BMC Bioinformatics

Methodology article

SeqHound: biological sequence and structure database as a platform for bioinformatics research

Katerina Michalickova[1,2], Gary D Bader[1,2], Michel Dumontier[1,2], Hao Lieu[2], Doron Betel[1,2], Ruth Isserlin[1,2] and Christopher WV Hogue*[1,2]

Address: [1]Department of Biochemistry, University of Toronto, Toronto, Ontario, Canada M5S 1A8 and [2]Samuel Lunenfeld Research Institute, 600 University Avenue, Toronto, Ontario, Canada M5G 1X5

E-mail: Katerina Michalickova - katerina@mshri.on.ca; Gary D Bader - gary.bader@utoronto.ca; Michel Dumontier - micheld@mshri.on.ca; Hao Lieu - lieu@mshri.on.ca; Doron Betel - betel@mshri.on.ca; Ruth Isserlin - ruth.isserlin@utoronto.ca; Christopher WV Hogue* - hogue@mshri.on.ca

*Corresponding author

Abstract

Background: SeqHound has been developed as an integrated biological sequence, taxonomy, annotation and 3-D structure database system. It provides a high-performance server platform for bioinformatics research in a locally-hosted environment.

Results: SeqHound is based on the National Center for Biotechnology Information data model and programming tools. It offers daily updated contents of all Entrez sequence databases in addition to 3-D structural data and information about sequence redundancies, sequence neighbours, taxonomy, complete genomes, functional annotation including Gene Ontology terms and literature links to PubMed. SeqHound is accessible via a web server through a Perl, C or C++ remote API or an optimized local API. It provides functionality necessary to retrieve specialized subsets of sequences, structures and structural domains. Sequences may be retrieved in FASTA, GenBank, ASN.1 and XML formats. Structures are available in ASN.1, XML and PDB formats. Emphasis has been placed on complete genomes, taxonomy, domain and functional annotation as well as 3-D structural functionality in the API, while fielded text indexing functionality remains under development. SeqHound also offers a streamlined WWW interface for simple web-user queries.

Conclusions: The system has proven useful in several published bioinformatics projects such as the BIND database and offers a cost-effective infrastructure for research. SeqHound will continue to develop and be provided as a service of the Blueprint Initiative at the Samuel Lunenfeld Research Institute. The source code and examples are available under the terms of the GNU public license at the Sourceforge site [http://sourceforge.net/projects/slritools/] in the SLRI Toolkit.

Background

We have developed an integrated database system, SeqHound, that contains similar information to that found in Entrez [1], yet that is capable of being locally hosted and daily updated like the Sequence Retrieval System (SRS) [2]. SeqHound can be easily installed as a tool in

any laboratory and used as a resource for a number of bioinformatics projects. It is fast, robust and primarily supports numerical-based retrieval of information (e.g. by accession number or GenInfo identifier (GI)), rather than text-based retrieval. The system supports a wide range of information and is provided as a high-quality, portable package. SeqHound has been well tested over the past two years in our own laboratory and is the main Bioinformatics information server system in use at MDS Proteomics Inc.

For data access and mining, an extensive application programming interface (API) has been developed for Perl, C and C++ access and a streamlined WWW interface is available. This kind of generic integrated resource is not widely available in the bioinformatics community. One of the benefits of using a daily updated integrated database is that one does not need to use specialized data sets in various formats that are often difficult to keep up to date. Computations that derive numeric results, predictions or scoring functions can be automatically updated as the databases grow.

We have previously made use of the National Center for Biotechnology Information's (NCBI) interface for remote network querying of the Entrez databases, but we and others have found that with a very high number of queries this method of access is not always successful. NCBI limits access to the Entrez service to reasonable amounts of automated requests, however as databases continue to grow, it becomes increasingly difficult to perform bioinformatics analysis with remote database resources. For large-scale automated queries, we have encountered problems with data transmission; interrupted network connections and low transmission speeds that slowed tasks down substantially. In addition, the Entrez servers and administrators are not tolerant of large numbers of automated queries.

We created the SeqHound database system to overcome these obstacles and to serve as a platform for our research [3–6]. While the Entrez servers are not presented by NCBI in an exportable format, the NCBI FTP site has a number of relevant and daily updated data sets that can be used to reconstruct a functional in-house database resource that behaves like an Entrez server. The only part of Entrez that seems to not be available in export format on the NCBI site is the sequence neighbour information calculated by BLAST [7], however we have recently shown that we can compute protein sequence neighbors on our own internal cluster computer using a variant of NCBI Blast called NBLAST [8].

We parse NCBI's native ASN.1 sequence and structure data files to reproduce link information and store it together with the original full data annotation in the Se-

qHound database. This paper describes in detail the SeqHound database system and gives examples of its use in the form of pseudo-code which is supplemented with actual source code (in Perl, C and C++) at the Sourceforge site [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slri-tools/slri/seqhound/examples/].

Results
System specification
The system overview of SeqHound is shown in Figure 1 by a Unified Modeling Language (UML) [http://www.rational.com/uml/] diagram. This figure indicates the dependencies on outside sources such as the NCBI data and programming toolkit [http://ncbi.nlm.nih.gov/IEB], the database management system (DBMS) and the freely distributed bzip library [http://sources.redhat.com/bzip2/].

Data from the NCBI FTP site is used to initialize and update the database. The database is accessed using the local API, remote API and WWW pages. Individual components of the system are described in the next sections.

Database tables
The core of the SeqHound database system holds the equivalent of the fully annotated sequence and structure databases hosted by Entrez [1]. These databases include GenBank [9], the RefSeq database [10], the Protein Data Bank (PDB) [11], Swiss-Prot [12], the European Molecular Biology Laboratories (EMBL) nucleotide sequence database [2], the dbEST database [13], the Molecular Modeling Database [14], the Protein Information Resource (PIR) [15], the Protein Research Foundation [http://www.prf.or.jp/], the Taxonomy database [http://www.ncbi.nlm.nih.gov/Taxonomy/], LocusLink database [10], Conserved Domain Database (CDD) [16]. The Gene Ontology (GO) vocabulary for functional annotation [17] is also included.

The primary key for all the SeqHound tables is the GenInfo (GI) identifier. GI identifiers are integers and are guaranteed to be unique within NCBI data space, hence providing a very reliable set of primary keys for the SeqHound database.

To facilitate linking between different data resources, SeqHound keeps tables of GIs cross-referenced to identifiers to different databases such as MMDB, MEDLINE, Entrez Taxonomy, GO vocabulary, CDD, Online Mendelian Inheritance in Man (OMIM), LocusLink and Vector Alignment Search Tool (VAST) 3-D structural domains. Nucleotide GIs are linked to protein GIs.

