BMC Bioinformatics

BioMed Central

Methodology article

FOUNTAIN: A JAVA open-source package to assist large sequencing projects

Jean-Marie Buerstedde* and Florian Prill

Address: Department of Cellular Immunology, Heinrich-Pette-Institute, Martinistr. 52, 20251 Hamburg, Germany

E-mail: Jean-Marie Buerstedde* - buersted@genetics.hpi.uni-hamburg.de; Florian Prill - fprill@gmx.de

*Corresponding author

Published: 21 September 2001

BMC Bioinformatics 2001, **2**:6

Received: 12 July 2001

Accepted: 21 September 2001

This article is available from: http://www.biomedcentral.com/1471-2105/2/6

Abstract

Background: Better automation, lower cost per reaction and a heightened interest in comparative genomics has led to a dramatic increase in DNA sequencing activities. Although the large sequencing projects of specialized centers are supported by in-house bioinformatics groups, many smaller laboratories face difficulties managing the appropriate processing and storage of their sequencing output. The challenges include documentation of clones, templates and sequencing reactions, and the storage, annotation and analysis of the large number of generated sequences.

Results: We describe here a new program, named FOUNTAIN, for the management of large sequencing projects [http://genetics.hpi.uni-hamburg.de/FOUNTAIN.html] . FOUNTAIN uses the JAVA computer language and data storage in a relational database. Starting with a collection of sequencing objects (clones), the program generates and stores information related to the different stages of the sequencing project using a web browser interface for user input. The generated sequences are subsequently imported and annotated based on BLAST searches against the public databases. In addition, simple algorithms to cluster sequences and determine putative polymorphic positions are implemented.

Conclusions: A simple, but flexible and scalable software package is presented to facilitate data generation and storage for large sequencing projects. Open source and largely platform and database independent, we wish FOUNTAIN to be improved and extended in a community effort.

Introduction

Over the last two decades, the automation of template preparation, sequencing reaction and sequence data collection greatly facilitated large scale sequencing projects. Most modern sequencers have the capacity to analyze at least 96 reactions per day using either gel or capillary electrophoresis systems. The increased sequencing capacity was matched by a decrease in the price of a single sequence reaction, leading to an almost exponential increase in the monthly world wide output of new sequences [http://www.ncbi.nlm.nih.gov:80/] .

Although these advances culminated in the determination of the whole genome sequence of several model organisms [1–4] including the human genome, the overall sequencing activity has not slowed down significantly. Free capacities are now being shifted to document sequence variation within species as well as between species [5] and produce a complete catalog of all expressed

sequences [6]. In addition, new applications for large scale sequencing in connection with mutant or two hybrid screens arise [7,8].

The impressive increase in sequencing efficiency means that the bottle neck for a large scale sequencing project is often not the accumulation of the raw data, but their appropriate storage and analysis. This is particularly true for laboratories outside of genome resource centers which may possess valuable expertise to address biological questions but lack experience in computer science.Although good sequence assembly programs like the Staden package [9] are available, an free and open-source program to manage the storage of the sequences in a database is lacking. To ameliorate this situation we have written a software package named FOUNTAIN to assist sequence data collection, storage, and analysis. By choosing JAVA [http://java.sun.com/] as the programing language and channeling all access to the relational database through the JDBC application interface (API), the code is largely platform and relational database management system independent. Structured in packages and centered around JAVA interactions with the database, FOUNTAIN should be easy to understand and to improve as a community effort. We are for example currently adding a work package for the storage and presentation of micro array expression data.

Results and Discussion

The overall design

A number of design options were pondered before starting the FOUNTAIN project. The highest priority was to make the code easy to understand, to maintain and to extend even by a novice programer. Other objectives were platform and database independence and reliance on open source solutions whenever feasible.

Important decisions were the choice of the programming language and the data storage system. We initially experimented with the PERL language [http://www.perl.com/] , but then opted for using JAVA due to inherent object orientation, platform independence and superior debugging features.

We also became convinced that a mature relational database management system (RDMS) is needed to assure correct data storage, even if this introduces additional complexity for database installation and maintenance.

The user interface was based on HTLM forms, although applets would have allowed more elegant graphics and faster error checking. However, it was felt that this would make the code more complex and introduce dependence on the JAVA version of the web browsers.

The combination of JAVA and a relational database gives FOUNTAIN the advantage of being easy to maintain and scalable. No additional skills are needed apart from a basis knowledge of JAVA, SQL and the administration of a relational database, and even a novice programer should be able to customize the software. This is perhaps best explained by the fact that the main author of FOUNTAIN is himself a biologist, who only recently started programming.

The network connections

All parts of FOUNTAIN – the database, the web server and the JAVA classes – can be installed on the same computer. However FOUNTAIN is well suited for installation within a network as all connections to the database rely on the JDBC API and web browsers.

Fig. 1 shows the configuration of the network which was used for the Bursal EST sequencing project [10]. We preferred Oracle8i over MySql as the RDMS in the intranet due to the availability of foreign key constraints and transactional control. In our hands, Oracle8i is not difficult to install and to run on Linux and the fee for an intranet license is moderate. The Sequence, Cluster and Polymorphism work packages are presented on the internet web server using MySql as the RDMS.

