Storage and visualisation of genomic information

Master of Science Thesis in Computing Science

Linda Wiklund

10th September 2004

Supervisors:

Pär Larsson - *Swedish Defence Research Agency, NBC Defence Division, Department of NBC Analysis, SE-901 82 Umeå.*

Michael Minock - *Department of Computing Science, Umeå University, SE-901 87 Umeå.*

Abstract

The development of new technologies within the area of biological research has lead to a massive increase in the amount of biological data. The need for computerized methods for managing and analyzing such data has led to new areas of research, such as bioinformatics, and a large number of new tools. Since several aspects of biology often need to be considered to be able to draw conclusions to biological questions, the integration of different tools, approaches and data are important aspects of bioinformatics. This report gives an example of the automation of information flow between two tools, BASE, a tool for managing expression data, and GBrowse, a tool for visualising genomic information. General aspects on how to store and represent biological data are also discussed.

Sammanfattning

Utvecklingen av nya teknologier inom biologiska forskningsområden har lett till en avsevärd ökning av mängden biologiska data som produceras. Nya forskningsområden, till exempel bioinformatik, liksom en mängd nya verktyg har utvecklats som svar på ett ökat behov av datoriserade metoder för att hantera och analysera sådana data. Då flera olika biologiska infallsvinklar ofta behöver beaktas innan slutsatser kan dras om biologiska frågeställningar, är integration av olika verktyg, tillvägagångssätt och data viktiga aspekter inom bioinformatik. Denna rapport ger ett exempel på automation av informationsflöde mellan två verktyg, BASE som hanterar microarray-analysdata och GBrowse som används för att visualisera genomisk information. Lagring och representation av biologiska data ur ett mer allmänt perspektiv diskuteras också i rapporten.

Contents

1 Introduction

Advances in biological methodologies, particularly during the last decade, has made it possible to generate massive amounts of data for use in biological research. New fields of research has developed, for instance bioinformatics and computational biology, in response to the increased need for and use of computerized methods for handling biological data. An effect of this is the development of a large number of tools and standards for handling and representing such data. These tools can be used to analyze many different aspects of biology, which often depend on and influence each other. To draw analytical conclusions it is often needed to consider several associated aspects of the data, often using different tools for each aspect. This report gives an example on how to automate the information flow between tools that are used for handling different aspects of genetic information, thereby creating a closer association between these aspects for the feature under investigation. It also gives some insight to how genetic information can be stored and retrieved in general.

1.1 The outline of the report

This report will first give a short theoretical background for areas within biology and biological methodology that are interesting in the context of this study. After that the requirements for the task in question and an introduction to relevant systems and techniques are given. This is followed by a description and discussion of the system design and the resulting solution. A user manual is present in the appendix, made primarily with the users at the Swedish Defence Research Agency in Umeå in mind.

Because of the fact that biological data and systems are prone to change often, the design choices and sometimes alternatives to design choices will be explained quite thoroughly. If changes in the system, or upgrades of the applications that the system works against, are to be done in the future the description of made choices and alternatives might make it easier to judge the suitability of different alternatives for making these changes.

2 Introductory microbiology

The passed several years have witnessed an exponential increase in the amount of biological data [20]. As stated in [21] the key issue during the late 1990s shifted from "how to sequence a genome" to "how to handle the sequenced data". The field of bioinformatics developed, to handle and analyse the vast amounts of information that was generated by large sequencing projects such as the sequencing of the human genome [21]. As the cost of genome sequencing has come down, an increasing number of organisms are either in the process of being sequenced or are likely to be sequenced in the near future [27]. During the 21st century, the emphasis has shifted back towards biological questions [21]. To try to answer questions such as how components of the genome and different gene products interact, a new set of tools is used in combination with the established sequencing methodologies. This way of studying the structure and function of the genome has lead to a branch of biology called genomics [21]. Other related fields are proteomics, the study of the full set of proteins encoded by a genome, and metabolomics which is the use of genome sequence analysis to determine the capability of a cell, tissue or organism to synthesize small molecules [21]. The event and development of the Internet should also be mentioned when talking about advances in the field of genetics, since it has eased the sharing of both tools and data for scientists around the world.

2.1 The transcription and translation process

The complexity in biological information processing in part comes from the dynamics of biological systems, with many interacting parts and many different timescales [24]. The basic information unit in the cell is DNA (deoxyribonucleic acid). Genes, the hereditary material of the chromosomes, are essentially long stretches of double-helical DNA [1]. With DNA as a template, RNA (ribonucleic acid) can be produced through a process called transcription. A certain type of RNA can in turn be used as a template for producing proteins, molecules with key roles in cell function, through the process of translation.

Before looking further into the processes of transcription and translation a distinction needs to be made between two different type of cells, prokaryotic cells and eukaryotic cells. Prokaryotes and eukaryotes are the two basic cell types based on the internal structure of cells [17]. There are only two evolutionary lineages of prokaryotes, Bacteria and Archaea, whereas algae, fungi, protozoa as well as all multicellular life forms (plants and animals) are constructed of eukaryotic cells [17]. The internal structure of these two cell types differ in a number of ways but one of the major differences is the arrangement of DNA within the cell. In eukaryotes the DNA is contained within the nucleus, a special membrane-enclosed structure, in the form of linear pieces of DNA present in individual chromosomes [17]. A typical bacterial genome[1] (prokaryotes) consists of a single closed circular molecule of DNA in the cytoplasm of the cell [17].

[1]A genome is the total complement of genes in a cell or virus [17].

2.1.1 Transcription

Transcription is the process of synthesizing RNA from a DNA template using the rules of complementarity [28]. The monomers[2] of nucleic acids are nucleotides which consists of a nitrogenous base, a sugar and a phosphate [28]. There are four types of bases in DNA, adenin (A), thymine (T), guanine (G) and cytosine (C). RNA differs from DNA in that it has the base uracil (U) instead of thymine, it has the sugar ribose instead of deoxyribose and it is, except in certain viruses, not double stranded [17]. The double-helical structure of DNA comes from the fact that hydrogen bonding occurs between the bases thymine and adenine and between cytosine and guanine. The bases that can form bonds are called complementary to each other and the two strands in a DNA helix is also called complementary strands [28]. Since these bases also exist in RNA, except for thymine, RNA bases can also form bonds with the bases of single stranded DNA, with the difference that uracil instead of thymine binds to adenine. Thereby the sequence of A, T, C and Gs in the DNA determines the sequence of A, U, C and Gs in the transcribed RNA. Three main types of RNA can be produced from DNA by transcription, they all serve different roles in the transcription process. Messenger RNA (mRNA) carries sequence information of DNA to particles known as ribosomes, where this information is translated into protein sequences [28]. Ribosomal RNA (rRNA) has a structural and functional role as a part of the ribosome and transfer RNA (tRNA) bring the amino acids that proteins are composed of to the ribosomes [28]. For each DNA strand there exists a polarity due to the fact that one end ends with a phosphate group, this is called the 5' end, and the other end, called the 3' end, ends with a hydroxyl group [28]. The transcription process is said to proceed in the 5' to 3' direction, since the RNA is formed from the 5' to the 3' end [1]. The process starts with the binding of an enzyme[3] to the DNA, which after binding moves along one of the strands in the 3' to 5' direction and catalyzes the formation of bonds between RNA ribonucleotides using the DNA strand as a template [17, 1]. The base sequence of the DNA contains signals for initiation and termination of RNA synthesis [1]. Most prokaryotic transcripts contain several genes whereas virtually all transcripts from higher eukaryotes contain just one gene [28]. Polycistronic mRNA is a single mRNA molecule for a group of genes that code for enzymes that are often functionally coupled [17]. Genes that code for polycistronic DNA or ribosomal RNA are the type of genes that an operon often consists of. An operon is a cluster of genes whose expression is controlled by a single operator, which is a specific region of the DNA adjacent to the coding region of the first gene in the operon [17]. The operator exhibits transcriptional control of the genes by acting as a binding site for a repressor protein which when bound interferes with the binding of the enzyme that is needed to start transcription [28]. Another difference between prokaryotes and eukaryotes is that eukaryotic DNA contain segments

[2] Monomers are the smaller molecules from whichi, if put together in sequence, larger biological molecules can be built [1].

[3] An enzyme is a protein that act as a catalyst for reactions in biological systems [1].

of DNA which are transcribed into RNA but never translated into proteins, these are called introns [28]. The segments between introns, which are transcribed and translated, and hence expressed, are called exons [28]. Introns are removed from RNA transcripts before they are transported out of the nucleus.

2.1.2 Translation and gene regulation

Translation is the process of synthesizing proteins from mRNA. In eukaryotes, mRNA is transported out of the nucleus where protein synthesis takes place at the ribosomes, whereas for prokaryotes, which lack a nucleus, translation of mRNA at ribosomes can begin even before transcription is completed [28]. The ribosomes attach to the mRNA strand and move along the strand as it elongates. There are also other major differences between the transcription and translation process in prokaryotes compared to eukaryotes [28, 1]. At the ribosomes, amino acids are added together in sequence to form proteins using the mRNA as a template. The sequence of amino acids is determined by the sequence of bases in the mRNA. Three bases at a time, so called codons, code for a certain amino acid or constitute a stop codon, which signals that the translation should end [1]. The functional properties of a protein is determined by its three dimensional structure, which is ultimately determined by the amino acid sequence [1]. Many proteins are enzymes that catalyze different reactions in the cell, thus regulation of biochemical reactions in response to for instance changing growth conditions can be done by regulating the expression (the amount of protein produced) of these proteins [17]. This can be done either by regulating the transcription or translation of the protein or regulating its activity after it has been translated [17]. One technology that can be used to analyse gene expression is the microarray technolgy.

2.2 The microarray technology

Microarrays is a technology that has been utilized in various applications within biological research and is today accessible to even relatively small laboratories [22]. This technique can be used for instance to analyse gene expression by quantifying large number of messenger RNA transcripts [23]. The technique includes having a large number of DNA sequences, also called reporters, immobilized on a solid surface in an ordered array. Messenger RNA from a given cell line or tissue can then be hybridized[4] in parallel to the DNA sequences in the array [23]. Arrays typically contain 5000 to 45000 DNA reporters, each of which has dozens of biological and quality control annotations[5]. The reporters are most commonly complementary DNA (cDNA), which is DNA gotten as a product

[4]Hybridization is when the RNA binds in to a corresponding single stranded DNA sequence that is complementary to that RNA [28].

[5]Annotation is the derivation of structural or functional information from unprocessed genomic DNA sequenced data [21].

from PCR[6] , or oligonucleotides (oligos), which are short sequences synthesized in situ. By using different fluorescent dyes (such as Cy3 and Cy5) mRNAs from two different cell populations or tissues, for instance where one of them has been subject to a particular chemical and one is as a control, can be labelled in different colours and then mixed and hybridized to the same array. This results in competitive binding of the target to the arrayed sequences [23]. After hybridization and washing, the slide can be scanned using two different wavelengths, which correspond to the dyes that were used. By comparing the intensity of the same spot of DNA in both channels the ratio of transcript levels or expression for each gene represented on the array can be measured [23]. Microarrays facilitate monitoring hundreds or even thousands of genes on a single chip giving scientists a better picture of simultaneous interactions among these genes [5]. Such holistic views are hard to obtain using conventional molecular biological technologies in which one gene is investigated in one experiment, providing an incomplete picture of how genes function and interact [5]. Since this technique generates a massive amount of data, computerized tools have been developed as an aid in structuring and analyzing the data from microarray analysis, one example of such a tool is BASE, described in section 4.2.1 on page 13.

2.3 The grouping of proteins into families

Comparative genomics is the study of the differences and similarities in genome structure and organization in different organisms [21]. The driving force behind this can be to gain a more detailed understanding of the process of evolution or, to translate DNA sequence data into proteins of known function [21]. DNA sequences encoding important cellular functions are more likely to be conserved between species than sequences encoding dispensable functions or non-coding sequences [21]. Homology between genes means that the given genes are related in sequence due to common descent from an ancestral gene that is present in a shared ancestor [8]. Usually, the hypothesis of homology is concluded on the basis of sequence similarity and the physical location of the gene in the genome for closely related taxa [26]. When talking about homologous genes one should distinguish between different types of homology. Orthologous genes are homologous genes in different species [10], they have in other words arisen from a speciation event [26]. Paralogous genes are homologous genes that have arisen through the process of gene duplication [26], and they can therefore be present in the same species. It should be mentioned that the usage and definition of the orthologue and paralogue concept has been subject to some debate [13, 6]. The grouping of genes into orthologue groups can be an aid in analysing gene function [15] and several projects exists that has public releases of their orthologue grouping, for instance the database of Clusters of Orthologous Groups of proteins (COGs) [30, 75] and the TIGR Orthologous Gene Alignment (TOGA) database [15, 88]. Since orthologues typically have the same function, the grouping of genes into orthologue groups can yield a number of functional predictions for poorly characterized genomes [29]. The

[6]PCR is a method for amplifying DNA.

mentioned databases constitute examples of public biological data storage, which is one of the subjects to be discussed in the next section.

3 Storage and retrieval of biological data

Massive amounts of biological data can be generated by use of the new technologies that have evolved mainly during the last decade, for instance automated sequencing, gene expression microarrays and mass spectrometry to characterize DNA, mRNA and proteins [20]. Computerized methods for analysing and handling this data are important to make this information useful. To develop good tools and applications an understanding of the nature of biological data is necessary.

3.1 Characteristics of biological data

Elmasri and Navathe [4] brings up nine characteristics of biological data, most of which contribute to the management of such data being a particularly challenging problem. In short these characteristics are the following;

- ⋆ Biological data is highly complex when compared with most other domains or applications. The definitions of such data therefore must be able to represent a complex substructure of data as well as relationships, and also ensure that no information is lost during biological data modelling.
- ⋆ The amount and range of variability in data is high, requiring systems that are flexible in handling data types and values. Placing constraints on data types might need to be limited since this might exclude unexpected values.
- ⋆ Schemas in biological databases change at a rapid pace. Therefore schema evolution and data object migration should be supported, the ability to extend the schema is not well supported though in most relational and object database systems.
- ⋆ Representations of the same data by different biologists will likely be different (even when using the same system), the results often reflect the particular focus of the scientist.
- ⋆ Most users of biological data do not require write access to the database, read-only access is adequate.
- ⋆ Most biologists are not likely to have any knowledge of the internal structure of the database or about schema design. There might be a tradeoff between usability and the need to reflect the underlying data structure, when designing biological database interfaces.
- ⋆ The context of data gives added meaning for its use in biological applications, which stresses that context must be maintained and conveyed to the user when appropriate.

⋆ Defining and representing complex queries is extremely important to the biologist, hence, biological systems must support complex queries, for instance through a tool for building such queries.

⋆ Users of biological information often require access to "old" values of the data, particularly when verifying previously reported results.

According to [4] these characteristics point to the fact that today's DBMSs do not fully cater to the requirements of complex biological data and that a new direction in database management systems is necessary. Some of the above given characteristics are discussed further, within the context of this study, in section 6.1 on page 46. Several projects have evolved with the aim to ease the sharing and handling of biological data. These projects can include for instance the gathering and publication of large amounts of data on the Internet, the production of reusable and adaptable software or repositories of information for a specific organism.

3.2 Databases and digital libraries

The need to integrate heterogenous information is a common problem not only within biology. Several projects aim to facilitate the integration of heterogeneous information sources accessible over the web, one example of such a project is the TSIMMIS (The Stanford-IBM Manager of Multiple Information Sources) project [54, 7]. The design of middleware for increasingly diverse systems, query optimization across diverse data structures and heterogenous databases and algorithms for executing complex[7] queries over large databases are other related areas that has been the focus of several projects and reports [12, 9, 3]. The survey in [9] includes query techniques and algorithms for executing complex queries over large databases for both relational and postrelational database systems. A biological example of a postrelational approach is the Human Protein Reference Database (HRPD) [20] which is an object database that integrates a lot of different information relevant to the function of human proteins in health and disease.

Within biology so called digital libraries have been an active area of research. Digital libraries collect, store, and provide data in digital forms, and also organize the information and provide for ways to search, retrieve and process data [4, 39]. Unlike the related data in a database, digital libraries include data from a multitude of sources, often unrelated [4]. Databases can though logically be components of digital libraries [4]. A driving force behind the development of digital libraries has been the Digital Library Initiative (DLI) [39] which has the focus of advancing the means to collect, store, and organize information in digital forms, and make it available for searching, retrieval, and processing via communication networks, in a user-friendly manner. Within DLI six large projects are ongoing at six different universities with research in network information systems primarily [39].

[7]Complex in the sense that executing the query requires a number of query-processing algorithms to work together [9].

There exists a multitude of biological database projects available on the Internet. The rapid expansion of nucleotide sequence data in public databases have a variety of uses, for instance the discovery of novel genes, identification of homologous genes and chromosomal localization of genes [19]. Three of the major organisations that handle such projects are EBI (the European Bioinformatics Institute) [64], NCBI (National Center for Biotechnology Information) [73] and TIGR (The Institute for Genomic Research) [87]. The EMBL Nucleotide Sequence Database at EBI constitutes Europe's primary nucleotide sequence resource and is produced in an international collaboration with Genbank (USA) at NCBI and the DNA Database of Japan (DDBJ) [61]. Each of the three groups collects a portion of the total sequence data reported worldwide, and all new and updated database entries are exchanged between the groups on a daily basis [64]. TIGR is a not-for-profit research institute with research interests in structural, functional and comparative analysis of genomes and gene products, from a wide variety of organisms [87]. TIGR has a collection of databases containing for instance DNA and protein sequences, gene expression, cellular role, protein family, and taxonomic data for microbes, plants and humans. The Swiss-Prot database is also a large database worth mentioning. It is a curated protein sequence database that strives to provide a high level of annotation, minimal redundancy and a high level of integration with other databases for protein information [65]. Numerous databases are necessary to support the various needs of the scientific community [80]. One particular type of databases is model organism system databases.

