

BArray: A Microarray Data Warehouse and
Web Based User Interface

by

Bryan Olmstead


A Master's Thesis

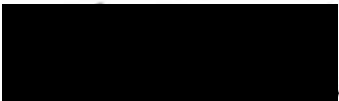
Presented to the Division of Medical
Informatics and Outcomes Research
in the Oregon Health Sciences University
School of Medicine
in partial fulfillment of
the requirements for the degree of
Master of Science
June 2001

School of Medicine
Oregon Health Sciences University

CERTIFICATE OF APPROVAL

This is to certify that the
Master's thesis of
Bryan Olmstead
has been approved


Professor in charge of thesis


Member


Member

Table of Contents

Abstract.....	iii
Introduction.....	1
Background & Significance.....	1
Figure 1: Gene expression pathway.....	4
Figure 2: Microarray printing robot.....	6
Figure 3: Microarray image sample.....	7
Methods.....	9
Requirements and Design Summary.....	13
Table 1: Requirements and descriptions.....	14
Results.....	14
Evaluation.....	28
Table 2: Requirements and solutions.....	29
Summary.....	31
Appendix.I - BArny Schema Figures	
Figure 4a: Microarray specific portion.....	34
Figure 4b: Public Database Integration Project.....	35
Figure 4c: PDI continued.....	36
Appendix II - Web UI Figures	
Figure 5: BArny schema example.....	37
Figure 6: Microarray search page.....	38
Figure 7: Microarray search results.....	39
Figure 8: Gene library search results.....	40
Figure 9: Archive search results.....	41
Appendix III - System Administrators Documentation..	42
Appendix IV - Rossetta Inpharmatics GEML XML dtd....	61
Bibliography.....	65

I would like to thank Christopher Dubay, Srinivasa Nagalla, Mark Turner and Susan Price for their help in making this project a reality.

Abstract

BARry is a data warehouse designed to store data produced by the microarray. The microarray is a tool recently developed for use in Functional Genomics that allows a researcher to examine the expression of genes within a cell on the genomic scale. That is, instead of examining gene expression a few genes at a time, an entire organism's genome can be examined in one experiment.

Through interaction and work with the microarray three main problems with the storage of microarray data were discovered, that are in turn solved with BARry. These problems are: 1) storage of microarray expression data, 2) storage of information about laboratory gene libraries and 3) storage of the location of archived image and data files. These data are then made accessible via a Web based interface, which also includes a mechanism to export expression data in various formats, including the Extensible Markup Language (XML), to aid in the sharing of data.

To ensure scalability and ease of administration of BARry a suitable operating system (OS), relational database management system (RDBMS) and development language needed to be used. To satisfy the OS and RDBMS requirements the Linux operating system using Oracle Corporation's database

management system was chosen. To satisfy the development language requirement the Web interface and scripts used throughout BArri are written in Perl.

Currently, an initial implementation of BArri exists that contains data on some 600 microarrays totalling more than 7.7 million records, solves the three problems mentioned above, ensures scalability and ease of administration, and possesses the capability to expand to meet future requirements.

Introduction

This thesis is concerned with the design and implementation of BARRY, a microarray data warehouse. The warehouse stores microarray expression data, information about laboratory gene libraries, and the location of archived data and image files from past experiments. Lab technicians and other users can access the data through an easy-to-use Web based interface. A suitably reliable and robust Operating System, Relational Database Management System and a hardware configuration capable of handling the large amounts of data to be collected over time were chosen for the implementation of the software. The system provides an initial implementation of the warehouse to serve a laboratory's current needs, as well as those expected in the future.

Background and Significance

In this section some of the differences between genetics and genomics will be highlighted, a brief overview of a cell's expression pathway and components will be

discussed, followed by a description of the microarray. Finally, the significance of the thesis is discussed.

Two basic fields of study concern themselves with an organism's genome: genetics and genomics. The field of genetics concerns itself with every aspect of a single gene, or gene set, from patterns of inheritance and distribution to the gene's biochemistry and how the gene is expressed[1]. In short, genomics is genetics, but on a much grander scale, as genomics concerns itself with an organism's entire set of genes which comprise its genome[2]. Functional Genomics is an extension of genomics that concerns itself with the expression and function of an entire organism's genome[3]. The distinction between genetics and functional genomics is important. For genetic research there are many tools available to the researcher, but they are not capable of easily scaling to the level needed for functional genomics. The microarray is a tool used for analyzing gene expression and is capable of scaling to measure the expression of an entire genome[4, 5], as opposed to a single gene, or small gene set as has been done with preceding techniques.