The protein sequence neighbour information was reconstructed using the BLAST algorithm [7]. Unlike the Entrez system, information about redundancy within the protein
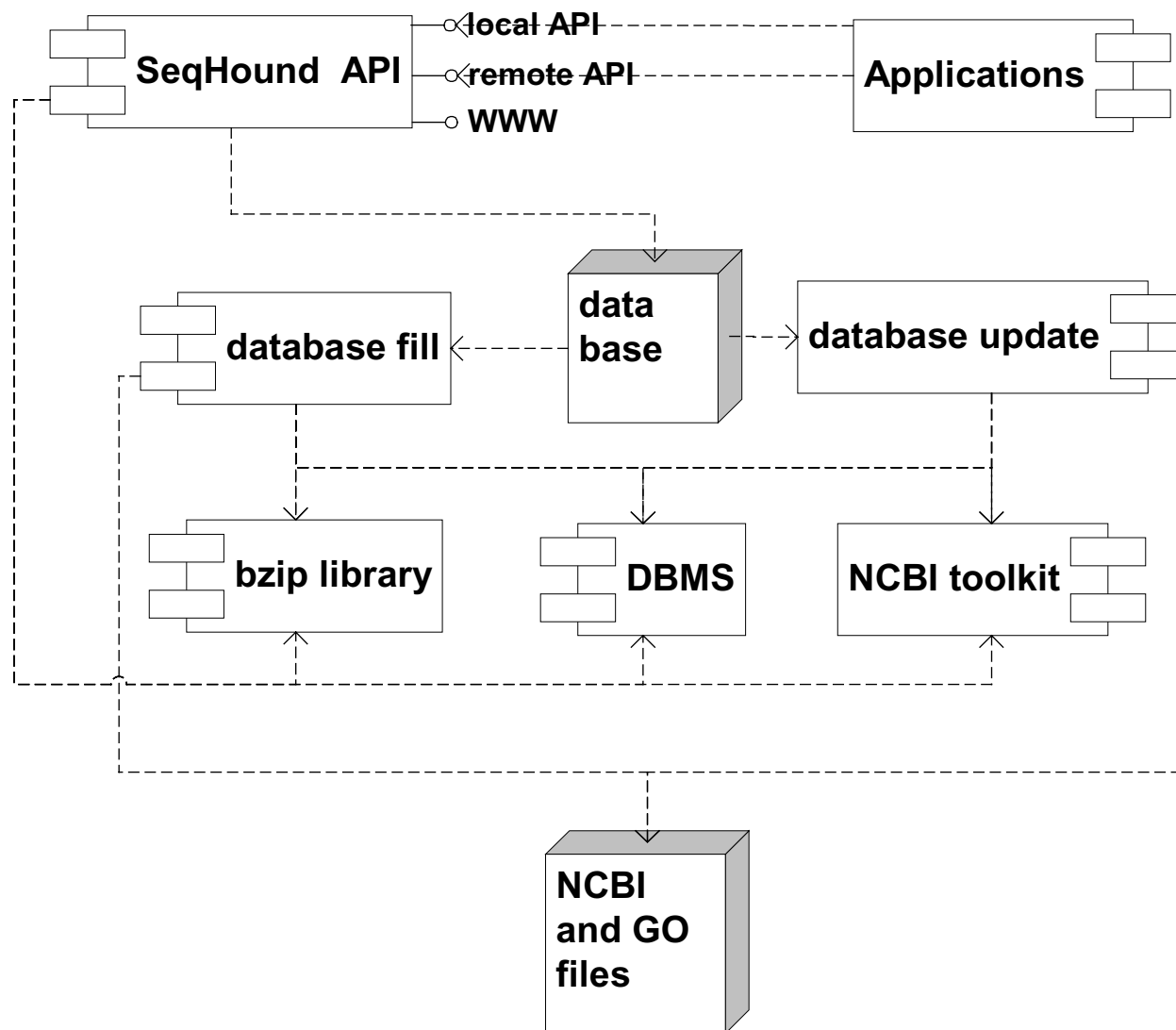
Figure 1
The SeqHound database system in UML. From the bottom up: the system relies on data provided by the NCBI FTP site and the Gene Ontology resource. It uses the NCBI programming toolkit, the database management system (DBMS) and the bzip compression scheme as programming tools. The database is filled and updated using SeqHound parsers, programming tools and NCBI data as input. The database is searched using the SeqHound query interface which is usable in three forms – as CGI-based web pages, as a local API and as a remote API. All applications (top right) are written using the SeqHound API.

sequence database is kept as well as structured data recording the assignment of entries to sets of sequences constituting complete genomes [http://www.ncbi.nlm.nih.gov:80/PMGifs/Genomes/org.html]. The SeqHound database schema and description are presented in Figure 2 and in Table 1, respectively.

Data resources and the system build process
The system is initially filled and subsequently updated from the sequence data files posted on the NCBI FTP site.

Few other data files are needed to build the system. Data for complete genomes are downloaded and parsed to mark all proteins and DNA constituting the complete genomes in our database. The non-redundant protein BLAST database (nr) is utilized to obtain information about strictly redundant protein sequences. This provides a table for the SeqHound API interface that allows the retrieval of non-redundant sequences. Additionally, the hierarchical taxonomy database, the MMDB structural database, the Locus Link resource file and GO release are

PUBSEQ

PK	GI
	MUID

NUCPROT

PK	GI protein
	GI NA

ACCDB

PK	GI
	DB name **accesssion** chain release version title

REDUND

PK	GI
	order group

BLASTDB

PK	ID
	E value ASN.1 NBlast-Result

NBLASTDB

PK	ID
	GI ASN.1 NBlast-GiAndEval

LL_LLINK

PK	GI
	Locus Link ID Chrom. location

LL_OMIM

PK	GI
	OMIM ID

PARTI

PK	GI
	division

DOMNAME

PK	CDD ID
	label MMDB ID ASN.1 CDD

RPSDB

PK	GI
	CDD ID from length E value missing N missing C total domains

DIVDB

PK	division ID
	ASN.1 SLRI-taxon-div

GCODEDB

PK	gcode ID
	ASN.1 SLRI-taxon-gencode

TAXDB

PK	tax ID
	ASN.1 SLRI-taxon

TAXGI

PK	GI
	tax ID chrom ID molecule type

CHROM

PK	tax ID
	chrom ID chrom flag

ASNDB

PK	GI
	ASN.1 Bioseq division release molecule type

SENGI

PK	GI
	SEID division

SENDB

PK	SEID
	ASN.1 Seq-Entry

MMDB

PK	MMDB ID
	ASN.1 Biostruc PDB id bwhat models molecules size bzsize

MMGI

PK	GI
	MMDB ID

LL_CDD

PK	GI
	CDD ID E value

GO_PARENT

PK	GO ID
	parent of GO ID

GO_REFERENCE

PK	GO ID
	cross-reference DB

GO_SYNONYM

PK	GO ID
	synomym name

GO_NAME

PK	GO ID
	name DB level

LL_GO

PK	GI
	GO ID PMID evidence evidence code

DOMDB

PK	GI
	MMDB ID PDB ID chain ASN.1 SLRIDomain domno domall domain ID representative

Figure 2
The database schema in UML. Each box depicts one table within the SeqHound system. The grey areas contain the table names. PK stands for "primary key". For the majority of the tables, the primary key is the GenInfo (GI) identifier. Each subsequent entry in each of the boxes indicates a field of information stored in the tables. Required fields are in bold. ASN.1 schema in these tables can be found at [http://ncbi.nlm.nih.gov/IEB] (for the Bioseq, Seq-Entry, Cdd and Biostruc) and at [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slritools/slri/seqhound/asn] for the rest of the objects.

Table 1: Summary of contents of SeqHound database tables.