3.2.1 Model organism system databases

Model organism system databases (MODs) are an important help in genomics when facing the challenges of a flood of new data, rapid growth in data diversity, and the complexity of data produced by cross-disciplinary investigation [80]. Model organisms often have strong advantages for experimental research, for instance quick breeding [36]. Since many aspects of biology are similar in most or all organisms, model organisms, in which it is often much easier to study particular aspects, can be very useful for investigating different biological aspects [36]. By studying model biological systems in great depth, the gained knowledge can be used for instance to design appropriate studies of non-model systems, to answer important questions about their biology [56]. The key role of model organism databases is to connect genomic features to the classical biology of the organism [27]. This role distinguishes them from whole genome annotation projects, such as EnsEMBL and they also have an interpretative and curatorial role that distinguishes them from strictly archival databases, for instance Genbank [27]. Examples of important eukaryotic model organisms and corresponding model organism databases are *Drosophila melanogaster* (fruit fly) and Flybase [41], *Saccharomyces cerevisiae* (yeast) and the Saccharomyces Genome Database (SGD), *Mus musculus* and the Mouse Genome Database (MGD) and *Caenorhabditis elegans* and the WormBase database [70]. An important prokaryote model organism is *Escherichia coli* [89].

3.3 File formats

The existence of a multitude of online biological databases has lead to a number of more or less standardized formats for representing biological data. The existence of several competing standards, or some times the lack of standards complicates the issue of designing tools to handle biological data, and there is a widespread desire for interoperability and integration [2, 14]. Also, to be able to draw conclusions to biological questions, a vast array of diverse information often need to be considered [32]. The BioMOBY project [58] is an example of a project that adresses these issues by aiming to design and deploy platforms that enable and simplify biological database interoperability. Attempts to standardize file formats and ease the exchange of data of different formats are also gaining momentum [2]. One example of this is the Distributed Annotation System (DAS) project [57], which is an open source client-server system for exchanging annotations on genomic sequence data. Some of the important formats for storing sequence and annotation information are the Genbank, EMBL, FASTA and GFF file format.

3.3.1 The Genbank and EMBL flat file format

The Genbank flat file format is one of the most common formats for annotated sequence information. Genbank files can contain a large number of data elements or fields, the function of which is further described in [79]. The file begins with a file header where a large amount of information can be given, for instance an accession number which is an identifier for the file in question, version, organism name, references to articles etc. The header section is followed by a list of sequence entries of one of a large number of possible entry types, for instance gene, exon, CDS[8] or mRNA. Other information is also given for each sequence entry, for example start and end position in the reference sequence, strandedness and notes. For the full Genbank flat file release specification see [33].

The EMBL flat file format is used when submitting sequence and annotation data to the EMBL database. As with the Genbank format the EMBL format also includes a number of different fields and elements that are further described in [63]. The entries in the database are structured so as to be usable by human readers as well as by computer programs [63].

3.3.2 The FASTA format

A sequence in the FASTA format begins with a description or sequence identifier on a headerline that starts with the '>' sign. After that line one or more lines of sequence data can follow. Several sequences (DNA or protein sequences) can be given after each other, each beginning with a new header line. An example taken from [74] of a sequence in FASTA format is as follows.

[8]A CDS is a sequence coding for amino acids in protein

```
>gi|532319|pir|TVFV2E|TVFV2E envelope protein
ELRLRYCAPAGFALLKCNDADYDGFKTNCSNVSVVHCTNLMNTTVTTGLLLNGSYSENRT
QIWQKHRTSNDSALILLNKHYNLTVTCKRPGNKTVLPVTIMAGLVFHSQKYNLRLRQAWC
HFPSNWKGAWKEVKEEIVNLPKERYRGTNDPKRIFFQRQWGDPETANLWFNCHGEFFYCK
MDWFLNYLNNLTVDADHNECKNTSGTKSGNKRAPGPCVQRTYVACHIRSVIIWLETISKK
TYAPPREGHLECTSTVTGMTVELNYIPKNRTNVTLSPQIESIWAAELDRYKLVEITPIGF
APTEVRRYTGGHERQKRVPFVXXXXXXXXXXXXXXXXXXXXXXVQSQHLLAGILQQQKNL
LAAVEAQQQMLKLTIWGVK
```

3.3.3 The GFF format

GFF is another format for describing genes and other features associated with DNA, RNA and protein sequences. [85]. A GFF record is an extension of a basic (name, start, end) tuple that can be used to identify a substring of a biological sequence, for example (ChromosomeI, 2000, 3000) specifies the third kilobase of the sequence named "ChromosomeI" [85]. The primary format uses a record-based structure to be easily parsed and processed by a variety of programs and tools. Each feature is described on a single line with a number of tab-delimited fields. These fields include the name of the sequence, the source of the feature (normally indicating the program making the prediction), the feature type, start and end position for the feature (sequence numbering starts at 1), a floating point score value, strand (indicated by a '+', '-', or if strandedness is not relevant, a '.'), frame and an optional attribute field where several attributes can be given separated by ';' [86]. Some example record are

```
SEQ1 EMBL atg 103 105 . + 0
SEQ2 Genbank CDS 100 540 . -  0
SEQ3 Genbank gene 500 1503 . -  0 Note "strain L4" ; Note "Complete genome".
```

4 Automation of information flow

The task in this study was to find a way to place experimental data sets, in this case microarray results, in their genomic context. By also automating the information flow between the systems that handle these different aspects of biological data, it becomes easier to analyse the data and to draw conclusions to biological and functional questions. One way of placing microarray results, in this case from BASE, in their genomic context is to dynamically visualize the region of the genome where different reporters from the microarray experiment are positioned. The genes positionally close to the gene that the chosen reporter represents are interesting for further analysis, particularly for prokaryotes where small groups of neighbouring genes can be part of the same operon. If the visualisation is done in an interactive manner, with possibilities to further investigate questions about the neighbouring and other genes, this might speed up the pace with which analytical questions can be answered. More specific requirements for the task in question are given below.

4.1 Requirements

The requirements were the following:

- ⋆ Find a way to display the entire genome of different species, that is browsable and interactive.
- ⋆ The viewer used for displaying the genomes should be database-based or be possible to use on top of a database, to make information about the features and other information that the database contains searchable.
- ⋆ One should be able to store/load seqence-annotation files of the Genbank file format, both locally generated Genbank files and publically released Genbank files fetched from the NCBI website [73], which is one of the standard formats for storing such information, into the database.
- ⋆ The information flow between the chosen viewer and a system called BASE should be automated so that analytical pathways can be investigated with as little interference as possible.
- ⋆ From the system called BASE one should be able to click on a link for a reporter and the genomic feature that this reporter represents is then to be automatically viewed using the chosen viewer within the correct context of the genome it belongs to.
- ⋆ One should be able to right click a feature in this view and have a menu come up with choices for 1, displaying more information about the selected feature and

its sequence, 2, displaying expression data from the system called BASE for that feature and 3, displaying so called orthologues for that feature.

⋆ If the orthologue choice is selected these orthologues are to be displayed in the context of each genome they belong to, also in an interactive manner, with the selected viewer. For each of the orthologue views one should be able to continue to interact with the features.

⋆ If orthologues are found they are to be displayed and aligned below each other so that one can easily compare the genomic environments which the features reside in.

⋆ If the expression data choice is selected, the available expression data for reporters that represent the feature of choice is to be fetched from BASE and displayed. Previous experiments where that gene has been involved (or rather reporters representing that gene) in BASE should be displayed and it should be indicated if the feature was up- or down-regulated according to the found data.

4.2 Relevant systems and techniques

This section will introduce some of the techniques and systems that were used to fulfill the above requirements. The microarray analysis tool that was used is a system called BASE. For displaying genomes, two genome viewers in particular were investigated, GBrowse and the light weight genome viewer. Out of these two GBrowse was chosen for the final solution and was used on top of a MySQL database schema called Bio::DB::GFF. To connect these two systems and to handle and automate the flow of information between the systems PHP, HTML and to a smaller extent also Perl and JavaScript was used. The PHPDoc style for commenting [82], which is a documentation convention similar to Javadoc, was chosen for the PHP files, and CSS [91, 90] was used to control the style of interface pages.

4.2.1 BASE

BioArray Software Environment (BASE) is a freely available tool for management and analysis of areas of microarray experimentation [22, 34]. BASE handles biomaterials, raw data and images, and provides integrated and plug-in tools for normalization, data viewing and analysis [53]. It also has array production features which can be integrated with the data analysis [35]. The primary database platform used in BASE is MySQL, although it has been modified to support a few other database systems as well [35]. To use BASE two databases need to be created, typically called 'base' and 'basedynamic'. An ER diagram for the database schema can be found in the documentation at [34]. The base database contains for instance the Reporter table which contains information for each reporter in the database. The basedynamic database is initially empty and is gradually filled with data as different results are achieved, for instance after running plug-ins on the raw microarray data. Per-site versions of the Reporter and RawBioAssayData (result file data) tables are supported in BASE 1.2 [34]. This means that if another column is needed in one of those tables, a column can be added. By indicating this in BASE the added column will be used in file formats, be displayed in various places, and so on [34]. These things are defined in two files in include/local under the BASE source directory, to add for instance extra reporter columns the reporter_columns.inc.php file is modified. The currently released 1.2.12 version of BASE is based mainly on PHP, but a 2.0 version is planned which is to be Java based.

4.2.2 GBrowse

GBrowse (the Generic Genome Browser) [27] is a web-based application for displaying genomic annotations and other features. It is part of the GMOD-project (the Generic Model Organism System Database Project) [66] which seeks to develop reusable software components for model organism system databases (MODs), see more on MODs in section 3.2.1 on page 9. The GMOD-project started as a cooperation between the four MODs mentioned on page 9, which had reflected on the fact that a major component of the cost of creating a new MOD is the development of database schemata, middleware and visualisation software [27]. The intent was to make reusable components available to the community free of charge under an open source license [27]. GBrowse was the second component to be released within the project. It has a web-based display that can be used to display an arbitrary set of features on a nucleotide or protein sequence and it can accommodate genome-scale sequences, megabases in length [27].

The GBrowse system GBrowse is used together with several software modules. An overview of the GBrowse system is given in figure 1, note that the Bioperl library, the Apache Web Server, Oracle, MySQL and Flat Files parts are not part of the GBrowse project. The top level gbrowse.cgi script is responsible for managing the user interface and form requests from the user [27]. The Bio::Graphics module takes care of rendering genome images [27]. Beneath this module lies the Bioperl library which contains for instance the Bio::DB::GFF module. This module provides fast indexed access to a sequence annotation database and supports multiple database types and schemas through a system of adaptors and aggregators [52, 40]. The adaptor handles the details of fetching information from databases, and is specified during Bio::DB::GFF construction [52]. Aggregators help to rebuild aggregation properties of individual features and are used by GBrowse for the representation of complex features [43, 52]. These properties can be indicated in the "attribute" field of the GFF flat file format, which the data model that Bio::DB::GFF uses is compatible with [52, 85], see section 3.3.3 on page 11 for a more thorough description of the GFF format. The Bioperl distribution also includes several scripts that make it easier to work with Bio::DB::GFF databases, for instance bp_load_gff.pl, bp_bulk_load_gff.pl and bp_gb2gff.pl, the use of which will be explained later in this section [52].

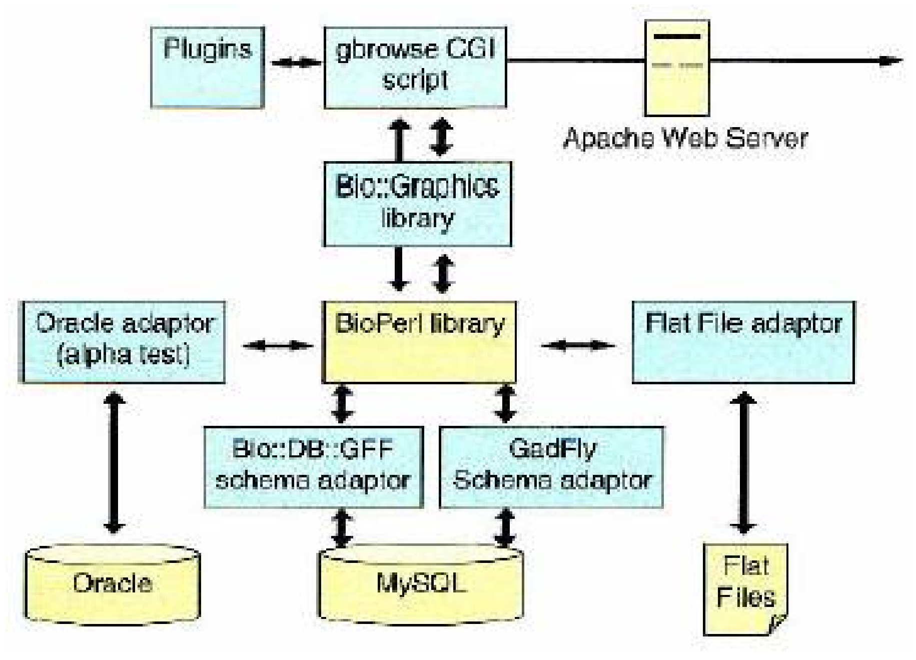


Figure 1: Basic layout of the GBrowse system together with components from other systems that GBrowse needs [27].

GBrowse can be used on top of for instance a MySQL, PostgreSQL or Oracle database and also on top of an in memory database (stored in flat files) [27]. The support for and maturity of the adaptors for the schemas in use for these DBMSs can differ though, also between different schemas for the same DBMS. GBrowse initially supported two distinct schemata within MySQL, the Bio::DB::GFF schema which is designed for simplicity and speed and GadFly which is a somewhat richer schema and thereby also slower in performance [27]. Other interesting schemas that has been mentioned for instance on the GBrowse mailing list during the scope of this study are the Chado schema (GMOD Modular Schema) [68], which really is a set of schema modules for building a model organism relational database that is under development, and the BioSQL schema [47, 37].

Bio::DB::GFF The Bio::DB::GFF schema uses a minimal schema, see figure 2, to represent features on sequences, and is optimized for queries that retrieve features by ID, by their type or by the region of the genome they overlap [27]. The last type of query is an example of a positional range query, which can be quite inefficient if conventional relational database indexes are used, particularly when searching for a small feature in the middle of for instance a large chromosome [27]. If one queries a relational database for all features present between a certain start and stop position on a particular reference sequence, for instance a certain chromosome, an SQL optimizer can eliminate rows whose

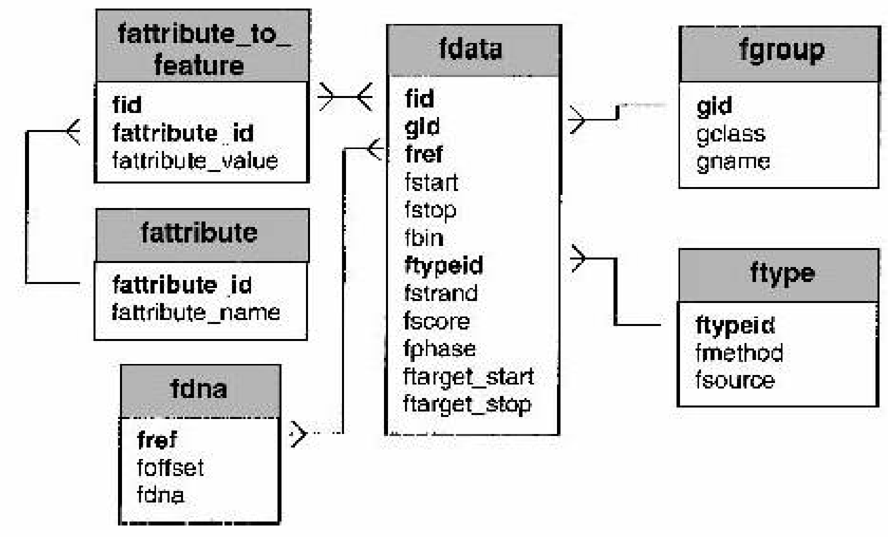


Figure 2: Overview of the tables in the Bio::DB::GFF schema. Fdata contains for instance the start and stop position, within its reference sequence, for each feature, fgroup tracks certain grouping of subfeatures into features and fdna stores the raw DNA sequence. Attributes are used for storing textual information such as for instance notes and ftype contain information of the method and source of the feature [27].

start and stop positions does not satisfy the criteria for start and end [27]. However, the remainder of the feature table, which can be quite large, must then be inspected by a linear search [27]. A composite B-tree index on the triple of (segment, start, end), where feature locations are stored in one table and start and end represent the respective coordinates of a location, can provide sufficient support for queries against smaller genomes. It however suffers from a huge variance of the response time due to its inherent dependency on how well the first attributes in the index slice the index tree that needs to be scanned for the other attributes (and therefore needs to be scanned from disk) [71]. Occasionally, the performance for this solution is exceptionally poor, for instance for queries into the middle of a segment, and genome browsers often also need to respond to many queries simultaneously for example if they are components of genome portals [71].

The solution that Bio::DB::GFF uses for this problem is a special case of the R-tree algorithm. An R-tree is a dynamic index structure for efficient retrieval of data items according to their spatial locations [11, 4]. They are the "de facto" reference structures for indexing spatial data, that is minimum bounding rectangles of spatial objects. The tuples of a spatial database represent spatial objects and has a unique identifier which can be used to retrieve it [11]. Since spatial data objects often cover areas in multi-dimensional spaces and are not well represented by point locations, classical one dimensional database indexing structures are not appropriate to multi-dimensional spatial searching [11]. R-tree leaf nodes contain index record entries with an n-dimensional bounding box of the spatial object and a tuple-identifier that refers to a tuple in the database. Non-leaf nodes contain a bounding box that covers all bounding boxes in the entries of the lower nodes together with a child-pointer which is the address of a lower node in the tree [11]. When searching an R-tree, the algorithm descends the tree from the root in a manner similar to a B-tree [11], however since the keys in R-trees do not have a linear ordering and can overlap [69] more than one subtree under a node visited can match the query predicate and may therefore need to be searched [11]. Therefore, the placement of keys in the search tree is critical for efficient performance using an R-tree [69]. An R-tree always shows the current state of the data set and does thereby not allow the user to query the spatial database with respect to past states [16]. However, there has been development towards extending the R-tree structure to preserve old states, allowing the user to query them [16].