Definition of the tasks and work packages

FOUNTAIN was developed to handle the following tasks for large scale sequencing projects: 1) Generation and storage of data related to libraries, clones, templates, primers and sequence reactions. 2) Import and annotation of sequence data. 3) Clustering of sequences. 4) Definition of single nucleotide polymorphism (SNIP) or sequencing errors. 5) Storage and retrieval of user defined variables.

FOUNTAIN is divided into different work packages based on these tasks. This division is reflected in the JAVA package and sub package structure and in the table structure (schema) of the database. In this way the code of one work package can be easily modified without affecting the functionality of the other work packages as long as the foreign key restrictions of the database scheme are respected. In addition, it is possible to simplify or exchange entire work packages after deletion or modification of the foreign key restrictions.

The BLAST algorithm as a work horse

The Sequence, Cluster and Polymorphism work packages rely on homology searches of the database sequences against the public databases or against each other. The most popular and tested algorithm for this task is the BLAST algorithm [11]. Advantages of BLAST are that 1) searches against the latest version of the public databas-

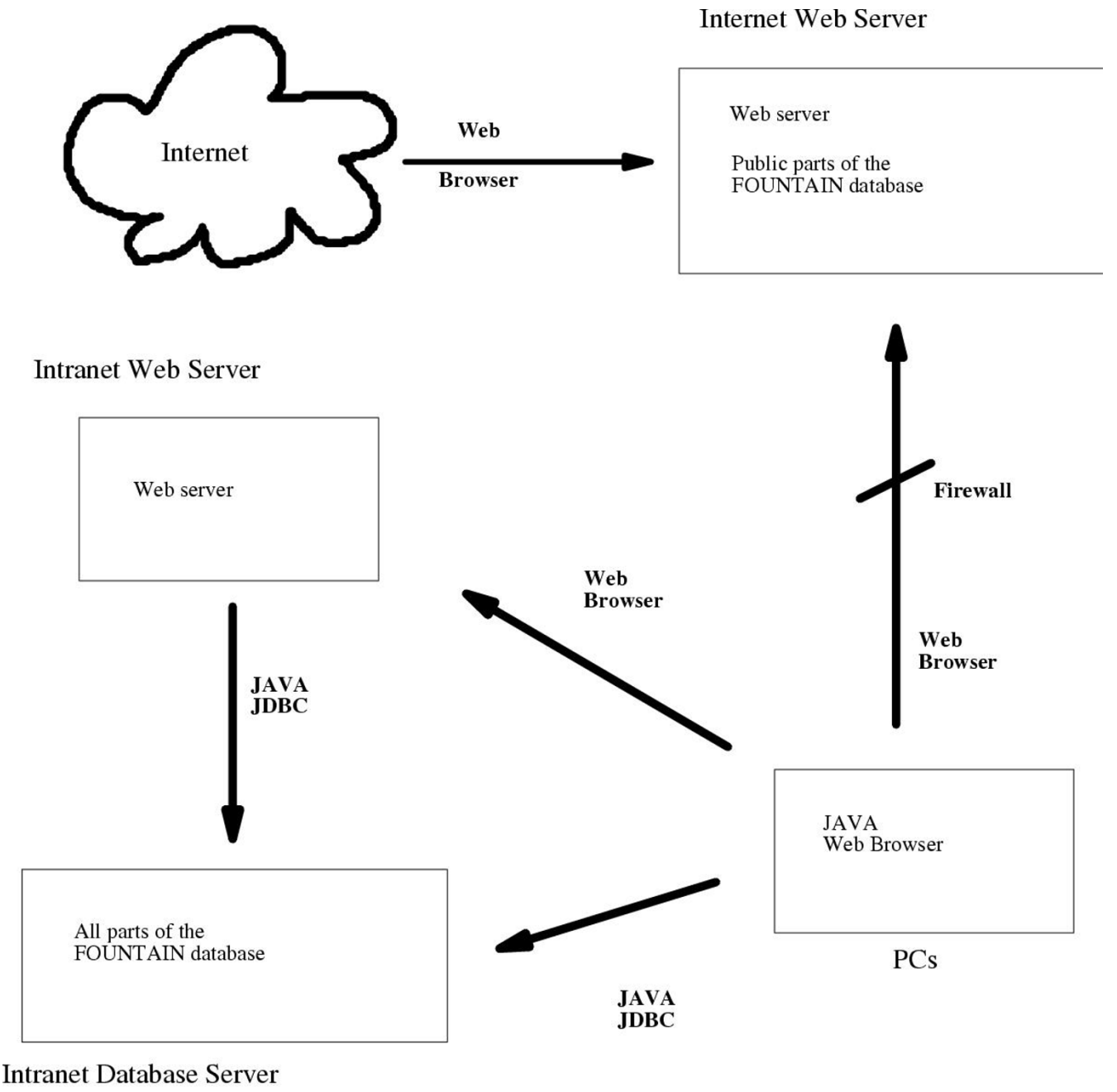

Figure 1
Possible network configuration for FOUNTAIN.

es can be done over the internet, 2) the output format is structured and can be easily parsed and 3) further information about the homologous query sequences can be obtained by following URLs to the NCBI server [http://www.ncbi.nlm.nih.gov:80/entrez] . For these reasons, the FOUNTAIN work packages heavily rely on BLAST search analysis.