Database Table	Contents Summary
ASNDBs	GI and ASN.1 Bioseq – partitioned into divisions
PARTITION	GI and division – master list
ACCDB	GI, accession number, and database specific identifier (if available)
NUCPROT	protein GI and DNA GI
PUBSEQ	GI and MEDLINE identifier
REDUND	GI and redundant group identifier
SENGI	GI and SeqEntry identifier – partitioned into divisions
SENDBs	SeqEntry identifier, ASN.1 SeqEntry and division
CHROM	GI and chromosomal identifier from complete genome
TAXGI	GI and taxonomy identifier
TAXDB	Taxonomy hierarchy
GCODEDB	Taxonomy identifier and genetic code
DIVDB	Entrez division
MMGI	GI and MMDB identifier
MMDB	MMDB identifier and ASN.1 for 3-D structures (Biostruc)
DOMDB	GI, ASN.1 structural domain and chain redundant set tag
NRBLASTDB	GI and ASN.1 list of neighbours
BLASTDB	Hashed pair of GIs and ASN.1 alignment from BLAST comparison
GO_PARENT	Gene ontology
GO_NAME	GO identifier and function name
GO_SYNONYM	GO identifier and synonym
GO_REFERENCE	GO identifier and reference to other databases (e.g. Enzyme Consortium)
LL_GO	GI and GO identifier
LL_OMIM	GI and Online Mendelian Inheritance in Man identifier
LL_LLINK	GIs, Locus Link identifier and chromosomal location
LL_CDD	GI and Conserved Domain Database identifier extracted from Locus Link database
RPSDB	GI and Conserved Domain Database identifier, details of alignment
DOMNAME	Conserved Domain Database annotation

utilized. The source files, their origin, corresponding executables and the affected SeqHound database tables are listed in Table 2.

The SeqHound system is divided into one required core module and several optionally configured modules. Modules are groups of tables and API calls that are filled using a common data resource, for example the 3D structures. The purpose of this division is to give the user an option to control hardware resources and complexity of system administration when parts of the SeqHound system are not required. The list of SeqHound modules and their data resources is contained in Table 2. After a system build, the module information is recorded in the configuration file which is then utilized by the API to determine if certain operations can be achieved with the current setup.

We import and use sequence data in binary form structured with the Abstract Syntax Notation (ASN.1) data description language [http://asn1.elibel.tm.fr/]. The data is parsed in order to index links between related records. The

NCBI programming toolkit [http://ncbi.nlm.nih.gov/IEB] provides the complete functionality needed to read and traverse the binary ASN.1 data objects in the computer main memory. In addition flat-file dumper code created and maintained by NCBI is used to generate GenBank, GenPept, XML and PDB formatted flat files.

In the NCBI data model [18], each biological sequence is represented by an ASN.1 object called a Bioseq. Relationships between different sequences (for example RNA and their encoded proteins) are captured in an object called a Bioseq-set that may contain a group of separate Bioseqs; for example one protein Bioseq for each of several coding sequence on a piece of genomic DNA. A Bioseq-set contains descriptive information that apply to all contained Bioseqs.

The basic data unit in SeqHound is the Bioseq data type. Bioseqs are removed from inside the Bioseq-set record and stored in the core database table (ASNDB) to provide for fast protein sequence and annotation retrieval. This normalization of the underlying ASN.1 binary objects

Table 2: Parsers and resource files needed to build and update SeqHound.

Input File	Resource	Parser	Tables Modified	Module
ASN.1 sequences	[ftp://ncbi.nlm.nih.gov/ncbi-asn1/*.aso] [ftp://ncbi.nlm.nih.gov/refseq/cumulative/*.bna]	mother	ASNDB, PARTI, NUCPOT, ACCDB, PUBSEQ, TAXGI, SENDB, SENGI	core
ASN.1 sequences	[ftp://ncbi.nlm.nih.gov/ncbi-asn1/daily-nc/*.aso] [ftp://ncbi.nlm.nih.gov/refseq/daily/*.bna]	update	ASNDB, PARTI, NUCPOT, ACCDB, PUBSEQ, TAXGI, SENDB, SENGI	core
FASTA nr database	[ftp://ncbi.nlm.nih.gov/blast/db/nr]	redund	REDUND	redundb
List of complete genomes (flat file)	[http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slritools/slri/seqhound/genomes/chromff]	chrom	CHROM	gendb
ASN.1 for complete genomes	[ftp://ncbi.nlm.nih.gov/genomes/*/*.asn]	comgen	TAXGI, ACCDB	gendb
Taxonomy release (flat file)	[ftp://ncbi.nlm.nih.gov/pub/taxonomy/taxdump.tar]	importtaxdb	TAX, GCODE, DIV, DEL, MERGE	taxdb
ASN.1 MMDB release	[ftp://ncbi.nlm.nih.gov/mmdb/mmdbdata/*.val]	cbmmdb	MMDB, MMGI	strucdb
MMDB (database table)	MMDB table	vastblst	DOMDB	strucdb
3-D chain BLAST sets (flat file)	[ftp://ncbi.nlm.nih.gov/mmdb/nrtable/nrpdb.*]	pdbrep	DOMDB	strucdb
FASTA nr database	[ftp://ncbi.nlm.nih.gov/blast/db/nr]	nblast	nrB	neigdb
BLAST ASN.1 results	nrB table available nrB and nrN tables availables at [ftp://ftp.mshri.on.ca/pub/NBLAST]	nbraccess	nrN	neigdb
LL_tmpl (flat file)	[ftp://ncbi.nlm.nih.gov/refseq/LocusLink/LL_tmpl]	llparser	LL_OMIM, LL_GO, LL_LLINK, LL_CDD	lldb
gene_associaton.com pugen.GenBank/Swissprot (flat files)	[http://www.geneontology.org]	addgoid	LL_GO	lldb
function.ontology process.ontology component.ontology (flat files)	[http://www.geneontology.org]	goparser	GO_PARENT, GO_NAME, GO_REFERENCE, GO_SYNONYM	godb
CDD database	[ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/]	domname	DOMNAME	rpsdb
FASTA nr database and CDD database	[ftp://ncbi.nlm.nih.gov/blast/db/nr] [ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/] DOMNAME and RPSDB tables available at [ftp://ftp.mshri.on.ca/pub/RPS]	rpsdb	RPSDB	rpsdb

makes FASTA formatted protein sequence retrieval much more efficient, as it obviates parsing Bioseq-sets to locate individual protein sequences.

The Bioseq-set descriptive units stripped of their Bioseqs are also saved in a separate database (SENDB). Bioseq-set records are reconstructed with their constituent Bioseqs when required at run time and they are used as an input for NCBI toolkit GenBank flat file generators. These functions guarantee the authenticity of the GenBank flat file generated from the ASN.1 binary data.

SeqHound was originally developed using the CodeBase® system from Sequiter® Software Inc. [http://www.sequiter.com], a commercial database management system. We selected CodeBase because it is a fast, lightweight database engine that allows for royalty-free distribution of executables. Further, the system does not require a Database Administrator trained in management of large table systems in order to operate and install SeqHound. It is easily linked with the NCBI toolkit on a number of platforms and provides a streamlined database subsystem capable of

holding the sequence and structural databases. System design includes a modular, abstracted database layer that allows for the database back-end to be switched. Recently, the SeqHound database has been successfully back-ended to DB2® RDBMS [http://www-4.ibm.com/software/data/db2/]. Other RDBMS back-ends may be used by customizing the SeqHound source code.