To optimize the Bio::DB::GFF schema in GBrowse the genome is divided into a hierarchical set of bins, with the size of 10 Mb at the top level followed by a tier with bins the tenth of that size and so on down to 1000bp [27]. Each bin is assigned a floating point identifier and as features are entered into the database, they are assigned to the smallest bin that will entirely contain them [11]. The Bio::DB::GFF module automatically takes advantage of this binning scheme to optimize the queries [27]. The solution uses ideas from the domain of text processing [18] and includes translating the interval overlap query into a two-dimensional point-in-box geometric query supported by an R-tree index [71]. Unfortunately there does not yet exist any publication on the exact details of the special case R-tree solution in Bio::DB::GFF, but a short description and example is given in

[27] and a poster abstract is available online [71]. After GBrowse has been installed and the feature database created [67], feature data is to be loaded and GBrowse needs to be configured for use with that data. The data to be loaded needs to be available in the GFF flat file format, see section 3.3.3 on page 11, before trying to upload it. Since several flat file formats are available for storing sequence and annotation data [9] the existence of several Bioperl scripts for converting from different formats into the GFF format might be of good use [67, 52]. The bp_bulk_load_gff.pl Bioperl script initializes a new Bio::DB::GFF database with a fresh schema, deleting anything that was there before, and then loads the GFF file [67]. After this first load the database can be incrementally loaded by use of the bp_load_gff.pl script, see also appendix A.1.3 for examples on how to use the different scripts.

Configuration Each datasource that GBrowse is to be used with needs to have a corresponding configuration file in the gbrowse.conf directory, with a name of the form sourcename.conf [43]. The sourcename is a short descriptive word which can also be used to select the datasource when linking to the browser in the following way [43]

```
http://your.site.org/cgi-bin/gbrowse/sourcename.
```

Several parameters can also be sent by the URL, for instance to link directly into a specific area in a reference sequence. The configuration file contains a number of sections, the GENERAL section contains settings for the entire application while the other sections define tracks that are to be displayed. A large number of options are recognized in these sections and only a few of them, the ones interesting for this study, will be mentioned here, for more details on the different options see [43]. In the GENERAL section the first options are for the datasource. The db_adaptor option tells GBrowse what type of database adaptors to use, currently the only stable type of adaptors are in Bio::DB::GFF, which is a Bioperl module that contains a standard set of adaptors in Bioperl. The db_args are arguments to pass to the chosen adaptor type. For instance, to use Bio::DB::GFF on top of a MySQL database use

```
db_adaptor    = Bio::DB::GFF
db_args       = -adaptor dbi::mysql
                -dsn dbi:mysql:database=<dbname>;host=<dbhost>
```

and if the database requires login with user name and password the db_args -user <username> and -pass <password> can be added, alternatively the two configuration options user and pass can be used in the same purpose. To use the Oracle or in-memory version of Bio::DB::GFF the adaptor arguments need to be changed [43]. There are several options that allow the insertion of HTML in the GBrowse page at strategic places. The

[9]Each model organism database has its own flat file format for representing the data [67], and several larger online databases, for instance Genbank [78] and EMBL [62] also use different formats.

value for the head option is content to be inserted into the HTML <head></head> section, for instance JavaScript code. The header and footer options can contain any valid HTML for the header to print at the top of the browser page and the footer to print at the bottom of the browser page respectively. We also have the options html1, html2, html3, html4, html5, html6 which allow insertion of HTML in different places in the GBrowse page, see [43] for details. Through the examples option a list of examples of interesting regions can be specified in the GBrowse configuration file for the datasource. The value of the examples option is set to a space delimited list of interesting regions. The ones given in the current solution are only there as an example of the syntax and should be changed.

For the TRACK sections we have several interesting options. The glyph option controls which glyph (graphical icon) that is used to represent features in a track. Some of the glyphs are inherently stranded and for the others the strand_arrow option can be set to true if one wants the glyph to indicate the strandedness of the feature. Usually this is done by drawing some sort of arrow pointing in the direction of transcription for the feature, which in turn depends on which of the DNA strands that the feature resides in. A large variety of glyphs are available and more are being added as the Bio::Graphics module grows [43]. The feature option allows one to choose what kind of features to be displayed in a certain track, if wished for several feature types can be displayed on a single track. Each feature has a method and a source, and the feature types are given in the form method:source. The type of a feature with the method ”gene” and the source ”Genbank” would have the type ”gene:Genbank”. The method part can not be omitted but if one omits the source part, for the feature above this would mean having the type ”gene”, all features with method gene will be displayed in the track regardless of the source field [43]. The link, link_target and title options for tracks are identical to similarly named options in the GENERAL section, but can be used to override the global rules on a track-by-track basis. The link option creates a default rule for creating outgoing links from the GBrowse display when the user clicks on a feature of interest. The value for that option should be a URL containing one or more of a number of variables that are recognized [43]. It is possible to compute the URL dynamically using a subroutine, which will be called with a single argument consisting of a Bio::DB::GFF::Feature object. The subroutine should examine this object to determine what URL to link to and return the URL as a string, or undef if no link should be made [43]. To make links open in new windows when features are clicked in the detailed view,

```
link_target = blank
```

can be added to the GENERAL section. The title option controls the tooltip text that is displayed when hoovering with the mouse over a glyph. If the value is set to AUTO or if the option is missing, a default description will appear.

Semantic zooming[10] is another property that can be implemented in the configuration file. This is done by adding length qualified stanzas to a track description where the different stanzas contain option-value pairs that apply for the specific zoom level of the stanza. Also worth mentioning is that a branch of the BioMOBY project mentioned in section 3.2 on page 8 has an API that is now being used by data providers worldwide to publish data in an interoperable manner [43]. With a small amount of changes in the configuration file a MOBY browser written for GBrowse can be used, which allows the user to explore other data related to the genomic features displayed in GBrowse, for more details see [43].

In subsections to section 5 on page 26 the use of some of the above options within this study is described. GBrowse has a large number of functionalities and configuration options which have not been described here, for more information about the functionalities and configuration options that GBrowse offer see [27, 42, 43].

4.2.3 Lwgv

Lwgv (light weight genome viewer) is an application used to ease the display of genomic information [46, 44]. It uses an executable, genomeview.cgi, to visually present the content of annotation files written in a particular format. If genomeview.cgi is run without any parameters, all files ending with .ann in the current directory will be displayed. You can specify a particular annotation file with the parameter ann, for instance

```
http://www.myHomePage/genomeview.cgi?ann=myAnnFile.ann
```

displays the annotation file myAnnFile.ann. To change the directory in which the script looks for annotation files, the anndir parameter can be used, for instance

```
http://www.myHomePage/genomeview.cgi?ann=myAnnFile.ann&anndir=/tmpAnn
```

will show the file myAnnFile in the directory /tmpAnn.

The language one uses in these files can contain configurations, genome information and commands and it also allows for the inclusion of additional annotation information from external files and arbitrary text to be printed to the screen by use of the statements #include and #insertFile respectively. All configuration options have the form

```
#config  PARAMETER_NAME  PARAMETER_VALUE.
```

If you for example would want to perform smoothing on the graph, set the number of nucleotides to print per line to 70 or change the graph type to line instead of histogram you can do this by

```
#config GRAPH_SMOOTH true
#config SEQ_LINE_LENGTH 70
#config GRAPH_TYPE line
```

[10]Semantic zooming is when the appearance of a track is changed in reflection of the user having zoomed out or in beyond a certain level in the view [43].

respectively. Approximately 28 configurations are available having boolean, integer or string values for all of which a default value is set. All commands have the form

```
type variable command(value)
```

where, if type and variable are not given, the last appropriate type and variable are used. If you for instance would like to remove the graphs myGraph1 and myGraph2 from the genome myGenome, and set the color of track myTrack1 to white you could achieve that by

```
genome myGenome removeGraph(myGraph1, myGraph2)
track myTrack1 setTrackColor(255, 255, 255).
```

A number of other commands are of course also available. Many of the commands allow a comma separated list of parameters to be passed to them. The type and variable indicate what type of genomic feature data that the command is to be used on. Currently three different types of genomic feature data is accepted by lwgv, tracks, graphs and sequences. The order of which features are added is not important. Note that the software currently supports only one genome per file. Genomes are started with

```
begin genome myGenomeName
```

and are finished with the

```
end genome
```

command. Features can be added to a track by the addPairs function with the start and end locations given as colon separated integers. For instance

```
track myTrack1 addPairs(1212:3455, 0:1110)
```

adds two features to the track myTrack1. Two types of links can be added to a feature on a track. A normal link that uses the key word link and displays a small tooltip when the mouse is held over it on the image can look like this

```
track myTrack2 addPairs(0:1110:link(''linkName'',''url''))
```

, and an example of a super link that uses the key word slink and produces a JavaScript popup, which can display an arbitrary amount of information when the mouse is held over the link, is

```
track myTrack2 addPairs(1212:3455:slink(''Link popup title'' =>
''linkName1'',''url1'',''linkName2'',''url2'',...)).
```

Points are added to a graph by

```
graph myGraph1 addPoints(x:y)
```

, where x and y are real numbers specifying a point in the graph. Finally, sequences can be added by

```
sequence mySeq addSeq(ATGTTCGAAAGC...).
```

A minimal example of an annotation file taken from [46] is the following

```
% initialize a genome
begin genome Hello_World_Genome
% add a feature to the track first_track from 150bp to 200bp
track first_track addPairs(150:200)
end genome
%create the webpage and image for the genome
showGenome(Hello_World_Genome)
```

, which produces the output of figure 3. For a nicer looking, more advanced example of the use of lwgv see [45]. Lwgv is according to the lwgv documentation [46] primarily useful if

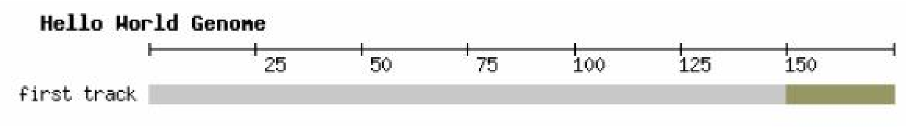


Figure 3: This very simple example of the use of lwgv, taken from [46], shows only one track with one feature in this track.

you have a reasonably small amount of data to visualise, if you want to get started fast without a lot of system and library requirements before installation (which for instance GBrowse has), if you do not need a large relational database system, if you want to create annotation files ”on the fly” and if your annotations change frequently. It is in other words not meant to be used with a database or with large amounts of data, but rather when working actively in determining annotations, when the data changes a lot and when the amount of data is quite small so that it can easily and manually be used to create annotation files for lwgv to display.

4.2.4 Bioperl

The Bioperl project is an open source collaboration of biologists, bioinformaticians and computer scientists that has lead to the development of a library of Perl modules for managing and manipulating biological information [25]. These Perl modules facilitate the development of Perl scripts for bioinformatics applications [59]. Bioperl can for instance be used to execute analyses and processing results from a number of biological tools, and to provide access to data stores such as Genbank, SwissProt and the emerging common sequence data storage format of the Open Bioinformatics Database Access Project [25]. An interesting discussion on the advantages of having an open source approach for the Bioperl project is given in [25].

4.2.5 MySQL

MySQL is the DBMS that was used for the task in question [72]. MySQL has a number of different storage engines, the default storage engine as of MySQL 3.23 is MyISAM which is fast but non-transaction safe. Other MySQL storage engines include the Merge, ISAM, HEAP, InnoDB and BDB storage engines, only the InnoDB storage engine is transaction safe (ACID compliant[11]) [31]. Transaction safe tables have the advantage of being safer, in case of hardware or software problems lost data can be recovered by automatic recovery or from a backup and the transition log. ROLLBACK and COMMIT operations are also supported for InnoDB tables as well as foreign key constraints (which only InnoDB support) [31]. Due to no transaction overhead, non-transaction-safe tables are faster and use less disk space, but they of course have the obvious disadvantage of being non-transaction-safe [31]. A MERGE table is a collection of identical MyISAM tables that can be used as one. If selects, deletes and updates are done they apply to all tables in the collection. ISAM tables are deprecated since MyISAM tables are a better implementation of those. HEAP tables use a hashed index and are stored in memory, which makes them very fast, but if MySQL crashes, all data stored in them will be lost [31]. They might be useful for instance for temporary tables. BDB (BerkeleyDB) tables has support for COMMIT and ROLLBACK, but are not fully ACID-compliant. The BASE and GBrowse versions used in this study both use the MyISAM storage engine for the database tables, which can be indicated by adding ENGINE = MyISAM or TYPE = MyISAM at the end of the CREATE TABLE block for the table. It is not necessary to use these options as long as the default storage engine is MyISAM, but it might be considered a good coding practice in case the default storage engine type changes in the future.

Some of the MySQL types are particularly interesting in the context of this study. The BLOB and TEXT types can both be used to store sequence information, in BASE the type text is used for a sequence field, and in GBrowse the type longblob is used to store entire reference sequences. A BLOB is a binary large object that can hold a variable amount of data. The only difference in use between the blob and the text types is that sorting and comparison are performed in case-sensitive fashion for blob values and case-insensitive fashion for text values [31]. Blob and text values are represented internally by a separately allocated object in contrast to all other column types, for which storage is allocated once per column when the table is opened [31]. The text and blob types can be prefixed with TINY, MEDIUM or LONG, varying in the maximum length of the value they can store, for instance longblob can store a 2^{32} bases long sequence [31]. Another type worth mentioning is the ENUM type, which in GBrowse is used to indicate to which DNA strand a feature belong. An ENUM type is a string object with a value chosen from a list of allowed values, which are enumerated explicitly in the column specification at table creation time [31]. For instance ENUM('+', '-', '.') assigns the index values 0, 1 and

[11]An ACID compliant DBMS should enforce the properties of Atomicity, Consistency preserving, Isolation and Durability or permanency to transactions [4].

2 respectively to '+', '-' and '.' and the index values can later on be used in for instance select queries on that field. An integer column (TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT) can have the additional attribute AUTO_INCREMENT, which can be used to generate a unique identity for new rows [31]. When tuples are inserted, an index in a column with the auto_increment attribute set, will be increased automatically by one for each new tuple. The used auto_increment key can be retrieved by the SQL function LAST_INSERT_ID() or the mysql_insert_id() API function [31].

Indexes are structures used to find rows with specific column values fast, most MySQL indexes are stored in B-trees [31]. All MySQL column types can be indexed, also the blob and text types (but only on a prefix in that case, not on the entire column), and according to [31] use of indexes on the relevant columns is the best way to improve the performance of SELECT operations. Indexes can be created by use of the CREATE INDEX syntax, also indexes on multiple columns. If the indexes are used or not, when querying the database, depends on how the query is constructed [31]. One can check if a created index gets used for a specific query by preceding the SELECT statement with EXPLAIN.

Saving storage space is essential when dealing with large amounts of data, and therefore one should choose types with smaller storage requirements, if there exists such types and the use of them does not comprise a poorer data storage design. If for instance SMALLINT can be used instead of INT, some storage space can be saved. However, if the amount of features in the database is predicted to increase dramatically in the near future the use of INT or BIGINT instead of SMALLINT for an index field might of course be better.

4.2.6 Overlib

Overlib is a JavaScript library created by Erik Bosrup to ease the use of small popup information boxes on web pages [60]. It is well documented and has a command reference with examples for all the commands that are available in the distribution.

4.2.7 PEAR DB

For the sake of database portability the PEAR DB package was used when connecting to and querying different databases in the system. This was done in particular to make it easier for a possible shift to Oracle instead of MySQL in the future, since that was discussed as a long term intent at the site of development. PEAR (PHP Extension and Application Repository) is a framework and distribution system for reusable PHP components [48]. DB is a database abstraction layer which contain for instance portability features for making programs written for one DBMS work with other DBMSs also [49]. It layers itself on top of PHP's existing database extension and currently has support for the extensions dbase, fbsql, interbase, informix, msql, mssql, mysql, mysqli, oci8, odbc, pgsql, sqlite and sybase [49].

4.2.8 HTML

HTML was used to display information that was retrieved from databases in the system. One particular issue that needed adressing was how to be able to display different orthologues in their genomic environments, under each other and keeping the different views interactive independently from each other. To accomplish this inline frames were used. The iframe element allows for the insertion of HTML documents within other HTML documents [92]. In this case that property could be used to run a genome viewer cgi script in each iframe, with different URL parameters sent to the script for each view. This allowed for different genomic surroundings to be displayed in the views, more information on how iframes were used within the current solution is given in section 5.4 on page 39.

5 System design

In order to achieve a functioning system a number of areas had to be looked into, and the suitability of using each investigated alternative had to be judged. The first area to be investigated was how to be able to display genomic information.

5.1 Choice of viewer

At first a number of possible genome viewers were discussed but some of them were quickly discarded, for instance the commercial ones, since there existed open source alternatives which, besides from not costing any money, are easier to modify for your particular needs and also lets the people that use the technology and know what would be useful, to extend its functionality by developing their own code which can be shared with others. Two open source genome viewers were chosen for further investigation, the Generic Genome Browser (GBrowse) [27] and the Light Weight Genome Viewer (lwgv) [46].

