AN APPLICATION FOR DOWNLOADING AND INTEGRATING MOLECULAR BIOLOGY DATA

Burr R. Fontaine, MD, MS

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Master of Sciences
in the School of Informatics
Indiana University
July 2004

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Master of Sciences.

Tatiana Foroud, Ph.D.

Snehasis Mukhopadhyay, Ph.D.

Masters Committee

Narayanan B Perumal, Ph.D.

July 6, 2004

ACKNOWLEDGEMENTS

First and foremost, the successful completion of this project would not have been possible without the assistance and guidance I received from my masters committee: Dr. Tatiana Foroud, Dr. Snehasis Mukhopadhyahy, and Dr. Narayanan Perumal.

End-user involvement is critical to ensuring that a software development project does a successful job of meeting their needs. Kenneth White, Dan Koller, Nathan Pankratz, Dongbing Lai, Lixiang Liu, Subha Krishna, and Leah L. Flury attended numerous meetings and contributed a substantial amount of valuable input over the course of this project. This project also received outstanding technical support from Joseph Urbanski in the department of Medical and Molecular Genetics and Andrew Arenson in Centralized Life Sciences Data Services at Indiana University.

This project received grant funding from NIH/NIDS R01-NS37167 and NIH/NIA P01-AG18397. This research was also supported in part by the Indiana Genomics Initiative (INGEN). The Indiana Genomics Initiative (INGEN) of Indiana University is supported in part by Lilly Endowment Inc.

ABSTRACT

INTRODUCTION AND BACKGROUND

Integrating large volumes of data from diverse sources is a formidable challenge for many investigators in the field of molecular biology. Developing efficient methods for accessing and integrating this data is a major focus of investigation in the field of bioinformatics.

In early 2003, the Hereditary Genomics division of the department of Medical and Molecular Genetics at IUPUI recognized the need for a software application that would automate many of the manual processes that were being used to obtain data for their research. The two primary objectives for this project were: 1) an application that would provide large-scale, integrated output tables to help answer questions that frequently arose in the course of their research, and 2) a graphic user interface (GUI) that would minimize or eliminate the need for technical expertise in computer programming or database operations on the part of the end-users.

In early 2003, Indiana University (IU), IBM, and the Indiana Genomics Initiative (INGEN) introduced a new resource called Centralized Life Sciences Data Services (CLSD). CLSD is a centralized data repository that provides programmatic access to biological data that is collected and integrated from multiple public, online databases.

METHODS

1. an in-depth analysis was conducted to assess the department's data requirements and map these requirements to the data available at CLSD
2. CLSD incorporated new data as necessary
3. SQL was written to generate tables that would replace the targeted manual processes
4. a DB2 client was installed in Medical and Molecular Genetics to establish remote access to CLSD
5. a graphic user interface (GUI) was designed and implemented in HTML/CGI
6. a PERL program was written to accept parameters from the web input form, submit queries to CLSD, and generate HTML-based output tables
7. validation, updates, and maintenance procedures were conducted after early prototype implementation

RESULTS AND CONCLUSIONS

This application resulted in a substantial increase in efficiency over the manual methods that were previously used for data collection. The application also allows research teams to update their data much more frequently. A high level of accuracy in the output tables was confirmed by a thorough validation process.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

BLAST	Basic Local Alignment Search Tool
bp	base pair
CGI	Common Gateway Interface
CLSD	Centralized Life Sciences Data
cM	centiMorgan
DAS	Distributed Annotation Service
GUI	Graphic User Interface
HTML	Hypertext Markup Language
INGEN	Indiana Genomics Initiative
IU	Indiana University
IUPUI	Indiana University Purdue University Indianapolis
JDBC	JAVA Database Communication
MGI	Mouse Genome Informatics
NCBI	National Center for Biotechnology Information
RDBMS	Relational Database Management System
RGD	Rat Genome Database
SNP	Single Nucleotide Polymorphism
STS	Sequence Tagged Site
UCSC	University of California Santa Cruz
UITS	University Information Technology Services
XML	Extensible Markup Language

INTRODUCTION

INTRODUCTION TO SUBJECT

A major focus of research in the Hereditary Genomics division of the department of Medical and Molecular Genetics at IUPUI is the identification and characterization of genes that cause human disease. This research requires large volumes of data from a number of online databases at the National Center for Biotechnology Information (NCBI), including LocusLink, UniGene, UniSTS, and dbSNP. Data are also obtained from Mouse Genome Informatics (MGI) and the Rat Genome Database (RGD).

Collecting and integrating this data manually requires the user to access multiple web sites and copy data from numerous web pages into spreadsheets, one element at a time. This process is tedious, time-consuming, and prone to human error. Automating these processes is a desirable goal because of the repetitive nature of these tasks and the volume of data that is required. However, the size and complexity of the sources from which the data are drawn make it difficult to automate these processes. This is especially difficult when data must be accessed and integrated from multiple sources that have been developed independently of each other. Automating access methods for many of these databases is also complicated by data models and file formats that are updated and revised on a regular basis.

IMPORTANCE OF SUBJECT

Integrating large volumes of data from diverse sources is a formidable challenge for many investigators in the field of molecular biology. The exponential increase of new information in this field in recent years has had a profound impact on medicine and biology. Managing this information so that its potential value can be fully realized is a critical component of the infrastructure that supports medical and biological research, and a major focus of investigation in the field of bioinformatics.

There are a large number of online biological databases that are freely available to the research community on the Internet, but the web sites for these databases are often difficult to navigate because of their size, complexity, and interface design. Web pages are a good way to access online databases when looking for specific pieces of information, but they are poorly suited for accessing large volumes of data on a regular basis. Many online biological databases also make programmatic access available, but a high level of technical expertise is frequently necessary to make use of these resources.

Another problem that is frequently encountered by molecular biologists is that data must be integrated from multiple sources. Many biological databases have been developed in relative isolation, and there are significant differences in the ways that the data is represented, organized, and managed. In the absence of close coordination, the knowledge domains covered by these databases may also have gaps and redundancies. As a result, accessing and integrating information from different databases is often a time-consuming manual process that is difficult to automate. These are important issues that must be addressed to realize the full potential of the large volume of data and information that is currently being generated in the field of molecular biology.

KNOWLEDGE GAP

In early 2003, researchers in the Hereditary Genomics division of the department of Medical and Molecular Genetics department at IUPUI recognized the need for a software application that would automate many of the manual processes that were required to obtain the data necessary for their research. The two primary objectives for this project were: 1) an application that would provide large-scale, integrated output tables to help answer questions that frequently arose in the course of their research, and 2) a graphic user interface (GUI) that would minimize or eliminate the need for technical expertise in computer programming or database operations on the part of the end-user.

In early stages of this project, it was not clear what technical and programming resources would be necessary or how the project would be implemented. After discussing the requirements with the School of Informatics, a decision was made to develop this application as part a master's project in Bioinformatics.

BACKGROUND

RELATED RESEARCH

Sujansky (2001) and Stein (2003) have described three traditional approaches to the integration of heterogeneous biological and medical databases. For databases that are available online, the first and most basic approach is URL linking for end-users that are using a web browser. NCBI maintains an extensive network of URL links between its component databases as well as external resources such as MGI. This approach to database integration is relatively simple to implement, but it requires regular maintenance and close coordination to keep the URL links up to date whenever data is updated or reorganized.

A more sophisticated approach is to create a centralized data repository. This involves collecting data from numerous sources and integrating it into a single data warehouse under a global data model. The primary advantages of the data warehouse approach are local control of the data and internal consistency. However, this approach requires a substantial investment in resources to develop and maintain.

Another high-level approach is to access multiple databases with a global query engine, which translates user queries into different formats that can be processed by each database (Lacroix 2002). The query engine also integrates the results that are obtained from multiple sources into a final form that is delivered as output to the end-user. The primary advantage of this method is that the data and its organization remain under the control of individuals who are experts in their respective knowledge domains. One disadvantage of this approach is that longer processing times are frequently required because of the need to translate queries and integrate query results. Writing the drivers that communicate between the central query engine and component databases also requires a substantial investment of resources.

Stein (2003) has identified a number of database integration issues which must be addressed by all three of these approaches: 1) differences in data structures, 2) naming differences, 3) semantic differences, and 4) information which is present but not explicitly represented. Structural differences refer to the forms in which data is stored and organized, such as flat files, relational tables, or web pages. Relational databases may differ in the degree to which the tables have been normalized, and the manner in which equivalent data fields are organized into different tables. All of these structural forms may differ in the degree to which they allow free text vs. structured data fields to represent the same information.

Secondly, databases may have different names for the same data fields. For example, two medical facilities may both use social security number for patient identification, but one database might call this field PATIENT_ID and another might call it MED_REC_NUM. These similarities may be difficult to identify unless careful attention is paid to the comparison of data definitions and field values.

A third problem is semantic differences between fields and values that are similar but not equivalent. For example, one institution might quantify blood culture growth on a scale of 0-1-2-3-4+ and another might use no-low-mod-lg growth. This problem is also reflected in subtle differences in the conceptual framework that is defined for different databases. For example, LocusLink defines a gene as a nucleotide sequence with start and stop codons that can be translated into a known or predicted protein product. UniGene defines a gene as a series of inter-related mRNA sequences. When these two databases are integrated, some of the LocusLink id's cannot be mapped to a UniGene id because an mRNA sequence for that gene has not been observed and reported in at least one tissue or organ system.

Finally, databases may be difficult to integrate because of information that is present but not explicitly represented. Data that is not explicitly represented may be implicit, derivable, or missing. An example of implicit data is the species identifier in a single genome database. This field can be omitted within a single-genome context, but it becomes critical when that information is integrated with data from other species. An example of data that might be derivable but not explicitly represented is the use of chromosome vs. contig coordinates, in base pairs (bp), to indicate the physical position of a gene on a chromosome. At NCBI, the position of genes is recorded in contig coordinates, because contigs are the basic unit by which a genome is sequenced and annotated. The position of a gene on a chromosome can be derived by linking to a second table, which stores the starting and ending points for the contig on a chromosome. Finally, databases may have different definitions for missing values. "NULL" can represent negative, missing, an erroneous value, or a value that is known but not available. All of these considerations must be taken into account when integrating data from multiple sources.

A number of alternatives to the three traditional approaches to database integration have been proposed (Chicurel 2002). The first of these is a web-based service called the Distributed Annotation System[1] (DAS), which is an extension of the URL linking approach (Stein 2003, Dowell 2001). In this system, a user submits parameters that define a chromosome segment to a DAS server. The server then accesses one or more "third-party" servers and retrieves the desired annotations (e.g. introns, exons, SNP's, etc.) for that chromosome segment. This data is then integrated and returned to the end-user in the form of a table or a graphic display. The DAS data exchange is based on an XML standard. The major advantage of DAS is that the annotation data remains under the ownership of domain experts at the third-party databases. This arrangement also requires relatively little maintenance on the part of the DAS server and database owners to make the data available in a form that can be easily accessed and integrated by the DAS server and the end-user.

The primary limitation of DAS is that it is can only represent data that can be aligned with base-pair coordinates on a linear map of the genome. This framework does not lend itself well to representing data like cM positions on a genetic map, three dimensional protein structure, or metabolic pathways. However, DAS does allow each annotation to include links back to its source database, where additional information of this nature can

[1]http://biodas.org

be accessed. In its present form, DAS servers do not require that annotations be peer reviewed, and DAS does not enforce explicit control over semantics or naming conventions. The end user assumes responsibility for resolving any inconsistencies or ambiguities that result from these differences. Biological ontologies, or controlled vocabularies, can be helpful in this regard, but only to the extent that they are accepted and enforced by the third party data providers. DAS architecture has been successfully implemented by UCSC Genome Bioinformatics, Ensembl, WormBase, and FlyBase.

Two initiatives that are attempting to extend the DAS concept are BioMOBY and myGrid. BioMOBY[2] exists as a centralized registry of web-based biological resources that are defined in terms of the services that they provide (Stein 2003, Wilkinson 2003). Conceptually, each resource, or service, in the registry is defined in terms of the data that is accepted as input and the new output that is generated. The ultimate objective is to feed user input into a series of linked operations that eventually produces the information that the user needs. For example, the individual steps in a BioMOBY transaction might include: 1) accepting a human gene identifier as input, 2) retrieving the nucleotide sequence for that gene, and 3) performing a BLAST search on a mouse genome database to identify one or more homolog genes. Each step in this process is executed by the most appropriate resource, which is selected on the basis of the information in the BioMOBY registry. One of the biggest problems BioMOBY has encountered to date is the lack of a uniform syntax and ontology for naming and defining data fields. Another is that similar services performed by different resources, such as BLAST algorithms, may differ in

[2] http://biomoby.org

subtle but important respects that are not reflected in the service registry. Another potential concern is that if BioMOBY becomes widely accepted, the number of requests submitted to the central server and the volume of data that is transmitted is likely to require a substantial investment in hardware, software, and bandwidth. BioMoby is currently in the design and prototype stage.

myGrid[3] is based on information processing technologies that have been developed for grid computing (Stein 2003, Stevens 2003). One important objective of this project is to develop a much more precise and detailed description of biological information services that are available at each web-based resource. Towards this end, myGrid is based on an ontology that is capable of describing the bioinformatics capabilities of each resource in considerable detail. The ultimate goal of this project is a system that is capable of automated, complex information processing by making appropriate use of multiple, online biological resources. myGrid is currently in the design and prototype stage.

Stein (2003) makes a convincing argument that the challenge of integrating biological data is as much a sociological problem as it is a technical one. Molecular biology has traditionally been a reductionist science, and many existing databases have been developed to meet the specific needs of a relatively narrow segment of the research community. However, molecular biology is rapidly becoming an integrative science, where research depends on large volumes of complex and heterogeneous data drawn from multiple sources. As a result, the user base for many databases is becoming much

[3] http://www.mygrid.org.uk

more diverse, and the ability to integrate data from different sources has become a critical requirement for a growing number of end-users. To meet these needs, database owners will need to redefine their traditional roles and responsibilities.