To build the SeqHound database, several parser executables are needed. They operate with different input data files to load and index the system with specific information. Since the volume of data involved is very large, the execution of the major parser requires an extensive amount of time. The individual parsers have been kept separate to allow for better control over the system build and to provide a future option for parallelization of the indexing process. The entire build process including the download of all the data is executed and controlled by shell and Perl scripts which are customized for the local servers' directory and web structure. The programmer's manual can be found at [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slritools/slri/seqhound/shreadme].

Updating the database

Once a release version of the system is built, the core of the SeqHound system is updated on a daily basis by a "cron job" on a Unix platform and by a "scheduled task" on a Windows platform until the next database release. Currently, the NCBI sequence data files are released bimonthly. Between the releases, daily update files are posted on the NCBI FTP site in the same ASN.1 binary format as the release files. These files provide the source for the SeqHound system daily updates. The update executable, source data files and their origin are shown in Table 2.

In the first step, the update algorithm identifies sequence entries in the update file that have been updated and deletes them from the current system. There are two types of updates; first those sequences that undergo minor annotation updates are re-released with the same GI and the same accession and version number. Sequences with changes to the sequence itself are released with a new GI and with the same accession supplemented with an incremented version number. Both types of changes are identified and marked in the system by the update process. Second, all new entries in the update file are appended to the existing databases.

Additionally, we implemented regular updates for LocusLink, complete genomes and redundant sequences information. These modules are rebuilt from freshly downloaded source files on a daily basis. Finally, every day after the system is updated, the largest taxonomy queries are pre-computed.

At the end of each two month daily update cycle – with a new release of the Entrez databases – the SeqHound system is rebuilt semi-manually using the parsers listed in Table 2 and several scripts which download the release and execute the appropriate parsers on each of the data files. The core module is built from the new ASN.1 release of the Entrez databases. The optional SeqHound modules (godb, lldb, taxdb, strucdb, gendb and redundb) are built using the original parsers and freshly downloaded source files described in Table 2. The tables constituting the neighbours and RPS-BLAST modules (neigdb and rpsdb) can be downloaded from [[ftp://ftp.mshri.on.ca/NBLAST] and RPS].

Alternatively, the core module can be updated continuously from daily files without rebuilding every couple of months. In this case, all optional modules should be rebuilt regularly from freshly downloaded data sources.

The Application Programming Interface (API)

The UML for the API specification is depicted in Figure 3. The API provides extensive SeqHound database access via a number of functions from C, C++ and Perl. These are designed to build up complex non-textual queries. The interface can operate locally from the C interface on a computer which hosts the database subsystem as well as remotely through an HTTP interface from a client networked computer. This is useful within a local bioinformatics lab setting. An additional benefit of the HTTP based API lies in its accessibility to any programming language capable of executing HTTP calls such as Perl, Java, Visual Basic, C and C++. The Perl and C++ APIs are available together with SeqHound source code. The server component is implemented as a Fast CGI application [http://www.fastcgi.com/] and is optimized for performance of API calls.
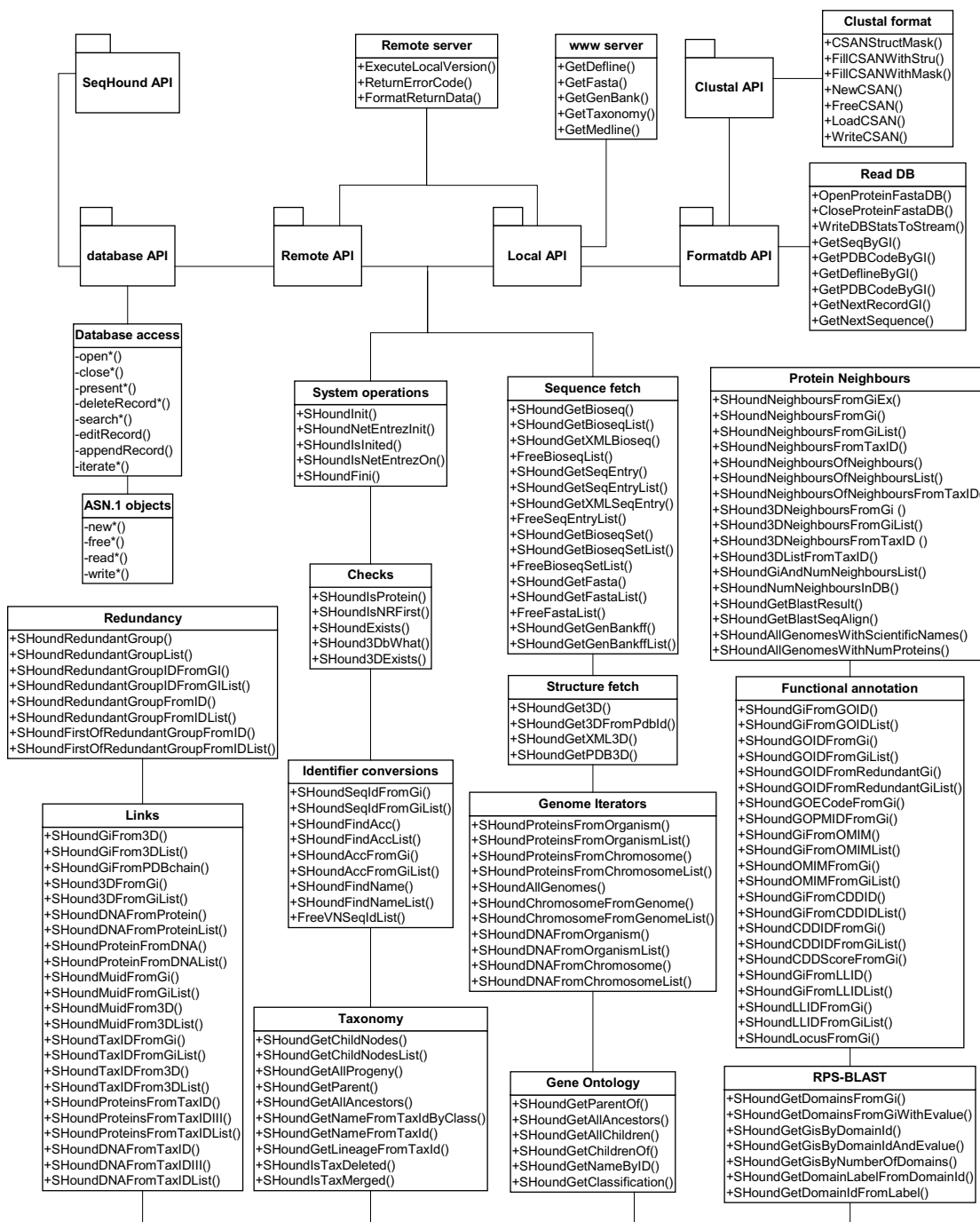
The database layer performs abstracted database administration operations such as initialization, closing, searching, editing, presenting and deleting a record. The API on the top of the database layer deals with SeqHound system opening, testing and closing. The system can also be optionally opened to use the Web Entrez NCBI service for retrieval of obsolete GI identifiers that are not contained in the local SeqHound database. Versions of sequence records that were updated prior to the most recent database release are referred to as obsolete GI identifiers. These GIs are only accessible through the Entrez service and are not part of regular Entrez database releases.