The Clone work package

The Clone work package generates and stores information prior to the import of the sequencing data. The tables shown in Fig. 2 and all packages starting with 'fountain.genetics.clone' belong to this work package. A particular sequencing project is presented by an entry in the 'LibrarySet' table. A 'LibrarySet' can include one or more entries in the 'Library' table which might corre-

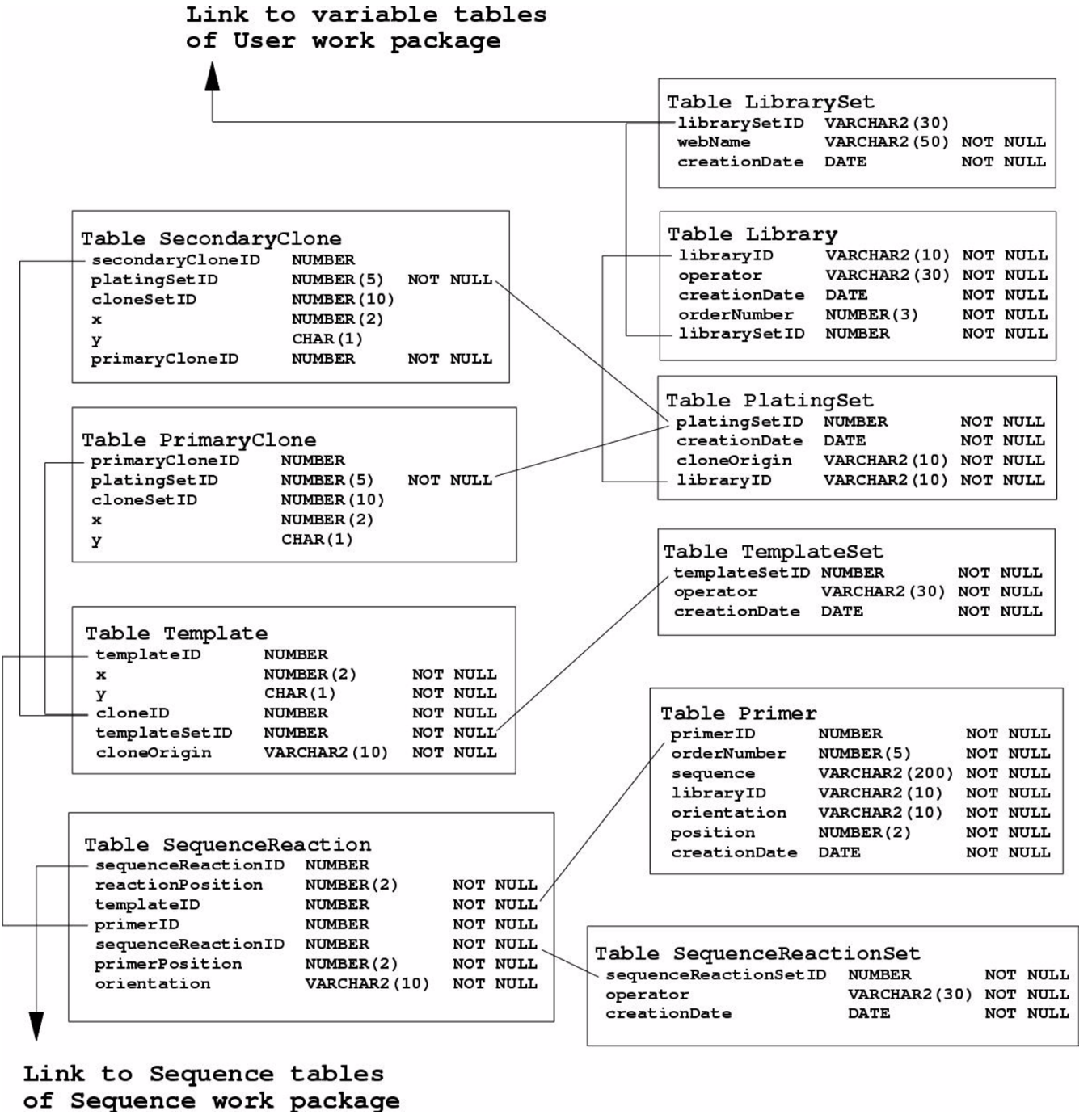

Figure 2
The tables of the Clone work package. Primary – foreign key relationships are indicated by lines between columns.

spond to different cDNA libraries. Any name can be used for the library, but it should not contain '_' or '.' characters, which interfere with parsing of the sequence names. When clones from these libraries are transferred into plates to make a permanent stock, this is documented by an entry in the 'PlatingSet' table and entries for each clone in either the 'PrimaryClone' or the 'SecondaryClone' tables. The 'SecondaryClone' table is needed, if one wants to sequence a subset of replated primary clones. This can for example be used, when primary clones containing inserts of already known genes are identified within a 'PlatingSet' and the remaining clones

are transferred into new microtiter plates for further sequencing. The meaning of the other tables should be clear from the table and column names.