CURRENT UNDERSTANDING

Research in the Medical and Molecular Genetics department at IUPUI currently uses data from two primary sources, the National Center for Biotechnology Information (NCBI) and Mouse Genome Informatics (MGI) (Benson 2003, Blake 2003, Pruitt 2003, Thorisson 2003, Wheeler 2003). At NCBI, data is most frequently accessed from five component databases: RefSeq, UniGene, LocusLink, UniSTS, and dbSNP. All of these databases are based on flat files except for dbSNP, which is a relational database. NCBI web pages for most of these databases can be accessed using Entrez, an integrated search engine. NCBI’s databases are also connected by an extensive network of URL crosslinks. NCBI’s databases also have URL cross links with MGI, but MGI is not included in the Entrez search engine. Data from NCBI and MGI are also available by ftp download, but re-creating portions of these databases from ftp files requires substantial time, effort, and technical expertise.

In the early stages of this project, a considerable amount of time was spent evaluating the relative advantages and disadvantages of implementing a centralized data repository or a global query engine for this project. The two options considered for the centralized data repository were 1) using PERL or JAVA scripts for automated web page mining of the NCBI and MGI web sites, and 2) using the ftp download directories at NCBI and MGI to mirror relevant portions of those databases in the Medical and Molecular Genetics department. PERL and JAVA scripts were also considered for implementing a global query engine that would pull data off of the NCBI and MGI web sites on a real-time, as needed basis. Storage requirements were a significant concern with the centralized data repository, and response times were a significant concern with the global query engine. It was also anticipated that any of these options would require a project team of at least 6-8 persons and 9-12 months for successful implementation.

In May 2003, University Information Technology Services (UITS) at Indiana University (IU) introduced a new resource called Centralized Life Sciences Data Services (CLSD) that resolved many of these issues. CLSD mirrors portions of NCBI’s databases at IU and IUPUI in a mainframe environment that supports integrated access to multiple data sources, complex SQL programming, and the rapid downloading of large volumes of data to a desktop computer. CLSD also has the potential to incorporate additional data from NCBI and other public, online databases as the need arises within the university community.

CLSD is a joint collaboration between IU and IBM, with financial support from the Indiana Genomics Initiative (INGEN). CLSD was created to facilitate access to online biological data for researchers at IU by automating and centralizing processes that are now being done manually and replicated in many research departments across the university.

UITS supports CLSD by providing the hardware and programming that is necessary for implementation and maintenance. IBM supports CLSD by providing the database software on which the project is based. CLSD is a high-profile project for IBM because it is the company’s first major life sciences initiative. CLSD is rapidly becoming a critical part of the infrastructure that supports medical and biological research at IU.

CLSD was initially implemented using DiscoveryLink for IBM's DB2 Relational Database Management System (RDBMS) version 7. This has since been upgraded and renamed Information Integrator for DB2 version 8. Information Integrator is a DB2 add-on that uses “wrappers” to translate DB2 queries into a format that can be interpreted by multiple external data sources. The wrapper also integrates the query results from external sources and translates them back into DB2 format. Although Information Integrator is capable of acting as a global query engine, at CLSD its primary function is to automate the process of transferring data from multiple external sources into an integrated, centralized data repository. In the initial stages, the data sources at CLSD that were relevant to this project included LocusLink, UniGene, and dbSNP. LocusLink and UniGene are downloaded as hierarchical text files, which are parsed and uploaded into a relational schema that is unique to CLSD. dbSNP is downloaded as a series of relational tables that are parsed and uploaded into the CLSD database with only minor modifications to accommodate DB2 requirements, such as date formats. UniGene and LocusLink are updated twice weekly. dbSNP is updated on an irregular basis when new builds are released by NCBI, usually every one to two months. NCBI does not add data to dbSNP in between builds. The code for loading LocusLink was written by programmers at IBM; the code for UniGene and dbSNP was written by UITS. CLSD exists as a single database, but data from each external source is downloaded, parsed, and loaded into it's own schema. CLSD can be accessed remotely using a DB2 client, which can be based on PERL/DB2, Java Database Communication (JDBC), or Windows Open Database Communication (ODBC). In the early stages of this project, CLSD did not have all of the data required by Medical and Molecular Genetics, but UITS indicated a strong willingness to provide whatever additional data were necessary for this project.

INTENDED PROJECT PLAN

1) identify and prioritize the manual processes in the Department of Medical and Molecular Genetics that need to be automated
2) define the context in which these processes take place:
    a. how is research in the department organized?
    b. what types of data are needed at each stage?
    c. what are the most frequent and important questions that are asked when using NCBI and other web sites?
3) translate end-user requirements from web-page based processes into discrete queries that can be defined in terms of input parameters and output tables
4) install a DB2 client in the Medical and Molecular Genetics department that will allow remote, real-time access to the CLSD database
5) write and implement program code that will submit queries to CLSD and return data from different sources in the form of integrated tables

6) design and implement a graphic user interface that does not require technical or in-depth knowledge of computer programming, database operations, or the underlying sources from which the data is drawn
7) early implementation of a working prototype with basic functionality that can be extended over time, in response to user feedback
8) user and maintenance manuals will be provided for end-users and technical support, respectively

METHODS

MATERIALS AND INSTRUMENTS

Hardware: CLSD runs on the IU Research Supercomputer that is maintained by Research and Technical Services at UITS. Access requires two accounts: one for the Research Supercomputer and another for CLSD. The Medical and Molecular Genetics department used a Sun Ultra 60 computer running UNIX Solaris 8 OS for this project.

Software: MS Access and Excel were used for data modeling in the development stage. The end-user interface was implemented as a Web browser using hypertext markup language (HTML), common gateway interface (CGI) code, and Apache 1.3.26 web server software. The HTML code was written to be compatible with both Netscape and MS Internet Explorer. A PERL/DB2 client, version 7, was installed in Medical and Molecular Genetics to establish a remote connection with CLSD. Programming in PERL version 5.8.0 was used to implement an interface between the web browser and the DB2 client.

PROCEDURES AND INTERVENTIONS

Definition of end-user requirements: In the early stages of this project, a substantial amount of time was devoted to meeting with people in the department to understand how research in the department was organized, what types of data were needed by different projects at different stages, and how online resources like NCBI were being used.

A major focus of research in the Department of Medical and Molecular Genetics is identifying the location and nature of genes and mutations that cause human disease. This research is broadly organized into two stages: 1) family linkage studies and 2) linkage disequilibrium studies. Family linkage studies are used to do whole genome screens that can localize a disease gene to a chromosome segment measuring up to 20 centiMorgans (cM) in length. These are conducted by tracking the inheritance of microsatellite chromosome markers through affected and unaffected members of families with a genetically determined disease. Affected family members will usually have the same alleles for microsatellite markers that are located close to a disease gene.

Once a disease gene has been localized to a shorter chromosome segment defined by a pair of markers, the next step is to gather as much information as possible about all the known genes on that segment. This is usually done by accessing online databases at NCBI and other web sites. This information is then used to select a small number of genes for further study. Selecting genes for further investigation is a critical decision, because substantial resources must be committed to evaluating each gene. To make this decision, as much relevant data as possible is gathered for each candidate gene. This is a time consuming process when done manually, because a typical chromosome segment may contain 300 or more genes, and the relevant data for each gene may be distributed over multiple web pages. There was a strong consensus in the department that this was the most important process that needed to be automated.

Once the above information is gathered and a small number of high-priority candidate genes have been identified, linkage disequilibrium studies are used to determine the likelihood that one or more of these genes is related to the disease in question. These studies use single nucleotide polymorphisms (SNP’s), or point mutations, to identify haplotype blocks, which are a series of SNP’s that are located in or adjacent to a target gene and inherited as a group in individuals with the disease. The selection of high-quality SNP’s is a critical decision in the design of a linkage disequilibrium study. Collecting as much relevant data as possible for each candidate SNP from one or more online databases is another time-consuming, manual process in the department. There was a strong consensus in the department that automating this process was the second highest priority.

The end-user requirements for data collection at the marker-to-gene selection stage were defined as follows:

Gene ID	abbreviation	symbol (PCDH12)
	short description	gene name (protocadherin 12)

	long description	paragraph describing gene function
Gene Size	DNA	full length of open reading frame
	mRNA	full length of mRNA transcript
		processed mRNA (exons only)
	protein	number of amino acids
Gene location	chromosome number	
	chromosome coordinates (bp)	
Tissue expression	filter genes expressed in specific organs or tissues	

Tissue expression is particularly important because a disease gene is usually expressed in the tissue or organ that is affected by the disease. Filtering on tissue expression can reduce the number of candidate genes for further investigation by 70% or more. The department expressed a strong desire for a GUI that would allow filtering on this field.

The end-user requirements for data collection at the gene-to-SNP selection stage were defined as follows:

SNPs	SNP ID
	allele ID’s (nucleotide substitution)
	allele frequencies
	contig and chromosome coordinates
	function: intron or exon, amino acid substitution, etc.
	% heterozygosity
	sample size statistics
	validation

Allele frequencies and percent heterozygosity are important because SNP’s that are extremely rare are of little value in defining haplotype blocks that are more frequently observed in individuals with the disease. Sample size statistics and validation status are of interest because many of the SNP’s that are currently available are based on preliminary reports, and require further investigation to confirm that they are not artifacts or errors of the genome sequencing process. SNP location and function are important because SNP’s that are located in exons and cause amino acid substitutions are more likely to be related to disease expression if and when the disease gene is identified.

End-user requirements for the GUI were also discussed at this stage. The department expressed a strong preference for an HTML-based GUI that would require little or no training on the part of the end-users. There was also a strong preference for output tables that could easily be moved into MS Excel or Access.

Data Model and SQL: The end-user requirements were initially defined in terms of information that was being obtained manually from the NCBI web pages. The next stage in the development process was to translate these requirements into data elements that could be found in, or brought into CLSD, and organized into queries with well-defined input and output fields.

Most, but not all, of the data elements listed in the initial user requirements were incorporated into the final version of the software. For a number of reasons, some of the data elements that were originally requested were difficult to obtain from NCBI and/or extract from CLSD. In all of these cases, however, substitutes were found that were acceptable to the department.

The data model for the marker-to-gene selection query is shown in Figure 1. A data dictionary with field definitions is provided in Appendix A.

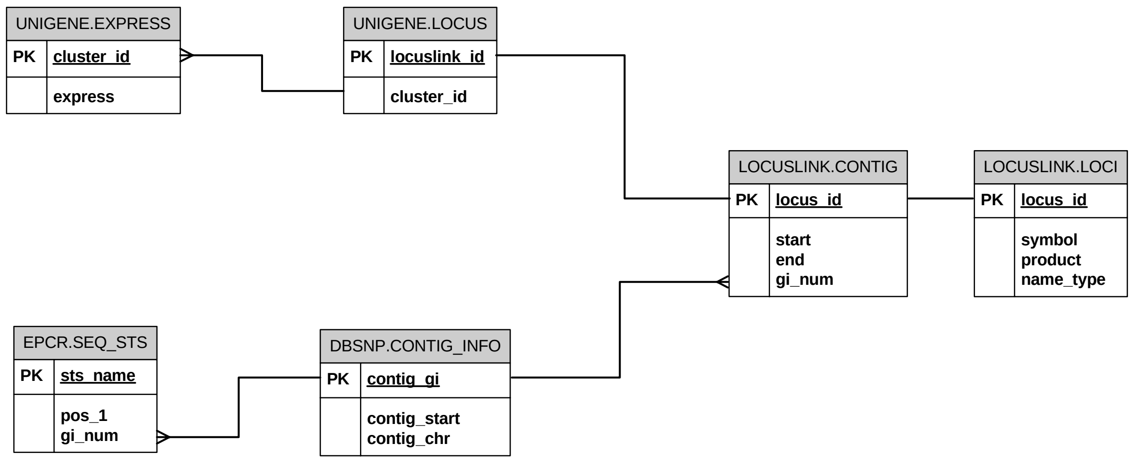


Figure 1. Data model for the marker-to-gene selection query.

The output table is generated by a compound query. The first SQL statement retrieves the chromosome number and coordinates for the two markers entered by the user:

```
select sts_name,
      contig_chr as chr,
      pos1+contig_start-1 as chrpos

from epcr.Seq_STS,
      dbsnp.contiginfo

where (sts_name in ([marker input] , [marker input]))
            and (gi_num=contig_gi)

order by chrpos;
```

The chromosome positions of the markers, along with user input from the tissue filter field, are passed on to a second SQL statement, which retrieves the chromosome position, gene length, symbol, product name, name type, and tissue expression for all known genes on the chromosome segment defined by the two markers:

```
select pos.locus_id as locus_id,
      pos.chrpos as chrpos,
      gene_length,
      ll.symbol as symbol,
      ll.product as product,
      ll.name_type as n,
      express as tissue

from (select c.locus_id,
            contig_start+start-1 as chrpos,
            end-start as gene_length

      from locuslink.contig as c,
            dbsnp.contiginfo

      where (gi_num=contig_gi)

            and (((contig_start+start - 1 > [chr position 1])
            and (contig_start + start - 1 < [chr position 2]))
            or ((contig_start + end - 1 > [chr position 1])
            and (contig_start + end - 1 < [chr position 2])))

            and (source = [chr number])) as pos

left outer join locuslink.loci as ll
on pos.locus_id=ll.locus_id

left outer join unigene.locus as ul
on pos.locus_id=ul.locuslink_id

left outer join unigene.express as ue
on ul.cluster_id=ue.cluster_id

where (express like '%[tissue filter input]%')
order by chrpos;
```

To support these queries, it was necessary to add the Epcr.Seq_Sts and LocusLink.Contig tables to the CLSD database. Epcr.Seq_Sts is a table from the UniSTS database that gives contig positions for chromosome markers. The LocusLink.Contig table gives contig positions for the beginning and end of each gene. CLSD also modified the LocusLink.Loci table to include both official and unofficial (preferred) terms in the gene symbol and product name fields. Initially, these fields had included only terms that were officially approved by the HUGO Gene Nomenclature Committee, which resulted in a significant number of missing values. A Name_Type field was also added to this table to indicate whether the nomenclature was official (O) or Preferred (P).

The data model for the gene-to-SNP selection query is shown in Figure 2. A data dictionary with field definitions is provided in Appendix B.