Functional Genomics and tools such as the microarray allow a researcher to probe an organism looking for differences in gene expression that could be responsible for disease, or other harmful conditions. For example, variations in gene expression could lead to the ultimate development of cancer. At one point in time nothing

differentiated two cells from each other; then, due to some unknown change, one of the cells began to divide and replicate at an uncontrolled pace, creating a cancerous tissue. The unknown changes that led to the development of the cancer are assumed to be changes that occurred at the genetic level on a genomic scale. That is, a change in one gene's expression is not likely to be solely responsible for the cancer; however, a change in the expression level of many genes may be.

To study gene expression on a genomic scale, and thus what is happening within a cell, one must look at the expression pathway of the genes of interest. This pathway begins with the genetic blueprint for an organism (a cell's DNA) and ends with the production of the molecules responsible for virtually every activity within a cell (proteins).

It is these proteins that in turn reflect the level to which a gene is being expressed. If no proteins from a gene exist within a cell, then the gene is not being expressed, while if proteins are present within a cell, the gene is being expressed. For example, if "normal" gene expression means that 10x proteins from a gene are present within the cell at any one time, and it is determined that 20x proteins are present it can be said that this gene is being over expressed, while 5x proteins would lead to the conclusion that the gene is being under expressed.

To determine the amount of proteins within a cell, and thus the expression level of a gene, the expression pathway must be examined. Within a cell's nucleus, portions of the genetic blueprint are transcribed into messenger RNAs (mRNAs). The mRNAs then travel out of the cell's nucleus into the cytoplasm where they are translated into proteins. The amount of protein within a cell is thus related to the amount of mRNA produced. Because of the fact that mRNA can be reverse transcribed back to DNA called copy DNA (cDNA), and because the amount of mRNA can be directly related to the expression level of a gene, quantifying the amount of cDNA is widely used to determine the expression level of genes within a cell.[6,7,8]

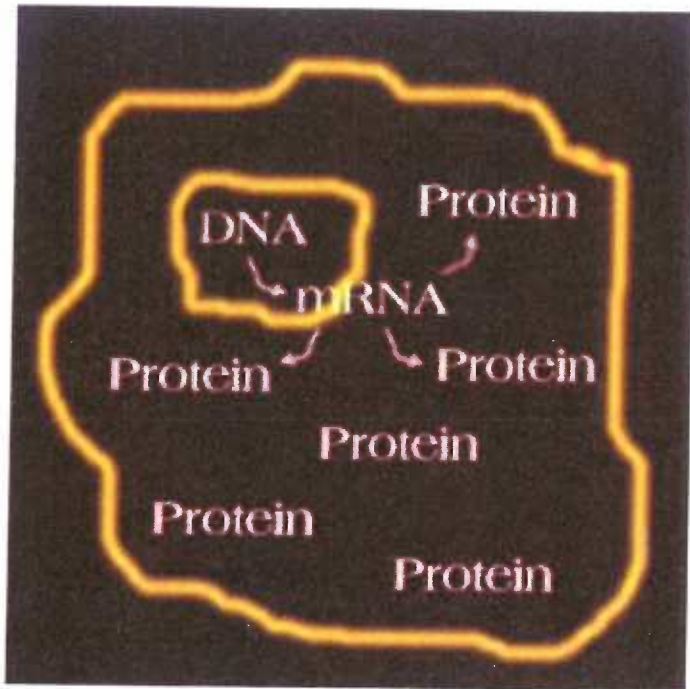


Figure 1: Gene expression pathway from DNA within the nucleus, to mRNA, to proteins within a cell.

Based on the level of mRNA reported by the cDNAs, if we know that gene X is expressed at a certain level, then how do we know what protein, or proteins, are related to gene X? As the Human Genome Project comes to an end we will know the sequence of approximately 95%[3,9] of

the human genome, however that does not completely answer the question of which proteins are related to gene X. Within the known DNA sequence there are genes whose proteins have been established, and there are sequences that are thought to be genes, but have no known related proteins. Clones of these unknown "genes" which have been partially sequenced are called Expressed Sequence Tags (ESTs). Currently, the estimates of genes within the human genome range from 30,000 to 35,000[10,11,12,13], and the majority of these are uncharacterized[14]. The use of microarrays allows a researcher to determine to what degree genes are being expressed, however for the uncharacterized genes determining what proteins they produce requires the application of other methods.