The program is flexible in that sets of sequencing reactions can be either produced batch wise or individually and the order of the sequencing reaction can reflects manual loading of 96 well gels or analysis by a capillary sequencer. Automatic primer design to extend already available sequences is also implemented and takes into account the sequence quality as judged from the PHRED scores.

Although not reflected in the generic database scheme, a number of assumptions has been made in the JAVA code of the Clone work package: 1) The clones are stored in 384 format plates. 2) Four sets of 96 template reactions are derived from each clone plate. 3) Only a single template and a single sequencing reaction is present for each primary or secondary clone.

The last assumption is needed to permit a simple and unambiguous assignment of clones to templates, templates to sequencing reactions and sequencing reactions to sequences. If different versions of template and sequencing reactions were allowed, one would need to indicate the version name. This would make the names and the name assignments far more complicated without being of much value for the end user of the sequence.

The mentioned assumptions were useful for the Bursal EST sequencing project, allowing data entry with minimal user input. However changes may be needed for other sequencing projects. As the assumptions are not fixed in the database scheme, modifications can be introduced to make the code more flexible and future versions of FOUNTAIN may address this issue.

The Sequence work package

The Sequence work package (Fig. 3) includes all classes and sub packages of 'fountain.genetics.sequence'. It performs the tasks to import the sequencing data in form of nucleotides and scores into the database and to annotate the sequences based on BLAST searches against the public sequence databases. To improve performance of the database, the data of each 'LibrarySet' are stored in separate tables owned by the 'LibrarySet' owner.

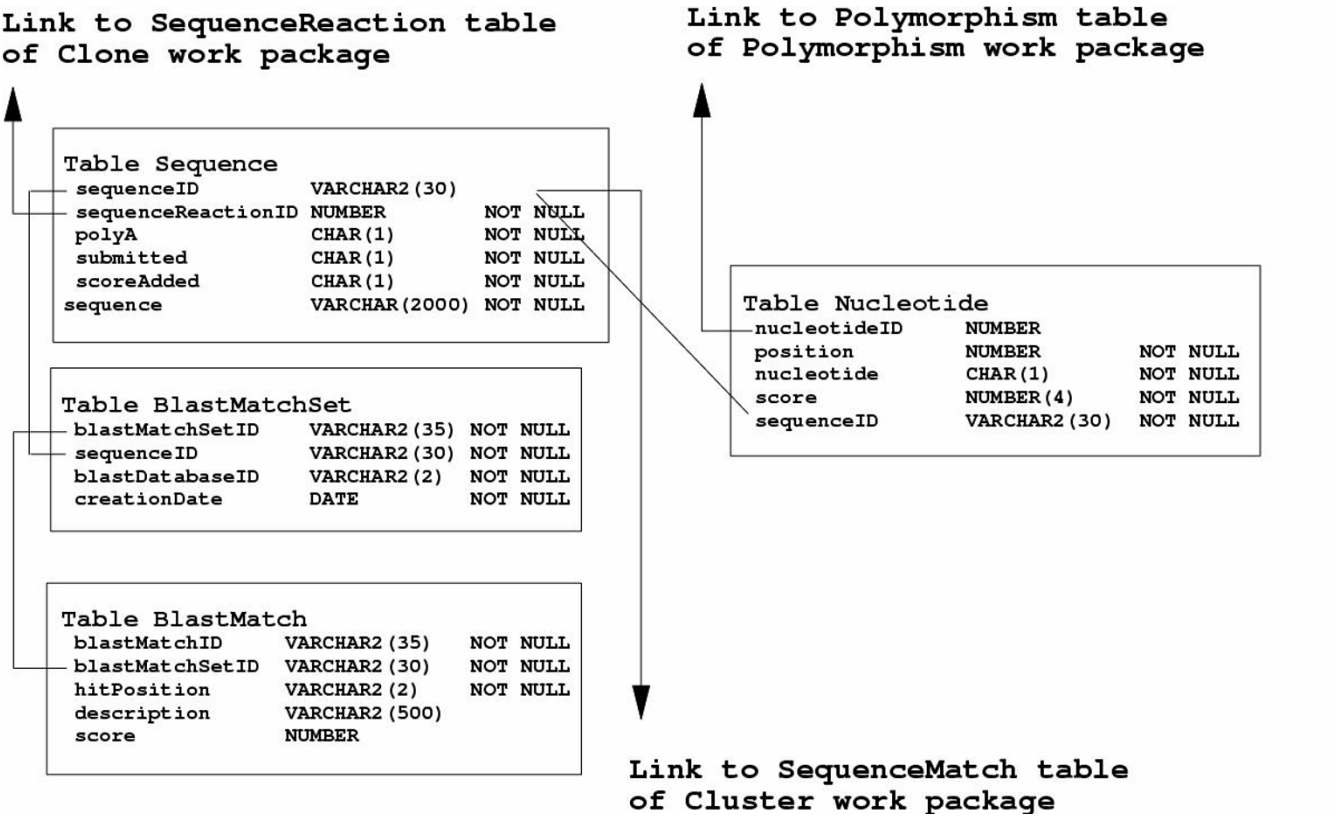

Figure 3
The tables of the Sequence work package. Primary – foreign key relationships are indicated by lines between columns.