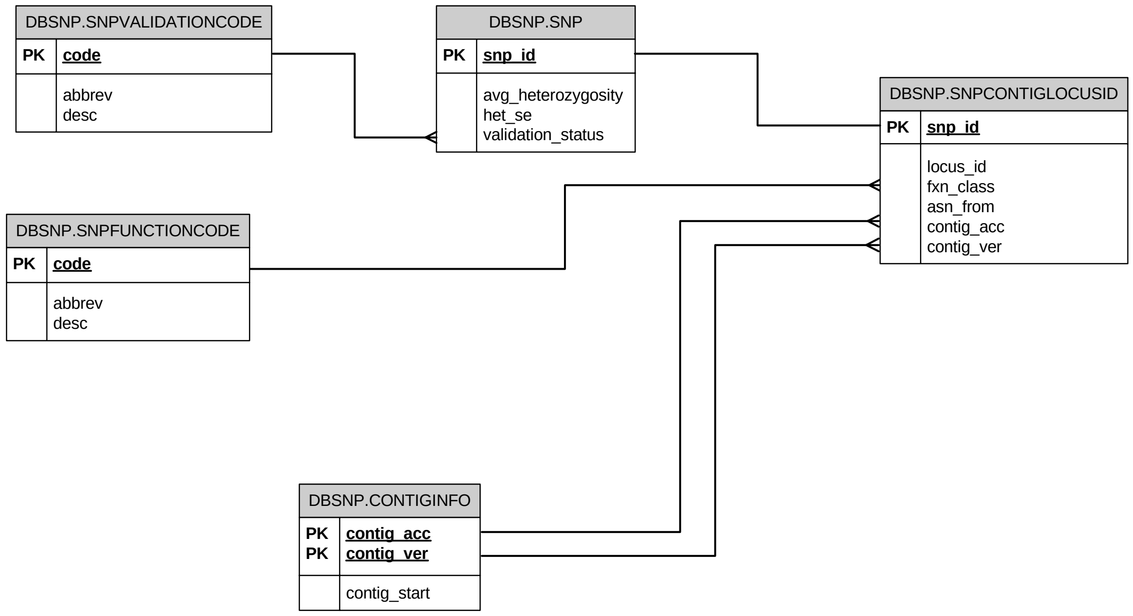


Figure 2. Data model for the gene-to-SNP selection query.

This query requires a locus id input from the end-user, and returns the SNP id, chromosome position, average heterozygosity, heterozygosity standard error, functional class, and validation status for all SNP's within 2000 base pairs (bp) upstream or 500 bp downstream from the gene boundaries:

```
select scl.locus_id,
      scl.snp_id,
      fxn_class,
      asn_from + ci.contig_start - 1 as chrpos,
      avg_heterozygosity as avg_hz,
      het_se,
      snp.validation_status as val

from dbsnp.snpcontiglocusid as scl,
      dbsnp.contiginfo as ci,
      dbsnp.snp as snp

where (locus_id= ? )
      and (scl.contig_acc=ci.contig_acc)
      and (scl.contig_ver=ci.contig_ver)
      and (scl.snp_id=snp.snp_id)

order by chrpos;
```

Installation of the DB2 client and PERL programming. The data model and SQL were developed in a DB2 environment using SSH Secure Shell to establish a direct connection with CLSD. The next stage in the development process was to incorporate the SQL into a PERL program that could interface with both the DB2/PERL client and an HTML-based GUI (Quigley 2002, Meltzer and Michalski 2002, Hanegan 2001). The DB2 client was installed and tested using DB2 commands that were executed from the command line in SSH Secure Shell. The PERL program and GUI were then developed in parallel, starting with a single input field, a simple test query, and a small output table. Functionality was added to the PERL code and the GUI in small increments, with error checking after each modification. Modifications to the PERL code were tested in two stages, once with input parameters supplied via the command line of the SSH shell/DB2 client interface, and a second time with input parameters submitted from the GUI. The code was structured so that new queries as well as further modifications to the existing queries could easily be incorporated at any stage of the development process.

The first section of the PERL program opens a connection to CLSD through the DB2 client. The rest of the program consists of two subroutines, one for each of the two queries available on the GUI input form. The first section in each subroutine assigns values from the GUI input form to variables in the PERL program. These values are then incorporated into an SQL statement that is submitted to CLSD. The last section of each subroutine converts the data retrieved from CLSD into HTML code for an output table that is displayed on the web browser. Output tables in the web browser can be moved into

MS Excel using the standard EDIT/COPY and EDIT/PASTE commands for Windows applications.

A working prototype with basic functionality for the marker-gene selection table was placed online in August 2003, about 3½ months after the initial planning for this project was started. Basic functionality for the gene-to-SNP selection query was added a few days later. The PERL code for the prototype is given in Appendix C. The HTML code and a screen print of the web input form for the prototype are given in Appendices D and E, respectively.

PROJECT EVALUATION

Both formal and informal methods were used for project evaluation. The Department of Medical and Molecular Genetics holds weekly lab meetings where the progress of all of the projects in the department is reviewed. These meetings were very helpful in eliciting feedback and suggestions from the end-users. They were also helpful in keeping the department informed of progress, problems, and other developments. When the prototype was placed online, the department was advised that identifying and correcting errors in the prototype was a normal and important part of the development process, and they were encouraged to communicate any questions or problems that came to their attention. Feedback obtained from using the prototype was very helpful in guiding subsequent work on this project. As a result of the weekly lab meetings and the early prototype implementation, the end users in the department were intimately involved in the development of this application over the entire course of the project.

Formal evaluation consisted of manually crosschecking the data in the output tables against data obtained from CLSD and NCBI. When the prototype was placed online, the department was cautioned that the application was still under development and had not yet been fully validated. End-users in the department submitted a number of output tables for manual validation before basing research decisions on the results. These crosschecks were a critical part of the evaluation process.

EXPECTED RESULTS

The primary objective of this project was to automate the process of obtaining online biological data for the Medical and Molecular Genetics department. A substantial increase in efficiency was anticipated over the manual processes that were previously used. An increase in data quality was also anticipated, because human error is inherent in repetitive, manual processes, and a thorough validation phase was planned after the prototype was placed online. It was also anticipated that automating these processes would allow more frequent access to updated data.

RESULTS

INTRODUCTION

Overall, the accuracy of the output tables generated by the prototype was very good. A number of errors and other problems were identified during the validation process, but most of these were minor. In all cases, the errors were corrected or an alternative solution was implemented to correct the problem. Overall, the end-users expressed a high level of satisfaction with the GUI, the output tables and the validation process.

IMPORTANT HIGHLIGHTS

A substantial number of improvements and other modifications were implemented after the prototype was placed online. Most of these fell into two general categories: 1) those implemented as a result of user feedback, and 2) those implemented as a result of manual cross-checking of the data in the output tables against data obtained from CLSD and the NCBI web site.

A number of additional modifications were necessary because of maintenance, updates, and other changes at CLSD and NCBI. The SNP database at NCBI is updated every 1-2 months; these updates frequently include changes to the data model. NCBI also implements new whole-genome builds on a less frequent basis. CLSD upgraded their DB2 software from version 7 to version 8 in November 2003.

SPECIFIC FINDINGS

This section is divided into four parts: 1) manual validation of the marker-to-gene output tables, 2) manual validation of the gene-to-SNP output tables, 3) modifications based on user feedback, and 4) maintenance and updates.

Manual validation of marker-to-gene output tables.

1) The spheroid body myopathy project has localized a gene for this disease to a segment on chromosome 5 bordered by markers D5S2057 and D5S436. The prototype was used to generate an output table using these markers and the term “muscle” in the tissue expression field. The data in the prototype table was compared with a spreadsheet that was created manually using data collected from the NCBI web site in May 2003, as part of the initial planning for this project.

The browser correctly identified all of the 67 genes on this chromosome segment that were expressed in muscle tissue. There were no errors in the locus ID, gene symbol, or product name fields. The chromosome start position for each of the genes was off by one base pair. This error occurred because the origin for the chromosome and contig coordinate systems is base pair number 1, not 0. This error was corrected by making a minor modification in the SQL and PERL code.

Six genes were present in the August prototype output that were not on the May 2003 spreadsheet:

LOCUS ID	GENE SYMBOL
6879	TAF7
55374	PRO1580
348944	not available
340061	LOC340061
51128	LOC51128
3308	HSP04

The prototype output table was consistent with the data in CLSD and NCBI as of August 2003. These genes appear to have been added to NCBI's database between May and August of 2003.

Another locus ID had different nomenclature in the prototype output than May 2003 spreadsheet:

LOCUS ID	DATE	SYMBOL	DESC
9879	5-03	KIAA0801	RNA helicase
9879	8-03	DDX46	DEAD Asp-Glu-Ala-Asp box peptide 46

The nomenclature in the prototype output table was consistent with the data in CLSD and the NCBI web site as of August 2003. The symbol and description reported in May 2003 were listed on the NCBI web site as alternate nomenclature in Aug 2003. This discrepancy appears to be due to a revision in the nomenclature for this gene sometime between May and August 2003.

Another locus ID, 84105, was present on the May 2003 spreadsheet, but not the prototype output table generated in August, because the tissue expression field included leiomyosarcoma, but not muscle. The Medical Genetics department did not consider this gene to be a good candidate for further investigation because tumors frequently express many genes that are not active in normal tissue. However, this discrepancy did raise the possibility that in some cases, searching on a single term in the tissue expression field might not be sufficient. This problem is discussed in greater detail in the next example.

2) The osteoporosis project is investigating a gene for this disease that has been localized to chromosome 15 between markers D15S1507 and D15S131. The prototype was used to generate an output table using these markers and the term “bone” in the tissue expression field. The data in the prototype table was compared with data obtained directly from NCBI and CLSD.

The browser correctly identified 23 genes on this chromosome segment that were expressed in bone. All of the data in the prototype table for these genes were correct. Four additional genes were found on the NCBI web site that were not in the prototype output because tissue expression was reported using a term other than bone:

SYMBOL	TISSUE EXPRESSION
DPP8	osteosarcoma, myeloma, Ewing’s sarcoma
FLJ10036	Ewings sarcoma
MADH6	osteosarcoma
FLJ11506	trabecular meshwork, osteosarcoma

Three of these genes were expressed only in tumors and considered low priority by the osteoporosis project. The remaining gene, FLJ1156, was a more serious concern because its expression had been reported in trabecular meshwork, or normal bone tissue. The terms in this field are a free-text vocabulary, which is selected by the investigators that report the tissue expression in the scientific literature. These terms are not converted to a controlled vocabulary when the data is entered into the UniGene database at NCBI.

As a result of these findings, the end-users in the department were advised that in some cases, searching on a single term in the tissue expression field is not sufficient. After further discussions within the department, a link to the Medical Subject Headings

(MeSH) database at the National Library of Medicine (NLM) was added to the prototype input form to help identify additional search terms that might be relevant. MeSH is a standardized, hierarchical medical vocabulary that can be used to identify synonyms. It can also help the user identify additional search terms that are more specific or general. For example, if a search is conducted on "bone", MeSH will return a number of related medical terms, including osteoblasts, osteoclasts, and osteocytes. A second prototype query using the same chromosome markers and the tissue expression term "osteo" would have detected three of the four genes that were missed in the output table that was generated using only the term "bone". A detailed discussion of this issue and instructions for using the MeSH browser were included in the user manual (Appendix F).

One missing URL link was found on the NCBI web site while reviewing this table. The prototype reported one additional gene, Locus ID 338946, on this chromosome segment with expression in bone, but there was no link to UniGene on the LocusLink web page for this gene. However, when the UniGene ID for this gene was retrieved from CLSD and entered into the standard UniGene URL, NCBI returned a web page with tissue expression data that was consistent with the CLSD and the prototype output table.

3) The osteoporosis project is investigating another gene for this disease that has been localized to a segment on chromosome 14 between markers D14S588 and D14S592. The prototype was used to generate an output table using these markers with the term "bone" in the tissue expression field. The data in the prototype table was compared with data

obtained directly from NCBI and CLSD. The browser correctly identified all but one of the 18 genes on this chromosome segment that were expressed in bone.

NCBI and CLSD both confirmed that MNAT1 lies within the marker interval and is expressed in bone, but this gene was not included on the browser output because the gene straddles the boundary of the chromosome segment defined by the two markers. The SQL was initially written to select genes where the gene start point was located between the two chromosome markers. MNAT1 straddles the lower boundary of this chromosome segment: the endpoint of the gene is included in the marker interval but the starting point is not. This problem was corrected by modifying the PERL and SQL code to select genes where any portion of the gene segment falls between the two chromosome markers. No other new problems were identified in reviewing this table.

4) The Parkinson’s disease project is investigating another gene for this disease that has been localized to a segment between markers D2S396 and D2S338 on chromosome 2. Browser output was generated using these markers along with the term “brain” in the tissue expression field. The data in the prototype table was compared with data obtained directly from NCBI and CLSD.

The browser correctly identified 28 of 37 genes on this segment that are expressed in brain tissue. Nine genes on this chromosome segment were missed because tissue expression was reported using a term more specific than “brain”:

SYMBOL	TISSUE EXPRESSION
LOC93349	hypothalamus and medulla

LOC348761	medulla and glioblastoma
MGC35154	medulla
LOC344562	medulla
CHRND	neuroblastoma
LOC339766	astrocytoma
B3GNT7	neuroblastoma
DKFZp762E1312	neuroblastoma
LOC150933	nervous tumor

Of these nine genes, five were reported only in tumors and were considered to be low priority by the Parkinson’s disease project. The other four genes were reported in hypothalamus and/or medulla. Due to the complexity of brain anatomy, a MeSH search did not readily suggest that these might have been useful terms for repeating the marker-genetable query.

One error was also found on the NCBI web site while validating this table. The LocusLink web page for TIGD1 links to the same UniGene web page as EIF4EL3, so the reported tissue expression for these two genes is identical. The data on the browser output for both TIGD1 and EIF4EL3 was consistent with the data on NCBI and CLSD.

Marker-to-gene table validation summary. The validation process identified a number of minor errors in the PERL and SQL code that were easily corrected. The most significant problem with this table was genes that were not included in the output tables because their tissue and organ expression was reported under alternative or non-standard terms in the NCBI and CLSD databases. This problem seems to be most significant for tissues and organs with complex anatomy and physiology. Of the 152 “target” genes on these four chromosome segments, the browser correctly identified 136 using standard search terms, for an accuracy rate of 89%. An additional 4 genes could have been detected by repeating the search with additional tissue expression terms obtained from MeSH, which raises the accuracy to 92%. Of the remaining 11 genes that were not retrieved by commonly used terms for tissues and organ systems, 7 were expressed only in tumors and were not considered good candidates for further investigation by research projects in the department. The number of potentially significant genes that were missing from the output tables was therefore 5/152, or 3%.