Lwgv and GBrowse The first intent was to use lwgv to display the genomes. Even though lwgv, as previously mentioned in section 20, is not really intended to be used with a database, it would be possible to do so if one had intermediary steps that extracted a suitable amount of data from the database and presented that data in the appropriate annotation file format for lwgv to display. In this case a PHP or Perl script could have been used to fetch information from the database and to create an annotation file from that information of the format that lwgv needs to be able to display the genomic information. However, the more research being done for GBrowse, the benefits of using that viewer instead became more and more evident. GBrowse did already have an architecture that was based on a database and had a much more vivid developer community, accessible through the GBrowse mailing list, than lwgv did. Even though lwgv might not need an active developer community, for the sake of developing the technique further (since lwgv is quite simplistic), it is important for the sake of future support for the users of the system that they can easily get in touch with the developers and other users. Other advantages with GBrowse is that it seems to have many users, it makes use of the Bioperl library (which in turn is also very spread throughout the biotech community) and is part of a larger project, the GMOD project [66], with a clearly stated goal[12]. This is important because, since GBrowse has these qualities, there is a much bigger chance that, if you need to do something that might be useful for others that work with similar things, someone has already done something similar, or you can get help and support in doing it. This later showed to be the case also within this study, since a much needed script for turning Genbank files into the GFF format, which is the format needed by GBrowse to be able to load files into the database, already had been done and was available for everyone to use.

[12]The goal of GMOD (The Generic Model Organism System Database Project) is to develop reusable software components for model organism system databases [66].

Another gain with using formats and schemas that many others use, and that is publicly available and described is that useful tools and applications might be developed in the future that are compatible with this schema and released code is also being subject to testing, bug checking and often further development by the users.

Adaptability in GBrowse In my opinion one of the biggest advantages with GBrowse is that the configuration file that is needed for each datasource, offer a lot of options which can be used to adapt the use of GBrowse to ones own purposes. Normally, it might be considered a bit strenous to have that many configuration options to consider for an application, but there are example files to look at included in the installation, and once you have made a configuration file for one datasource you can use that as a template for other configuration files. The configuration file also have several places where custom code, for instance html, can be added and the value of some options can be computed at run time by the use of Perl subroutines using the following format (where continuation lines must begin with white space)

```
option name = sub {
            some perl code;
            more perl code;
            more perl code;
            }
```

, some examples of such options are given in section 4.2.2 on page 18. Since many adaptations to the system can be made in the configuration file the risk of having to edit other files, if GBrowse is to be adapted to use with other components in a system, decreases. This separation of adaptable parts of the system from the rest of the system makes it easier to be able to upgrade GBrowse when new versions are released while at the same time easily save and copy local adaptations to the newly installed version.

The simplicity of lwgv Possible pros for using lwgv include that it produces a nice and simple interface and simple installation and usage. All of the functionality that GBrowse offer and all of the things inherent in the interface GBrowse produces is really not necessary for fulfilling the requirements of the task in question. Lwgv also have inherent constructs for making JavaScript popups, with title and links in them, which could be used to solve the requirement to have a menu displayed when right clicking a feature in the viewer (see section 4.2.3 on page 20). In lwgv though the popup is displayed when hovering with the mouse over the feature and not when right clicking a feature. Since GBrowse is much more complex it also takes a lot of time just to understand how it works and which parts that are essential for the task in question. This extra time is easily compensated for though, if one can make use of scripts and code that has already been developed by others for GBrowse, as a help in completing the task. The extra functionality that GBrowse offer might also prove to be useful for resolving other issues in the future.

Further comparisons If lwgv had been used a database would have had to be designed and created, from which one could get the information for the annotation files that lwgv can interpret. One of the pros of designing a new database from scratch is that the database could then with ease have been completely designed and optimized for the specific purpose in question, at least up to the point when the applications and interfaces that are to work against this schema has been done. After this, modifying the schema to reflect these changes in the rest of the system, might demand more work. GBrowse on the other hand has certain adaptors that allow it to work with a small number of schemas, and of those schemas mainly one has a relatively stable and mature adaptor, that is the Bio::DB::GFF schema. Therefore, if GBrowse is used one have to sort out if all the information one needs to make the entire system work does already exist in that schema. If not, the possibility to modify it in such a way that everything which needs to be stored, can be stored in a consistent way that does not interfere with the functionality of the viewer, must be investigated. Finally, one should also consider if it would be a better option to separate the content that is needed for the viewer to work, from the content one needs to complete the system, into different databases. Before this has been thoroughly investigated and a complete solution found one can not irreversibly decide upon using GBrowse and discard lwgv completely. An alternative to use lwgv with a database designed from scratch would have been to use the same database schema that GBrowse uses but to use lwgv on top of it instead of GBrowse. This has the advantage that if one at a later time would like to change into using GBrowse as the viewer instead, this might be done relatively painless. However, you still then have to investigate whether all information that is needed can be stored in this schema and you still miss out on a number of pros with using GBrowse. A problem with using lwgv together with files dynamically created from a database might also be that one then needs to handle the dynamic creation, naming and removal of all the annotation files needed. It might for instance sometimes be wise to save the files that are created for some time, if the same files are to be displayed quite often during a period of time, depending on if it takes a disturbingly long time to generate them from the database or not.

GBrowse was finally chosen to be used as viewer, on top of its standard database schema for MySQL, that is the Bio::DB::GFF schema. GBrowse can be run on top of different kinds of databases, see section 4.2.2, but in the scope of this task MySQL was used as DBMS. This was both due to the fact that MySQL was already installed at the site and used also for the BASE system, and to the fact that GBrowse has good support and works relatively stable with MySQL since MySQL was the first DBMS that GBrowse supported [27]. The Bio::DB::GFF schema was chosen because the support for and maturity of adaptors for other possible schemas, see section 4.2.2 on page 15, was somewhat experimental or currently under a lot of development at the time of the completion of the task or did not suite well for this task. The chosen schema was then modified for the task in question, according to the database modification description in section 5.4 on page 39,

and a number of test files from the public Genbank database [78] was loaded into the database. For further information on how to load the database with data, see the user manual in appendix A.1.3 on page 62.
Now when the choice of viewer has been accounted for, let's turn the attention to the rest of the system, starting with BASE.

5.2 Linking out from BASE

To be able to make a connection from BASE to GBrowse one first has to decide how to connect the reporters in BASE, which represent different features, to the corresponding features which they represent in the GBrowse database. Since reporters can be oligos[13], which for instance can originate from different regions within a gene, several reporters in BASE can represent the same gene feature in GBrowse. Or, oppositely put, one gene feature in the GBrowse database can have several different reporters in BASE that represent it. How then, is the connection between reporters in the BASE database and gene features in the GBrowse database, to be achieved?

5.2.1 Alternatives for connecting databases

One alternative for connecting the BASE and the GBrowse databases would be to add a certain id field to both the Reporter table in the BASE database and to the fdata table in the GBrowse database, that is those tables that represent reporters and genomic features respectively. That way, when linking from BASE, this id could be fetched for the selected reporter and sent as a URL parameter to GBrowse. In GBrowse, or rather in an intermediary script that handles the sent parameter from BASE, a connection could be done to the GBrowse database, fetching the information needed to display the feature that has the same id in the GBrowse database. When a connection is to be made in the opposite direction, to display expression data from BASE for a selected feature in GBrowse, the information needed to do this for reporters in BASE with the same id as the selected feature could be fetched by querying the BASE database. This would mean making minor changes in the schemas for both the BASE database and the GBrowse database. Although not particularly desirable from an upgradeability point of view, this was tested and could be done without affecting the functionality of any of the systems. This solution works fine for small amounts of data where an id can be assigned to different reporters and features manually, however when dealing with larger genomic projects with possibly thousands of reporters and features, a number of issues arise. The ids would need to be generated in a more automated fashion for this alternative to be a feasible solution. If we had some inherent information in the Reporter field in the BASE database and the fdata field in the GBrowse database, that was shared between the reporters in BASE and the features in GBrowse that they represented, and that at the same time was

[13]Oligos are shorter sequences of a certain length, usually composed of 25 or fewer nucleotides.

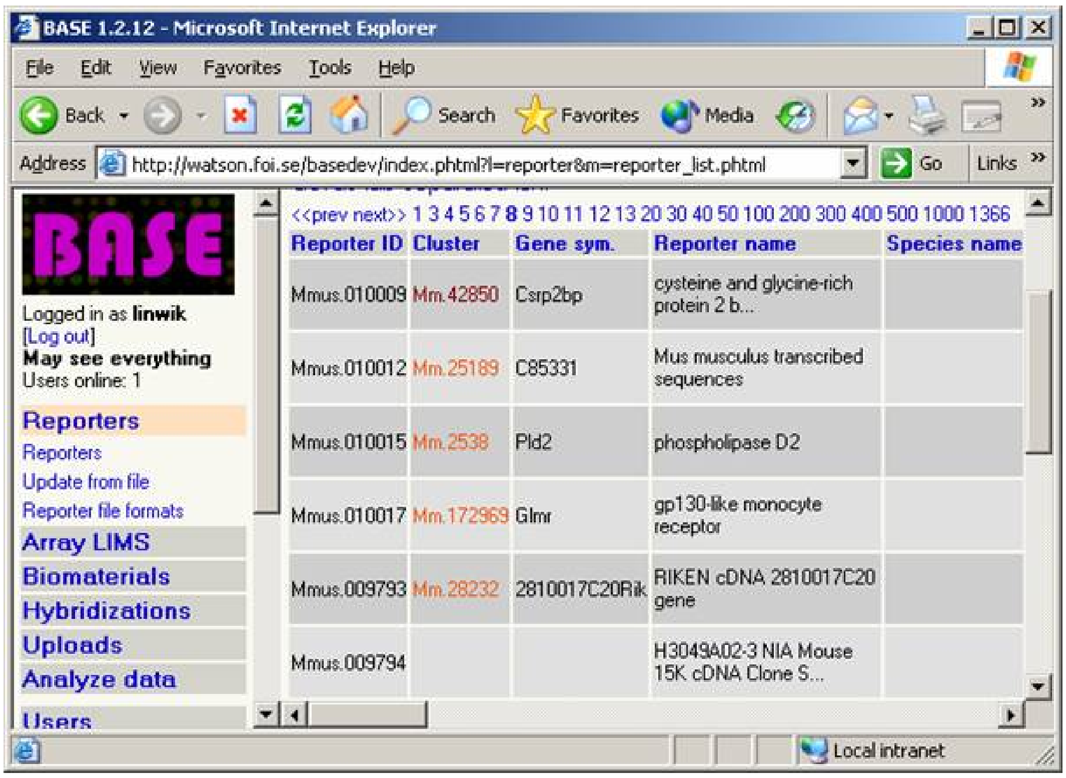
unique for each feature in GBrowse, this would be ideal to use as an id. If the reporters in BASE always had reached from the beginning of a gene to the end of a gene one might have used the reference sequence name and the start and stop position for the reporters, separated by some delimiter to create an id for the reporters, and done the same in the GBrowse database for the corresponding genes [14]. The generation of this id might then have been scripted for the BASE database and the GBrowse database separately, removing the need for manually checking and assigning which id from GBrowse to be assigned to which reporters in BASE. However, this is unfortunately not the case since, as previously mentioned, a reporter in BASE can start and end with a certain offset from the start and end of the feature it is to represent. Also when designing the sequences for the reporters in BASE one might not be entirely sure where all suggested features for the chromosome in question start and end. The sequences might be positioned right across the border between two genes. Another important issue would be that the annotation information for genomes can be changed after a while, due to the slight uncertainty in deciding where genes actually start and end. How would one handle ids that were composed of start and end information, if that is later changed? Even if all other issues with this solution would be resolved it would still take some work to automate the generation and loading of ids into both BASE and GBrowse respectively. For GBrowse, the id would need to be added to the converted GFF files for the entries that belong to a specific track, and thereby modified copies of the loading scripts would be needed as well, to load the new id field into the right place in the database.

A better approach A much better approach, if one wanted to use a common id to connect reporter and feature information in the two databases, would be to use the inherent feature id in the fdata table in the GBrowse database, the fid column. This id is unique to each feature in GBrowse and by using this id no modifications would need to be done for any of the GBrowse associated scripts. Since the fid column is an auto increment column that is not assigned to a specific start value by the loading scripts, the ids are in the form of integers ranging from 1 and up. What is needed to be done for BASE then is to create two extra Reporter table columns, an id column and one column for a reference sequence identifier, which are both loaded from a tabbed text file together with other BASE reporter column data. The reference sequence is to be set to the same reference sequence identifier that is used in the GBrowse database to identify which reference sequence a feature belongs to. If the Bioperl bp_genbank2gff.pl script is used to convert Genbank files into GFF files before loading them into the database, the reference sequence name for each feature will be set to the accession number in the Genbank file. This makes it quite easy to assign the value in the reference sequence identifier column for BASE since all reporters belonging to the same reference sequence will have the same

[14]There is a minimal risk that one would get the same values for those three fields for genes that reside in the same reference sequence but on opposite strands, due to the nature of the apparatus around how genes are transcribed.

value for that column. One might argue that it would be a better database design to have an integer value instead of a whole reference sequence name in that column and to store the reference sequence names in a separate table together with the corresponding integer indexing value, thereby saving storage space and reducing the redundancy of represented data. However, the way that BASE allows for adding information in the form of Reporter table columns in a way that is consistent with its functionality, interface and database design, see more in section 4.2.1 on page 13, makes this a more attractive solution despite the database design issue. The same goes for the next solution mentioned, which is discussed further down. To generate the right feature id for the reporter entries in the text file, a script is needed which opens a database connection to the GBrowse database and then, for each reporter entry in the file, sets the feature id to the id of the feature in the GBrowse database that has the same reference sequence as the given reporter and whose start and stop positions are such that at least part of the reporter starts within these positions. After this we have ids in BASE that also exists in the GBrowse database. One drawback of this is that one, before creating these ids for BASE, has to have the reference sequence with annotation information present in the GBrowse database. Not all reporters might belong to a valid feature whose annotation has been decided on, even though the reference sequence is present. For those reporters there might be difficulties in creating a link from BASE, since they have no particular feature id assigned to them.

The chosen solution In the end a different solution without any ids was chosen, which got rid of a lot of the previously mentioned issues. As for the previously discussed solution, an extra Reporter table column for storing the reference sequence identifier was added in BASE for the test data set. To add an extra column in BASE modifications need to be done in the reporter_columns.inc.php file under the BASE source directory. Entries for the extra column need to be added in two places as indicated by the structure of, and comments in the code. For all experiments for which one want to be able to use this solution, the extra column must be present in the tabbed text file with reporter column data to be loaded into BASE. In the GBrowse database no changes were needed for the connection between the two databases to work. In the reporter list displayed under the Reporters menu in BASE a list of reporters present in the BASE database, one for each row, is displayed together with the values that are present for all the Reporter table columns, see figure 4. One of those columns is the accession column displaying the reference sequence identifier for each reporter. As previously mentioned, see section 4.2.1 on page 13, BASE has a couple of files where modifications can be done to handle different aspects of the extra Reporter table columns that one can add in BASE. One of these files is reporter_columns.inc.php which contains functions for making links for the reporter column values in the reporter list in BASE. Unfortunately none of those functions could be used to make the accession column links that were to be created. The accession column links were to include three parameters, the reference sequence identifier, the reporter start position and the reporter end position within the reference sequence. None of the available linking functions where



http://watson.foi.se/basedev/index.phtml?l=reporter&m=reporter_list.phtml

Logged in as **linwik**
[Log out]
May see everything
Users online: 1

<<prev next>> 1 3 4 5 6 7 8 9 10 11 12 13 20 30 40 50 100 200 300 400 500 1000 1366

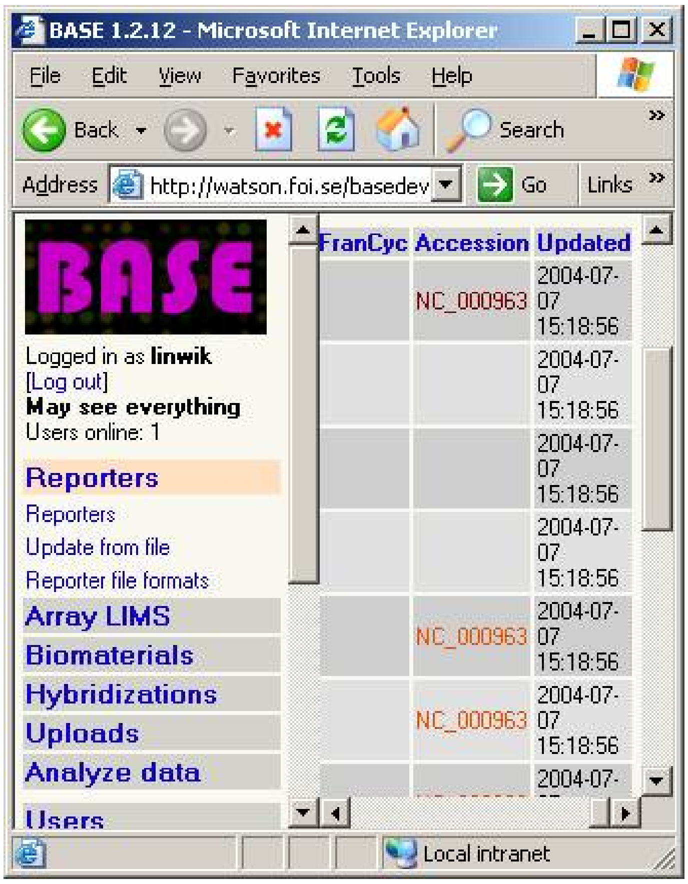
Reporter ID	Cluster	Gene sym.	Reporter name	Species name
Mmus.010009	Mm.42850	Csrp2bp	cysteine and glycine-rich protein 2 b...	
Mmus.010012	Mm.25189	C85331	Mus musculus transcribed sequences	
Mmus.010015	Mm.2538	Pld2	phospholipase D2	
Mmus.010017	Mm.172969	Glmr	gp130-like monocyte receptor	
Mmus.009793	Mm.28232	2810017C20Rik	RIKEN cDNA 2810017C20 gene	
Mmus.009794			H3049A02-3 NIA Mouse 15K cDNA Clone S...	