Most of the API function calls come in two versions, one that operates on a single input identifier and another version that accepts a list of identifiers. The API performs conversions of accession numbers and other sequence identifiers (e.g. from PDB, Swiss-Prot or PIR databases) to GI identifiers which serve as input for querying the database. As the TrEMBL database [12] is not a part of the Entrez database collection, searches for TREMBL identifiers are not currently supported.

The next category of basic API functions provides system content checks such as "Is the sequence in the system?", "Is the sequence a protein?" or "Is there a structure for the sequence?" etc.

SeqHound links

Sequence identifiers link to many additional resources within SeqHound such as taxonomy, MEDLINE, MMDB, VAST domains, GO, LocusLink, CDD and OMIM. Additionally, the sequences are inter-linked based on sequence identity, on sequence similarity, on links between DNA and proteins as implied by coding sequence features. All listed links are accessible by numerous API functions in both directions i.e. using GI as an input and collecting GI as an output of a function.

SeqHound API

Remote server
+ExecuteLocalVersion()
+ReturnErrorCode()
+FormatReturnData()

www server
+GetDefline()
+GetFasta()
+GetGenBank()
+GetTaxonomy()
+GetMedline()

Clustal API

Clustal format
+CSANStructMask()
+FillCSANWithStru()
+FillCSANWithMask()
+NewCSAN()
+FreeCSAN()
+LoadCSAN()
+WriteCSAN()

database API   **Remote API**   **Local API**   **Formatdb API**

Read DB
+OpenProteinFastaDB()
+CloseProteinFastaDB()
+WriteDBStatsToStream()
+GetSeqByGI()
+GetPDBCodeByGI()
+GetDeflineByGI()
+GetPDBCodeByGI()
+GetNextRecordGI()
+GetNextSequence()

Database access
-open*()
-close*()
-present*()
-deleteRecord*()
-search*()
-editRecord()
-appendRecord()
-iterate*()

ASN.1 objects
-new*()
-free*()
-read*()
-write*()

System operations
+SHoundInit()
+SHoundNetEntrezInit()
+SHoundIsInited()
+SHoundIsNetEntrezOn()
+SHoundFini()

Sequence fetch
+SHoundGetBioseq()
+SHoundGetBioseqList()
+SHoundGetXMLBioseq()
+FreeBioseqList()
+SHoundGetSeqEntry()
+SHoundGetSeqEntryList()
+SHoundGetXMLSeqEntry()
+FreeSeqEntryList()
+SHoundGetBioseqSet()
+SHoundGetBioseqSetList()
+FreeBioseqSetList()
+SHoundGetFasta()
+SHoundGetFastaList()
+FreeFastaList()
+SHoundGetGenBankff()
+SHoundGetGenBankffList()

Protein Neighbours
+SHoundNeighboursFromGiEx()
+SHoundNeighboursFromGi()
+SHoundNeighboursFromGiList()
+SHoundNeighboursFromTaxID()
+SHoundNeighboursOfNeighbours()
+SHoundNeighboursOfNeighboursList()
+SHoundNeighboursOfNeighboursFromTaxID()
+SHound3DNeighboursFromGi ()
+SHound3DNeighboursFromGiList()
+SHound3DNeighboursFromTaxID ()
+SHound3DListFromTaxID()
+SHoundGiAndNumNeighboursList()
+SHoundNumNeighboursInDB()
+SHoundGetBlastResult()
+SHoundGetBlastSeqAlign()
+SHoundAllGenomesWithScientificNames()
+SHoundAllGenomesWithNumProteins()

Checks
+SHoundIsProtein()
+SHoundIsNRFirst()
+SHoundExists()
+SHound3DbWhat()
+SHound3DExists()

Redundancy
+SHoundRedundantGroup()
+SHoundRedundantGroupList()
+SHoundRedundantGroupIDFromGI()
+SHoundRedundantGroupIDFromGIList()
+SHoundRedundantGroupFromID()
+SHoundRedundantGroupFromIDList()
+SHoundFirstOfRedundantGroupFromID()
+SHoundFirstOfRedundantGroupFromIDList()

Structure fetch
+SHoundGet3D()
+SHoundGet3DFromPdbId()
+SHoundGetXML3D()
+SHoundGetPDB3D()

Functional annotation
+SHoundGiFromGOID()
+SHoundGiFromGOIDList()
+SHoundGOIDFromGi()
+SHoundGOIDFromGiList()
+SHoundGOIDFromRedundantGi()
+SHoundGOIDFromRedundantGiList()
+SHoundGOECodeFromGi()
+SHoundGOPMIDFromGi()
+SHoundGiFromOMIM()
+SHoundGiFromOMIMList()
+SHoundOMIMFromGi()
+SHoundOMIMFromGiList()
+SHoundGiFromCDDID()
+SHoundGiFromCDDIDList()
+SHoundCDDIDFromGi()
+SHoundCDDIDFromGiList()
+SHoundCDDScoreFromGi()
+SHoundGiFromLLID()
+SHoundGiFromLLIDList()
+SHoundLLIDFromGi()
+SHoundLLIDFromGiList()
+SHoundLocusFromGi()

Links
+SHoundGiFrom3D()
+SHoundGiFrom3DList()
+SHoundGiFromPDBchain()
+SHound3DFromGi()
+SHound3DFromGiList()
+SHoundDNAFromProtein()
+SHoundDNAFromProteinList()
+SHoundProteinFromDNA()
+SHoundProteinFromDNAList()
+SHoundMuidFromGi()
+SHoundMuidFromGiList()
+SHoundMuidFrom3D()
+SHoundMuidFrom3DList()
+SHoundTaxIDFromGi()
+SHoundTaxIDFromGiList()
+SHoundTaxIDFrom3D()
+SHoundTaxIDFrom3DList()
+SHoundProteinsFromTaxID()
+SHoundProteinsFromTaxIDIII()
+SHoundProteinsFromTaxIDList()
+SHoundDNAFromTaxID()
+SHoundDNAFromTaxIDIII()
+SHoundDNAFromTaxIDList()

Identifier conversions
+SHoundSeqIdFromGi()
+SHoundSeqIdFromGiList()
+SHoundFindAcc()
+SHoundFindAccList()
+SHoundAccFromGi()
+SHoundAccFromGiList()
+SHoundFindName()
+SHoundFindNameList()
+FreeVNSeqIdList()

Genome Iterators
+SHoundProteinsFromOrganism()
+SHoundProteinsFromOrganismList()
+SHoundProteinsFromChromosome()
+SHoundProteinsFromChromosomeList()
+SHoundAllGenomes()
+SHoundChromosomeFromGenome()
+SHoundChromosomeFromGenomeList()
+SHoundDNAFromOrganism()
+SHoundDNAFromOrganismList()
+SHoundDNAFromChromosome()
+SHoundDNAFromChromosomeList()

Taxonomy
+SHoundGetChildNodes()
+SHoundGetChildNodesList()
+SHoundGetAllProgeny()
+SHoundGetParent()
+SHoundGetAllAncestors()
+SHoundGetNameFromTaxIdByClass()
+SHoundGetNameFromTaxId()
+SHoundGetLineageFromTaxId()
+SHoundIsTaxDeleted()
+SHoundIsTaxMerged()

Gene Ontology
+SHoundGetParentOf()
+SHoundGetAllAncestors()
+SHoundGetAllChildren()
+SHoundGetChildrenOf()
+SHoundGetNameByID()
+SHoundGetClassification()

RPS-BLAST
+SHoundGetDomainsFromGi()
+SHoundGetDomainsFromGiWithEvalue()
+SHoundGetGisByDomainId()
+SHoundGetGisByDomainIdAndEvalue()
+SHoundGetGisByNumberOfDomains()
+SHoundGetDomainLabelFromDomainId()
+SHoundGetDomainIdFromLabel()

Figure 3
The application programming interface (API) in UML. The SeqHound API consists of the database administration API, the local and remote query APIs, the formatdb API and the Clustal API. The remote server executes remote API requests using local API and returns results to a client. The WWW server utilizes the local API to present WWW pages to the user. Each box contains a group of programming functions with similar purpose. The individual functions are used to retrieve a set of data from the SeqHound system.