Unfortunately, this error rate still presents a significant problem if the tissue expression for a "target" disease gene is reported unusual or non-standard terms that are not included in the MeSH vocabulary. However, the consensus in the department was that the error rate for this application was an acceptable tradeoff for the substantial increase in efficiency that was achieved by automating this process.

Manual validation of gene-to-SNP output tables.

1) The osteoporosis project is conducting linkage disequilibrium studies on the frizzled-related gene, FRZB, on chromosome 2. The locus ID for this gene, 2487, was used to generate an output table that was compared with data obtained directly from NCBI and CLSD.

The browser correctly identified all of 67 the SNP's listed for this gene in the CLSD database. The data in the locus ID, SNP ID, functional class, average heterozygosity, heterozygosity standard error, and validation status fields were all consistent with the data in the CLSD database. The chromosome positions for the SNP's were initially off by one base pair. This error was corrected in a similar manner to that described for the marker-to-gene output table.

There were some significant discrepancies between the data in the output table and the NCBI web site. At the time the prototype was implemented, CLSD was using data based on build 114 of the SNP database (dbSNP). The data on the NCBI web site at that time reflected build 116. New dbSNP builds frequently include changes in the data model, which requires a significant amount of programming for CLSD to incorporate. CLSD was eventually able to automate some of the work that is required to incorporate dbSNP updates, but in the early stages of this project they did not have the resources to incorporate all of these updates manually.

For locus ID 2487, CLSD and the prototype output table captured 68 of the 87 SNP's (78%) that were listed on the NCBI web site for build 116. A review of the new SNP's revealed that most of these were poorly characterized with respect to validation status. Furthermore, most of the new SNP's had no reported values for average heterozygosity. For this reason, the department felt that most of these SNP's would not have been good candidates for further investigation, even if they had been reported in the output table. However, of the SNP's that were present in both builds, about 35%, had a revised validation status. In most cases, the validation status was improved or upgraded between the two builds. This was considered to be important information for selecting SNP's for linkage disequilibrium experiments. For this reason, the users in the department were advised to submit all SNP output tables for cross-validation before making any important decisions based on the results, until the data in CLSD had been updated.

2) The Parkinsons Disease project requested a review of a gene-to-SNP table generated with locus ID 81618. This gene lies on chromosome 2 and codes for integral membrane protein 2 (ITM2C).

This table correctly reported all of the SNP’s listed for this gene in the CLSD database. The data in the locus ID, SNP ID, functional class, average heterozygosity, heterozygosity standard error, and validation status fields were all consistent with the data in the CLSD database. For this gene, CLSD and the prototype output table captured 55 of the 62 SNP’s (88%) that were listed on the NCBI web site for build 116. No new problems were identified when validating this table.

On October 29, 2003, the PERL code was modified to use dbSNP tables from build 117, which had been incorporated by CLSD several days prior. CLSD has since been able to automate most of the processes necessary to update dbSNP, so that new dbSNP data can be incorporated into CLSD within 2-3 days after it is released from NCBI. Once the new build is available from CLSD, the SQL and PERL code on our web server can generally be updated to take advantage of the new data within another 2-3 days.

Gene-to-SNP table validation summary. The validation process identified one minor error in the SNP positions that that was corrected by modifying the PERL and SQL code. In the early stages of prototype implementation, there was a significant delay between the release of new dbSNP builds and the incorporation of the new data into the CLSD database. CLSD has since automated most of these processes, so that new dbSNP data is now available within a matter of days. These issues are discussed in greater detail in the maintenance and updates section.

Modifications based on user feedback.

The following modifications in the marker-to-gene selection query were implemented based on feedback from users in the department:

1. The marker and tissue expression fields were modified to be case insensitive. Input was accepted as upper, lower, and title case letters. For the tissue expression field, the PERL code searches the CLSD database for terms in upper, lower, and title case, regardless of the input.
2. The tissue expression field was made optional by modifying the PERL code to accept null values.
3. HTML links to the STS and LocusLink home pages at NCBI were added to the GUI input form, to simplify the process of looking up marker and gene ID's.

The following modifications to the gene-to-SNP selection query were also implemented:

1. Heterozygosity standard error and validation status fields were added to the output table. Measures of SNP quality were included in the user requirements but not available in the first version of the prototype because the user requirements had not yet been reconciled with the data available at NCBI and CLSD.

Specifically, some of the measures available on the NCBI web pages were not explicitly represented in dbSNP. When SQL was added to derive these values the queries took much longer to run. Alternate measures were subsequently identified that were acceptable to the department.

2. A lookup table for the validation status codes was added to the GUI input form.
3. The descriptors for the function class code table on the GUI input form were improved, based on additional information obtained from NCBI.
4. The data values displayed in the heterozygosity and heterozygosity standard error fields were changed from scientific notation to decimal format.
5. An HTML link to the SNP home page at NCBI was added to the GUI input form, to simplify the process of looking up SNP ID’s.

Two additional modifications were implemented based on recommendations from UITS:

1. Password protection was added to the web input form, because the data in CLSD is not licensed for use outside the university.
2. New departmental accounts for exclusive use by the browser were created at CLSD and the Research SP. Student and staff passwords in the PERL code were felt to represent a higher risk if the security of the web server was ever compromised. Another reason for having a departmental account is that the username and password in the PERL program does not have to be changed when students or staff leave the department or the university.

User feedback and modification summary. A significant number of enhancements and modifications were implemented based on user feedback after the prototype was placed online. It would have been difficult to anticipate the need for many of these improvements without the involvement of the end-users.

Maintenance and updates.

The first few months after the prototype was placed online, a significant amount of maintenance was necessary to keep up with changes at CLSD and NCBI. On October 13, 2003, CLSD upgraded DB2 from version 7 to version 8. This upgrade required the installation of a new DB2 client in our department. There were a number of technical problems affecting both CLSD and the DB2 client that took several weeks to resolve. User access was uninterrupted over most of this time because the browser remained connected to a version 7 DB2 client in the department and the old version of CLSD, which UITS continued to support. During this time, CLSD also incorporated dbSNP build 116 into CLSD running on DB2 version 8. We were unable to take immediate advantage of this because the installation of our version 8 DB2 client had not yet been completed. CLSD subsequently implemented dbSNP build 117 on October 22. The browser was successfully transitioned to DB2 version 8 and dbSNP build 117 on October 29.

A number of modifications were made to the PERL code at this time. The DB2 connect statement was redirected to CLSD2, the new name given to CLSD running on DB2 version 8. The prefixes on all of the dbSNP table names were changed from dbsnp114 to dbsnp117. The gene-to-SNP query was also modified because NCBI eliminated the validation status table and moved the validation status code to another table. NCBI also modified the validation status codes and descriptors. This did not change the data model, so it did not affect the PERL/SQL code, but it did require updating the reference table on the web input form.

In October, we also became aware of a problem with relational integrity when the web input form stopped working for some, but not all, input values. On further investigation, it was found that a number of genes in the LocusLink.Contig table had contig id numbers with no corresponding values in the dbSNP114.contiginfo table. This problem was traced to a new genome build that NCBI released in October. Whenever a new genome build is created, a number of chromosome contigs are revised or consolidated, and given new id numbers. CLSD updates the data in LocusLink twice weekly, so the new contig id numbers were reflected immediately in this database at CLSD. However, these contig id numbers were not yet reflected in the dbSNP database at CLSD. CLSD implemented a short-term solution for this problem by putting the older LocusLink tables that were compatible with dbSNP114 back online so they could be used on an interim basis. These were no longer needed once the browser was updated to dbSNP build 117 on October 29. CLSD has since automated most of the processes for incorporating new dbSNP builds, so this can now be done within 2-3 days. In addition, CLSD and Medical Genetics now subscribe to email lists from both dbSNP and NCBI. These provide advance notice whenever a new dbSNP or genome build is planned, so that appropriate updates for CLSD and the CLSD browser can be implemented as quickly as possible.

Another new problem that emerged with dbSNP build 117 was the response time for the gene-to-SNP query. This was running at 2-3 minutes on build 114 and increased to 12-15 minutes with build 117. The response time for the marker-to-gene table was unchanged. This problem was apparently related to the design of build 117. NCBI released dbSNP build 118 about the same time that build 117 was incorporated into the CLSD. Build 118 included a substantial increase in the size of the database, which we suspected was due to indexing. dbSNP build 118 was incorporated into CLSD about one week after its release by NCBI; this reduced the response time for the gene-to-SNP query to 2-3 seconds.

Maintenance and updates summary. For the first 2-3 months after the prototype was placed online, a considerable amount of time and effort was necessary to keep up with changes at both CLSD and NCBI. Valuable experience was gained during this period. It will never be possible to anticipate all of the issues raised by changes at NCBI and CLSD, but at the present time, many of the issues related to these processes have been addressed at least once, and good solutions are in place or evolving to meet these needs. It is anticipated that maintenance and upgrades will continue to become more efficient as further experience is gained. A maintenance manual which summarizes this experience to date is included in Appendix G.

VALIDATION SUMMARY

The final form for the PERL program, HTML code, and the web input form are given in Appendices H, I, and J, respectively. A high level of accuracy was achieved in the final version of this application. The biggest problem encountered during the validation of the marker-to-gene table output was the lack of a controlled vocabulary for the tissue expression field in the UniGene database. Although this problem was not eliminated, an acceptable solution for the department was found. The primary problem that was identified while validating the gene-to-SNP output tables was the delay in incorporating new dbSNP builds from NCBI into the database at CLSD. The time currently required to incorporate new dbSNP builds has been reduced to 2-3 days by CLSD and another 2-3 days in the genetics department. Maintenance will continue to become easier and more efficient as more experience is gained with the issues involved. Early implementation of a working prototype and end-user involvement was very helpful in guiding further development of this project. This approach requires good, two-way communication with end-users who understand that further development and validation is necessary before the application becomes fully functional. The end-users expressed a high level of satisfaction with the development process and the final version of this application.

CONCLUSION

Improved methods for accessing and integrating large volumes of biological data from multiple, complex sources is a critical need for research in molecular biology and a formidable challenge for the field of bioinformatics. This project demonstrates that it is possible to develop software tools that allow integrated access to large, complex data sources that require only minimal technical expertise or training on the part of the end-users. However, significant involvement by the end-users in the development process is necessary to ensure that these applications will meet their needs. The successful development these tools also requires a substantial amount of time and effort for successful implementation. The success of this project was heavily dependent on the availability of a centralized data repository. Large, integrated resources like these require a substantial investment of resources and close coordination between the relevant databases, software vendors, and the organizations where the data will ultimately be used. Automating and increasing the efficiency of database integration and improved methods for accessing these resources remains an active area of research and development in the fields of molecular biology and bioinformatics.

DISCUSSION

LIMITATIONS OF THE STUDY

The primary limitation of this project is the number and flexibility of the queries that are currently available. This project required a substantial investment of time and effort in an evaluation of the department's data requirements and the development of an infrastructure that would support programmatic access to a centralized data repository. This infrastructure is scalable; once it is in place, it is relatively easy to add new queries and extend the functionality of the queries that are already available.

RECOMMENDATIONS FOR FUTURE DEVELOPMENT

A number of enhancements and improvements for continued development of this application are planned or in progress. These include:

1) multiple tissue filter fields with Boolean AND/OR options for the marker-to-gene table

2) homologue tables to identify similar genes in multiple species, including humans, mice, and rats

3) a marker table to aid in the selection of chromosome markers for human pedigree studies

4) further work to increase the efficiency of maintenance, updates, and upgrades

5) transition of primary responsibility for the application to a permanent, full-time staff member in the Medical and Molecular Genetics department.

REFERENCES

1. Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, and Wheeler DL. GenBank. Nucl. Acids. Res. 31(1), 28-33. 2003.
2. Blake JA, Richardson JE, Bult CJ, Kadin JA, and Eppig JT. MGD: the Mouse Genome Database. Nucl. Acids. Res. 31(1), 193-195. 2003.
3. Chicurel, M. 2002. Bioinformatics: bringing it all together. Nature 419(6908):751-757.
4. Hanegan, Kevin. Custom CGI Scripting with PERL. Copyright 2001 by John Wiley and Sons, Inc.
5. Karolchik D, Baertsch RM, Diekhans TS, Furey A, Hinrichs YT, Lu KM, Roskin M, Schwartz CW, Sugnet DJ, Thomas RJ, Haussler WD, and Kent WJ. The UCSC Genome Browser Database. Nucl. Acids. Res. 31: (1), 51-54. 2003.
6. Lacroix, Z. Biological data integration: wrapping data and tools. 2002. IEEE Trans Inf Technol Biomed. 6(2):123-8.
7. Meltzer, Kevin and Michalski, Brent. Writing CGI Applications with PERL. Copyright 2002 by Addison-Wesley.
8. Pruitt KD, Tatiana T, and Maglott DR. NCBI Reference Sequence Project: update and current status. Nucl. Acids. Res. 31(1), 34-37. 2003.
9. Quigley, Ellie. PERL by Example, Third Edition. Copyright 2002 by Prentice-Hall, Inc.
10. Thorisson GA and Stein LD. The SNP Consortium web site: past, present and future. Nucl. Acids. Res. 31(1), 124-127. 2003.

11. Stein LD. Integrating Biological Databases. Nature Genetics 4, 337-345. 2003.

12. Stevens RD, Robinson AJ, Goble CA. 2003. myGrid: personalized bioinformatics on the information grid. Bioinformatics 19 (Suppl 1):i302-i304.

13. Sujansky, W. Heterogeneous database integration in biomedicine. Journal of Biomedical Informatics 34, 285-298. 2001.

14. Wilkinson, MD, Gessler, D., Farmer, A., Stein, L. 2003. The BioMOBY Project Explores Open-Source, Simple, Extensible Protocols for Enabling Biological Database Interoperability. Proc Virt Conf Genom and Bioinf (3):16-26.

15. Wheeler DL, Church DM, Federhen S, Lash AE, Madden TL, Pontius JU, Schuler GD, Schriml LM, Sequeira E, Tatusova TA, and Wagner L. Database resources of the National Center for Biotechnology. Nucl. Acids. Res. 31(1), 28-33. 2003.