Figure 4: By clicking on Reporters in the BASE menu, a list of the reporters and their attributes can be displayed.

sent the appropriate parameters that were needed to create this kind of link, when called from the file reporter_list.phtml. Therefore a new linking function was added, which took a different set of parameters that could be used to retrieve the desired information from other reporter columns for that reporter. This however, demanded a small change in the file where the calls to linking function were made in order to create the links under the Reporters menu in BASE, that is in reporter_list.phtml. Alternatives to this were carefully investigated since it was desirable not to make any changes to the BASE code outside the two files that were intended for making changes in. This is due to the fact that if changes are limited to these files it becomes easier to upgrade to newer versions without the need to remember making changes in a lot of different files. The conclusion was though, that the linking needed could not be done without minor changes in the reporter_list.phtml file, so the small changes that were needed were accepted as necessary. The new function is useful for making links, for values in a reporter column, which are dependent on the value of other reporter columns. In the future this type of links can be created also for other columns than the Accession column by use of the new function. All that is needed is to add the name of the column in question, at the place in reporter_list.phtml where the link functions are called for the different columns. After the addition of this function, links to a receiving PHP script can be generated for reporters that has a value in the Accession column, see figure 5. Three parameters were sent in the address field of the URL, the reference sequence identifier, the start position for the reporter and the end position for the reporter. The start and end position values in the BASE database has a default value of zero and if positions are given they are set to a value larger than or equal to one. Therefore, if a zero value was come across for any of the positions, the corresponding parameter value was set to -1 to indicate the non existence of the value to the receiving PHP script.

5.2.2 The intermediary script

The intermediary script receives the reference sequence information together with start and end positional information for the clicked reporter in BASE. The script makes a number of checks and modifications to the start and end values before it finally makes a redirection to a GBrowse view with the modified parameters sent along. First, the validity of the parameters is checked and then a number of other checks are performed on the parameters. If the start or end position for the reporter is less than one then they might not be set in BASE and GBrowse is called with parameters that set it to display the first 5000 bp of the given reference sequence. For the case where the start and end position for the reporter has been given, the script will have to be adapted to the way in which these values are given. There are several possibilities for how the reporter positions are set in BASE. In figure 6 a 100 kbp hypothetical reference sequence is given, with two genes and three shorter reporter regions indicated. It might feel natural to assign the position to the far left in the upper strand to one and the position at the far right to



FranCyc	Accession	Updated
	NC_000963	2004-07-07 15:18:56
		2004-07-07 15:18:56
		2004-07-07 15:18:56
		2004-07-07 15:18:56
	NC_000963	2004-07-07 15:18:56
	NC_000963	2004-07-07 15:18:56
		2004-07-

Figure 5: The links for the reporters in BASE was made in the Accession column under the Reporters menu in BASE.

100 kbp, giving a start position of 10 and an end position of 38 kbp for the gene in the top strand. For the two reporters the positions would be 20 to 30 and 70 to 80 kbp. One might also consider giving the reporter positions as an offset from the feature they reside in, for instance a gene. However, there might be uncertainties about where genes really start and end, and a reporter might overlap a border between two gene features, which makes this convention a bit confusing. For the lower strand there are several alternatives

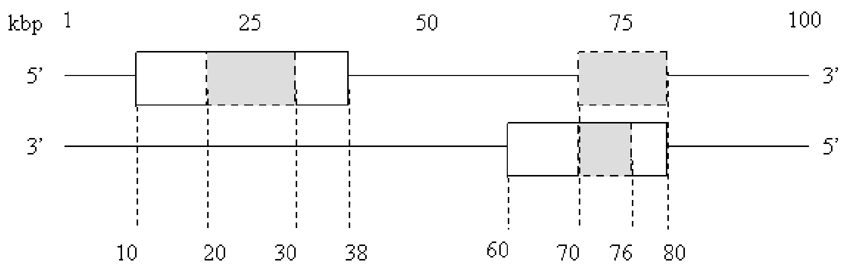


Figure 6: A 100 kbp hypothetical reference sequence with two longer regions, representing genes, and four shorter regions, grey in colour, representing reporters.

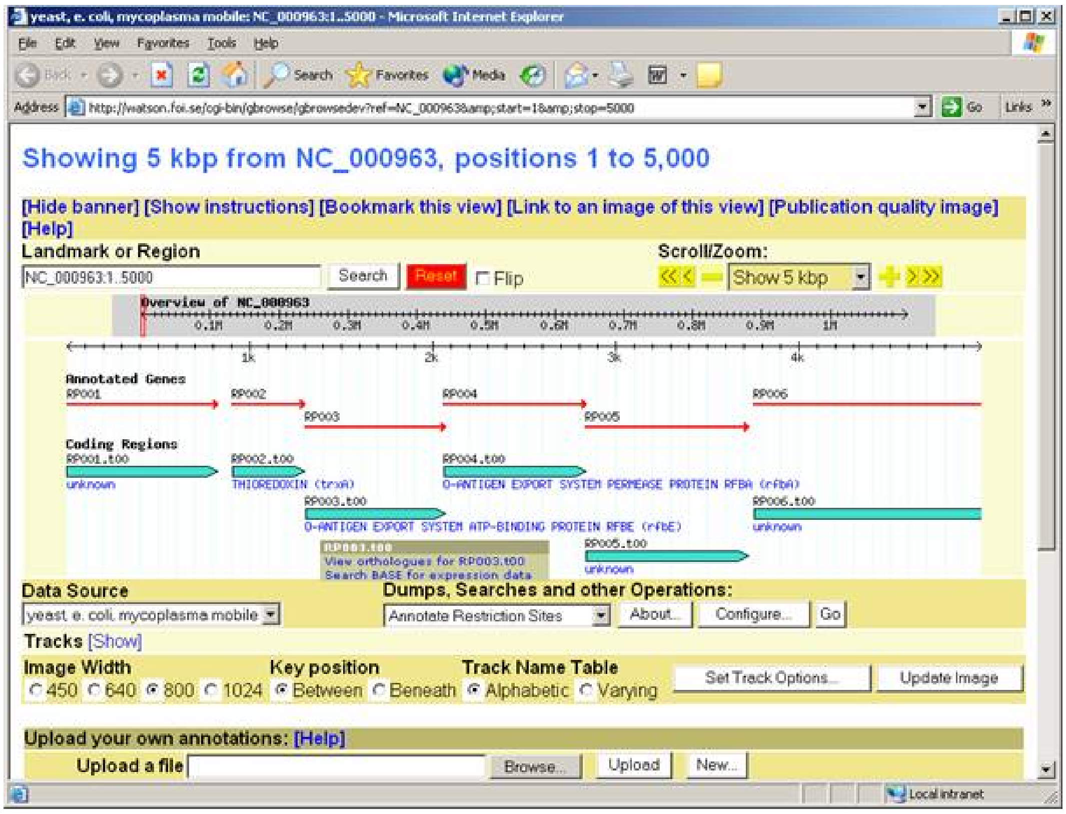
for how to assign the positions. One might consider it most natural to count the positions from right to left, since the polarity of the lower strand is opposite to the upper strand, and this is also the way things are done in some tools. If this convention is used, then the gene in the lower strand would have a start position of 20 and an end position of 40 kbp. Or, one might use the same coordinate system for both strands, counting from one in the far left to 100 kbp in the far right and indicate the strandedness of the feature in some other way. This is how the Bio::DB::GFF schema for the GBrowse database does. In this schema the strandedness is indicated in a special column. However in the BASE database, where the reporter start and end positions are given, there is no extra column for this purpose, so the strandedness should be indicated by the start and end position values them selves. Since these values are set manually before the data is loaded into BASE, the convention on how to set these values has to be agreed upon before the system is used. For the current solution the positions in the lower complementary strand should be assigned by still using the coordinate system of the main upper strand, but setting the start value to the right boundary of the feature and the end value to the left. The start position of the right most reporter in the lower strand would then be set to 75 and the end position to 70 kbp. This way, the start value is larger than the end position value for all reporters in the lower complementary strand and thereby the strandedness of the feature is indicated by the start and end positions by them selves. The fact that the Bio::DB::GFF schema in GBrowse assigns all positions counting from left to right in the coordinate system of the main strand, but in BASE this is done differently for reporters of the complementary strand, does not constitute a problem. In the script that receives these parameters from BASE one can check if the start position is larger than the end position and if so, switch the start and end position before sending them as parameters to GBrowse. In the intermediary script the area that is to be displayed by GBrowse is finally widened by lessening the start position and increasing the end position parameter by a certain amount before they are sent as parameters to GBrowse. This is done to show the environment which the reporter is in, both the feature it resides in and also some of the surrounding features, since the genetic environment around the gene that the reporter represent is important for analytical purposes. However, if the interval to display gets too large, above a certain threshold value, the end position parameter to be sent to GBrowse is lessened in order to lessen the interval to display. This is done because in GBrowse, if an interval that is above a certain value in size is given, the user will not see anything in the detailed view, where all the tracks are displayed, from the beginning. Instead the user will be prompted to click on a particular area in the overview before details are shown. This is not desireable since it might be difficult for the user to choose the exact location of the reporter one has chosen in BASE, which in turn would make the connection between this reporter and the view in GBrowse less meaningful. The threshold size to be displayed in the detailed view is dependent on the max segment option in the configuration file for GBrowse. When GBrowse is called with the parameters ref (the reference sequence identifier), start (a start position in that reference sequence) and stop (a stop position in that reference sequence), it displays the selected region as in figure 7. How to create the JavaScript menu that is also shown in the figure, is further explained in section 5.3.1 on page 38.

5.3 The GBrowse view

Before any data can be displayed with GBrowse the datasource one links to in GBrowse must have been properly created. It must also contain the data for the reference sequence that one links to, see the user manual in appendix A.1.3 for more information on how to achieve this. The configuration file for GBrowse can be used to alter the GBrowse view in several ways [43]. Only two tracks were displayed in the detailed view of GBrowse in this solution, a track for coding sequences (the CDS track) and a track for named genes (the GENE track). Coding sequences are portions of an mRNA or genomic sequence that encodes for a protein sequence. The order of which the tracks are displayed in the detailed view is the same as the order of the tracks in the GBrowse configuration file. To turn these tracks on by default the default features option in the configuration file was set to

```
default features = CDS GENE.
```

Genbank files loaded with the bp_genbank2gff.pl script automatically get a certain value for the source field in the database, but one might also want to load and display data from files that have not been converted to GFF-format by using this script, and thereby



yeast, e. coli, mycoplasma mobile: NC_000963:1..5000 - Microsoft Internet Explorer

Address: http://watson.foi.se/cgi-bin/gbrowse/gbrowsedev?ref=NC_000963&amp;start=1&amp;stop=5000

Showing 5 kbp from NC_000963, positions 1 to 5,000

[Hide banner] [Show instructions] [Bookmark this view] [Link to an image of this view] [Publication quality image] [Help]

Landmark or Region: NC_000963:1..5000

Scroll/Zoom: Show 5 kbp

Data Source: yeast, e. coli, mycoplasma mobile

Dumps, Searches and other Operations: Annotate Restriction Sites

Tracks [Show]

Image Width: 450 640 [✓ 800] 1024

Key position: [✓ Between] Beneath

Track Name Table: [✓ Alphabetic] Varying

Upload your own annotations: [Help]

Upload a file

Figure 7: If sent the appropriate parameters GBrowse displays a particular selected region e.g. as above. The tracks that are shown have been configured in the configuration file for the given datasource.

might have a different source value. If one wants a particular track to be able to display features with a particular method, for instance "CDS", regardless of the source for the features one needs to remove the source part of the feature type that is given as value for the feature option of that track in the configuration file. For instance, instead of writing feature = gene:Genbank one just writes feature = gene. This was done for both tracks that were to be displayed in this study. The max segment option in the configuration file was set to 1000000. This sets an upper bound on the maximum size DNA segment that will be displayed in the detailed view. A large number of configuration options can be set in the GBrowse configuration file, some of which were mentioned in section 4.2.2 on page 18.

5.3.1 JavaScript popup menu

To be able to display a JavaScript menu when hoovering over a feature in the detailed view, several things need to be done in the GBrowse configuration file. In this case the Overlib library was used to create the JavaScript popup menu, for more information on Overlib see section 4.2.6 on page 24. For being able to use Overlib the html in the head option of the GBrowse configuration file was set to include the overlib.js file. Also, the header option was set to include a html div tag that is needed for Overlib to work properly. The style rule given for this tag was slightly modified to include also a width for the resulting JavaScript menu, without that modification the menu looked a bit odd. The menu was to be displayed for different features in a track, and the links in the menu needed to be constructed with different parameters depending on which feature that was selected. Since the link option for different tracks can be dynamically computed by the use of Perl subroutines of a particular form, see also section 4.2.2 on page 18, and include information for the specific feature to link from, this option was used for creating the JavaScript menu. The link option for the CDS track was set to a Perl subroutine which investigated the feature object that is sent to it as a parameter, and retrieved the information, which was needed to create the links, from that object. Normally this subroutine should return a URL which is linked to when the selected feature is clicked, but in this case a JavaScript Overlib call was returned instead. The Overlib call contained a menu title and several menu links, in part composed of feature information retrieved from the feature object. One link was for displaying orthologue information for the selected feature and that link was to a PHP script to which the feature id for the selected feature was sent as a URL parameter. The other link was for displaying expression data from BASE for the selected feature. That link was also made to a PHP script and the reference sequence identifier, the feature start position and the feature end position were sent along as URL parameters. According to the requirements there were also supposed to be a third link, for displaying information in the database for the selected feature. However, this requirement was fulfilled in a different way. Setting the link option in the GBrowse configuration file to AUTO, allows for an inherent GBrowse script, the gbrowse_details script, to be run when a feature is clicked in the detailed view. This script automatically generates a summary, of the information for the clicked feature that exists in the GBrowse database. This option was used in the GENERAL section of the configuration file, which makes this a global rule for all track features. Also, the link target option was set to _blank making these summary pages pop up in a new browser window. Further down in the configuration file, for the CDS track, this rule was overridden by the use of the Perl subroutine described above. This means that clicking a feature in the GENE track generates a summary window for that feature, while hoovering with the mouse over a feature in the CDS track creates a JavaScript menu with two links. This menu is "sticky" in nature, which means that it sticks until the user has either clicked one of the links in the menu or has moved the mouse first into the menu area and then out again. A tooltip is automatically displayed when holding the mouse over a feature in the detailed view. It might feel unnecessary to display the tooltip for the CDS track when a JavaScript menu is also displayed for that track, to control what is displayed in the tooltip the title option in the configuration file can be set to override the default text. Note that to be able to use Overlib the overlib.js file mentioned in section 4.2.6 was downloaded and placed in the same directory as the other extra files which were needed for the automation of information flow to work. To handle what to be done if any of the links in the JavaScript menu were clicked a webbased database interface was constructed using mainly PHP and HTML.

5.4 Handling the orthologue choice

When the orthologue link is chosen in the JavaScript menu for a feature in the GBrowse view, the feature id of that feature is sent to the PHP script that handles the request to view orthologues for a feature. To be able to store the information needed to show orthologues for a feature, some modifications to the original Bio::DB::GFF schema, in figure 2 on page 16, were needed, see modifications in figure 8. First an orthologue id was added to the fdata table, which was thereby coupled to a table containing the orthologue families which the ids correspond to. The orth_fam attribute is used as an orthologue family identifier. The value of these identifiers differs depending on which kind of grouping that has been used, an example of this is that COG families [75] are conventionally named COG followed by four numbers. This way, the name of the orthologue families also indicate what kind of grouping scheme that was used for them. If one would want to use a locally developed grouping scheme, a different prefix and numbering convention might be used for those orthologue groups, see also section 6.5 on page 49 for a discussion of possible extra tables that might ease the tracking of different versions and grouping schemes. The orth_descr attribute is a description of the given orthologue family. The forth_fam_func table couples orthologue families to different functionalities, described in the forth_func table. The three extra tables were designed with the COG orthologue grouping in mind [75], where orthologue families have a short identifying name and a description and each belong to one or more functionality groups which each has a one letter abbrevation [76, 77].