Taxonomy database

The NCBI taxonomy database is stored as a hierarchical graph. Each node in the hierarchical taxonomy database is referred to by an integer identifier – taxonomy identifier. The API enables the user to fetch full taxonomy names, taxonomy lineages and child or parental nodes in the taxonomy tree using the taxonomy identifiers. Correspondingly, the hierarchical database enables retrieval of sequences from whole taxonomy sub-trees rather than from individual taxons. This functionality may be used in combination with API functions that return FASTA formatted files in order to create customized, daily updated BLAST formatted databases of certain taxon branches, for example FASTA files from all non-redundant mammalian protein sequences, or all proteins from *archaea*.

Complete genomes

Another set of API functions operates on complete genome sets as they are made available through the Entrez service. This layer is aware of all completed genome sequencing projects in SeqHound and of the contents of individual completed genomes in terms of individual chromosomes and other DNA elements. The functions can return lists of DNA/proteins from a single chromosome (or another DNA element such as plasmid, mitochondrion or extrachromosomal element) and subsequently build up all DNA or all non-redundant protein lists from the entire sequenced genome. Such functionality is useful in bioinformatics research projects that investigate a specific organism or family of organisms. We are using this functionality to explore protein thermostability in thermophilic genomes and have computed high quality structural alignments from 27,370 out of 219,848 predicted open reading frames from 78 complete genome sequencing projects for a project examining the utility of species specific sequence composition-based scoring functions (Dumontier *et al.*, submitted).

Data formats

Once a desired subset of sequences is obtained, a full sequence annotation can be fetched in several formats including FASTA, GenBank flat file, Bioseq and Seq-Entry in ASN.1 and the Extensible Markup Language (XML) [http://www.w3.org/XML/]. The 3-D structures can be obtained in either PDB or MMDB Biostruc format (ASN.1 and XML).

Protein sequence neighbours

The sequence neighbour information was reconstructed using the BLAST blastpgp executable from the NCBI toolkit [8]. The blastpgp executable was customized so it can be implemented on a Beowulf cluster [http://www.beowulf.org/] to allow parallelization of the task. Each sequence from the non-redundant protein BLAST database (nr) was compared to the nr database to generate an exhaustive database of alignments. From all possible N × N alignments only $N(N - 1)/2$ alignments were computed. Sequence neighbours to protein sequences in SeqHound can be retrieved together with the BLAST E-value statistics for individual matches. Upon retrieval, an E-value cutoff can be applied.

CDD domain mapping

We used the reverse position specific (RPS) BLAST algorithm from the NCBI toolkit to identify common motifs/domains in proteins. The program rpsblast was modified to store RPS-BLAST output in the form of a database table. All proteins from the nr database were compared to domains in the CDD database and the results were stored in SeqHound tables together with the full domain annotation. The API for this module retrieves all pre-computed domains for a given protein based on an E-value cutoff. The information about domain mapping contains E-value, position, length of the alignment, N and C-terminal residues of the consensus domain missing and the total number of domains identified on the protein. The domains in CDD possess unique identifiers derived from Pfam [19], SMART [20] and NCBI internal databases. Additionally, domains are identified by popular, non-unique labels such as SH2, MATH etc. SeqHound accepts both kinds of identifiers for searching.

Clustal and Formatdb API

Two additional local interfaces are implemented in the C programming language. The Clustal W program [21,22] is a multiple sequence alignment program for DNA or proteins. It produces biologically meaningful multiple sequence alignments of divergent sequences. A seldom used, but powerful feature in the Clustal package is the ability to use position specific gap penalties, seeded with information from secondary structure derived from protein 3D structures. In SeqHound, the Clustal API is used to generate secondary structure masks in the Clustal format from protein sequences that originate from MMDB structure records. The Clustal program accepts the secondary structure mask for one sequence in multiple alignments. The mask is then used to compute position specific gap penalties which force gaps into loop regions and enhance the quality of the alignments.

The formatdb API is based on the formatdb and readdb interface from the NCBI toolkit. This is part of NCBI BLAST that provides sequences and definition lines to the BLAST core. The formatdb executable transforms a FASTA database into a binary searchable database. The database can be queried with the readdb interface that allows simple software projects in C to compute using BLAST style formatted and indexed databases that are optimized for sequence reading functionality.

WWW interface

A WWW interface to SeqHound [http://seqhound.mshri.on.ca/] was originally designed for testing purposes and is now used in connection with other bioinformatics projects in the laboratory. The interface facilitates database searches for sequences based on GI identifiers, accession numbers, NCBI taxonomy identifiers, MEDLINE identifiers and sequence names in other databases such as PDB, Swiss-Prot and PIR. Nucleic acid and protein sequences are displayed either as a short definition line, FASTA, GenBank/GenPept flat file, XML or ASN.1 print format. No text-based searches are currently provided as the system is designed mainly as a computational resource, but will follow as we complete the fielded text-query indexing system for SeqHound.

WWW user queries trigger a search for linked nucleic acid or proteins, taxonomy, MEDLINE and protein sequence neighbour links. The MEDLINE and taxonomy searches offer a brief summary page with a direct link to the NCBI web site to obtain information about a taxon or an abstract of a published article concerning the sequence of interest. If available, the web page displays a listing of the protein redundant group. The interface has been optimized for fast execution of taxonomy queries; requests for sequences from an organisms which generally take longer than 10 seconds have been pre-computed and the lists of GI identifiers are stored on the server for immediate retrieval. These pre-computed lists are updated on a daily basis.

Examples

Four examples of pseudo-code programs have been included in Appendix A to demonstrate the use of the SeqHound API. The full code is available at [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slritools/slri/seqhound/examples] in the C programming language.

The first example called "gen2fasta" illustrates how SeqHound can be used for assembling protein sequences from a complete genome as a FASTA flat file database. All proteins from a complete genome are fetched and the output list is converted into a list of FASTA entries. In the last step, the FASTA entries are written out in a desired line length.

The FASTA database can be subsequently used for creating a local searchable database of sequences using the "formatdb" executable and the formatdb API. This is demonstrated in the second example, "readdb". The same FASTA database format can be used as input for local BLAST searches using the BLAST executables provided by the NCBI.