APPENDIX A: DATA FIELD DEFINITIONS FOR THE MARKER-TO-GENE SELECTION QUERY

<u>EPCR.SEQ_STS</u>

sts_name	unique id for marker in NCBI UniSTS database (e.g. D14S588)
gi_num	unique id for the contig that the marker is located on, in NCBI RefSeq database
pos_1	starting position for marker, in contig coordinates (bp)

<u>DBSNP.CONTIG_INFO</u>

contig_gi	unique contig id
contig_chr	contig chromosome
contig_start	starting position for contig, in chromosome coordinates (bp)

<u>LOCUSLINK.CONTIG</u>

locus_id	unique id for gene in NCBI LocusLink database
start	start position for gene, in contig coordinates (bp)
end	end position for gene, in contig coordinates (bp)
gi_num	unique id for the contig that the gene is located on, in NCBI RefSeq database

<u>LOCUSLINK.LOCI</u>

locus_id	unique id for gene in NCBI LocusLink database
symbol	abbreviation for gene (e.g. PCDH12)
product	short description of gene product (e.g. protocadherin 12)
name_type	indicates whether gene symbol and product are official nomenclature approved by HUGO (O), or unofficial but preferred (P); this field is not represented as a distinct field in NCBI LocusLink, but is derived by CLSD from the LocusLink flat file

<u>UNIGENE.LOCUS</u>

locuslink_id	unique id for gene in NCBI locuslink database
cluster_id	unique id for EST/mRNA cluster in NCBI UniGene database

<u>UNIGENE.EXPRESS</u>

cluster_id	unique id for EST/mRNA cluster in NCBI UniGene database
express	free-text field that lists all tissues and organs in which expression has been reported

APPENDIX B: DATA FIELD DEFINITIONS FOR THE GENE TO-SNP SELECTION QUERY

DBSNP.SNP

snp_id	unique id for SNP in NCBI dbSNP database
avg_heterozygosity	average heterozygosity, expressed in percent, to two decimal places
het_se	heterozygosity standard error, expressed in percent, to three decimal places
validation_status	code with value from 0 to 15 that describes what combination of methods was used for validation

DBSNP.CONTIGLOCUSID

snp_id	unique SNP id
locus_id	unique id for gene in NCBI LocusLink database
fxn_class	code with value between 1 and 9 which describes location of SNP in reference to gene, exon, or intron boundaries; also indicates whether the SNP results in an amino acid substitution in the gene product
asn_from	starting point for the gene on the contig, in bp
contig_acc	contig accession number, part of compound id for this table
contig_ver	contig version number, part of the compound id for this table; if a contig is revised between genome builds, the version number increases by one, but the accession number is unchanged

DBSNP.CONTIGINFO

contig_acc	part of compound id for this table; see above
contig_ver	part of compound id for this table; see above
contig_start	starting point for the contig on the chromosome, in bp; this value is used, together with asn_from, to calculate the starting position for the gene on the chromosome

DBSNP.SNPVALIDATIONCODE

[lookup table for validation status codes used in dbsnp.snp]

code	validation status code
abbrev	short description
desc	long description

DBSNP.SNPFUNCTIONCODE

[lookup table for function class codes used in dbsnp.contiglocusid]

code	function class code
abbrev	short description
desc	long description

APPENDIX C: PERL CODE FOR PROTOTYPE

```
#!/usr/local/bin/perl -w
# snp2.pl
# use snp.pl as template
# this program generates genetable with markers and tissue submitted
from browser
# generates snptable from with input=sgene and gene=locus_id submitted
from browser

use strict;

use DBI;
use DBD::DB2::Constants;
use DBD::DB2;
use CGI qw(:standard);
use CGI::Carp qw(fatalsToBrowser);

my $q=new CGI;

my $dbh = DBI->connect("dbi:DB2:clsd", "username", "password")
       or die "Couldn't connect to database: " . DBI->errstr;

my $input = $q ->param("input");
my $code = $q->param("code");

if ($input eq "marker") {
   markergenetable();
} elsif ($input eq "sgene") {
   snptable();
}

############################# GENETABLE SUBROUTINE ##############

sub markergenetable {
my $marker1 = $q->param("marker1");
my $marker2 = $q->param("marker2");

my $tissue = $q ->param("tissue");
my ($tissue1, $tissue2, $tissue3);
if ($tissue ne "") {
  $tissue1 = lc $tissue;
  $tissue2 = uc $tissue;
  $tissue3 = $tissue;
  $tissue3 =~ s/(\w+)/\u\L$1/g;
} else {
  die "Cannot run query without tissue field term";
}

my $stmt = "select STS_NAME, contig_chr as chr, pos1+contig_start as
chrpos
   from epcr.Seq_STS, dbsnp114.contiginfo
   where (sts_name in (? , ?))
      and (GB_ACC_NUM=contig_acc)
   order by chrpos";
```

```
my $sth = $dbh->prepare($stmt)
       or die "Can't prepare SQL statement: ", $dbh->errstr(), "\n";

$sth->execute($marker1,$marker2)
       or die "Can't execute SQL statement: ", $sth->errstr(), "\n";

my @row1  = $sth->fetchrow_array;
my ($mkr1,$chr1,$chrpos1) = ($row1[0],$row1[1],$row1[2]);
#print "chr: $chr\n\n";
#print "chrpos1: $chrpos1\n\n";

my @row2 = $sth->fetchrow_array;
my ($mkr2,$chr2,$chrpos2) = ($row2[0],$row2[1],$row2[2]);

$sth->finish();

my $stmt2 = "select pos.locus_id as locus_id,
                      pos.chrpos as chrpos,
                      gene_length, ll.symbol as symbol,
                      ll.product as product,
                      express as tissue,
                      ll.name_type as n

from (select c.locus_id, contig_start+start as chrpos, end-start as
gene_length
from locuslink.contig as c, dbsnp114.contiginfo
where (gi_num=contig_gi)
and (contig_start+start > ? )
and (contig_start+start < ? )
and (source = ? )) as pos

left outer join locuslink.loci as ll
on pos.locus_id=ll.locus_id

left outer join unigene.locus as ul
on pos.locus_id=ul.locuslink_id

left outer join unigene.express as ue
on ul.cluster_id=ue.cluster_id

where (express like '%$tissue1%')
   or (express like '%$tissue2%')
   or (express like '%$tissue3%')

order by chrpos";

my $sth2 =  $dbh->prepare($stmt2)
       or die "Can't prepare SQL statement: ", $dbh->errstr(), "\n";

$sth2->execute($chrpos1,$chrpos2,$chr1)
       or die "Can't execute SQL statement: ", $sth->errstr(), "\n";

print header();

#print "Chromosome interval lower limit: $chrpos1\n<br>";
#print "Chromosome interval upper limit: $chrpos2\n<br>";
#print "Chromosome: $chr1\n<br>";
```

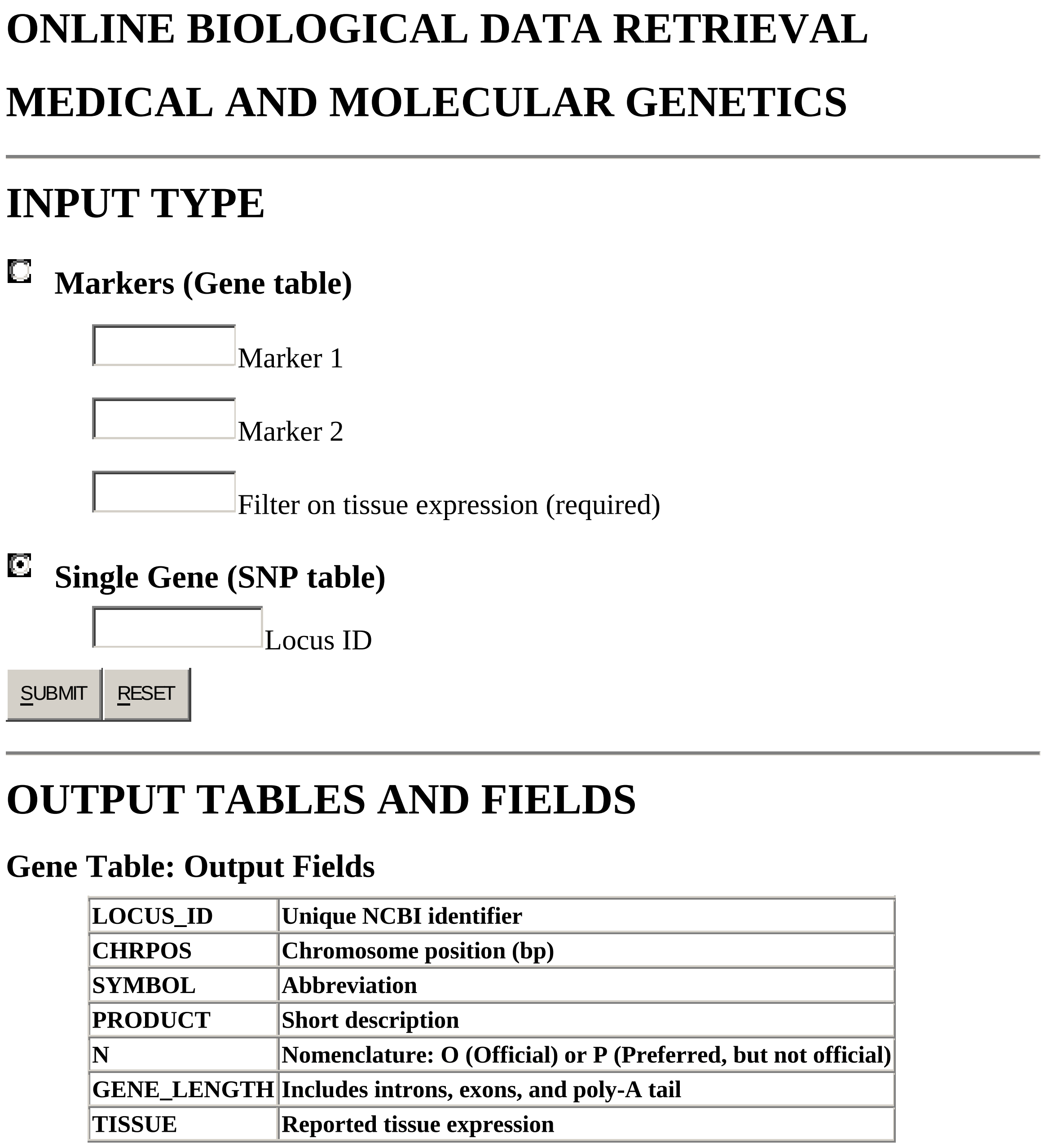
```
print <<HTML;
      <html><head><title>Query Results</title></head>
       <body>
        <table border="1" cellspacing="0">
         <tr>
          <td><b>MARKER</b></td>
          <td><b>CHR</b></td>
          <td><b>CHRPOS</b></td>
         </tr>
HTML

  print qq(<tr>\n);
  print qq(<td>$mkr1</td>\n);
  print qq(<td>$chr1</td>\n);
  print qq(<td>$chrpos1</td>\n);
  print qq(</tr>);

  print qq(<tr>\n);
  print qq(<td>$mkr2</td>\n);
  print qq(<td>$chr2</td>\n);
  print qq(<td>$chrpos2</td>\n);
  print qq(</tr>);
  print qq(<P>);

print <<HTML;
      <html><head><title>Query Results</title></head>
       <body>
        <table border="1" cellspacing="0">
         <tr>
          <td><b>LOCUS_ID</b></td>
          <td><b>CHRPOS</b></td>
          <td><b>SYMBOL</b></td>
          <td><b>PRODUCT</b></td>
          <td><b>N</b></td>
          <td><b>GENE_LENGTH</b></td>
          <td><b>TISSUE</b></td>
         </tr>
HTML

while(my $data = $sth2->fetchrow_hashref){
  print qq(<tr>\n);
  print qq(<td>$data->{LOCUS_ID}</td>\n);
  print qq(<td>$data->{CHRPOS}</td>\n);
  print qq(<td>$data->{SYMBOL}</td>\n);
  print qq(<td>$data->{PRODUCT}</td>\n);
  print qq(<td>$data->{N}</td>\n);
  print qq(<td>$data->{GENE_LENGTH}</td>\n);
  print qq(<td>$data->{TISSUE}</td>\n);
  print qq(</tr>);
}

$sth2->finish();
}

########################## SNPTABLE SUBROUTINE ####################
sub snptable {
```

```
my $gene = $q->param("gene");

#my $stmt = "select locus_id, snp_id, fxn_class
#from dbsnp114.snpcontiglocusid
#where locus_id= ? ";

my $stmt = "select scl.locus_id, scl.snp_id, fxn_class,
asn_from+ci.contig_start as chrpos, avg_heterozygosity as avg_hz
from dbsnp114.snpcontiglocusid as scl, dbsnp114.contiginfo as ci,
dbsnp114.snp as snp
where (locus_id= ? )
   and (scl.contig_acc=ci.contig_acc)
   and (scl.contig_ver=ci.contig_ver)
   and (scl.snp_id=snp.snp_id)
order by chrpos";

my $sth = $dbh->prepare($stmt)
       or die "Can't prepare SQL statement: ", $dbh->errstr(), "\n";

$sth->execute($gene)
       or die "Can't execute SQL statement: ", $sth->errstr(), "\n";

print header ();
#print "gene1: $gene1\n";

print <<HTML;
      <html><head><title>Query Results</title></head>
       <body>
        <table border="1" cellspacing="0">
         <tr>
          <td><b>LOCUS_ID</b></td>
          <td><b>SNP_ID</b></td>
          <td><b>CHRPOS</b></td>
          <td><b>FXN_CLASS</b></td>
          <td><b>AVG_HZ</b></td>
         </tr>
HTML

while(my $data = $sth->fetchrow_hashref){
  print qq(<tr>\n);
  print qq(<td>$data->{LOCUS_ID}</td>\n);
  print qq(<td>$data->{SNP_ID}</td>\n);
  print qq(<td>$data->{CHRPOS}</td>\n);
  print qq(<td>$data->{FXN_CLASS}</td>\n);
  print qq(<td>$data->{AVG_HZ}</td>\n);
  print qq(</tr>);
}

$sth->finish();

}

$dbh->disconnect;
```