The Sequence work package is linked to the Clone work package by using the primary key of the 'SequenceReaction' table as a foreign key in the 'Sequence' table. The primary key of the 'Sequence' table reflects the name of the DNA sequence. To keep the sequence name user friendly, it is not numeric and includes the library name, the clone plate number and the x- and y-coordinates as well as the position and the orientation of the primer used for sequencing.

The annotation of each sequence is based on the descriptions of the most homologous subject sequences in the public databases. The idea behind this is, that even if the gene from which the sequence was derived is not in the public domain, homologous sequences from other species should provide hints to its function. This annotation system has the advantage that it is fully automated and the annotations can be easily updated by extracting subject sequence descriptions from new BLAST searches. The name of the query sequence and the public database against which the search was performed can be derived from the 'blastMatchSetID' of the 'BlastMatch' table. The description of the query sequence, the hit position and the score of the BLAST match are stored in separate columns of the same table.

It was considered worthwhile to add a measure for the quality of the base calls to the 'Nucleotide' table. The most commonly used program to assign quality scores to base calls using sequence chromatogram trace files is the PHRED program from the University of Washington [12]. Working with the Bursal EST database we faced the problem that our sequences were independently derived from the chromatogram files and did not completely match the PHRED derived sequences. It was therefore decided to perform BLAST searches of our sequences against the database of PHRED derived sequences and assign the PHRED score based on the alignment of the corresponding sequences. This procedure worked well and may be of generic value for similar problems.

The Cluster work package

The Cluster work package (Fig. 4) including all classes and sub packages of 'fountain.genetics.cluster' generates and stores information about possible sequence and gene clusters. This is particularly useful for EST sequencing projects where this information can be used to estimate the redundancy of the database sequences.

The program determines the maximal length of identical sequence overlap between the sequences by analysis of the BLAST searches of each sequence against all other sequences of the same 'LibrarySet'. If the length of overlap between the query and subject sequence exceeds a certain user defined limit, this is taken as evidence that both sequences belong to the same sequence cluster. To reduce false clustering due to the presence of repetitive elements, these elements are masked in the sequences before the BLAST searches are performed.

Sequence which do not overlap may still be derived from different parts of the same gene. To define such gene clusters, sequence clusters whose members showed high BLAST scores to a sequence in the public databases which is derived from the same species, were combined.

Although these clustering methods are simple and may produce a certain percentage of false positives and false negatives, they proved very useful for grouping the over 15 000 ESTs of the Bursal EST database into about 8000 distinct sequence clusters [http://genetics.hpi.uni-hamburg.de/dt40Est.html] .

The Polymorphism work package

The Polymorphism work package (Fig. 5) tries to define possible polymorphism or errors in the sequences. It includes all classes and sub packages of 'fountain.genetics.polymorphism'.

The analysis relies on a comparison of the alignment between a database sequence and a likely orthologous sequence in the public database. BLAST matches were considered to be between likely orthologues, if the query sequence and subject sequence in the public database belong to the same species and the score exceeds a user defined limit. The link to the accession number of the subject sequence is subsequently followed to determine whether the change occurs in the coding region and results in an amino acid change.

Graphical user interfaces

Many of the FOUNTAIN commands process files and table entries batch wise with no user input and the software does not include sophisticated graphical user interfaces. However HTML based forms and servlets play an important role for the following tasks:

1) Addition of connection parameters or changes of user defined variables. This information is stored in the variable tables of the User work package (Fig. 6). The correct settings of the connection variables is critical, since these variables are retrieved by all connections classes belonging to the Sequence, Cluster and Polymorphism work packages. Management of the connection parameter and the user defined variables is done by the WebUser servlet. This servlet also allows to add LibrarySet and Library entries to the tables of the Clone work package.

2) Addition of PlatingSets, ClonePlates, TemplateSets, Primers and SequencingReactions to the tables of the