If the choice to view orthologues for a feature is clicked in the JavaScript menu in GBrowse

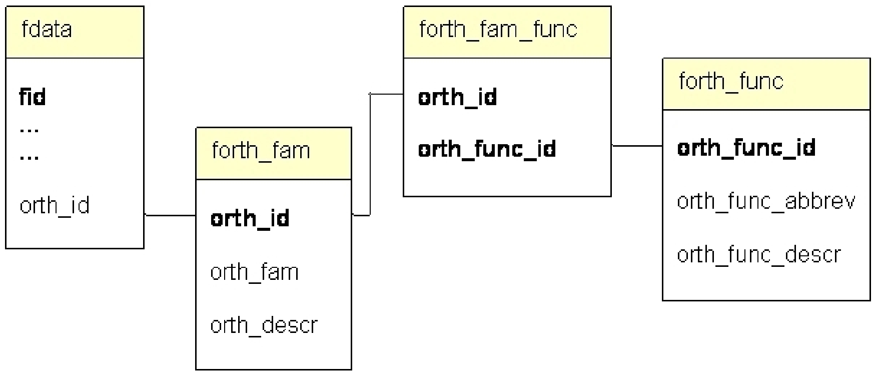
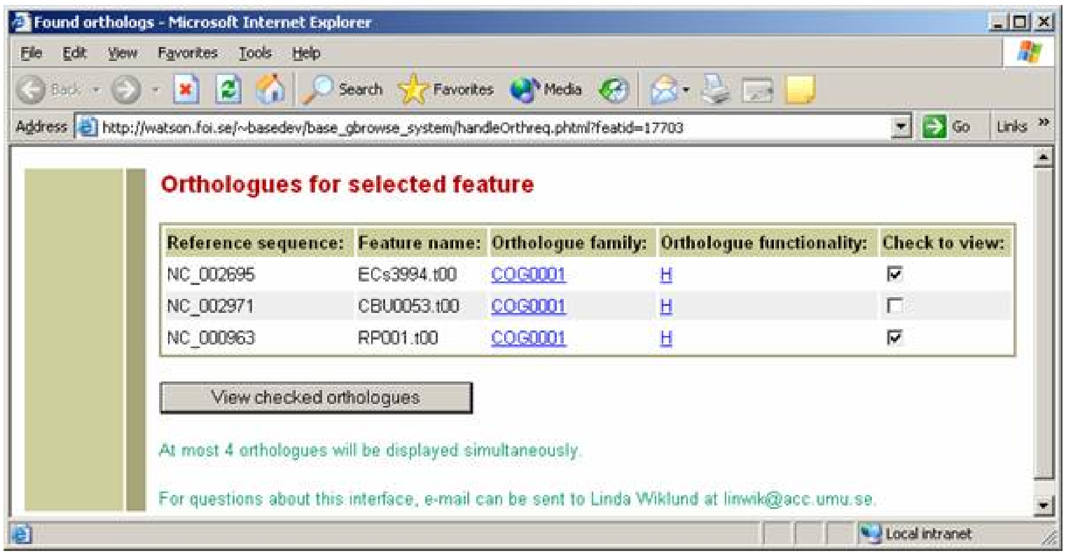


Figure 8: A few modifications of the original Bio::DB::GFF schema was needed to be able to store information about to which orthologue family a certain feature belong. An orthologue id was added in the fdata table and the addition of three other tables was also needed to accomplish this.

the orthologue id and the reference sequence identifier for that feature is used to search the fdata table for orthologues to the selected feature. Features that belong to the same orthologue family but does not reside in the same reference sequence as the clicked feature are concluded to be orthologuous to the selected feature. Section 6.2 on page 47 contain a discussion of circumstances when this might not be the case. If orthologues are found, a checklist form is produced with some information for each found orthologous feature, see figure 9. Except for the reference sequence and feature name two links are displayed for each ortholog. The orthologue family link is to a page with descriptions and links for that particular orthologue family at the COG website [75]. The orthologue functionality link is set to a page, also at the COG website, which describes the different functionality groupings for each abbrevation. It would be possible, using URL parameters, to link to a more specific page where a description of a particular functionality is given instead of the overview page for functionalities. However, as one orthologue family can be connected to many functionalities, the overview page was chosen to link to instead. See also section 6.5 on page 49 for a discussion on how to adapt these links if several grouping schemes are to be used and not just COGs. Already when the checklist is generated, the reference sequence, start position and end position of the orthologous features are retrieved from the GBrowse database. These are checked and modified in a number of ways and are after that used as parameters in the URL links to GBrowse which are generated for each found feature. When the orthologues that the user wishes to display has been checked and the form is submitted, these URLs are sent along to the script that handles the form data, along with some other information, such as the maximum number of orthologues to be displayed. In the web page that is produced an iframe is created for each checked feature,



Found orthologs - Microsoft Internet Explorer

Address http://watson.foi.se/~basedev/base_gbrowse_system/handleOrthreq.phtml?featid=17703

Orthologues for selected feature

Reference sequence:	Feature name:	Orthologue family:	Orthologue functionality:	Check to view:
NC_002695	ECs3994.t00	COG0001	H	☑
NC_002971	CBU0053.t00	COG0001	H	☐
NC_000963	RP001.t00	COG0001	H	☑

View checked orthologues

At most 4 orthologues will be displayed simultaneously.

For questions about this interface, e-mail can be sent to Linda Wiklund at linwik@acc.umu.se.

Figure 9: When orthologous features are found for a selected feature, information about these features is first displayed in the form of a checklist, so that the user can choose which of them to display the genomic environment for.

and in each iframe a specific GBrowse view is opened, displaying the next feature in order and some of its genomic surroundings. This is done by using URLs, which were sent with the form, to specify a reference sequence and region, see figure 10. By using iframes each orthologue feature can be opened in a fully interactive GBrowse window of its own, with all the functionality of a normal GBrowse view such as the JavaScript popup menu and different link outs, retained. The use of iframes to retain interactivity is also discussed in section 6.2 on page 47. The height of the iframes containing the GBrowse windows, was set to a quite small value. This was due to the fact that if the orthologue option in the JavaScript menu is chosen for a particular feature and several orthologs are found, it would not be desireable to display a too large portion of each view. To be able to overview and align the different orthologue regions one would want at least some of them to be visible in the browser window without the need to scroll.

5.5 Handling the expression data choice

When the choice is made to view expression data for a feature, the reference sequence identifier and the start and stop position for that feature are sent to an intermediary PHP script. A connection is then made to the BASE database and the ids and other information of reporters that belong to the same reference sequence as the selected feature and that overlap the region of the feature. The information about these reporters is currently displayed in a table on a web page, see figure 11, but the ids are in the future to be sent as parameters to an expression data search interface which is currently under development

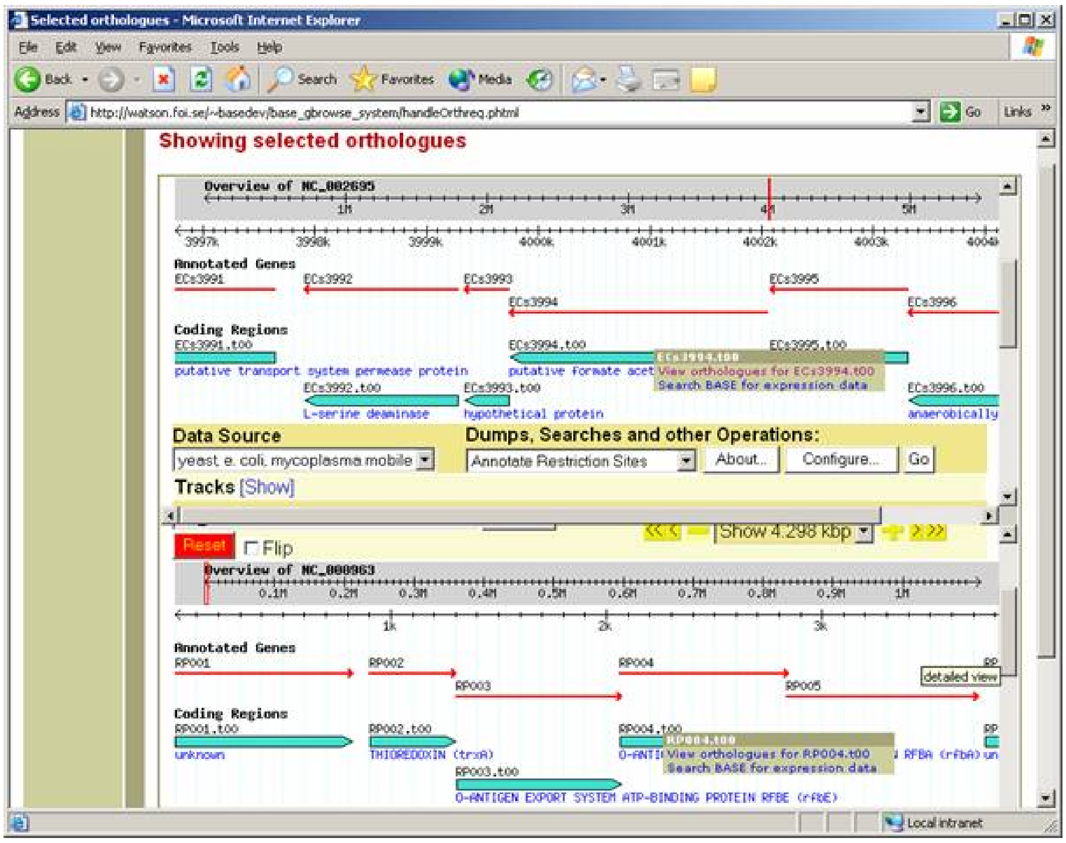
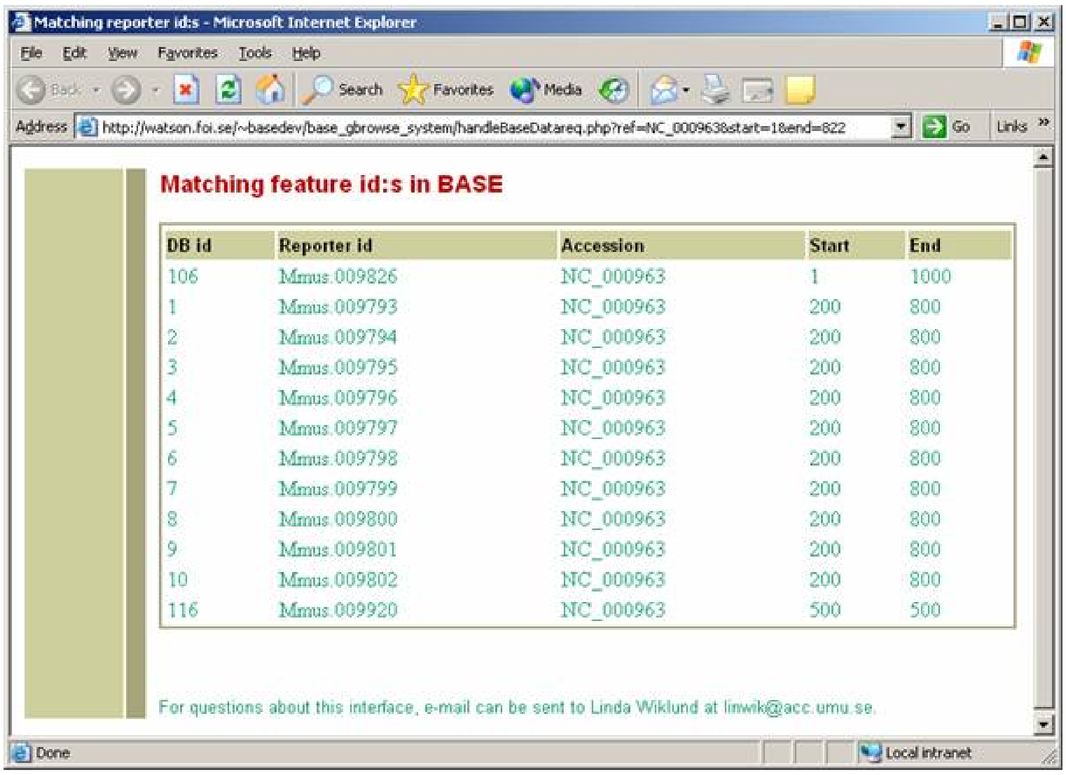

Selected orthologues - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://watson.foi.se/~basedev/base_gbrowse_system/handleOrthreq.phtml
Go Links
Showing selected orthologues
Overview of NC_002695
Annotated Genes
ECs3991 ECs3992 ECs3993 ECs3994 ECs3995 ECs3996
Coding Regions
ECs3991.t00 ECs3994.t00 ECs3995.t00
putative transport system permease protein
putative formate acet
ECs3994.t00
View orthologues for ECs3994.t00
Search BASE for expression data
ECs3992.t00 ECs3993.t00 ECs3996.t00
L-serine deaminase
hypothetical protein
anaerobically
Data Source
yeast e. coli, mycoplasma mobile
Dumps, Searches and other Operations:
Annotate Restriction Sites
About... Configure... Go
Tracks [Show]
Show 4.298 kbp
Reset Flip
Overview of NC_000963
Annotated Genes
RP001 RP002 RP003 RP004 RP005
detailed view
Coding Regions
RP001.t00 RP002.t00 RP004.t00
unknown
THIOREDOXIN (trxA)
RP004.t00
View orthologues for RP004.t00
Search BASE for expression data
RFBA (rfbA)
RP003.t00
O-ANTIGEN EXPORT SYSTEM ATP-BINDING PROTEIN RFBE (rfbE)
Local intranet

Figure 10: Orthologues that are chosen to be displayed are aligned under each other in a separate fully interactive GBrowse view for each orthologue.

by another party, the reasons for this are discussed in section 6.3 on page 48. A composite


Matching reporter id:s - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://watson.foi.se/~basedev/base_gbrowse_system/handleBaseDatareq.php?ref=NC_000963&start=1&end=822
Go Links
Done
Local intranet

Matching feature id:s in BASE

DB id	Reporter id	Accession	Start	End
106	Mmus.009826	NC_000963	1	1000
1	Mmus.009793	NC_000963	200	800
2	Mmus.009794	NC_000963	200	800
3	Mmus.009795	NC_000963	200	800
4	Mmus.009796	NC_000963	200	800
5	Mmus.009797	NC_000963	200	800
6	Mmus.009798	NC_000963	200	800
7	Mmus.009799	NC_000963	200	800
8	Mmus.009800	NC_000963	200	800
9	Mmus.009801	NC_000963	200	800
10	Mmus.009802	NC_000963	200	800
116	Mmus.009920	NC_000963	500	500

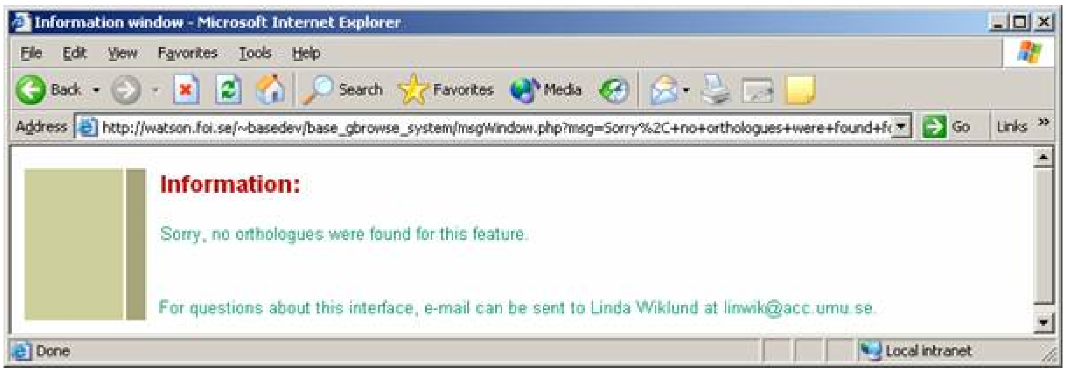
For questions about this interface, e-mail can be sent to Linda Wiklund at linwik@acc.umu.se.

Figure 11: When the expression data choice is made in the JavaScript menu in GBrowse, information about the reporters from BASE that overlap the region of the selected feature is displayed in a table.

index on the reference sequence identifier, the start and the stop position was added to the Reporter table in the base database and the use of the index when querying was checked using the EXPLAIN syntax, see also section 4.2.5 on page 23. This is not required but can be useful for improving the performance of the query to retrieve reporter ids.

5.6 Error messages

If errors occur or if a query returns an empty set at some point, within any of the PHP scripts that handle the link choices of the JavaScript menu in the GBrowse view, a message is displayed in a message window, see figure 12. Messages are displayed for instance if no matching reporters are found in BASE for a selected feature, if no orthologues are found for a particular feature or if the feature id of a feature can not be retrieved. Alternatively the message could be displayed in a JavaScript popup window instead, by use of the open.window JavaScript function.


Information window - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://watson.foi.se/~basedev/base_gbrowse_system/msgWindow.php?msg=Sorry%2C+no+orthologues+were+found+fc
Go Links
Done
Local intranet

Information:

Sorry, no orthologues were found for this feature.

For questions about this interface, e-mail can be sent to Linda Wiklund at linwik@acc.umu.se.

Figure 12: An example of a message window, in this case for explaining that no orthologues were found for the selected feature.

5.7 Overview of the system

Figure 13 shows an overview over the currently used solution. The reference sequence, start and end position for the reporter that is clicked in BASE is sent to an intermediary script that modifies the parameters before invoking GBrowse with the modified parameters sent along. By clicking a feature in the GENE track a summary page is displayed with a summary of the information for that feature that is present in the database (not shown in the figure). The functionality of the JavaScript menu for features in the GBrowse view includes displaying orthologues for a feature and fetching information about the reporters in BASE which represent the selected feature. To also display expression data for each reporter, an extra linkage to an expression data search interface which is under development by another party, will need to be done, see appendixA.1.4 on page 64 and section 6.3 on page 48 for more information. The orthologues that are found for a feature are first displayed in a checklist to allow the user to choose which to display before they are displayed under each other in a separate GBrowse view for each. The GBrowse database can be loaded with data from Genbank files if they are first converted to GFF files.

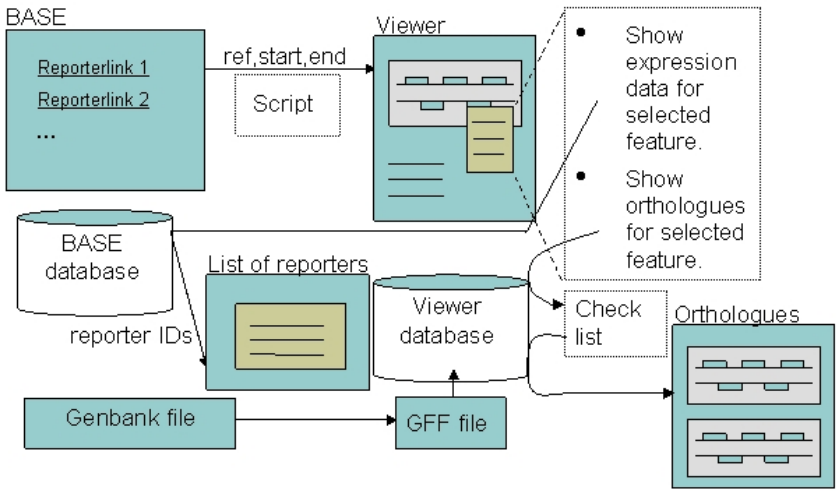


Figure 13: Overview of the current solution.

6 Discussion

Different parts and aspects of the current solution are discussed below.