APPENDIX D: HTML CODE FOR PROTOTYPE WEB INPUT FORM

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD><TITLE>cgi-test</TITLE>
<META http-equiv=Content-Type content="text/html; charset=iso-8859-1">
<META content="MSHTML 5.50.4934.1600" name=GENERATOR></HEAD>
<BODY>
<FORM action=http://www.medgen.iupui.edu/cgi-bin/binf/guidev/snp2.pl
method=post><BR>
<H2>ONLINE BIOLOGICAL DATA RETRIEVAL
<P>MEDICAL AND MOLECULAR GENETICS</H2>
<HR>
<H2>INPUT TYPE</H2>
<P>
<H3><INPUT type=radio CHECKED value=marker name=input> Markers (Gene
table)
<P></H3>
<UL>
  <P><INPUT name=marker1> Marker 1</P>
  <P><INPUT name=marker2> Marker 2</P>
  <P><INPUT name=tissue> Filter on tissue expression
(required)</P></UL>
<H3><INPUT type=radio value=sgene name=input> Single Gene (SNP table)
<P></H3>
<UL><INPUT name=gene> Locus ID
  <P></P></UL>
<P><INPUT type=submit value=SUBMIT><INPUT type=reset value=RESET>
<P></FORM></P>
<HR>

<H2>OUTPUT TABLES AND FIELDS</H2>
<P>
<H3>Gene Table: Output Fields
<P></H3>
<UL>
  <TABLE cellSpacing=0 border=1>
    <TBODY>
    <TR>
      <TD><B>LOCUS_ID</B></TD>
      <TD><B>Unique NCBI identifier</B></TD></TR>
    <TR>
      <TD><B>CHRPOS</B></TD>
      <TD><B>Chromosome position (bp)</B></TD></TR>
    <TR>
      <TD><B>SYMBOL</B></TD>
      <TD><B>Abbreviation</B></TD></TR>
    <TR>
      <TD><B>PRODUCT</B></TD>
      <TD><B>Short description</B></TD></TR>
    <TR>
      <TD><B>N</B></TD>
      <TD><B>Nomenclature: O (Official) or P (Preferred, but not
      official)</B></TD></TR>
    <TR>
      <TD><B>GENE_LENGTH</B></TD>
      <TD><B>Includes introns, exons, and poly-A tail</B></TD></TR>
    <TR>
      <TD><B>TISSUE</B></TD>
      <TD><B>Reported tissue
expression</B></TD></TR></TBODY></TABLE></UL>
<H3>SNP Table: Output Fields
<P></H3>
<UL>
  <TABLE cellSpacing=0 border=1>
    <TBODY>
    <TR>
      <TD><B>LOCUS_ID</B></TD>
      <TD><B>Unique NCBI identifier</B></TD></TR>
    <TR>
      <TD><B>SNP_ID</B></TD>
      <TD><B>Unique SNP identifier</B></TD></TR>
    <TR>
      <TD><B>CHRPOS</B></TD>
      <TD><B>Chromosome position (bp)</B></TD></TR>
    <TR>
      <TD><B>FXN_CLASS</B></TD>
      <TD><B>Functional class</B></TD></TR>
    <TR>
      <TD><B>AVG_HZ</B></TD>
      <TD><B>Average heterozygosity</B></TD></TR></TBODY></TABLE></UL>
<H3>SNP Table: Function Class Codes
<P></H3>
<UL>
  <TABLE cellSpacing=0 border=1>
    <TBODY>
    <TR>
      <TD><B>FXN_CLASS</B></TD>
      <TD><B>ABBREV</B></TD>
      <TD><B>DESC</B></TD></TR>
    <TR>
      <TD><B>1</B></TD>
      <TD><B>locus</B></TD>
      <TD><B>mrna_acc and protein_acc both null</B></TD></TR>
    <TR>
      <TD><B>2</B></TD>
      <TD><B>coding</B></TD>
      <TD><B>coding</B></TD></TR>
    <TR>
      <TD><B>3</B></TD>
      <TD><B>synonymous change</B></TD>
      <TD><B>synonymous change</B></TD></TR>
    <TR>
      <TD><B>4</B></TD>
      <TD><B>nonsynonymous change</B></TD>
      <TD><B>nonsynonymous change</B></TD></TR>
    <TR>
      <TD><B>5</B></TD>
      <TD><B>UTR</B></TD>
      <TD><B>untranslated region</B></TD></TR>
    <TR>
      <TD><B>6</B></TD>
      <TD><B>intron</B></TD>
      <TD><B>intron</B></TD></TR>
    <TR>
      <TD><B>7</B></TD>
      <TD><B>splice-site</B></TD>
      <TD><B>splice-site</B></TD></TR>
    <TR>
      <TD><B>8</B></TD>
      <TD><B>contig reference</B></TD>
      <TD><B>contig reference</B></TD></TR>
    <TR>
      <TD><B>9</B></TD>
      <TD><B>synonymy unknown</B></TD>
      <TD><B>coding: synonymy
      unknown</B></TD></TR></TBODY></TABLE></UL>
<P></H3></P></BODY></HTML>
```

APPENDIX E: SCREEN PRINT OF PROTOYPE WEB INPUT FORM



ONLINE BIOLOGICAL DATA RETRIEVAL

MEDICAL AND MOLECULAR GENETICS

INPUT TYPE

Markers (Gene table)

Marker 1

Marker 2

Filter on tissue expression (required)

[✓ **Single Gene (SNP table)**]

Locus ID

SUBMIT RESET

OUTPUT TABLES AND FIELDS

Gene Table: Output Fields

LOCUS_ID	**Unique NCBI identifier**
CHRPOS	**Chromosome position (bp)**
SYMBOL	**Abbreviation**
PRODUCT	**Short description**
N	**Nomenclature: O (Official) or P (Preferred, but not official)**
GENE_LENGTH	**Includes introns, exons, and poly-A tail**
TISSUE	**Reported tissue expression**

SNP Table: Output Fields

LOCUS_ID	Unique NCBI identifier
SNP_ID	Unique SNP identifier
CHRPOS	Chromosome position (bp)
FXN_CLASS	Functional class
AVG_HZ	Average heterozygosity

SNP Table: Function Class Codes

FXN_CLASS	ABBREV	DESC
1	locus	mrna_acc and protein_acc both null
2	coding	coding
3	synonymous change	synonymous change
4	nonsynonymous change	nonsynonymous change
5	UTR	untranslated region
6	intron	intron
7	splice-site	splice-site
8	contig reference	contig reference
9	synonymy unknown	coding: synonymy unknown

APPENDIX F: USER MANUAL

THE WEB INPUT FORM

The URL for the CLSD browser is: http://www.medgen.iupui.edu/binf/cgiproto.html

Access to the web input form is password protected because the data in CLSD is not licensed for use outside IU:

```
      username: clsdbrw
      password: ******
```

`[Note: the password is masked in this document but not the actual user manual].`

THE MARKER TO GENE TABLE QUERY

The marker to gene table query requires two chromosome markers and an optional tissue filter field. The query returns a table which summarizes data for all the known genes on the chromosome segment defined by the two markers. Occasionally a query may not work because the marker has more than one name. There is a link to the UniSTS home page at NCBI if you need help with this.

Another problem you may encounter when using this query is the tissue expression field in the UniGene database at NCBI uses a non-standardized vocabulary. So if you generate a table and filter the output on a term like “bone”, you may miss a few genes because their tissue expression is reported under a term like “osteocytes”. This tends to be more of a problem with tissues and organs that have complex anatomy and physiology, like the central nervous system.

The link to Medical Subject Headings Browser (MeSH) at the National Library of Medicine (NLM) can help alert you to the need for repeating the query on more than one tissue expression term. For example, if you enter the term “muscle”, you will get an output page that gives a number of related terms, including skeletal muscle, smooth muscle, and myocardium.

Another way to address this problem is to run your query using standard terminology like “bone” or “brain” for the tissue expression filter, and then scan the tissue expression field in the output table for unusual or less frequently used terms like “trabecular meshwork” or “medulla” that may not be included in the MeSH vocabulary.

The first output table gives chromosome positions for the two markers. Occasionally you may see more than one position reported for the same marker. The table we use to get marker positions is actually a primer table, so if more than one primer is available for a marker, the marker will be listed more than once. In most cases, the chromosome positions for these primers will be very close to each other. The SQL for this query sorts the first table by chromosome position, and then uses the first and last position to generate the gene table.

The table that we use to get chromosome positions also includes some markers with non-unique primers. If one of these markers is entered, the gene table will give inaccurate results, because the chromosome positions for the non-unique primers can vary widely. If you are using a marker which has at least one unique primer and one or more non-unique primers, the best solution is to use the UniSTS database and the map viewer at NCBI to select an adjacent chromosome marker that can be used as a proxy.

Most of the fields in the output table for this query are self-explanatory. An “O” in the nomenclature field indicates that the gene abbreviation and product name are officially approved by the HUGO Gene Nomenclature Committee. A “P” indicates that the nomenclature is preferred but not officially approved.

THE GENE ID TO SNP TABLE QUERY

The gene to SNP table query requires one Locuslink ID for input and returns a table that summarizes data for all of the known SNP’s in and adjacent to the gene. If you only know the name or abbreviation for the gene, you can use the link on the web input form to look up the Locuslink ID at NCBI. There is also a link to the dbSNP homepage at NCBI in case you need more information about a specific SNP.

Most of the fields in the output table for this query are self-explanatory. There are lookup tables on the web input form to provide descriptors for the function class and validation status codes.

The main problem we had while developing this query was the frequency of dbSNP updates at NCBI. We subscribe to NCBI mailing lists that usually, but not always, notify us when a new dbSNP build is available. If you see any results in the CLSD output that don’t agree with the NCBI website, please let us know as soon as possible.

MAINTENANCE AND DOWNTIME

It is not unusual for CLSD be unavailable for short periods due to minor problems or routine maintenance. CLSD usually lets us know in advance if service is going to be interrupted for more than a short time due to routine maintenance. We will try to pass these alerts on to you as soon as we hear about them from CLSD. If the CLSD browser is down for more than 2-3 hours, and you haven’t heard from us in advance, please let us know.

APPENDIX G: MAINTENANCE MANUAL

OUTLINE

The CLSD browser consists of four components: 1) the web input form, 2) PERL program, 3) DB2 client, and 4) CLSD. Each of these components will be discussed in turn, followed by a summary of update procedures and a general approach to troubleshooting.

THE WEB INPUT FORM

The web input form passes query parameters entered by the end-user to a PERL program that runs on a webserver in the Medical Genetics department. The host, path, and filename for the web input form HTML file is:

```
fisher.medgen.iupui.edu: /home/apache/htdocs/binf/cgiproto.html
```

The web input form URL for end-users is:

http://www.medgen.iupui.edu/binf/cgiproto.html

Note that the UNIX path and the URL for the web input form are different, even though they point to the same html file. The web input form passes the query parameters to the path and program specified in the <FORM> tag in the header of the html code. The currently active program is db120_2.pl:

```
<FORM action=http://www.medgen.iupui.edu/cgi-
bin/binf/prod/db120_2.pl method=post>
```

Access to the web input form is password protected because the data in CLSD is not licensed for use outside IU:

```
        username: clsdbrw
        password: ******
```

[Note: the password is masked in this document but not the actual user manual].

Joe Urbanski can help you if you ever need to change the password or username.

THE PERL PROGRAM

The PERL program is located on a webserver in Medical Genetics and serves two functions. The first is to accept input parameters from the web input form and generate SQL that is sent to CLSD. The second is to convert the raw data received from CLSD into HTML-based output tables that are sent back to the end-user. PERL programs are currently stored in two directories:

```
        fisher.medgen.iupui.edu: /var/apache/cgi-bin/binf/dev/
        fisher.medgen.iupui.edu: /var/apache/cgi-bin/binf/prod/
```

Note that the UNIX path to the active PERL program is different from the path in the HTML <FORM> tag, even though they both point to the same program.

The PERL program requires frequent modifications to keep up with changes and updates at CLSD and NCBI. The development directory (/guidev) is used for writing and debugging new PERL programs. A PERL program which has been finalized and is ready for use is copied to the production (/prod) directory. This arrangement minimizes the possibility of accidentally modifying a program that is actively being used. It also ensures that if the active program is ever corrupted, there will be a backup copy in the development directory.

To run the PERL program from the command line in SSH shell, you need to supply the input parameters that would normally be submitted by the web form. The syntax is as follows:

```
    perl db120_2.pl input=marker marker1=D14S588 marker2=D14S592
    perl db120_2.pl input=sgene gene=2487
```

The output from this command will be an html document, which may be large. You can usually tell if the output is OK by looking at the first page or so of code, but if necessary, the output can be copied into a text editor on your PC, saved as an html file, and opened with Netscape or Internet Explorer.

THE DB2 CLIENT

The DB2 client establishes a remote connection to CLSD from Medical Genetics. Joe Urbanski has assumed most of the responsibility for installing and maintaining the DB2 client. SSH Secure Shell can be configured so it will automatically connect to the DB2 client whenever you open SSH. You can ask Joe Urbanski for help if you'd like to set this up. Alternatively, you can connect to the DB2 client manually at the beginning of each session by entering the following at the command prompt (use your username and password, without quotation marks):

```
    db2 connect to clsd2 user "username" using "password";
```

CLSD

The CLSD homepage is: http://www.indiana.edu/~rac/clsd/. This homepage provides a list of available databases and the tables and field names for each database. Sometimes the table and field names at CLSD are different from the source databases. With the exception of dbSNP, CLSD does not maintain data dictionaries or entity relationship diagrams. For this information you will have to go to the source databases. Be forewarned that the documentation at NCBI may be incomplete and fragmented across many different web pages and ftp directories.

University Information technology Services (UITS) and Research and Academic Computing (RAC) maintain a number of other useful bioinformatics resources: http://www.indiana.edu/~rac/bioinformatics/

Our contact at CLSD is Andrew D. Arenson. Andy was recently promoted and CLSD is in the process of filling his old position. CLSD has only limited capability of making updates until this position is filled. Andy's new position includes supervising whoever is hired to replace him.

CLSD maintains a mailing list which is helpful in keeping up with service changes and planned downtime. Instructions for subscribing to this list are on the CLSD homepage.

To accounts are required to access CLSD. The department of Medical Genetics has one account on the university research supercomputer, and a second account for CLSD. To connect to the research supercomputer, open SSH Secure Shell Client and logon to orion17.uits.iupui.edu:

```
username    clsdbrw
password    ********
```

[Note: the password for clsdbrw account is masked in this document, but not in the actual maintenance manual]. To start CLSD, type the following commands at the command prompt:

```
cd /usr/local/bin
bash
source ~db2inst2/sqllib/db2profile
db2 connect to clsd2
cd /N/u/clsdbrw/SP
```

These commands are saved in a file called logon that is stored in the clsdbrw home directory: /N/u/clsdbrw/SP. I usually open this file and copy and paste the commands into the command prompt as a block so I don't have to type them in manually.

Here is the procedure for running SQL on a direct connection to CLSD. This bypasses the web input form, PERL program, and DB2 client. This is helpful for writing new SQL code or ruling out CLSD as the source of the problem when the CLSD browser goes down.

You can run simple queries from the command line as follows:

```
db2 “select * from dbsnp120.snpfunctioncode”
```

More complex SQL (anything longer than one line) has to be run from a file. I generally write the code in MS Word (or other text editor in Windows), and then copy and paste the code into vi (or other UNIX editor of your choice). I find this much easier than writing the SQL itself in vi. Sample sql:

```
select *
from dbsnp120.snpfunctioncode
where code = 1;
```

Save the file from vi with a filename like test.db2, and exit vi to the command prompt. To run this file from the command prompt, type:

```
db2 -tf test.db2
```

The -t tells DB2 that your SQL code ends with a semicolon. The -f is necessary when running SQL from a file instead of directly from the command line.

IBM has an extensive set of documentation manuals on their DB2 technical support page:

http://www-306.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v8pubs.d2w/en_main

NCBI

Each of the major databases at NCBI has a home page that is a good place to start when you are looking for documentation. NCBI has a lot of good documentation in addition to this, but some of it is very hard to find unless you know where to look for it. The following URL's are current as of this writing but are subject to change.

NCBI maintains a number of useful email lists. These usually, but not always, give you advance warning if NCBI is planning any updates or other modifications that may affect CLSD or the CLSD browser:

http://www.ncbi.nlm.nih.gov/Sitemap/Summary/email_lists.html

A good overview of ftp download resources at NCBI:

http://www.ncbi.nlm.nih.gov/Sitemap/index.html#FTPDatabases

There is a lot of useful documentation in some of the ftp readme.txt files that is otherwise very difficult to find.

A good summary of the genome sequencing and annotation process:
http://www.ncbi.nlm.nih.gov/genome/guide/build.html#sts

Description of fields in LocusLink flat files (these are converted to tables and fields when the flat file is uploaded to CLSD):
ftp://ftp.ncbi.nih.gov/refseq/LocusLink/README

Description of fields in the ePCR flat file that we use to get positions for chromosome markers:
ftp://ftp.ncbi.nih.gov/repository/UniSTS/UniSTS_ePCR.Reports/Homo_sapiens/README

Data dictionaries and entity relationship diagrams for dbSNP:
ftp://ftp.ncbi.nih.gov/snp/mssql/schema/

Description of fields in GenBank flat files:
http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html#AccessionB

A Nature Genetics article that does an excellent job of providing step-by-step instructions for task-specific, frequently asked questions at NCBI:

http://genomics.iupui.edu/G865/NatGenUserGuide/start.html

(Sept 2002, v. 32 supplement pp 1 - 79)

MAINTENANCE AND UPDATES

CLSD draws data from databases that are constantly being updated and modified. As a result, a significant amount of maintenance is required to keep CLSD and the CLSD browser functioning. We have relatively efficient processes for incorporating dbSNP updates because we've already done this five or six times. When NCBI makes substantial changes to other databases that we have not dealt with before, much more work may be required to incorporate these revisions the first time. As additional experience is gained, the new processes should become more efficient.

When NCBI changes the structure of a database, the first task is to determine whether or not these affect the tables and fields that are being used by the CLSD browser. Once you know what the changes are, you can usually just modify the SQL embedded in the PERL code. To make major modifications, you may need to write and test the new SQL on a direct connection to CLSD on orion17.uits.iupui.edu, and then incorporate the new SQL into the old PERL program.

The following is a general procedure for incorporating new dbSNP updates. The first step is to change the dbSNP table prefixes in the PERL/SQL code. The easiest way to do this is to copy the program into MS WORD and use the FIND/REPLACE function. For example, the prefix "dbsnp119."would be replaced by "dbsnp120." If the data model changes, you will have more work to do. Sometimes new tables are created to replace old tables, and sometimes data fields are moved from one table to another.

To modify the PERL program, I usually copy the active program in the /prod directory to the guidev directory under a new filename. For example, /prod/dbsnp119.pl would be used to create a new copy of the same file named /guidev/dbsnp120.pl. The new program in the /guidev directory is used for modification and debugging. When the new program is finished, it is copied to the /prod directory.

The lookup tables for validation status codes (dbsnpxxx.snpvalidationcode) and functional class codes (dbsnpxxx.snpfunctioncode) also need to be checked to see if they have been updated. These tables are not used in the PERL/SQL code, but they are used to provide lookup tables on the web input form that provide descriptions for the code numbers.

The HTML file for the web input form will also need to be modified to reflect the new target program in the <FORM> tag, as well as any revisions to the validation status and functional class code lookup tables. I usually copy the current HTML document to a text editor in Windows and save the file to my hard drive to modify and test the new code. When the new HTML document is finished, go back to the /home/apache/htdocs/binf/ directory on fisher and copy the active (old) HTML file to a new filename that reflects the backup date. For example, cgiproto.html would be copied to cgiproto-bkp-2004-05-14.html. The new HTML document in the Windows editor is then used to overwrite old cgiproto.html file. This process ensures that a working backup copy of the HTML file is always available if it is ever needed. It also provides a consistent URL for the end-users that does not have to be changed every time the web input form is updated.

TROUBLESHOOTING

It is not unusual for CLSD be unavailable for short periods due to minor problems or routine maintenance, so I do not usually take any action unless the CLSD browser is down for more than 2-3 hours. If you subscribe to their email list, CLSD will usually announce any major maintenance in advance.

If CLSD browser is down for more than 2-3 hours and no routine maintenance has been planned, I usually start with CLSD and work backwards towards the web input form in a stepwise manner to determine which component is causing the problem. Here is a general procedure for troubleshooting:

1) Login to orion17.uits.iupui.edu. The login screen will often give you up to date information on any planned or unplanned interruptions in service on the research supercomputer.

2) Login to CLSD and run some simple SQL to see if CLSD is working

3) If CLSD seems to be working OK, go to fisher.medgen.iupui.edu and try running one or more queries using the PERL program from the command line

4) If the PERL program is working OK, try running one or more queries from the web input form

5) At this point, you should have a pretty good idea which components are affected. The error messages will usually help determine what the problem is and how to start working on it. Sometimes you will need help from technical support in Medical Genetics, CLSD, or both.

APPENDIX H: PERL CODE FOR FINAL VERSION

```
fisher.medgen.iupui.edu: /var/apache/cgi-bin/binf/prod/db120_2.pl
```

The first section of this program contains comments, invokes PERL modules, and opens a connection to CLSD. The CLSD connection is called a database handle, or $dbh. Our departmental account at CLSD has username “clsdbrw”. The password for accessing the clsdbrw account is masked in this document but not the actual PERL program.

```
#!/usr/local/bin/perl -w
# db120.pl
# use db119.pl as template
# revise code for to convert dbsnp119 to 120

use strict;

use DBI;
use DBD::DB2::Constants;
use DBD::DB2;
use CGI qw(:standard);
use CGI::Carp qw(fatalsToBrowser);

my $q=new CGI;

my $dbh = DBI->connect(“dbi:DB2:clsd2”, “clsdbrw”, “********”)
       or die “Couldn’t connect to database: “ . DBI->errstr;
```

The next section determines which query the user wants to run. Depending on which radio button is selected on the web input form, the PERL variable $input is assigned a value of “marker” or “sgene”.

```
# which query does user want to run?
my $input = $q ->param("input");

# code is dummy cgi field for development only
my $code = $q->param("code");
```

This in turn invokes one of two subroutines, “markergenetable” or “snptable”.

```
# which query does user want to run?
if ($input eq "marker") {
   markergenetable();
} elsif ($input eq "sgene") {
   snptable();
}
```

The first step in the genetable subroutine is to pass the marker and tissue parameters submitted by the end-user to PERL variables.

```
############################# GENETABLE SUBROUTINE ##############

sub markergenetable {
my $marker1 = uc ($q->param("marker1"));
my $marker2 = uc ($q->param("marker2"));

my $tissue = $q ->param("tissue");
```

These variables are then used to generate an SQL statement, $stmt. The question marks are placeholders for the marker ids which are entered when the statement is executed.

```
my $stmt = "select STS_NAME, contig_chr as chr,
pos1+contig_start-1 as chrpos
   from epcr.Seq_STS, dbsnp120.contiginfo
   where (sts_name in (? , ?))
      and (gi_num=contig_gi)
   order by chrpos";
```

The statement handle, $sth, connects the SQL statement, $stmt, to the database handle ($dbh).

```
my $sth = $dbh->prepare($stmt)
       or die "Can't prepare SQL statement: ", $dbh->errstr(),
"\n";
```

The next command takes the current values of the marker variables and executes the statement handle. This sends the first query off to CLSD.

```
$sth->execute($marker1,$marker2)
       or die "Can't execute SQL statement: ", $sth->errstr(),
"\n";
```

The first query gets chromosome positions for the two markers from CLSD. The next block of code assigns these values to the variables $chrpos1 and $chrpos2. A small table is printed for the enduser which shows the chromosome positions of the two markers.

```
print header;

print '
      <html><head><title>Query Results</title></head>
       <body bgcolor="#E0E0E0">
        <table border="1" cellspacing="0">
         <tr>
          <td><b>MARKER</b></td>
          <td><b>CHR</b></td>
          <td><b>CHRPOS</b></td>
         </tr>
       ';

my ($chr1, $chrpos1, $chrpos2);
$chr1 = 0;
$chrpos1 = 0;
while ( my @row = $sth->fetchrow_array )
{
    if( $chr1 == 0 )
    {
        $chr1 = $row[1];
    }
    if ( $chrpos1 == 0 )
```

```
        {
            $chrpos1 = $row[2];
        }
        $chrpos2 = $row[2];
        print qq(<tr>\n);
        print qq(<td>$row[0]</td>\n);
        print qq(<td>$row[1]</td>\n);
        print qq(<td>$row[2]</td>\n);
        print qq(</tr>);
    }
    print qq(<P>);
    print qq(</table>);

    $sth->finish();
```

The next block of code uses the chromosome positions to generate a second query, which returns the raw data necessary for the gene output table. This query takes one of two forms, depending on whether or not tissue filter terms are submitted by the end-user. If a tissue filter term is submitted, the program will look for the term in upper, lower, and title case.

```
    my ($tissue1, $tissue2, $tissue3);
    my ($stmt2);

    if ($tissue ne "") {
      $tissue1 = lc $tissue;
      $tissue2 = uc $tissue;
      $tissue3 = $tissue;
      $tissue3 =~ s/(\w+)/\u\L$1/g;

      $stmt2 = "select pos.locus_id as locus_id,
                          pos.chrpos as chrpos,
                          gene_length, ll.symbol as symbol,
                          ll.product as product,
                          express as tissue,
                          ll.name_type as n

      from (select c.locus_id, contig_start+start-1 as chrpos, end-
    start as gene_length
      from locuslink.contig as c, dbsnp120.contiginfo
      where (gi_num=contig_gi)
      and (((contig_start+start-1 > ? ) and (contig_start+start-1 < ?
    ))
             or ((contig_start+end-1 > ? ) and (contig_start+end-1 < ?
    )))
      and (source = ? )) as pos

      left outer join locuslink.loci as ll
      on pos.locus_id=ll.locus_id

      left outer join unigene.locus as ul
      on pos.locus_id=ul.locuslink_id

      left outer join unigene.express as ue
      on ul.cluster_id=ue.cluster_id
```

```
  where (express like '%$tissue1%')
     or (express like '%$tissue2%')
     or (express like '%$tissue3%')

  order by chrpos";

} else {

  $stmt2 = "select pos.locus_id as locus_id,
                    pos.chrpos as chrpos,
                    gene_length, ll.symbol as symbol,
                    ll.product as product,
                    express as tissue,
                    ll.name_type as n

  from (select c.locus_id, contig_start+start-1 as chrpos, end-
start as gene_length
  from locuslink.contig as c, dbsnp120.contiginfo
  where (gi_num=contig_gi)
  and (((contig_start+start-1 > ? ) and (contig_start+start-1 < ?
))
       or ((contig_start+end-1 > ? ) and (contig_start+end-1 < ?
)))
  and (source = ? )) as pos

  left outer join locuslink.loci as ll
  on pos.locus_id=ll.locus_id

  left outer join unigene.locus as ul
  on pos.locus_id=ul.locuslink_id

  left outer join unigene.express as ue
  on ul.cluster_id=ue.cluster_id

  order by chrpos";

}

my $sth2 =  $dbh->prepare($stmt2)
    or die "Can't prepare SQL statement: ", $dbh->errstr(), "\n";

$sth2->execute($chrpos1,$chrpos2,$chrpos1,$chrpos2,$chr1)
    or die "Can't execute SQL statement: ", $sth->errstr(), "\n";
```

The final block of code in this subroutine converts the raw data from CLSD into HTML-based output tables, which are returned to the end-user.

```
print '
        <table border="1" cellspacing="0">
         <tr>
          <td><b>LOCUS_ID</b></td>
          <td><b>CHRPOS</b></td>
          <td><b>SYMBOL</b></td>
          <td><b>PRODUCT</b></td>
          <td><b>N</b></td>
          <td><b>GENE_LENGTH</b></td>
          <td><b>TISSUE</b></td>
         </tr>
       ';

while(my $data = $sth2->fetchrow_hashref){
  print qq(<tr>\n);
  print qq(<td>$data->{LOCUS_ID}</td>\n);
  print qq(<td>$data->{CHRPOS}</td>\n);
  print qq(<td>$data->{SYMBOL}</td>\n);
  print qq(<td>$data->{PRODUCT}</td>\n);
  print qq(<td>$data->{N}</td>\n);
  print qq(<td>$data->{GENE_LENGTH}</td>\n);
  print qq(<td>$data->{TISSUE}</td>\n);
  print qq(</tr>);
}

print '
        </table>
        </body>
        </html>
      ';

$sth2->finish();

}
```

The code for the snp table subroutine generates only one output table. Otherwise, it is very similar to the marker-gene table subroutine.