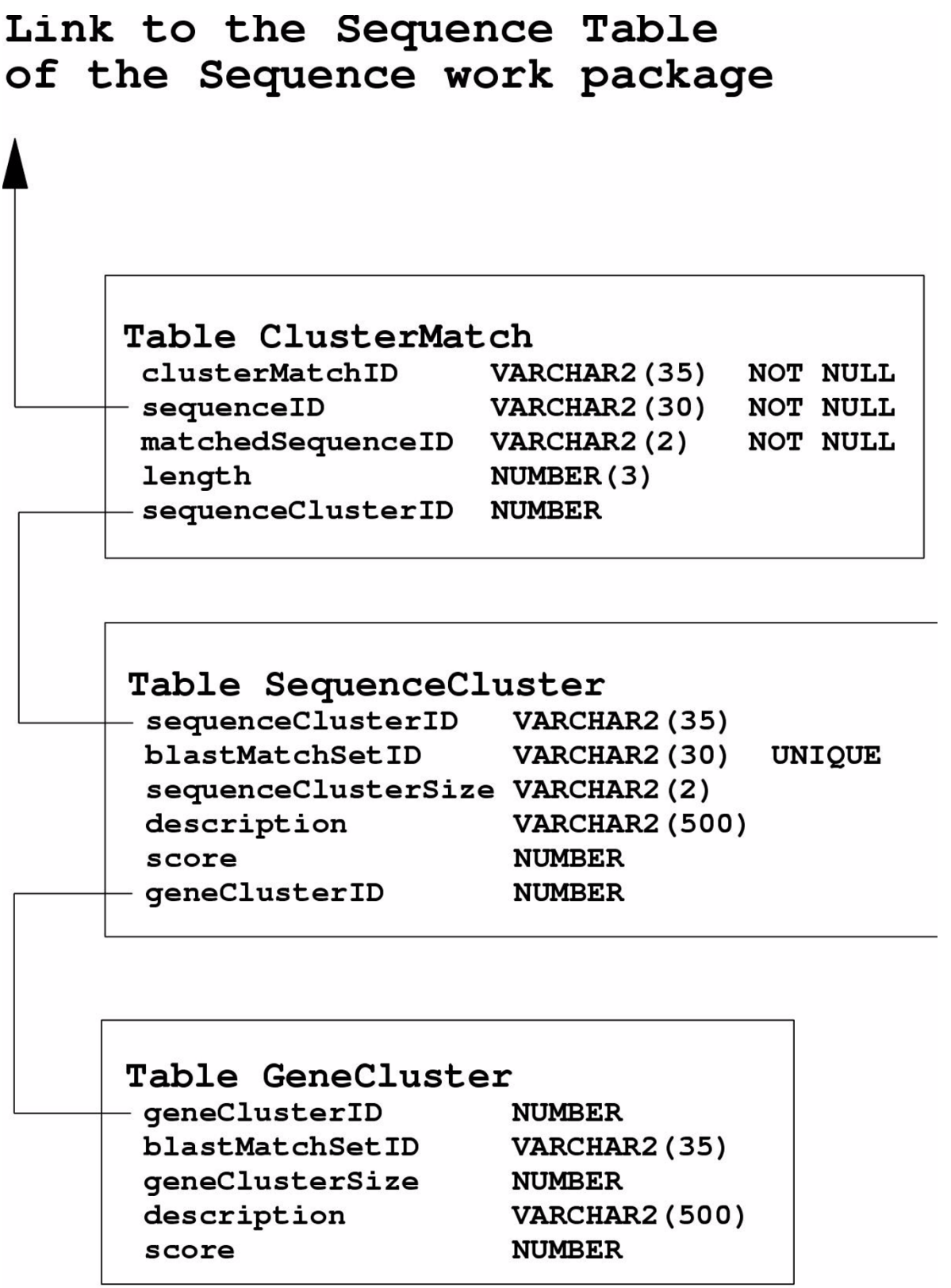

Figure 4
The tables of the Cluster work package. Primary – foreign key relationships are indicated by lines between columns.