6.1 Characteristics of biological data within this study

Many of the characteristics of biological data that Elmasri and Navathe [4] mention (summarized in section 6.1 on page 46) apply also to the context of this study. It has been particularly time consuming to investigate how data that is prone to change, complex, exists in massive amounts or, for the representation of which, there exists several competing standards, is to be handled. Since the database design was based on the relational model[15], which is not very flexible in this respect, and due to the fact that the current solution uses a number of tools and techniques that are more or less mature, some compromises were made in order to produce a functional solution within reasonable time. For instance, the Bio::DB::GFF schema for the GBrowse database did not have any inherent means for handling different versions of a reference sequence. This was also reflected in the bp_genbank2gff.pl script since it does not save any version information from the Genbank file that it converts into the resulting GFF file. Different versions of a reference sequence in the Genbank database has the same accession number (which is what is stored as the refernce sequence name in the database) but different version numbers. Therefore, if one would want to load a GFF file which represented a different version of a particular reference sequence that was already present in the database, the old one would first have to be removed, since it already has the reference sequence attribute for each feature set to the same accession number as is present in the new file to be loaded. There exists other schemas that have better support for version handling, for instance the BioSQL schema [37] where the version is saved in a separate column and which in my opinion has a better and more intuitive design than the Bio::DB::GFF schema does. The BioSQL schema also has interesting possible add-ons to the core schema, for instance bioentry date stamping, which makes the non-temporal database into a valid time database[16]. The support for the alternative schemas and their corresponding adaptors in GBrowse is however not as mature as for the Bio::DB::GFF schema, and the Bio::DB::GFF schema was therefore concluded to be the best choice at this point. Other important advantages of the Bio::DB::GFF schema, besides being well supported and stable, is that it is minimal and fast. A possible solution to the versioning problem in Bio::DB::GFF would be to make a version of the bp_genbank2gff.pl script that appended the version to the accession number in the resulting gff-file, so that version information is included in the database and different versions of reference sequences with the same accession number can be loaded.

[15]Using the relational model was the only feasible alternative under the conditions of this study.

[16]A valid time database is a temporal database which use the interpretation that a time point or time period associated with a fact or event in the database is the time that the event occured or the period during which the fact was considered to be true in the real world [4].

One reason why this is not particularly appealing though, is that it breaks first normal form. Even though it is possible to load different versions of the same reference sequence, since they get different names, the version information is not separated from the name of the reference sequence and one can not directly query the database for information that is based on name and version being separate entities. It might be a better choice to await more stable versions of other schemas instead.

Sometimes when there does not exist a clear convention for how data is to be entered or handled within certain tools, for instance for how start and end positions on the complementary strand are given in BASE, such a convention needs to be agreed upon before a system can be developed that is to handle this data. There is a tradeoff between allowing flexibility and the need for a more narrow specification to ease the development of software tools.

6.2 Handling orthologues

According to the definition of orthologues a speciation event should have separated orthologous genes. However, when the checklist with found orthologues for a selected feature is generated, the query used to retrieve the orthologues only make use of the fact that the reference sequence identifier for the found orthologues should not be the same as for the selected feature. For prokaryotes the DNA is organised in one circular chromosome, which makes it likely that if the reference sequences differ for features in the database, they reside in different genomes. In eukaryotes on the other hand, the genome consists of several chromosomes, which in the database are treated as different reference sequences. This means that in the case when a feature is found that is grouped into the same orthologue family and belongs to the same organism as the selected feature, but resides in a different chromosome in the same organism, it will be present in the checklist although it is strictly speaking not an orthologue. The same goes for features from other subspecies or strains that are grouped in the same orthologue family. However, these features might still be interesting to compare with the other ones in the checklist, even if they do not fully comply with the orthologue definition, and the user is free not to check them if they are considered not to be interesting. The definition and usage of the orthologue and paralogue concept has also, as previously mentioned, been subject to some debate [13, 6].

One might ask why the description of the orthologue functionality groups is included in the database design, since it is not really used in the interface. The functionality links available in the checklist of found orthologues, all link to the same page. However, if a locally adapted or other grouping scheme that is different from the COG grouping scheme is to be added in the future, the functionality link might want to be changed to for instance a PHP script that displays the functionality description from the database.

One important issue with displaying the orthologues in their genomic environment was how to align them under each other, for ease of comparison, and at the same time retain interactivity in the different views, so that analysis could be continued for each of the sequences. That is, the same requirements that apply to the genomic view which is opened when clicking a link in BASE should apply for all of these views also. The use of iframes suited this purpose perfectly and was of good use in the current solution. By displaying each orthologous sequence in a GBrowse view of their own, each of them could be investigated further and interacted with and all the views included the JavaScript menu and all other functionality that had been added to or configured in the GBrowse system. By opening these view in separate iframes, the views could be aligned under each other all in the same web page in a scrollable manner. Maintaining interactivity also for the orthologue views is important, both for being able to retrieve more information about the orthologous features and also for being able to dynamically explore the genomic environment that the features resides in.

6.3 Retrieving expression data from BASE

The requirement to be able to view expression data stored in the BASE database, for a feature selected in the GBrowse view, was not completely fulfilled within the scope of this task. At the start of the development of the current solution the expression data in question was only present in BASE in the form of intensity data for the array images that is loaded into BASE after an experiment (by comparing the intensity of different colours in the array images, the expression of the differently dyed sequences that bind in to reporters at the array spots can be compared). These raw data could be processed by an external program to get analysed expression data as a result. The data produced by such an analysis would be interesting to display for a feature, but could not at that time be loaded back into the BASE database. Instead a BASE R[17] plugin was counted on being available from another party, which would have the same functionality as this external program but be run from within BASE and store the results in the basedynamic database. Result sets produced from BASE plugins can be read into the basedynamic database if given in the correct file format. This plugin was however not completed during the scope of this study, and the lack of data in the database made it very difficult to design a solution for retrieval of data. Further complicating the issue was that the BASE system had quite recently been taken into use and the documentation for how to use and configure it was quite unsatisfactory. Adding to this, upgrades to newer versions of BASE was also done several times during the scope of this study, causing minor changes to the BASE database schema, the effect of which needed to be considered when developing a solution. There was also some confusion around exactly what data should be displayed as expression data and in what form. Quite recently a temporary solution for storing data back into BASE from the external program that produces analysed expression data has been made at the site of development. A type of ”fake plugin” was made in BASE which takes the result file from the external program and stores the data in the basedynamic database. This

[17]R is a language and environment for statistical computing and graphics similar to the S language [84]. There also exists a commercial version of S which is called S+.

makes the data visible from within BASE. It might be possible to retrieve expression data to display from this data by querying the basedynamic database, but the use of this solution was developed a bit too late to be considered in this study. It also involves a lot of manual handling of data, such as rearranging files to a format that is suitable to the external program or BASE respectively before any data can be handled by one or the other tool, the exact details on how this is done is not included here. This implies that it might be considered only a temporary solution. A solution that might be used in the near future is a search interface for retrieving expression data from within base, which is under development by Max Bylesjö at Umeå Plant Science Center. This interface is intended to be released as a part of the BASE GUI. Currently when the expression data choice is made for a feature in the JavaScript menu, the reporter ids and other facts about the reporters which has the same reference sequence as the clicked feature and that overlaps, at least in part, the region of the feature, are fetched from the BASE database. These reporters are then displayed in a table on a web page, see figure 11 on page 43. Instead of displaying these reporters, their ids can be sent as parameters to a redirect URL containing the adress of the search interface, which has the capability of accepting several reporter ids as parameters in the adress field. Comments has been made in the code for how to make this redirection, see details in appendix A.1.4 on page 64.

6.4 Limitations to the current solution

Limitations to the current solution include the lack of version handling in the Bio::DB::GFF schedule, see section 6.1 on page 46, the possibility for non-orthologues, according the strict definition of orthologues, to appear in the orthologue checklist, see section 6.2 on page 47, and that the choice of displaying expression data from BASE for a feature for different reasons could not be fully accomplished, see section 6.3 on page 48.

6.5 Future development

The performance of GBrowse can be increased by using GBrowse in conjunction with the mod_perl embedded Perl interpreter [27, 50].

In the future further restrictions to GBrowse might need to be added. The access to GBrowse can be restricted by IP address, host name, domain or username and password. Restrictions can apply to the database as a whole or to particular annotation tracks. To limit access to a whole database, Apache's standard authentication and authorization can be used, to limit access to individual tracks a restrict option can be added to the track. The value of the restrict option can be returned from a Perl subroutine, which is called with the host, IP adress and authenticated user as parameters. For more information about these issues see [43].

One might consider making an interface to ease the addition of orthologue information to the GBrowse database, both for assigning different features to an orthologue family and for loading a complete orthologue grouping version, for instance the current version of the COG families and their functionality [75]. The LOAD DATA INFILE syntax in MySQL might also be used, if one makes a textfile of the appropriate format with all the orthologue groups and functionalities, or with a number of ”feature id, orthologue id” pairs for assigning features in the database to orthologue families respectively [38]. The phpMyAdmin tool might also be used in this respect [83].

Features can, as previously mentioned, be grouped into orthologue families in different ways, for instance according to the COG database [75], the TOGA database [88] or according to a local grouping scheme. If one would want to load several of these groupings into the database, or different versions of them, it might be useful to add information about to which way of grouping and which version of that grouping the orthologue families present in the database belong. A suggestion would be to have a mapping table with orthologue family ids and ids that indicate the way of grouping and add a corresponding table with the grouping id as index and a description and version for the different groupings. This might be useful if one wants to keep track of to what grouping scheme and to what version of that grouping scheme that different features have been assigned to. Which type of grouping that is used for a feature is usually also evident from the orthologue family name, for instance COG families start with COG appended by four numbers, for a locally developed grouping scheme a different prefix and numbering scheme can be chosen.

One thing that needs to be changed if several grouping schemes are to be used is the orthologue family link and the orthologue functionality link that are generated in the checklist which is displayed when orthologues are found for a feature. One might use the naming convention of the different grouping schemes for the orthologue family name, and use a distinct naming convention for each locally adapted grouping scheme that is used. Then a simple check on the orthologue family name can be added in the file where these links are produced (that is in the printOrthRow in the htmlTempl.inc.php file), in order to generate different links for different schemes. As an example for the orthologue family link, if a locally adapted grouping scheme is used, with a naming convention of XCOG followed by four numbers, then this naming convention can be checked for and a link out made to a PHP script that takes the orthologue family id as a parameter. The link out to the PHP script will in other words be done for all features whose orthologue family name follows that convention. The script might then use the id to retrieve information for that orthologue family from the database and display it in a web page.

For more information on a number of other reconfiguring issues, such as how to use the interface under development of Max Bylesjö to view expression data for reporter ids fetched from the BASE database, or reconfigure issues to be dealt with, if the GBrowse database is to be changed from a MySQL database to an Oracle database, see appendix A.1.4 on page 64.

6.6 Conclusions

The developed framework should be quite extendable since new links in the JavaScript menu can easily be added in the GBrowse configuration file. In the Perl subroutine, where the links for this menu are created, one has access to several attributes of the selected feature through the feature object that is sent as a parameter to the subroutine. This information can be sent as parameters to for instance a PHP script, which opens up for a multitude of ways to extend the capabilities of the framework. This is an important property since the focus of a biological study may vary a lot depending on which type of questions that need to be answered before conclusions can be drawn. In respons to changes in functionality or data representation when upgrading to newer versions of the BASE or GBrowse system, changes might also be needed in the current solution, to reflect this fact.

A number of the characteristics which according to Elmasri and Navathe [4] makes management of biological data a particularly challenging problem has also been encountered during this study. To adress the issues that arise from these characteristics a closer connection between scientists in the fields of biological research and computer science is needed.

7 Acknowledgements

To:

Pär Larsson, external supervisor.
Swedish Defence Research Agency
NBC Defence Division
Department of NBC Analysis
SE-901 82 Umeå
For coming up with inspiring ideas.

Michael Minock, internal supervisor.
Senior Lecturer in Computing Science
Umeå University
SE-901 87 Umeå
For useful feedback on the report.

Kerstin Svensson.
Swedish Defence Research Agency
NBC Defence Division
Department of NBC Analysis
SE-901 82 Umeå
For introducing microarray methodologies in general and BASE in particular.

Johan Tegman.
Swedish Defence Research Agency
NBC Defence Division
Department of NBC Analysis
SE-901 82 Umeå
For help in setting up the developing environment.

Max Bylesjö
Umeå Plant Science Center (UPSC)
Umeå University
SE-901 87 Umeå
For letting in on the fact that an expression data interface is under development, which might later be used in conjunction with the solution presented in this report.

Employees of the Swedish Defence Research Agency in Umeå in general.
Swedish Defence Research Agency
NBC Defence Division
Department of NBC Analysis
SE-901 82 Umeå
For being nice and friendly to work with.

References

Books and articles

[1] Campbell, M.K. (1995). Biochemistry, 2nd ed. Saunders College Publishing, Orlando.

[2] Chicurel, M. (2002). *Bioinformatics: bringing it all together.* Nature 419:751-767.

[3] Cohen, W.W. (1998). *Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity.* International Conference on Management of Data. Proceedings of the 1998 ACM SIGMOD international conference on Management of data:201-212.

[4] Elmasri, R., & Navathe, S.B. (2000). Fundamentals of database systems, 3rd ed. Addison Wesley Longman Publishing Co., Inc., Boston, MA.

[5] Fadiel, A., & Naftolin, F. (2003). *Microarray applications and challenges: vast Array of possibilities.* International Archives of Bioscience 2003: 1111-1121.

[6] Fitch, W.M. (2000). *Homology a personal view on some of the problems.* Trends in Genetics, Volume 16, Issue 5:227-231.

[7] Garcia-Molina, H., & Papakonstantinou, Y., & Quass, D., & Rajaraman, A., & Sagiv, Y., & Ullman, J., & Vassalos, V., & Widom, J. (1997). *The TSIMMIS approach to mediation: Data models and Languages".* Journal of Intelligent Information Systems, Volume 8, Issue 2:117-132.

[8] Gaunt, S.J. (2002). *Evolutionary Developmental Biology: Homologoues Regulatory Genes and Processes.* Encyclopedia of life sciences, vol.7, pp. 20-27. Macmillan Publishers Ltd, Nature Publishing Group, London.

[9] Graefe, G. (1993). Query evaluation techniques for large databases. ACM Computing Surveys (CSUR), 25(2):73–169, 1993.

[10] Graves, J.A.M. (2002). *Gene mapping: Comparative.* Encyclopedia of life sciences, vol.7, pp. 625-630. Macmillan Publishers Ltd, Nature Publishing Group, London.

[11] Guttman, A. (1984). R-Trees: A dynamic index structure for spatial searching. In Proceedings of the ACM SIGMOD Conference, pp. 47-57. Boston, MA.

[12] Haas, L.M., & Kossmann, D., & Wimmers, E.L. & Yang, J. (1997). *Optimizing Queries Across Diverse Data Sources.* Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece. VLDB97:276-285.

[13] Jensen, R.A. (2001). *Orthologs and paralogs - we need to get it right.* Genome Biology, 2:interactions1002.1-1002.3.

[14] Jurisica, I., & Wigle, D.A. (2002). *Understanding biology through intelligent systems.* Genome Biology, 3(11):reports4036.1-4036.4.

[15] Lee, Y., & Sultana, R., & Pertea, G., & Cho, J., & Karamycheva, S., & Tsai, J., & Parvizi, B., & Cheung, F., & Antonescu, V., & White, J., & Holt, I., & Liang, F., & Quackenbush, J. (2002). *Cross-Referencing Eukaryotic Genomes: TIGR Orthologous Gene Alignments (TOGA).* Genome Research, Vol. 12, Issue 3:493-502.

[16] Nascimento, M.A., & Silva, J.R.O. (1998). *Towards historical R-trees.* Symposium on Applied Computing. Proceedings of the 1998 ACM symposium on Applied Computing, pp. 235 - 240. ACM Press, New York.

[17] Madigan, M.T., & Martinko, J.M., & Parker, J. (1997). Brock Biology of Microorganisms, 8th ed. Prentice-Hall, Inc., New Jersey.

[18] Miller, R.C. (2002). *Lightweight Structure in Text.* PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. Published as CMU Computer Science technical report CMU-CS-02-134 and CMU Human-Computer Interaction Institute technical report CMU-HCII-02-103.

[19] Pandey, A., & Lewitter, F. (1999). *Nucleotide sequence databases: a gold mine for biologists.* Trends in Biochemichal Sciences, 24(7):276-280.

[20] Peri, S. et al. (2003). *Development of Human Protein Reference Database as an Initial Platform for Approaching Systems Biology in Humans.* Genome Research 13:2363-2371.

[21] Primrose, S.B., & Twyman, R.M. (2003). Principles of Genome Analysis and Genomics, 3rd ed. Blackwell Publishing, Malden.

[22] Saal, L.H., & Troein, C., & Vallon-Christersson, J., & Gruvberger, S., & Bort, Å., & Peterson, C. (2002). *BioArray Software Environment (BASE): a platform for comprehensive management and analysis of microarray data.* Genome Biology, 3(8):software00003.1-0003.6.

[23] Schulze, A. (2001). *Navigating gene expression using microarrays - a technology review.* Nature Cell Biology, 3:E190-E195.

[24] Sejnowski, T. (2001). Complexity in biological information processing - No. 239, pp. 1-3. John Wiley & Sons Ltd, Chichester.

[25] Stajich, J.E., & Block, D., & Boulez, K., & Brenner, S.E., & Chervitz, S.A., & Dagdigian, C., & Fuellen, G., & Gilbert, J.G.R., & Korf, I., & Lapp, H., & Lehvaslaiho, H., & Matsalla, C., & Mungall, C.J., & Osborne, B.I., & Pocock, M.R., & Schattner, P., & Senger, M., & Stein, L.D., & Stupka, E.D., & Wilkinson, M., & Birney, E.

(2002). *The Bioperl Toolkit: Perl modules for the life sciences.* Genome Research, Oct;12(10):1611-8.