Determining the expression level of these genes is where the microarray plays a vital role. The microarray was developed by Dari Shalon for his 1995 PhD thesis[4], and had nearly immediate research acceptance and commercialization. Despite being used now for over six years, most of the literature about the microarray is of the "how-to" and "what-this-technology-can-bring-us" nature. A growing number of papers actually concern themselves with scientific studies in which the microarray was a central tool in providing a hypothesis.[15,16,17] However, the microarray is currently the most widely used technology capable of scaling to meet the needs of whole genome studies, and it is

a technology still in early development[2,5,18,19].

The microarray is composed of three basic pieces: a target, a probe, and a robotic printing device (figure 2). The targets are typically placed on a microscope sized slide

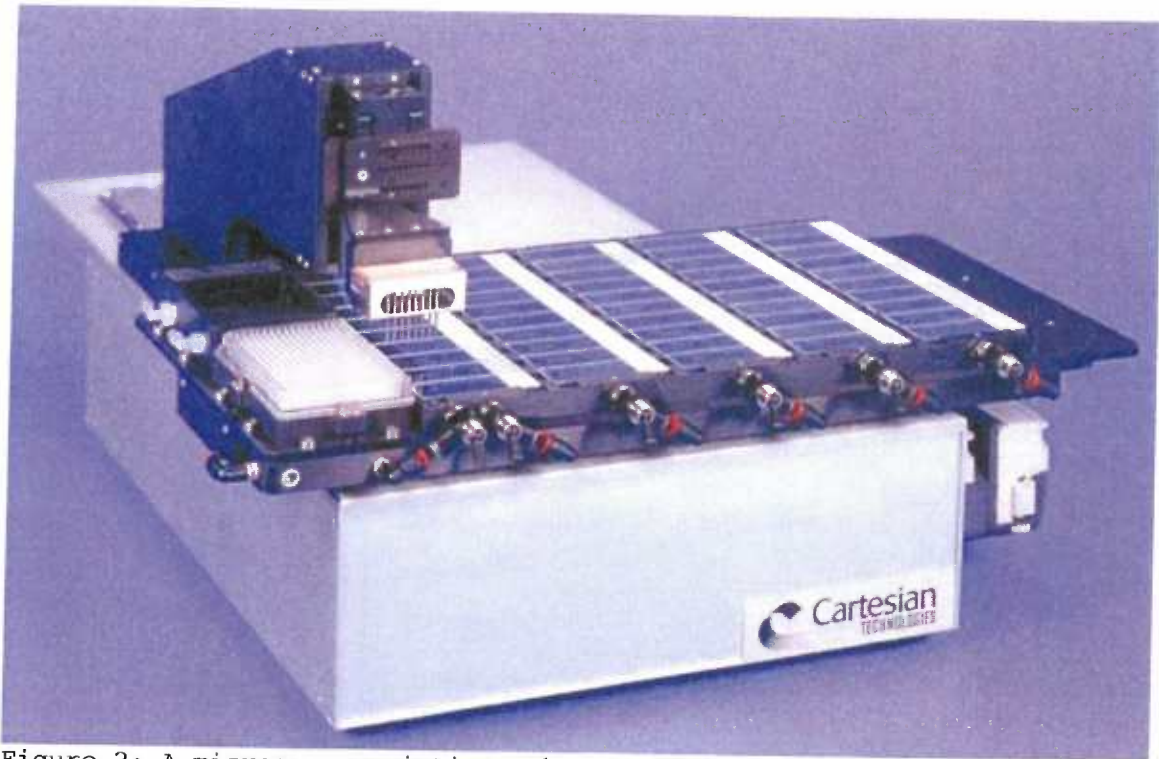


Figure 2: A microarray printing robot as pictured at, and sold by <http://www.cartesiantech.com>.