```
########################## SNPTABLE SUBROUTINE ##################
sub snptable {

my $gene = $q->param("gene");

#my $stmt = "select locus_id, snp_id, fxn_class
#from dbsnp114.snpcontiglocusid
#where locus_id= ? ";

my $stmt = "select scl.locus_id, scl.snp_id, fxn_class,
asn_from+ci.contig_start-1 as chrpos, avg_heterozygosity as
avg_hz, het_se, snp.validation_status as val
from dbsnp120.snpcontiglocusid as scl, dbsnp120.contiginfo as ci,
dbsnp120.snp as snp
where (locus_id= ? )
   and (scl.contig_acc=ci.contig_acc)
   and (scl.contig_ver=ci.contig_ver)
   and (scl.snp_id=snp.snp_id)
order by chrpos";


my $sth = $dbh->prepare($stmt)
       or die "Can't prepare SQL statement: ", $dbh->errstr(),
"\n";

$sth->execute($gene)
```

```
       or die "Can't execute SQL statement: ", $sth->errstr(),
"\n";

print header;

print '
      <html><head><title>Query Results</title></head>
       <body bgcolor="#E0E0E0">
        <table border="1" cellspacing="0">
         <tr>
          <td><b>LOCUS_ID</b></td>
          <td><b>SNP_ID</b></td>
          <td><b>CHRPOS</b></td>
          <td><b>FXN_CLASS</b></td>
          <td><b>AVG_HZ</b></td>
          <td><b>HET_SE</b></td>
          <td><b>VAL</b></td>
         </tr>
       ';

while(my $data = $sth->fetchrow_hashref){
  print qq(<tr>\n);
  print qq(<td>$data->{LOCUS_ID}</td>\n);
  print qq(<td>$data->{SNP_ID}</td>\n);
  print qq(<td>$data->{CHRPOS}</td>\n);
  print qq(<td>$data->{FXN_CLASS}</td>\n);
```

There are a significant number of missing values for the AVG_HZ and HETSE fields. The following code assigns these a value of ‘NA’. If this is not done, the HTML table will not show gridline borders for the empty cells.

```
  my $AVGHZ = $data->{AVG_HZ};
# if ($AVGHZ ne "") {
  if (defined($AVGHZ) && ("" ne $AVGHZ)) {
    $AVGHZ = sprintf("%1.2f", $AVGHZ);
  } else {
    $AVGHZ = "NA";
  }
  printf qq(<td>$AVGHZ</td>);

  my $HETSE = $data->{HET_SE};
# if ($HETSE ne "") {
  if (defined($HETSE) && ("" ne $HETSE)) {
    $HETSE = sprintf("%1.3f", $HETSE);
  } else {
    $HETSE = "NA";
  }
  printf qq(<td>$HETSE</td>);
  print qq(<td>$data->{VAL}</td>\n);


  print qq(</tr>);
}

print '
        </table>
```

```
                </body>
                </html>
            ';

    $sth->finish();

    }
```

The final command closes the connection to CLSD.

```
    $dbh->disconnect;
```

APPENDIX I: HTML CODE FOR FINAL VERSION

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD><TITLE>cgi-test</TITLE>
<META http-equiv=Content-Type content="text/html; charset=iso-8859-1">
<META content="MSHTML 5.50.4926.2500" name=GENERATOR></HEAD>
<BODY bgcolor="#C0C0C0" background="background.jpg">
<FORM action=http://www.medgen.iupui.edu/cgi-bin/binf/prod/db120_2.pl
method=post><BR>

<H2>ONLINE BIOLOGICAL DATA RETRIEVAL
<P>MEDICAL AND MOLECULAR GENETICS</H2>

<HR>

<H2>INPUT TYPE</H2>
<P>
<H3><INPUT type="radio" name="input" value="marker" CHECKED> Markers
(Gene table) </H3>
<P>
<UL>
  <P><INPUT TYPE="TEXT" name="marker1"> Marker 1 &nbsp;&nbsp;
      Search
      <a href="http://www.ncbi.nlm.nih.gov/genome/sts/query_tip.html">
      Markers
      </a> at NCBI)

  <P><INPUT TYPE="TEXT" name="marker2"> Marker 2

  <P><INPUT TYPE="TEXT" name="tissue"> Filter on tissue expression
  (optional)&nbsp;&nbsp;

      <P>Search <a href="http://www.nlm.nih.gov/mesh/MBrowser.html">
Medical Subject Headings (MeSH)</a> at NLM

</P></UL>

<H3><INPUT type="radio" name="input" value="sgene" > Single Gene (SNP
table)
<P></H3>

<UL>
  <P>
  <P><INPUT TYPE="TEXT" name="gene"> Locus ID &nbsp;&nbsp;(Search
     <a href="http://www.ncbi.nlm.nih.gov/LocusLink/help.html"> Locus
ID's</a>,
     <a href="http://www.ncbi.nlm.nih.gov/SNP/"> SNP's</a> at NCBI)
  </P>
</UL>

  <P></P></UL>

<P><INPUT type=submit value=SUBMIT><INPUT type=reset value=RESET>
<P></FORM></P>

<HR>
```

```
<H2>OUTPUT TABLES AND FIELDS</H2>
<P>
<H3>Gene Table
<P></H3>

<UL>
   <table border="1" cellspacing="0">
         <tr>
            <td><b>LOCUS_ID</b></td>
            <td><b>Unique NCBI identifier</b></td>
         </tr>
         <tr>
            <td><b>CHRPOS</b></td>
            <td><b>Chromosome position (bp)</b></td>
         </tr>
         <tr>
            <td><b>SYMBOL</b></td>
            <td><b>Abbreviation</b></td>
         </tr>
         <tr>
            <td><b>PRODUCT</b></td>
            <td><b>Short description</b></td>
         </tr>
         <tr>
            <td><b>N</b></td>
            <td><b>Nomenclature: O (Official) or P (Preferred, but not
official)</b></td>
         </tr>
         <tr>
            <td><b>GENE_LENGTH</b></td>
            <td><b>Includes introns, exons, and mRNA tail</b></td>
         </tr>
         <tr>
            <td><b>TISSUE</b></td>
            <td><b>Reported tissue expression</b></td>
         </tr>
    </table>
</UL>

<H3>SNP Table
<P></H3>

<UL>
   <table border="1" cellspacing="0">
         <tr>
            <td><b>LOCUS_ID</b></td>
            <td><b>Unique NCBI identifier</b></td>
         </tr>
         <tr>
            <td><b>SNP_ID</b></td>
            <td><b>Unique SNP identifier</b></td>
         </tr>
         <tr>
            <td><b>CHRPOS</b></td>
            <td><b>Chromosome position (bp)</b></td>
         </tr>
         <tr>
            <td><b>FXN_CLASS</b></td>
            <td><b>Functional class</b></td>
         </tr>
         <tr>
            <td><b>AVG_HZ</b></td>
            <td><b>Average heterozygosity</b></td>
         </tr>
         <tr>
            <td><b>HZ_SE</b></td>
            <td><b>Heterozygosity standard error</b></td>
         </tr>
         <tr>
            <td><b>VAL</b></td>
            <td><b>Validation status</b></td>
         </tr>
    </table>
</UL>
    </table>
</UL>

<UL>
   <table border="1" cellspacing="0">
         <tr>
            <td><b>FXN_CLASS</b></td>
            <td><b>ABBREV</b></td>
            <td><b>DESC</b></td>
         </tr>
         <tr>
            <td><b>1</b></td>
            <td><b>non-gene</b></td>
            <td><b>SNP lies within 2000 bp upstream or 500 bp
downstream from gene boundaries</b></td>
         </tr>
         <tr>
            <td><b>2</b></td>
            <td><b>coding</b></td>
            <td><b>refers to exons, appears to have been replaced by 3
and 4</b></td>
         </tr>
         <tr>
            <td><b>3</b></td>
            <td><b>synonymous change</b></td>
            <td><b>SNP changes mRNA codon but no amino acid
substitution</b></td>
         </tr>
         <tr>
            <td><b>4</b></td>
            <td><b>nonsynonymous change</b></td>
            <td><b>SNP changes mRNA codon with amino acid
substitution</b></td>
         </tr>
         <tr>
            <td><b>5</b></td>
            <td><b>UTR</b></td>
            <td><b>untranslated region: located on head or tail of
mRNA</b></td>
```

```
         </tr>
         <tr>
            <td><b>6</b></td>
            <td><b>intron</b></td>
            <td><b>intron</b></td>
         </tr>
         <tr>
         </tr>
         <tr>
            <td><b>7</b></td>
            <td><b>splice-site</b></td>
            <td><b>located on intron-exon border</b></td>
         </tr>
         <tr>
            <td><b>8</b></td>
            <td><b>contig reference</b></td>
            <td><b>SNP's with code of 3 or 4 are listed twice in the
database, once for each allele. The allele on the

reference contig gets a

 value of 8</b></td>
         </tr>
         <tr>
            <td><b>9</b></td>
            <td><b>synonymy unknown</b></td>
            <td><b>coding: synonymy unknown</b></td>
         </tr>
    </table>
</UL>

<UL>
   <table border="1" cellspacing="0">
         <tr>
            <td><b>VALIDATION CODE</b></td>
            <td><b>SHORT DESC</b></td>
            <td><b>LONG DESCRIPTION</b></td>
          </tr>
         <tr>
            <td><b>0</b></td>
            <td><b>not validated</b></td>
            <td><b>no validation</b></td>
         </tr>
         <tr>
            <td><b>1</b></td>
            <td><b>cluster</b></td>
            <td><b>validated by multiple, independent submissions to
the refSNP cluster</b></td>
         </tr>
         <tr>
            <td><b>2</b></td>
            <td><b>freq</b></td>
            <td><b>validated by frequency or genotype data: minor
alleles observed in at least two chromosomes</b></td>
         </tr>
         <tr>
            <td><b>3</b></td>
            <td><b>cluster, freq</b></td>
            <td><b>see above</b></td>
         </tr>
         <tr>
            <td><b>4</b></td>
            <td><b>submitter</b></td>
            <td><b>submitter submitted batch update with a second
validation method</b></td>
         </tr>
         <tr>
            <td><b>5</b></td>
            <td><b>submitter, cluster</b></td>
            <td><b>see above</b></td>

         </tr>
         <tr>
            <td><b>6</b></td>
            <td><b>submitter, freq</b></td>
            <td><b>see above</b></td>
         </tr>
         <tr>
            <td><b>7</b></td>
            <td><b>submitter, freq, cluster</b></td>
            <td><b>see above</b></td>
         </tr>
         <tr>
            <td><b>8</b></td>
            <td><b>doublehit</b></td>
            <td><b>all alleles have been observed in at least two
chromosomes apiece</b></td>
         </tr>
         <tr>
            <td><b>9</b></td>
            <td><b>doublehit, cluster</b></td>
            <td><b>see above</b></td>
         </tr>
         <tr>
            <td><b>10</b></td>
            <td><b>doublehit, freq</b></td>
            <td><b>see above</b></td>
         </tr>
         <tr>
            <td><b>11</b></td>
            <td><b>doublehit, freq, cluster</b></td>
            <td><b>see above</b></td>
         </tr>
         <tr>
            <td><b>12</b></td>
            <td><b>doublehit, submitter</b></td>
            <td><b>see above</b></td>
         </tr>
         <tr>
            <td><b>13</b></td>
            <td><b>doublehit, submitter, cluster</b></td>
            <td><b>see above</b></td>
         </tr>
         <tr>
```

```
            <td><b>14</b></td>
            <td><b>doublehit, submitter, freq</b></td>
            <td><b>see above</b></td>
         </tr>
         <tr>
            <td><b>15</b></td>
            <td><b>doublehit, submitter, freq, cluster</b></td>
            <td><b>see above</b></td>
         </tr>
   </table>
</UL>

</BODY></HTML>
```

APPENDIX J: WEB INPUT FORM FOR FINAL VERSION

ONLINE BIOLOGICAL DATA RETRIEVAL

MEDICAL AND MOLECULAR GENETICS

INPUT TYPE

[✓ Markers (Gene table)]

Marker 1 (Search Markers at NCBI)

Marker 2

Filter on tissue expression (optional)

Search Medical Subject Headings (MeSH) at NLM

Single Gene (SNP table)

Locus ID (Search Locus ID's, SNP's at NCBI)

SUBMIT RESET

OUTPUT TABLES AND FIELDS

Gene Table

LOCUS_ID	**Unique NCBI identifier**
CHRPOS	**Chromosome position (bp)**
SYMBOL	**Abbreviation**
PRODUCT	**Short description**
N	**Nomenclature: O (Official) or P (Preferred, but not official)**
GENE_LENGTH	**Includes introns, exons, and mRNA tail**
TISSUE	**Reported tissue expression**

SNP Table

LOCUS_ID	Unique NCBI identifier
SNP_ID	Unique SNP identifier
CHRPOS	Chromosome position (bp)
FXN_CLASS	Functional class
AVG_HZ	Average heterozygosity
HZ_SE	Heterozygosity standard error
VAL	Validation status

FXN_CLASS	ABBREV	DESC
1	non-gene	SNP lies within 2000 bp upstream or 500 bp downstream from gene boundaries
2	coding	refers to exons, appears to have been replaced by 3 and 4
3	synonymous change	SNP changes mRNA codon but no amino acid substitution
4	nonsynonymous change	SNP changes mRNA codon with amino acid substitution
5	UTR	untranslated region: located on head or tail of mRNA
6	intron	intron
7	splice-site	located on intron-exon border
8	contig reference	SNP's with code of 3 or 4 are listed twice in the database, once for each allele. The allele on the reference contig gets a value of 8
9	synonymy unknown	coding: synonymy unknown

VALIDATION CODE	SHORT DESC	LONG DESCRIPTION
0	not validated	no validation
1	cluster	validated by multiple, independent submissions to the refSNP cluster
2	freq	validated by frequency or genotype data: minor alleles observed in at least two chromosomes
3	cluster, freq	see above
4	submitter	submitter submitted batch update with a second validation method
5	submitter, cluster	see above
6	submitter, freq	see above
7	submitter, freq, cluster	see above
8	doublehit	all alleles have been observed in at least two chromosomes apiece
9	doublehit, cluster	see above
10	doublehit, freq	see above
11	doublehit, freq, cluster	see above
12	doublehit, submitter	see above

13	**doublehit, submitter, cluster**	**see above**
14	**doublehit, submitter, freq**	**see above**
15	**doublehit, submitter, freq, cluster**	**see above**