[26] Staton, J.L. (2002). *Homology in Character Evolution.* Encyclopedia of life sciences, vol.9, pp. 196-202. Macmillan Publishers Ltd, Nature Publishing Group, London.

[27] Stein, L.D., & Mungall, C., & Shu, S., & Caudy, M., & Mangone, M., & Day, A., & Nickerson, E., & Stajich, J.E., & Harris, T.W., & Arva, A., & Lewis, S. (2002). *The generic genome browser: a building block for a model organism system database.* Genome Research, 12:1599-1610.

[28] Tamarin, R.H. (1996). Principles of genetics, 5th ed. Times Mirror Higher Education Group, Inc, Dubuque.

[29] Tatusov, R.L., & Koonin, E.V., &Lipman, D.J. (1997). *A Genomic Perspective on Protein Families.* Science, VOL. 278:631-637.

[30] Tatusov, R.L., & Natale, D.A., & Garkavtsev, I.V., & Tatusova, T.A., & Shankavaram, U.T., & Rao, B.S., & Kiryutin, B., & Galperin, M.Y., & Fedorova, N.D., & Koonin, E.V. (2001). *The COG database: new developments in phylogenetic classification of proteins from complete genomes.* Nucleic Acids Research, Vol. 29, No. 1:22-28.

[31] Widenius, M., & Axmark, D., & MySQL AB. (2002). MySQL Reference Manual. O'Reilly & Associates, Inc., Sebastopol.

[32] Wilkinson, M.D., & Gessler, D., & Farmer, A., & Stein, L. (2003). *The BioMOBY Project Explores Open-Source, Simple, Extensible Protocols for Enabling Biological Database Interoperability.* Proceedings of the Virtual Conference on Genomics and Bioinformatics:17-27.

Web references

[33] ftp://ftp.ncbi.nih.gov/genbank/gbrel.txt
Genetic Sequence Data Bank. (2004). NCBI-GenBank Flat File Release 143.0. (2004-08-26).

[34] http://base.thep.lu.se
BASE - BioArray Software Environment. (2004-08-26).

[35] http://base.thep.lu.se/documentation/faq/faq/node3.html
Häkkinen, J. (2003). BASE Frequently Asked Questions. (2004-08-26).

[36] http://ceolas.org/VL/mo
Manning, G. (2002). The WWW Virtual Library: Model Organisms. (2004-08-26).

[37] http://cvs.bioperl.org/cgi-bin/viewcvs/viewcvs.cgi/biosql-schema/doc
Chris Dagdigian. biosql-schema/doc. (2004-08-26).

[38] http://dev.mysql.com/doc/mysql/en/LOAD_DATA.html
MySQL AB. LOAD DATA INFILE Syntax. (2004-08-26).

[39] http://dli.grainger.uiuc.edu/national.htm
Habing. T. (1998). Digital Libraries Initiative. (2004-08-26).

[40] http://doc.bioperl.org/releases/bioperl-1.4/Bio/DB/GFF.html
Bio::DB::GFF – Storage and retrieval of sequence annotation data. (2004-08-26).

[41] http://flybase.net
FlyBase A Database of the Drosophila Genome. (2004-08-26).

[42] http://flybase.net:8081/gbrowse
Stein, L.D. (2003). Generic Genome Browser. (2004-08-26).

[43] http://flybase.net:8081/gbrowse/docs/pod/CONFIGURE_HOWTO.html
Stein, L.D. & The GMOD development team. CONFIGURE-HOWTO. (2004-08-26).

[44] http://katahdin.cshl.org:9331/lwgv
Faith, J., & Katari, G., & Sachidanandam, R. lwgv. (2004-08-26).

[45] http://katahdin.cshl.org:9331/lwgv/lwgvShots.html
Lwgv screenshots. (2004-08-26).

[46] http://lwgv.sourceforge.net/lwgv.pdf
Faith, J., & Sachidanandam, R. (2003). light weight genome viewer. (2004-08-26).

[47] http://obda.open-bio.org
(2002). About OBDA. (2004-08-26).

[48] http://pear.php.net
The PHP Group. (2004). PEAR - PHP Extension and Application Repository. (2004-08-26).

[49] http://pear.php.net/package/DB
The PHP Group. (2004). Package Information: DB. (2004-08-26).

[50] http://perl.apache.org
(2004). Welcome to the mod_perl world. (2004-08-26).

[51] http://search.cpan.org/src/BIRNEY/bioperl-1.2.3/scripts/Bio-DB-GFF/bp_genbank2gff.pl
(2003). bp_genbank2gff.pl. (2004-08-26).

[52] http://search.cpan.org/src/BIRNEY/bioperl-1.4/Bio/DB/GFF.pm
(2003). GFF.pm. (2004-08-26).

[53] http://sourceforge.net/projects/basedb
(2004). Project: BASE: Summary. (2004-08-26).

[54] http://www-db.stanford.edu/tsimmis
Rys, M. (1998). TSIMMIS. (2004-08-26).

[55] http://www.apache.org
The Apache Software Foundation. (2004). The Apache Software Foundation. (2004-08-26).

[56] http://www.bio.umass.edu/biology/kunkel/modelsys.html
Kunkel, J. (1996). What Makes a Good Model System?. (2004-08-26).

[57] http://www.biodas.org/documents/spec.html
Stein, L.D., & Eddy, S., & Dowell, R. (2002). Distributed Sequence Annotation System (DAS). (2004-08-26).

[58] http://www.biomoby.org
moby. (2004-08-26).

[59] http://www.bioperl.org
bioperl.org. (2004-08-26).

[60] http://www.bosrup.com/web/overlib
Bosrup, E. (2004). What is overLIB. (2004-08-26).

[61] http://www.ddbj.nig.ac.jp
DDBJ. (2004). DNA Data Bank of Japan. (2004-08-26).

[62] http://www.ebi.ac.uk/embl
EBI Support. (2004). EMBL Nucleotide Sequence Database. (2004-08-26).

[63] http://www.ebi.ac.uk/embl/Submission/genomes.html
EBI Support. (2004). EMBL Genome Project Submission Account guidelines. (2004-08-26).

[64] http://www.ebi.ac.uk/Information
EBI Support. (2004). EMBL-EBI European Bioinformatics Institute . (2004-08-26).

[65] http://www.expasy.ch/sprot
ELG. (2004). Swiss-Prot Protein knowledgebase TrEMBL Computer-annotated supplement to Swiss-Prot. (2004-08-26).

[66] http://www.gmod.org
Cain, S. Generic Model Organism Database Construction Set. (2004-08-26).

[67] http://www.gmod.org/ggb/INSTALL.html#generic_genome_browser_installation
Stein, L. & The GMOD team. Generic Genome Browser Installation. (2004-08-26).

[68] http://www.gmod.org/schema
Generic Model Organism Database Construction Set. (2004-08-26).

[69] http://www.icsi.berkeley.edu/~ssinha/papers/Tech_CS262A_-_Spatial_Indexing_DNA.pdf
Harmon, C., & Sinha, S., & Yue, M. Spatial Indexing of DNA Sequence Feature Data. (2004-08-26).

[70] http://www.informatics.jax.org/mgihome//projects/overview.shtml#aboutMGD
Mouse Genome Database Overview. (2004-08-26).

[71] http://www.iscb.org/ismb2003/posters/hlappATgnf.org_326.html
Lapp, H., & Mungall, C., & Cain, S., & Stein, L. Optimizing Genome Interval Overlap Queries Using an R-Tree Index. (2004-08-26).

[72] http://www.mysql.com
MySQL AB. (2004). MySQL. (2004-08-26).

[73] http://www.ncbi.nlm.nih.gov
NCBI. National Center for Biotechnology Information. (2004-08-26).

[74] http://www.ncbi.nlm.nih.gov/BLAST/fasta.shtml
NCBI. FASTA format description. (2004-08-26).

[75] http://www.ncbi.nlm.nih.gov/COG
NCBI. Phylogenetic classification of proteins encoded in complete genomes. (2004-08-26).

[76] http://www.ncbi.nlm.nih.gov/COG/new/release/coglist.cgi
NCBI. List of COGs. (2004-08-26).

[77] http://www.ncbi.nlm.nih.gov/COG/new/release/cogvars.cgi
NCBI. COGs Functional Annotation. (2004-08-26).

[78] http://www.ncbi.nlm.nih.gov/Sitemap/#GenBank
NCBI. GenBank General Information. (2004-08-26).

[79] http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html
NCBI. (2004). Sample GenBank Record. (2004-08-26).

[80] http://www.nhlbi.nih.gov/meetings/modeldb
NHLBI. Report of the NHI Model Organism Database Workshop. (2004-08-26).

[81] http://www.php.net
The PHP Group. (2004). What is PHP?. (2004-08-26).

[82] http://www.phpdoc.org
Eichorn, J. (2004). phpDocumentor: The complete documentation solution for PHP. (2004-08-26).

[83] http://www.phpmyadmin.net/home_page
phpMyAdmin devel team. (2004). The phpMyAdmin Project. (2004-08-26).

[84] http://www.r-project.org
(2004). The R Project for Statistical Computing. (2004-08-26).

[85] http://www.sanger.ac.uk/Software/formats/GFF
The Wellcome Trust Sanger Institute. (2003). GFF: an Exchange Format for Feature Description. (2004-08-26).

[86] http://www.sanger.ac.uk/Software/formats/GFF/GFF_Spec.shtml
Durbin, R., & Haussler, D., with amendments proposed by: Stein, L., & Lewis, S., & Krogh, A. (2003). GFF (General Feature Format) Specifications Document. (2004-08-26).

[87] http://www.tigr.org
The Institute for Genomic Research. (2004). TIGR The Institute for Genomic Research. (2004-08-26).

[88] http://www.tigr.org/tdb/toga/toga.shtml
TIGR. (2002). Eukaryotic Gene Orthologs. (2004-08-26).

[89] http://www.uni-giessen.de/~gx1052/IECA/ieca.html
Kroeger-Block, A. (2004). IECA International E.coli Alliance E.coli Database Portal. (2004-08-26).

[90] http://www.w3.org/TR/REC-CSS1
W3C. (1991). Cascading Style Sheets, level 1. (2004-08-26).

[91] http://www.w3.org/TR/REC-CSS2
W3C. (1998). Cascading Style Sheets, level 2 CSS2 Specification. (2004-08-26).

[92] http://www.w3.org/TR/REC-html40/present/frames.html
W3C. (2004). Frames. (2004-08-26).

A Appendix

A.1 User manual

This manual describes in short the system requirements and the steps which are needed to use or reconfigure the system which is used in the current solution.

A.1.1 System requirements

The solution in question has only been tested in the presence of the following components:
Apache HTTP server version 2.0.
PHP version 4.3.3.
Generic Genome Browser version 1.61.
BASE version 1.2.12.
Bioperl version 1.4
MySQL distribution 4.0.16 version 12.22.

For more information on these components, see [55, 81, 66, 34, 59, 72] respectively.

A.1.2 System files

The base_gbrowse_system directory contains almost all the files needed for the current solution, and can be placed in for instance the public_html directory of a user. The file connectionInfo.inc.php is to be placed two directories up in the directory tree, since it contains possibly sensitive information about database users and passwords needed by the system. The functionality of the system also depends on certain code being present in the GBrowse configuration file for the datasource(s), and on previously described modifications to some of the BASE source files.

A.1.3 Using the system

To display a reporter in the reporter list in BASE within its genomic context, click the link in the Accession column for that reporter. This opens up a GBrowse window where, if the start and end positions for that reporter are correctly given in BASE, the features of the region around this reporter within the given reference sequence are displayed. If one or both of the start or end position for the clicked reporter is invalid, for instance negative or non existent in BASE, but the reference sequence for that reporter in BASE is valid and exists in the GBrowse database, then the start of the reference sequence (the first 5000bp) will be displayed in the GBrowse window.

To get a summary of the information that is present in the GBrowse database for a certain feature in the GBrowse view, click on that feature in the ”Named gene” track. This opens up a new window containing this information.

Placing the marker over a feature in the "CDS" track displays a popup menu with the name of the feature as title and two links below the name. The popup menu is "sticky" so when the mouse leaves the area it remains open until one moves the mouse in and then out of the area of the displayed menu. One of the links is used to check for orthologues for the given feature. If no orthologues are found when clicking this link a message window is displayed, explaining that no orthologues were found. If orthologues are found a checklist with the found orthologues is displayed, in which one can check the orthologues one wants to view and press the button to view them. Currently the maximum number of orthologues to be displayed is set to 4 in the handleOrthreq.phtml file, see section A.1.4 on page 64 on how to change this setting. If the other link, for retrieving expression data from BASE, is clicked this currently displays the reporterid, reference sequence, and start and stop position for all reporters that are contained within or overlap the region of the selected feature. This is to be changed in the future when the expression data interface mentioned in section 5.5 is released, for more information on how to do that see section A.1.4 below.

To update an existing GBrowse database with new data new data the bp_load_gff.pl Bioperl script can be used as follows

```
bp_load_gff.pl --accession NC_0002695 --stdout > test.gff.
```

If one wants to load data into an empty database using the Bio::DB::GFF schema or if one wants to start from scratch the bp_bulk_load_gff.pl Bioperl can be used, note that this will create a new schema deleting anything that was there before. For instance to load the file myGFFfile.gff into the myDatasource database do

```
bp_bulk_load_gff.pl --user <dbUser> --password <dbPassw> -d myDatasource
myGFFfile.gff
```

from within the directory where the .gff file is. Before data can be loaded it needs to be available in the GFF flat file format. To convert public or locally generated Genbank files the bp_genbank2gff.pl script can be used. If one fetches the Genbank file directly from the NCBI website [73], for instance the file with accession NC_0002695 one can use

```
bp_genbank2gff.pl --accession NC_0002695 --stdout > filename.gff
```

to convert the file into GFF format. If one wants to save the Genbank file first to the system and then convert it locally, or if one has a local Genbank file that needs to be converted the script can be used as follows

```
bp_genbank2gff.pl -f filename.gb --stdout > filename.gff.
```

Remember that the Genbank file to be loaded has to end on .gb. The bp_genbank2gff.pl script automatically converts the sequence, if that is given in the Genbank file to be converted. The converted file will then be of GFF3 format, which combines feature data with sequence/DNA data [22]. If one wants to load the sequence at a later time though, the bp_load_gff.pl script mentioned above can load DNA that is present in the FASTA file format [67], by using the -fasta option. An example of this syntax on a Unix system is

```
bp_load_gff.pl -d myDatabasename -fasta myFastaFilename.fasta.gz </dev/null.
```

For more information about how the above mentioned Bioperl scripts work see [43, 67, 51, 22].

For each datasource in GBrowse a corresponding configuration file in the gbrowse.conf directory is needed. If a new database has been created and loaded for use with GBrowse, one of the existing configuration files can be copied and modified as fit. The configuration file must be renamed in the following form

```
sourcename.conf
```

, where sourcename is a short word that describes the data source and can be used for selecting the datasource when linking to the browser, for instance by

```
http://your.site.org/cgi-bin/gbrowse/sourcename.
```

Be sure to also enter the appropriate adaptor and adaptor arguments in the beginning of the GENERAL section of the configuration file, see also section 4.2.2 on page 18 and [43] for more information on these configuration options.

Note that there will not be any links available in the Accession column in BASE for those reporters that do not have an accession given in the BASE database. The accession for a reporter is to be set to the accession number of the Genbank file that contains the whole reference sequence for that reporter. This is done in the same text file that is used for loading other Reporter table column data into BASE for different experiments. For instance, if a reporter belongs to chromosome2 in a genome, and the Genbank file with accession number NC_000111 contains chromosome2, then NC_000111 will become the reference sequence name for all features belonging to that chromosome in the GBrowse database. The value of the accession column in BASE for reporters belonging to this chromosome should then also be set to NC_000111. For reporters with lacking or invalid start and end positions, there will still be a link in BASE if they have a value for the accession column. However when clicked the GBrowse view will display the genome from the beginning of the reference sequence since the position of that reporter is not known. Some modifications to the files reporter_columns.inc.php and reporter_list.phtml in BASE also need to be done to be able to produce the links.

A.1.4 Reconfiguring the system

A list of examples of interesting regions can be specified in the GBrowse configuration file for the datasource in question. This is done through the examples option in the GENERAL section, the value of which is set to a space delimited list of interesting regions. The ones given in the current solution are only there as an example of the syntax and should be changed.

If new database connections are to be used within the system, or if connection information for present database connections are to be altered, changes need to be done in the connectionInfo.inc.php file. The comments in the code provide an example of how a new connection can be added.

New links to the JavaScript popup menu can be added in the .conf file for the datasource one is working with, located in the gbrowse.conf directory, see [43] for details. The new links are added in the TRACK section of the configuration file, more specifically in the Perl subroutine given for the link option of the CDS track.

The maximum number of orthologues to be displayed under each other is currently set to 4 in the handleOrthreq.phtml file. To change this setting change

```
\$maxOrthsDispl = 4;
```

to the desired number.

To later on connect to the interface for viewing expression data from BASE, mentioned in section 6.3 on page 48, the BASE reporter ids retrieved in the handleBaseDatareq.php file need to be sent as parameters to a URL instead of being displayed, as is currently done. The comments in that file contain a detailed description on how one might do that, but the exact syntax for how this interface accepts parameters through a URL will need to be checked by the users of the system when the interface has been publicly released. It would also be possible to have an intermediary step where the reporter ids were displayed in a checklist where one could choose which of the retrieved reporter ids to display expression data for, before sending them as parameters to the expression data interface. This however, would demand a bit more changes in the code, but the handleOrthreq.phtml, which handles the orthologue checklist, could be used as a template.

If the GBrowse database is to be changed from a MySQL to an Oracle database, the adaptor arguments in the GBrowse configuration file for that datasource need to be changed as follows

```
db_args = -adaptor dbi::oracle
          -dsn dbi:oracle:database=db_service
```

where db_service is replaced with the name of the desired database service definition [43], see the Perl dbd::Oracle database driver for more information about the -dsn format.