The third example, "gen2struc", retrieves all 3-D structures in PDB format from one organism. First, a list of protein GIs from an organism is retrieved; second, a redundant group of sequences (or sequence neighbours) is found for each protein and added to the list. The new list is used to search for structural identifiers using a database of links to MMDB structures. The identified structures are then printed out in the PDB format. Note that if the redundant group functionality is used, the program finds structures with an exact sequence match and if sequence neighbours are used the program finds more distant structural matches.

The fourth example facilitates multiple alignment input to the Clustal program. The example program "clustmask" retrieves a 3-D structural annotation given a PDB code and chain and outputs a sequence with its secondary structure mask formatted for Clustal. The resulting file can be passed directly to the Clustal executable. Example of tyrosyl tRNA synthetase (PDB code – 2TS1) sequence with a secondary structure mask is shown in Figure 4.

Discussion

We have developed an in-house database system that provides similar functionality as the NCBI Entrez service, but specifically for bioinformatics scripting and batch sequence retrieval for further processing. SeqHound stores full annotation for nucleic acids, proteins, 3-D structures and multiple links to additional resources. These subsections are all inter-linked.

We have added features that are not contemplated in the Entrez WWW interface and made them available for scripted access, such as information about strictly redundant sequences, Gene Ontology, the structural domain database and the Clustal file format functionality. The primary utility of SeqHound lies in the extensive Application Programming Interface that supports new bioinformatics applications and scripts. SeqHound is used by the Biomolecular Interaction Network Database (BIND) system [3,4], the PreBIND literature mining engine [http://bind.ca], in researching thermostability of proteins (Dumontier *et al.*, submitted) and by the Kangaroo regular expression search engine [5] and in other research laboratories in our area.

The SeqHound system can be set up as a local server on several platforms (Windows®, Linux, Solaris®, HP-UX®, AIX®, IRIX®). It is predominantly dependent on the underlying ASN.1 NCBI source file format thus circumventing complex parser maintenance. The API provides a query power without using a remote database federation system such as that used by the Kleisli system [23]. The local database source removes inherent Internet transmission problems.

```
CLUSTAL X


!SS_gi|230779|pdb|2TS1|        .AAAAAAAAA...bBBb.AAAAAAAAA..bBBBBBBb....bbbb..aAAAAAAAAAAA
gi|230779|pdb|2TS1|            MDLLAELQWRGLVNQTTDEDGLRKLLNEERVTLYCGFDPTADSLHIGHLATILTMRRFQQ


!SS_gi|230779|pdb|2TS1|        a.bBBBBBBb..aaa...............AAAAAAAAAAAAAAa..........bbBB
gi|230779|pdb|2TS1|            AGHRPIALVGGATGLIGDPSGKKSERTLNAKETVEAWSARIKEQLGRFLDFEADGNPAKI


!SS_gi|230779|pdb|2TS1|        BBbaaaa...aAAAAAAAaaaa..aaaaa..aaaa........aaaaaaAAAAAAAAAAA
gi|230779|pdb|2TS1|            KNNYDWIGPLDVITFLRDVGKHFSVNYMMAKESVQSRIETGISFTEFSYMMLQAYDFLRL


!SS_gi|230779|pdb|2TS1|        AAAa..BBBBBB...AAAAAAAAAAAAAAa....bBBBBB..................bb
gi|230779|pdb|2TS1|            YETEGCRLQIGGSDQWGNITAGLELIRKTKGEARAFGLTIPLVTKADGTKFGKTESGTIW


!SS_gi|230779|pdb|2TS1|        bb.....AAAAAAAAA......AAAAAAAA....AAAAAAAAAAAAA.....AAAAAAAA
gi|230779|pdb|2TS1|            LDKEKTSPYEFYQFWINTDDRDVIRYLKYFTFLSKEEIEALEQELREAPEKRAAQKTLAE


!SS_gi|230779|pdb|2TS1|        AAAAAA..AAAAAAAAAA.........................................
gi|230779|pdb|2TS1|            EVTKLVHGEEALRQAIRISEALFSGDIANLTAAEIEQGFKDVPSFVHEGGDVPLVELLVS


!SS_gi|230779|pdb|2TS1|        ..........................................................
gi|230779|pdb|2TS1|            AGISPSKRQAREDIQNGAIYVNGERLQDVGAILTAEHRLEGRFTVIRRGKKKYYLIRYA
```

Figure 4
Clustal formatted tyrosyl tRNA synthetase sequence. The letter "A" denotes an α-helix, "B" a β-strand. The capital letters indicate that the automated secondary structure assignment (as annotated in the MMDB database) and the assignment by authors agreed while the lower case letter indicates that there was a disagreement.

Future directions

SeqHound is still developing in many aspects and can be adapted readily, for example, by addition of more tables that index extra information. Similarly, new functionality can be readily added to the API to match new tables. Potential future developments could include indexing of DNA sequence features, for example, we have added an independent table of nucleotide FASTA database of coding regions generated from nucleotide entries to support a regular expression sequence search engine developed in our group, Kangaroo [5]. We are preparing text index table and text based retrieval functionality for SeqHound using an internally developed generic ASN.1 text-indexing engine [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slri-tools/slri/textindex], currently in use for the BIND project.

An additional feature request has been for the capacity of the SeqHound system to store and archive local protein and nucleotide sequences that are not yet present in public databases. We are working on a variant called SeqHound Archive Retrieval Format (SeqHound ARF) that adds additional tables, and creates local unique primary keys, replacing the use of GI numbers for internal indexing. This system should prove useful for genomics and proteomics discovery research efforts prior to public sequence submission.

As noted, TREMBL protein sequences are not currently supported, however the efforts at EBI to create an identifier translation system matching TREMBL identifiers with NCBI GI's from GenPept may provide a solution to integrating this information.

The NBLAST [8] and RPS-BLAST protein neighbor and domain annotation computations have added functionality to the SeqHound information systems thanks to our capacity for high-performance computing. However since these files are exportable, this design should allow users to set up SeqHound systems on relatively lightweight servers without requiring access to large clusters, but still have access to information like sequence neighbors or computed domain hits and updates through an FTP site. This is part of our future development goals. Our future plans include an addition of a full MEDLINE database. We are also in the process of developing a new WWW interface that takes full advantage of the robust API. Finally, we are testing a SeqHound Perl module built specifically for the Bioperl bioinformatics software collection [http://bioperl.org].

Methods

The SeqHound application was written in the ANSI C programming language using the NCBI programming toolkit [http://ncbi.nlm.nih.gov/IEB], the CodeBase® database

engine [http://www.sequiter.com], the bzip library [http://sources.redhat.com/bzip2/] and other libraries developed in our group.

The CodeBase database engine is an extended form of the Xbase database subsystem (e.g. FoxPro®, Dbase III) that is capable of supporting large 64-bit filesystems. A unique advantage of using this system is that databases and indexes are portable between platforms which use different byte order to store integers; for example, tables created on Intel ® Linux machines can be interchanged with other Unix platforms (Solaris®, Irix®, HP-UX®, AIX®) without being re-indexed or reformatted. This allows us to perform parallel calculations on lightweight Linux cluster nodes and copy the resulting database files directly onto a SUN® external server. This also implies that database tables and indexes may be obtained directly from our FTP server and used on various platforms.

