protein sequence between datasets Di and Dj in table Protein.

- ***Query Steps:***

1- Create a separate view containing the Name and sequence for all the records for each dataset in table Protein.

2- Join the selected dataset view (Di) with the comparable dataset view (Dj) based on equal names, but different protein sequences.

3- Count the number of records resulting from the join.

Create View Example:

```
• Create view Dj as select
  Name, AASequence
• From Protein
• Where DataSetWID=7;
```

Query Example:

```
• Select count(Distinct
  Dj.Name)
• From Dj, Di
• Where Dj.Name = Di.Name
  and
  • Dj.AASequence!=
    Di.AASequence;
```

Results:

Protein Datasets:

D_i \ D_j	1001	2	7	11	12	13	14
1001	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
7	0	0	0	0	7	7	0
11	0	0	0	0	0	0	0
12	0	0	7	0	0	245	0
13	0	0	7	0	245	0	0
14	0	0	0	0	0	0	0

- *Value V[Di,Dj] = # of Proteins common in names, but different in sequences between Di & Dj*

Query#7

- ***Description:***

  This query wrote to collect the records referring to the same Gene name. In order to show us the conflicts along multiple data sets.

- ***Query Steps:***

1- Pick a gene name that you wish to apply the query using it.

2- Select most important fields in gene table belong to the given gene name from every data set in the table.

Query Example:

```
• Select Name, NucleicAcidWID, SubsequenceWID, Type, GenomeID,
  CodingRegionStart, CodingRegionEnd, Direction, DataSetWID
• From Gene
• Where Name='thrL';
```

Name	NucleicAcidWID	SubseqWID	Type	GenomeID	CodingStart	CodingEnd	Direction	DataSet
thrL	NULL	NULL	NULL	NULL	190	255	forward	1001
thrL	NULL	NULL	polypeptide	NULL	NULL	NULL	NULL	7
thrL	NULL	NULL	NULL	EG11277	190	255	F	11
thrL	5121196	5121197	polypeptide	b0001	190	255	Forward	12
thrL	5133007	5133008	polypeptide	b0001	190	255	Forward	13
thrL	NULL	NULL	NULL	b0001	190	255	F	14

- ***Comments on Results:***

  As we notice in the above table the results for the same gene name along different datasets are not the same, but they have many similarities for example in CodingRegionStart and CodingRegionEnd. If we notice in field Direction five of the records have the same direction “Forward” but the representation of the word “Forward” was in two different shapes “Forward” and “F” which is considered as physical difference between the two record but not a conceptual difference, since “Forward” means the same as “F” but with different representation.

Query#8

- ***Description:***

  This query wrote to show us how user input can have different representations (which will lead to data conflict) but same meaning.

- ***Query Steps:***

  1. Pick a field name that you want to test its values.
  2. Select the distinct values presented in the selected field.

Query Example:

```
Select DISTINCT Direction From Gene;
```

Direction
NULL
Unknown
Reverse
R
Forward

- ***Comments on Results:***

  As we can see in the above column there are more than one representation for each value in field Direction, so we have **NULL** means **Unknown** and **Reverse** can be represented as **R** and finally we have **Forward** which is equivalent to **F**. One can think that this is not a problem, but if you tried to select the records where Direction=”Reverse” the query result will not include records where Direction=”r” where they are the same meaning in fact.

7- Proposed Solutions for Conflict Management

7.1 Taxonomy-Based Model

7.1.1 Overview of Existing Model: LITCHI

Very much like a spoken language, the names for different types of species are different all over the world. This can cause a lot of confusion when trying to gather information about a certain species. LITCHI, which stands for Logic-based Integration of Taxonomic Conflicts in Heterogeneous Information, is focused on detecting conflicts and errors within real taxonomic checklists. The goal of the LITCHI project that is funded by a grant from the BBSRC/EPSRC Bioinformatics Initiative is to create a tool that can be used by skilled taxonomic editors. This tool will help these editors identify and find solutions to conflicts that are within taxonomic checklists. A taxonomic checklist is an informative list of information about a certain taxa / species.

A good example of a conflict would be what was listed about the ILDIS World Database of Legumes. This database holds information on a total of 19,043 taxa but there are 37,394 names that have been applied to the said taxa. This results in an average two different names for each taxa. There are many reasons why this has happened. One reason may be that a certain species could have been discovered and named by two or more different biologists on their own. Conflicts can be found by seeking relationships between accepted and known names and synonyms.