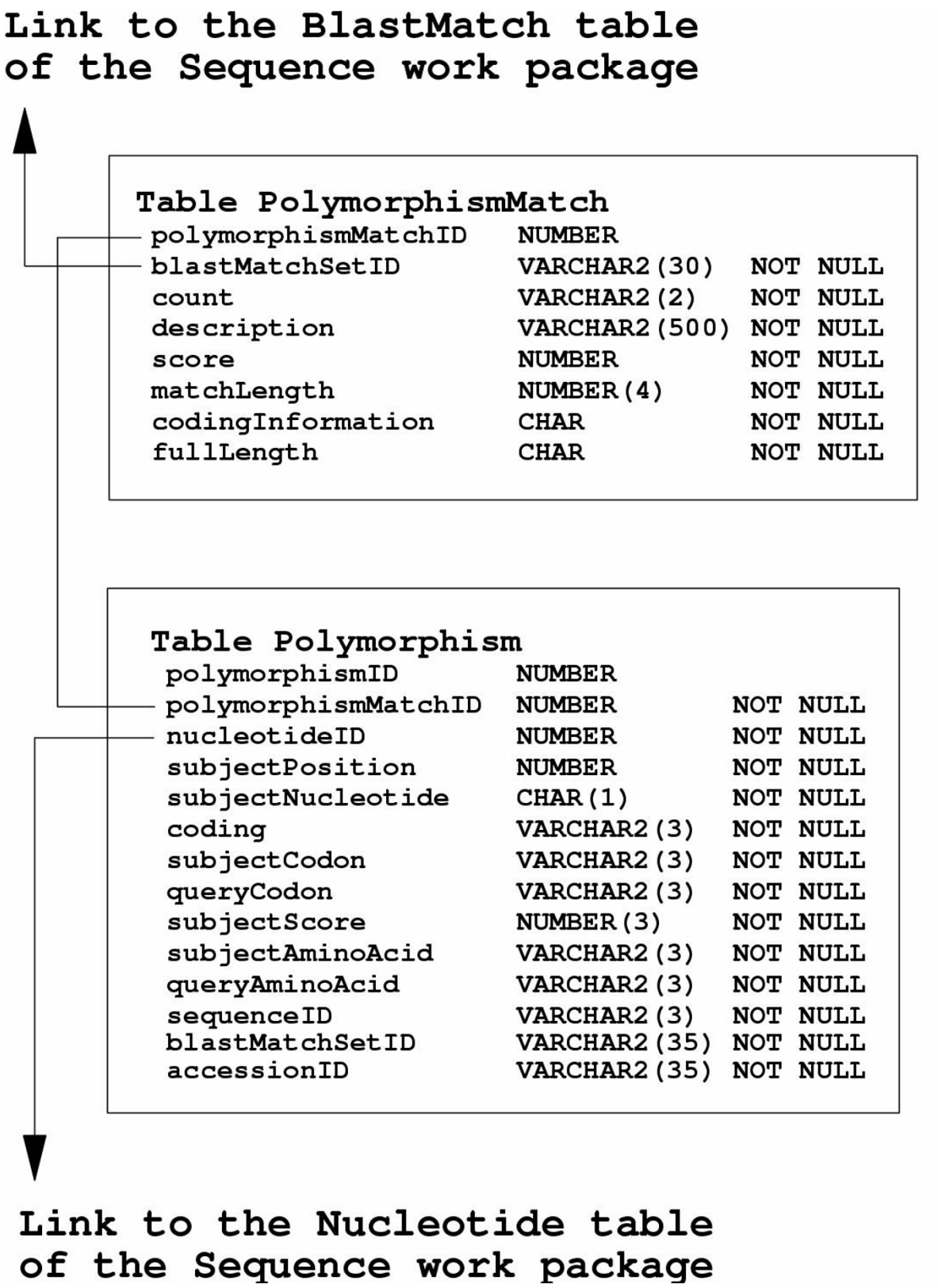

Figure 5
The tables of the Polymorphism work package. Primary – foreign key relationships are indicated by lines between columns.

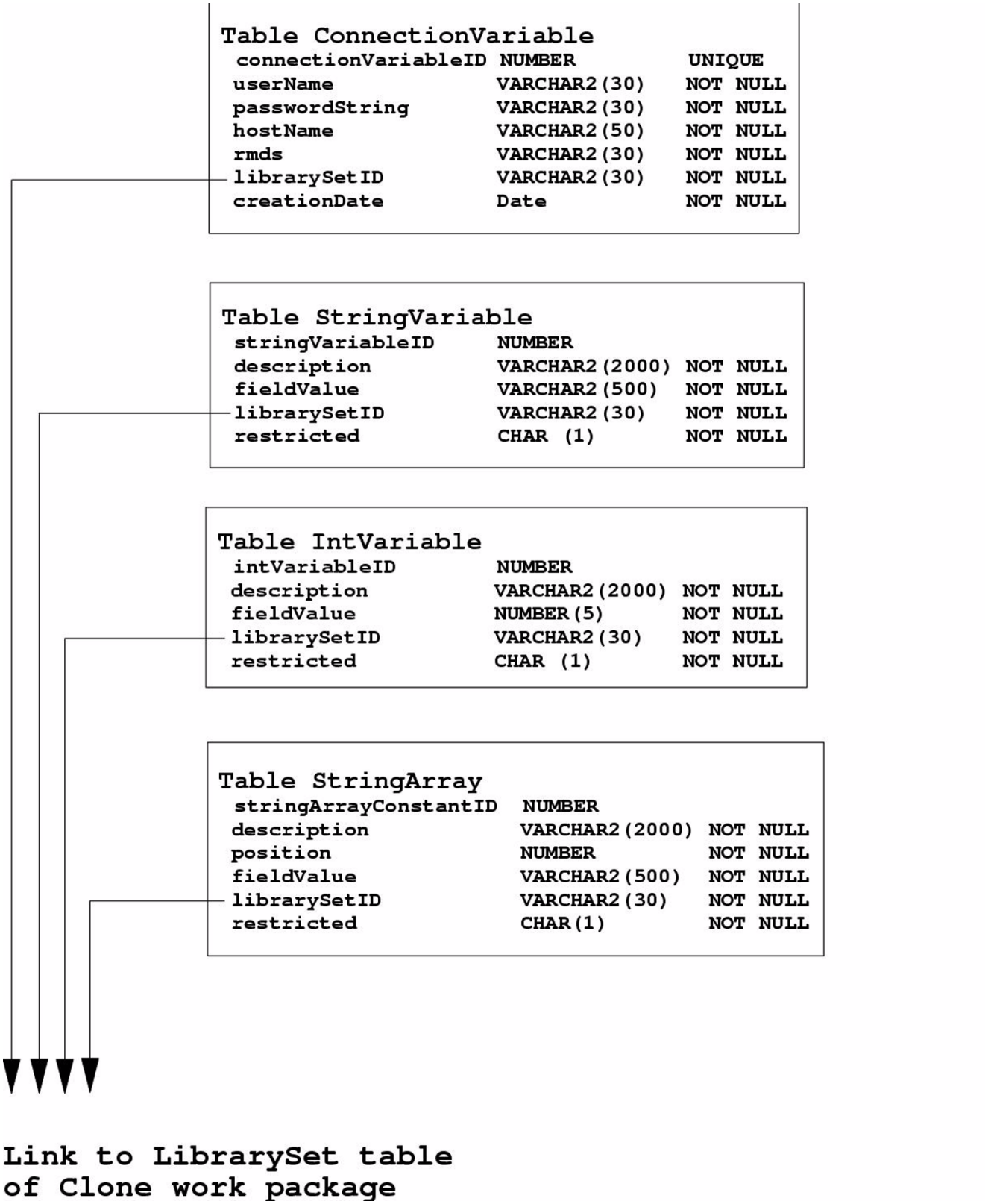

Figure 6
The tables of the User work package. Primary – foreign key relationships are indicated by lines between columns.