with as few as 1,000 to as many as 16,000+ target spots of DNA from a sequence verified and annotated collection of DNA (ie DNA gene libraries) printed on its surface. The probe used in the experiment is usually mRNA that has been extracted from a tissue or cell line of interest, and then reverse transcribed to cDNA. During the reverse transcription process a fluorescent label (direct labeling) or a primer (indirect labeling) is incorporated so that the

amount of probe (cDNA) can later be quantified. With indirect labeling the fluorescent label is added at a later time. The robotic printing device is responsible for the actual creation of the target. It accomplishes the task of printing each of the target spots onto the slide.[2,8,20,21] (Note: Some terminologies reverse the definitions of probe and target, as well as indirect and direct labeling)

Once the probe and target have been prepared, they are hybridized together, allowing the cDNA to bind with its complimentary DNA strand. After hybridization the slide is visualized by exciting the fluorescent label incorporated to the probe with a scanning laser. When the label is excited by the laser, it fluoresces, allowing the capture of an image using a confocal microscope and CCD camera. This image can then be analyzed using software to calculate the intensity of each target spot. From the intensity data it is then possible to calculate gene expression ratios. These ratios can be computed by dividing the normalized treated intensity by the normalized untreated intensity, or vice versa.

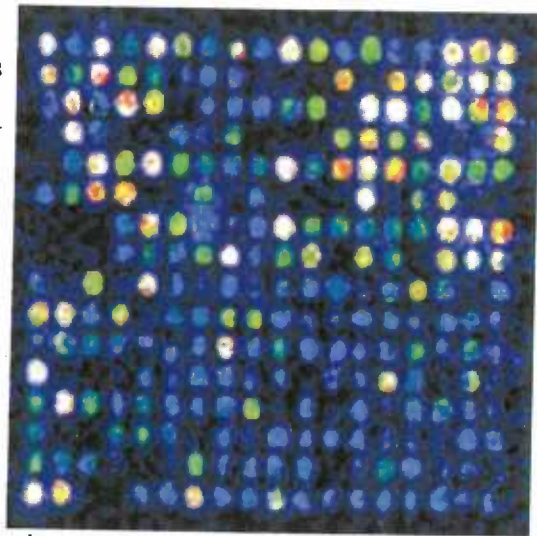


Figure 3: An example of the microarray image generated by the microarray scanner.

Normalization of spot intensity data is complex and beyond the scope of this thesis. From these ratios it is then

possible to compare gene expression between samples (e.g. treated and untreated), and use expression analysis software to make generalizations about gene expression and observations about possible interactions leading to, or preventing the disease or function of study.[2,8,21,22]

It is the image analysis software, and the data that it produces, that brings us to the significance of this thesis project. During image analysis there are a number of different values that are calculated, and as the complexity of the experiment grows, so does the amount of data.

From the literature and experience in the Nagalla laboratory it became apparent that a single microarray could have between 1,000 and 16,000 targets. As previously described, after hybridization the slide is scanned and one image file for each fluorescent label is produced. Generally, there are two labels per slide: one for the experimental probe, and one for the control probe. Each image file produced by the scanner averages 25 MB in size. These image files are then fed through analysis software to calculate expression data. The analysis of the image files generates some 20 data points per target, thus yielding on average between 20,000 and 320,000 data points, depending on the number of targets on the array. The text file created from this data extraction averages 5 MB in size. Associated with the image and text files are numerous configuration files that can double the amount of non-image data produced

per slide. Thus, a single project with 6 arrays having 10,000 targets per array could yield a data pool 660 MB in size. From this large data pool, the relevant expression data must be gathered from the appropriate file(s), cleaned and imported into the data warehouse.

Because the amount of data produced can, and does, grow quite rapidly, a data warehouse is employed, allowing researchers to browse and retrieve the desired data for the analysis of their work. During the 2nd International Meeting on Microarray Data Standards, Annotations, Ontologies and Databases (MGED2) in Heidelberg, Germany, some 16 differing schemas and warehouse implementations in development were discussed[22]. The following year, at MGED3, there were 29 posters presented under the category of databases[23]. One potential problem with these proposed databases is that all the schema incorporate a Laboratory Information Management System (LIMS), and many seem to ignore other aspects of the data that are important to the lab. BARRY attempts to fill these gaps by not incorporating a LIMS, as there are many good LIMS available, and instead focuses on solving some of the problems ignored by the other data warehouses.