Most scientific names do have to meet the international Code of Botanical Nomenclature. This code is a series of rules that set guidelines for the making of new botanical names. The taxonomic categories of which these names are comprised of are:

- Familia (family)
- Genus
- Species
- Subspecies
- Varietas (variety)

If any species is to be renamed the plan is to retain the specific epithet from the original given name (base name).

Regardless of all the inconsistencies and conflicts in scientific naming, there still has been no agreement in the taxonomic community about what needs to be done or what the "best practice" is when issuing and using scientific names. LITCHI biologists have started to suggest some potential rules. Two of these rules are

- A full name may not appear as an accepted name and as a synonym in the same checklist.
- A full name should not appear as an accepted name more than once in the same checklist.
- In no checklist is a full name a synonym of more than one taxon.
- Within any given checklist, two names may not contain the same Latin components unless labeled as a homonym or a misapplied name.
- Within any given checklist, every full name that is indicated to be misapplied by the form of its authority must be labeled as a misapplied name, and vice versa.

When a species is moved from one genus to another it is apparent that the name of this species may change. There is a rule that helps in aiding this conflict that states a checklist may not contain both a name and its basionym unless both names refer to the exact same taxon.

The LITCHI system will have repository style architecture. This means that the information will be communicated through a central database between a few different components. The three software components are the date import / export function (DIEF), the conflict reasoning engine (CRE), and the interface for Taxonomists (IfT). The most important part of this system is going to be the central database. The use of this central database will eliminate the need to constantly create and update large files and also eliminates some possible difficulties that relate to the communication of such large files.

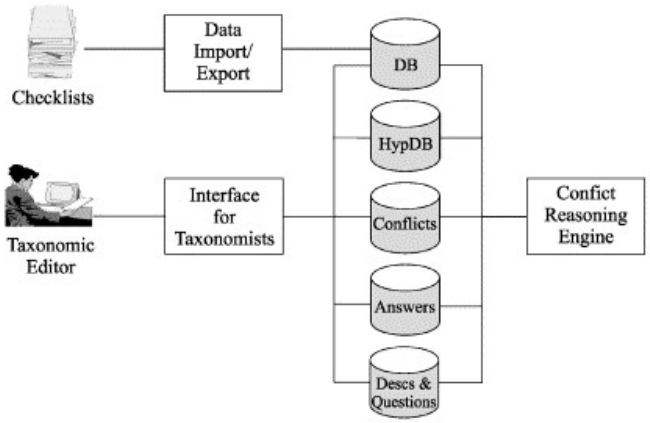


Figure 7.1: LITCHI System

The LITCHI system has already been able to produce results. For example; sets of conflicts generated from several checklists have been analyzed by LITCHI biologist, and have led to refinements of the formal model, they compared 5808 names in the trible Galegease from the ILDIS world database of Legumes with 1908 names from the Legumes of Northern Eurasis database, they detect 1500 potential conflicts.

Rule	Rule Description	Conflicts
C27	A full name may not appear as both an accepted name and a synonym in any given checklist	43
C4	A full name may not appear as the accepted name of more than one taxon in any given checklist.	372
C5	A full name may not appear as the synonym of more than one taxon in any given checklist.	112
W25	Within any given checklist, two names may not contain the same Latin components (but different authorities) unless labeled as a homonym or a misapplied name.	812
C26	Within any given checklist, every full name which is indicated to have been misapplied by the form of its authority must be labeled as a misapplied name, and vice versa.	32

The LITCHI system is still in need of some finishing touches and is not complete yet but taxonomists using the system have yielded some very good and useful results detecting conflicts in the ILDIS World Database of Legumes. Another benefit that was discovered is its role in the cleansing of data prior to it being integrated into the system. One can assume that the LITCHI system will be an extremely beneficial and useful tool when it is complete.

7.1.2 Exploitation and Extensions to LITCHI

In our case of study (Gene & Protein tables) and after analyzing the collected data we found that the reasons of data conflicts between datasets are many. Starting from data redundancy, where same genes and proteins records appear in different datasets, missing data values where some datasets have missing values for some fields either in some records or completely along all the records. Moreover, there are conflicts coming from having different representations of values, where these representations may refer to the same meaning. All these reasons combined in these tables maximized the occurrence of data conflict and make it harder for us to search for solutions that can fit in to our model and resolve conflicts.