Although the system is written in C, the remote interface can be addressed in different programming languages capable of performing HTTP requests. A conversion of the remote API to Perl and C++ languages was completed; the Perl module is available from [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slritools/slri/seqhound/perl] and the C++ API from [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slritools/slri/seqhound/include_cxx] and [http://src_cxx].

Given the vast amount of data, the system has to maintain very large data files. Currently, the SeqHound system runs and is regularly tested on true 64-bit platforms (HP-UX®, IRIX®, Solaris®, AIX®), Linux and on Windows® operating systems. The size of the SeqHound database grows exponentially as does Entrez. Currently, the hard disk space needed to build and update one cycle of the full SeqHound system is about 300 GB. Since the Entrez databases are released in many separate parts or divisions (which reflect the taxonomy or the experimental approach used to obtain the sequences), the size of the system can be reduced by omitting some of the divisions, which may not be needed by the user, in the original build. For example, for our 3-D structural research we have not been including the dbEST divisions; this reduced the system size by one half. The time needed to build the entire system varies on different platforms and with the growing amount of data; currently the parsers require about 60 hours to process one full release (built on a Sun E450 with 1.5 GB RAM).

List of abbreviations
ASN Abstract Syntax Notation

ANSI American National Standards Institute

API Application Programming Interface

BIND Biomolecular Interaction Network Database

BLAST Basic Local Alignment Search Tool

CDD Conserved Domain Database

DBMS Database Management System

DNA deoxyribonucleic acid

EST expressed sequence tag

EMBL European Molecular Biology Laboratories

E-value expectation value

FTP File Transfer Protocol

GO Gene Ontology

GI GenInfo identifier

HTTP Hypertext Transfer Protocol

MMDB Molecular Modeling Database

NCBI National Center for Biotechnology Information

OMIM Online Mendelian Inheritance in Man

PERL Practical Extraction and Report Language

PDB Protein Data Bank

PIR Protein Information Resource

RPS-BLAST reverse position specific BLAST

RNA ribonucleic acid

SRS Sequence Retrieval System

SMART Simple Modular Architecture Research Tool

UML Unified Modeling Language

VAST Vector Alignment Search Tool

WWW World Wide Web

XML Extensible Markup Language

Authors' contributions

Katerina Michalickova developed the first version of Se-qHound and continues to oversee its expansion in great detail. She takes care of daily maintenance, code updates and testing, user concerns and documentation.

Gary D. Bader developed the taxonomy module and provided invaluable input into building SeqHound.

Michel Dumontier developed the neighbours module and helped with grooming the source tree.

Hao Lieu developed the GO and Locus Link modules, Perl, Bioperl and C++ remote APIs.

Doron Betel developed the RPS-BLAST module.

Ruth Isserlin ported SeqHound to DB2, worked on the WWW user interface and developed the pre-computed taxonomy queries.

Additional material

Additional File 1

Examples of SeqHound usage
Click here for file
[http://www.biomedcentral.com/content/supplementary/1471-2105-3-32-S1.doc]

References

1. Schuler GD, Epstein JA, Ohkawa H, Kans JA: **Entrez: molecular biology database and retrieval system.** *Methods Enzymol* 1996, **266:**141-162
2. Stoesser G, Baker W, van den BA, Camon E, Garcia-Pastor M, Kanz C, Kulikova T, Leinonen R, Lin Q, Lombard V, *et al*: **The EMBL Nucleotide Sequence Database.** *Nucleic Acids Res* 2002, **30:**21-26
3. Bader GD, Hogue CW: **BIND-a data specification for storing and describing biomolecular interactions, molecular complexes and pathways.** *Bioinformatics* 2000, **16:**465-477
4. Bader GD, Donaldson I, Wolting C, Ouellette BF, Pawson T, Hogue CW: **BIND-The biomolecular interaction network database.** *Nucleic Acids Res* 2001, **29:**242-245
5. Betel D, Hogue CW: **Kangaroo – A pattern-matching program for biological sequences.** *BMC Bioinformatics* 2002, **3:**20
6. Michalickova K, Dharsee M, Hogue CWV: **Sequence analysis on a 216 processor Beowulf cluster.** *4th Annual Linux Showcase and Conference, Atlanta* 2000, **4:**111-119
7. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: **Basic local alignment search tool.** *J Mol Biol* 1990, **215:**403-410
8. Dumontier M, Hogue CW: **NBLAST: a cluster variant of BLAST for NxN comparisons.** *BMC Bioinformatics* 2002, **3:**13
9. Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Rapp BA, Wheeler DL: **GenBank.** *Nucleic Acids Res* 2002, **30:**17-20
10. Pruitt KD, Maglott DR: **RefSeq and LocusLink: NCBI gene-centered resources.** *Nucleic Acids Res* 2001, **29:**137-140
11. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE: **The Protein Data Bank.** *Nucleic Acids Res* 2000, **28:**235-242
12. Bairoch A, Apweiler R: **The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000.** *Nucleic Acids Res* 2000, **28:**45-48
13. Boguski MS, Lowe TM, Tolstoshev CM: **dbEST – database for "expressed sequence tags".** *Nat Genet* 1993, **4:**332-333
14. Wang Y, Anderson JB, Chen J, Geer LY, He S, Hurwitz DI, Liebert CA, Madej T, Marchler GH, Marchler-Bauer A, *et al*: **MMDB: Entrez's 3D-structure database.** *Nucleic Acids Res* 2002, **30:**249-252
15. Wu CH, Huang H, Arminski L, Castro-Alvear J, Chen Y, Hu ZZ, Ledley RS, Lewis KC, Mewes HW, Orcutt BC, *et al*: **The Protein Information Resource: an integrated public resource of functional annotation of proteins.** *Nucleic Acids Res* 2002, **30:**35-37
16. Marchler-Bauer A, Panchenko AR, Shoemaker BA, Thiessen PA, Geer LY, Bryant SH: **CDD: a database of conserved domain alignments with links to domain three-dimensional structure.** *Nucleic Acids Res* 2002, **30:**281-283
17. The Gene Ontology Consortium: **Creating the gene ontology resource: design and implementation.** *Genome Res* 2001, **11:**1425-1433
18. Ostell JM, Kans JA: **The NCBI data model.** *Methods Biochem Anal* 1998, **39:**121-144
19. Bateman A, Birney E, Cerruti L, Durbin R, Etwiller L, Eddy SR, Griffiths-Jones S, Howe KL, Marshall M, Sonnhammer EL: **The Pfam protein families database.** *Nucleic Acids Res* 2002, **30:**276-280
20. Letunic I, Goodstadt L, Dickens NJ, Doerks T, Schultz J, Mott R, Ciccarelli F, Copley RR, Ponting CP, Bork P: **Recent improvements to the SMART domain-based sequence annotation resource.** *Nucleic Acids Res* 2002, **30:**242-244
21. Higgins DG, Sharp PM: **CLUSTAL: a package for performing multiple sequence alignment on a microcomputer.** *Gene* 1988, **73:**237-244
22. Thompson JD, Higgins DG, Gibson TJ: **CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.** *Nucleic Acids Res* 1994, **22:**4673-4680
23. Chung SY, Wong L: **Kleisli: a new tool for data integration in biology.** *Trends Biotechnol* 1999, **17:**351-355