Methods

In this section the challenges raised by the microarray's enormous data production capability, along with

problems in data sharing and analysis will be examined. Simultaneously, the requirements that grew from these challenges and then were incorporated into BArri during development will be discussed. Finally, an overview of requirements for BArri will be reiterated. It should be noted that the development of the requirements along with the initial implementation took place in Dr. Srinivasa Nagalla's laboratory in the department of Pediatrics at Oregon Health Sciences University, Portland Oregon.

Gathering and cleaning data from the pool generated for importation into the warehouse presents one problem. In addition, since projects generally contain upwards of ten to twelve arrays, and are often scanned and analyzed multiple times in hopes of reducing background noise on the array and other factors detrimental to the analysis process. This results in a rapid decrease in available storage space on the server. Because data must be kept for future reference, and the server could quickly run out of storage space, the data need to be archived in some form.

As the Nagalla laboratory started to produce data, it was decided that due to the relatively low cost, the ease at which data could be archived, and the relative longevity of the media[24], that CD-ROMs would be used as the archive media. Archiving data to CD solves the storage problem, but creates another. As the data archive grows, it becomes difficult, if not impossible, for laboratory technicians to

keep track of which CDs contain their data. To solve this problem, the contents of each archive CD have been stored within the warehouse.

Experimental expression data and archive contents are not the only data that the warehouse needs to store. The information concerning the targets (the laboratory's gene libraries) printed on the arrays also need to be stored so that the laboratory has a central location to identify the targets for planning and interpreting experiments.

Gene library data, experimental expression data, and archiving have now been accounted for. However each experiment has its own set of parameters, procedures, and notes that need to be stored. As mentioned, other databases accomplish this by the inclusion of LIMS data in the data warehouse. Examples include the Gene Expression Omnibus[25], ArrayDB[26], the GATC Expression Database[27], the Stanford Microarray Database[28], and the European Bioinformatics Institute's ArrayExpress[29]. After some discussion we determined that since there are a multitude of quality LIMS products available, BARRY did not need to incorporate this data. Instead, BARRY would start out as a simple backend data warehouse by focusing on the microarray data, which could then be interfaced to a LIMS.

Once the various data have been imported into the warehouse, a user interface for browsing needs to be available. It was determined that the user interface needed

to fulfill three goals: 1) ease of use, 2) cross platform compatibility, and 3) ease of development. With the growth of technology the laboratory technicians have a seemingly endless need to learn how to use new software. Examples include software controlling the array printing robot, the liquid handler, microarray scanning software, microarray image analysis software, and gene clustering software. Thus, it was determined that the user interface for BARRY needed to be simple to use and require little or no training. Cross platform compatibility was determined essential, because laboratory technicians use Macintoshes, PC's and even Unix machines to complete daily tasks. The third goal, ease of development, was determined necessary so continued development by other people and groups would be feasible. The fewer programming languages and tools used to develop and implement BARRY, the quicker and easier it would be for collaborators to take over when the time came to continue this initial work.

In order to fulfill these goals, we determined that a Web based user interface would be best. Everyone in the Nagalla laboratory has experience with a Web interface, and the Web has been a driving force within the past few years behind platform independence. This, accompanied by the ease of web interface development, made it the best available choice.

Next, some way of exporting the data was needed. The

goal was to allow the exportation of data in some way so that it was not dependent upon software used specifically in the Nagalla lab. To fulfill this goal, we decided that the data should be exported in various formats. Because spreadsheets such as Microsoft Excel are often used within a laboratory, we determined that a simplistic exportation, such as tab and comma delimited, would suffice. However, such simplistic forms lack adequate ways of describing data once it is exported. This is important for two reasons. One, when data is generated within the laboratory, and then is being used in other collaborating laboratories, the data needs to be described in a more robust and standard fashion. Two, simplistic exportation would not facilitate the easy validation of experiments. In order to meet these requirements, it was decided that the UI should also allow the exportation of data according to some microarray data sharing standard.

Requirements and Design Summary

BARRY contains the expression data, the location of archived files and the information describing gene libraries used in array production. Unlike many other warehouses it is not LIMS oriented. Requirements include that it should be searchable using a Web-based interface that is platform independent and easy to develop for. The Web interface should not only allow one to browse data within the

warehouse, but to ease sharing among laboratories the expression data should be exportable in various formats. One of the formats implemented needs to be an industry wide standard for describing microarray expression data. Finally, all of the above requirements should be implemented using software and hardware capable of scaling to meet the demands of microarray laboratory data production to ensure speed, accessibility and maintainability.