LITCHI was a great model made by scientists and experts to resolve data conflicts along different datasets using special algorithms, taxonomies and checklists to resolve conflicts that can happen because of different representations for the same value or data redundancy. In our case LITCHI can be applied on table gene and protein by creating a taxonomy and check lists for the variety of representation for same values along table fields, so in this case for example in table **Gene** field **Direction** every time the taxonomy will read value ***F*** will match it to value ***Forward*** (Refer to Query 8 in Section 6). So now all records that have Direction value equal to ***Forward*** or ***F*** will be regarded as compatible fields and there record should appear in queries' results asking for either of the two values ***Forward*** or ***F***. Second, LITCHI system can resolve data redundancy, because we can add a new algorithm and alter our taxonomy in order to filter out records that already exist in our database. In this case, we can check if the Gene or Protein name is already exist, and we can run the comparison for all fields between the new coming record and the stored one. If they match, then we compare them if they are identical by comparing sequences and other fields, and if all tests get passed positively we can avoid inserting

the coming record into the table. Applying these changes will help resolving several of data conflicts, so the queries can return the correct results.

The LITCHI system, however, cannot solve other types of conflicts that may arise in the datasets. For example, if two records in Genes or Proteins tables have the same name and sequence but different in other fields, then LITCHI will consider them as different and will keep both of them in the database without noticing that they are conflicting with each other. When a user queries the database, the returned records will contain both records with different information. This is still considered as data conflicts, because for example, if we go back to Data Analysis section (Section 6) and specifically Query 6, we see that the gene name "thrL" have more than one occurrence along different datasets with variations along almost all fields between records, and this will cause confusion for users and in some cases can cause the users take wrong decisions because of this variations. In the following section, we suggest another approach complementary to LITCHI system to overcome this limitation.

7.2 Extended Relational Database Model

After we saw the exploitations which couldn't be resolved using the LITCHI system and avoid this type of conflicts. We came up with a suggestion to resolve this problem by building a ***history-tracking database system***, where the changes over the data can be tracked and if different versions exist in the database, the system can track which version is more recent.

Our suggestion build on the idea of having one up-to-date dataset of Genes or Proteins which will be in tables Gene or Protein that will be available for users to query. This dataset will be carrying unique and up-to-date Gene or Protein records with unique names and sequences. Basically we are going to add five columns for tables Gene and Protein and the fields are ***CurrRecordID, CurrDataSetID, UpdatedRecordID, UpdatedDataSetID***, and ***UpdatingDate***. Each of these fields will play a role in order to get to our final goal. ***CurrRecordID*** will be representing the ID of the current record after a new coming record updates it. ***UpdatedRecordID*** represents the ID of the current record before it gets updated. ***UpdatingDate*** represents the timestamp of updating the record. ***CurrDataSetID*** represents the ID for the dataset that the current record belongs to. And ***UpdatedDataSetID*** represents the datasetID of the updated record. The execution mechanism works as follows:

1- We will start with our initial Gene or Protein dataset, and initially UpdatedRecordID, UpdatedDataSetID, and UpdatingDate fields will be empty, and CurrRecordID, CurrDataSetID will be carrying the information for the current record.
2- We will receive a new Gene or Protein dataset.
3- We will apply LITCHI mechanisms on the coming dataset to avoid identical records.
4- Check if the coming records and the current records has the same name and sequence, but different in at least one filed
5- Store copy of the current record in table GeneDataSetStore or ProteinDataSetStore.
6- Update the current record fields values by the new coming values if they are different.
7- Store the coming record ID in CurrRecordID, and the coming dataset ID in CurrDataSetID and store the record ID before get upadated by the coming record in UpdatedRecordID and the record dataset ID before get upadated by the coming record in UpdatedDataSetID and the date of updating the record in UpdatingDate.