Clone work package. This is done batch wise for all PlatingSets by the WebClone servlet. Additions of randomly chosen SequencingReactions for a particular LibrarySet is implemented by the WebGelBursaest servlet for the example of the 'bursaest' LibrarySet. This servlet needs to be renamed and adjusted to reflect the LibrarySet of interest.

3) Keyword searches of the sequence, cluster and polymorphism annotations. This is done separately for each LibrarySet by a servlet whose name is composed of the prefix 'Web' followed by the value of the 'webName' column entry of the 'LibrarySet' table. A template class (WebBursaEst) for the default LibrarySet is provided which needs to be renamed to reflect the webName of the default LibrarySetID as specified in the LibrarySet table. If additional LibrarySet are added, the template class has to be copied, edited and renamed. The only values which need to be edited are the values of the librarySet and the host variables.

The advantages of using servlets and the JDBC API for database access is that the same servlet will work in the intranet and in the internet independent of the database system after adjusting the connection parameter. For our Bursal EST LibrarySet the servlet on the intranet web server accesses an Oracle8i database, whereas the servlet on the internet web server accesses a MySql database (Fig. 1).

Plans for the future

All the code of FOUNTAIN is free under the open source license agreement [http://www.fsf.org/copyleft/gpl.html] and the authors hope that it will be further developed in a community effort. Work will continue to improve the existing work packages and add new ones allowing for example integration of micro-array expression data. The ability to integrate and evaluate information stored in other biological databases is another goal of further development. The JAVA language as the basis of FOUNTAIN should be well suited for this endeavor.

Materials and Methods

The SQL database creation commands and the database scheme (FOUNTAIN_SQL.jar), the compiled and source JAVA classes (FOUNTAIN_JAVA.jar) and HTML pages (FOUNTAIN_HTML.jar) can be downloaded [http://genetics.hpi.uni-hamburg.de/Download] . The FOUNTAIN web site [http://genetics.hpi.uni-hamburg.de/FOUNTAIN.html] contains additional documentation and instructions how to install the FOUNTAIN package.

Acknowledgements

J.-M. B. would like to thank all members of the laboratory and Kay Förger for support and encouragement. This research was supported by grant Bu 631/2-1 from the Deutsche Forschungsgemeinschaft (DFG) and from the EU Framework V programs "Chicken Image" and "Genetics in a cell line".

References

1. Goffeau A, Barrell BG, Bussey H, Davis RW, Dujon B, Feldmann H, Galibert F, Hoheisel JD, *et al*: **Life with 6000 genes.** *Science* 1996, **274**:563-567
2. Adams MD, Celniker SE, Holt RA, Evans CA, Gocayne JD, Amanatides PG, Scherer SE, Li PW, Hoskins RA, Galle RF, *et al*: **The genome sequence of Drosophila melanogaster.** *Science* 2000, **287**:2185-95
3. Lander ES, Linton LM, Birren B, Nusbaum C, Zody MC, Baldwin J, Devon K, Dewar K, Doyle M, FitzHugh W, *et al*: **Initial sequencing and analysis of the human genome.** *Nature* 2001, **409**:860-921
4. Venter JC, Adams MD, Myers EW, Li PW, Mural RJ, Sutton GG, Smith HO, Yandell M, Evans CA, Holt RA, Gocayne JD, *et al*: **The sequence of the human genome.** *Science* 2001, **291**:1304-1351
5. Sachidanandam R, Weissman D, Schmidt SC, Kakol JM, Stein LD, Marth G, Sherry S, Mullikin JC, Mortimore BJ, Willey DL, *et al*: **A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms.** *Nature* 2001, **409**:928-33
6. Kawai J, Shinagawa A, Shibata K, Yoshino M, Itoh M, Ishii Y, Arakawa T, Hara A, Fukunishi Y, Konno H, *et al*: **Functional annotation of a full-length mouse cDNA collection.** *Nature* 2001, **409**:685-690
7. Liu D, Yang X, Yang D, Songyang Z: **Genetic screens in mammalian cells by enhanced retroviral mutagens.** *Oncogene* 2000, **19**:5964-5972
8. Uetz P, Giot L, Cagney G, Mansfield TA, Judson RS, Knight JR, Lockshon D, Narayan V, Srinivasan M, Pochart P, *et al*: **A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae.** *Nature* 2000, **403**:623-627
9. Staden R, Beal KF, Bonfield JK: **The Staden package, 1998.** *Methods Mol Biol* 2000, **132**:115-130
10. Abdrakhmanov I, Lodygin D, Geroth P, Arakawa H, Law A, Plachy J, Korn B, Buerstedde J-M: **A large database of chicken bursal ESTs as a resource for the analysis of vertebrate gene function.** *Genome Res* 2000, **10**:2062-2069
11. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: **Basic local alignment search tool.** *J Mol Biol* 1990, **215**:403-4010
12. Ewing B, Hillier L, Wendl MC, Green P: **Base-calling of automated sequencer traces using phred. I. Accuracy assessment.** *Genome Res* 1998, **8**:175-185