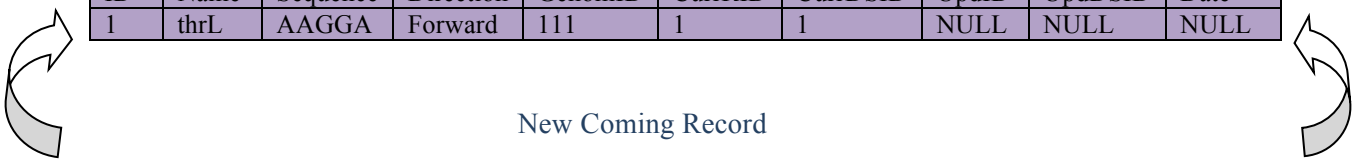
Special Case:

If the name of Gene or Protein in the coming record doesn't exist in the current database available to the user, then this record will be inserted as it is and it will be the initial record for this name and its dataset will be the initial dataset. After building this model we will end up having finalized dataset with most recent updated records and with unique record for each Gene or Protein. In case the user wants to go back for older versions he can go recursively by accessing the records in table GeneDataSetsStore or ProteinDataSetsStore using the values from CurrRecordID, CurrDataSetID, UpdatedRecordID, UpdatedDataSetID, and UpdatingDate.

Example:

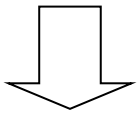
Current Record Before Update

ID	Name	Sequence	Direction	GenomID	CurrRID	CurrDSID	UpdID	UpdDSID	Date
1	thrL	AAGGA	Forward	111	1	1	NULL	NULL	NULL



New Coming Record

ID	Name	Sequence	Direction	GenomID	CurrRID	CurrDSID	UpdID	UpdDSID	Date
2	thrL	AAGGA	Reverse	111	2	2	NULL	NULL	NULL



Current Record After Update

ID	Name	Sequence	Direction	GenomID	CurrRID	CurrDSID	UpdID	UpdDSID	Date
2	thrL	AAGGA	Reverse	111	1	1	2	2	6:30 PM 9/29/2012

8- Conclusion

In this project, we studied several data integration issues that arise from collecting and sharing scientific data from different sources and datasets. We studied how scientist, over the past years, have been collecting and analyzing their data in isolation, e.g., storing and analyzing data in their individual laboratory, what systems they are using, and what file formats are mostly common among scientists. Our studies have focused on biological databases and we used several of the biggest biological databases available on the Internet, e.g., MetaBase, PortEco ,and GenBank, as our case studies. Our initial hypothesis is that different datasets will definitely contain conflicting data, e.g., same objects in the different datasets but with different attribute values. Based on the case studies that we used, we proved the correctness of our hypothesis through the analysis of the data and we reported significant percentage of conflicting data. We then studied possible solutions for avoiding/resolving such conflicts. We proposed two potential solutions that combined can reduce the effect of conflicts; the first solution is based on taxonomy-based models, and the second solution is based on extensions to database systems to track the history of evolving data.

In conclusion, we believe that scientists face a real problem when sharing data from many sources due to conflict issues. There must be efficient and systematic mechanisms to overcome this issue and help scientists focus on their research and experiments. We put our effort in this project to address this problem and this report summarizes our findings.

9- References

J. Gray, D. T. Liu, M. Nieto-Santisteban, and A. S. Szalary (2005).. *Scientific Data Management in the Coming Decade.* Microsoft Research

Peer Kroger, and Francois Bry (2003). *Computational Biology Database Digest: Data, Data Analysis, and Data Management.* Institute for Computer Science, University of Munich, Germany

W. Baker, A. van den Broke, E. Camon, P. Sterk, G. Stoesser, M.A. Tuli (2000). *The EMBL Nucleotide Sequence Database.* Nucleic Acids Research, Vol. 28, NO. 1, pp.19-23

A. Bairoch, R. Apweiler (2000). *The SWISS-PROT Database and its Supplement TrEMBL in 2000*. Nucleic Acids Research, Vol.28, No. 1, pp. 45-48.

H-W. Newes, D. Frishman, C. Gruber, B. Grire, D. Hasse, A. Kaps, K. Lemcke, G. Mannhaupt, F. Pfeiffer, C. Schuller, S. Stocker, B. Weil (2000). *MIPS: a Database for the Escherichia coli Genomes and Protein Sequences.* Nucleic Acids Research, Vol.28, No. 1, pp. 37-40

S. Letovsky, R.W. Cottingham, C.J. Porter, P.W.D. Li (1998). *GDB: the Human Genome Database.* Nucleic Acids Research, Vol. 26. No. 1, pp. 94-99

D. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, B.A. Rapp, D.L. Wheeler (2000). *GeneBank*. Nucleic Acids Research, Vol. 28, No. 1, pp.15-18.