Requirement	Description
Usable design	Understand lab workflow through lab interaction and experience
Ease of development	Limit development languages
Warehouse schema components	Archive, gene library, expression data. Not LIMS oriented.
Simple Web UI	Little training and insure platform independence
Data sharing	Allow collaboration and validation (export data in multiple formats)
Scalability	Maintain speed and ability to expand

Table 1: This is a list of the requirements for BArriy, along with a brief description of each requirement.

Results

This section will discuss the tools used to implement BArriy, along with how the previously discussed requirements were fulfilled.

To implement the warehouse it was important that a RDBMS be chosen that could fulfill the predefined requirements. That is, it would be capable of handling the

large amounts of data that a microarray produces, it would be platform independent, and implementation of the warehouse would be possible on other RDBMSs, regardless of which RDBMS was chosen for development. To fulfill these goals, Oracle 8i was chosen[30].

Within the Information Technology (IT) world Oracle Corporation's RDBMS is considered one of the best and most reliable databases on the market. Oracle is used at many corporations that require reliability and scalability, including OHSU. The software itself is not platform specific; it is capable of being run on many differing platforms including Sun Microsystems' Sparc processor under Solaris, Intel's x86 family of processors under either Windows NT/2000 or the open source OS Linux. Oracle also uses the standard Structured Query Language (SQL) to manage every internal aspect of the database. SQL is a defacto standard in the IT world for database interaction and manipulation. Thus, if a database is implemented using a RDBMS capable of understanding SQL, it can easily be implemented using other RDBMS's.

In order to populate the warehouse with the desired data the schema outlined in figure 4 was developed. Figure 4, part A, represents the portion of BARRY that stores microarray, archive and gene library information. Figure 4, parts B and C, outline the Public Database Integration project under development by Mark Turner within the Nagalla

laboratory, the goal of which is to link currently available knowledge from public databases such as UniGene, OMIM, and PubMed to data stored within BArray.

The tables *experiments*, *experimental_conditions*, *autogene_params*, and *autogene_raw_data* contain the microarray expression data. The tables *cd_archive_list* and *cd_contents* contain data describing the location of files that have been archived, while the *resgen* table contains the information pertaining to gene libraries. The remaining tables are for the purpose of future development and proof of concept, and are not fully implemented. The following paragraphs discuss the logic behind the layout of the schema described above and described in figure 4. (For table specifications please refer to Appendix III).

When a microarray experiment is performed, it generally consists of multiple slides. Each slide represents either a different variable within the experiment, such as treatment, or point in time. In turn, each of these slides is repeated a minimum of three times to insure validity. Thus, a way to group slides into various categories is needed.

Near the bottom of figure 4 is the *autogene_params* table. This table holds parameters specific to a microarray when it is scanned. In turn, the *autogene_raw_data* table holds information pertaining to each target within the microarray. The *autogene_raw_data* table is thus responsible for holding the raw expression data used by the researcher

during the analysis phase. It is important to note that while BARRY does not store LIMS information, it is able to interface with a LIMS. Each microarray entry within the *autogene_params* table contains a unique "array_id" entry. This "array_id" could then be used in a LIMS to link LIMS data to arrays within the warehouse.

The *experimental_conditions* table then groups a set of microarray slides into various experimental conditions. For example, if we were researching the gene expression of a cancer at various time intervals during treatment, the *experimental_conditions* table would be the table responsible for grouping arrays defined within *autogene_params* into groups such as day 1, day 2, day 7, and day 14. These groups then need to be defined as belonging to a specific experiment. This is done in the *experiments* table. An example of the grouping structure is shown in figure 5.

As discussed, data files accumulate and it becomes necessary to archive these files to CD-ROM. However, locating files in the archive at a future date is difficult, especially as the archive grows. To make searching feasible the tables *cd_archive_list* and *cd_contents* store the location of the archived files within the warehouse. The table *cd_archive_list* stores the name of each CD. In order to ensure that a unique name is used for each CD created, and that the CDs are easily categorizable, the naming format DayMonthYear-Number is used. The "Number" portion of the

name is three digits long, allowing 999 CDs to be created in one day before a name needs to be reused. Thus, the first CD created on January 3rd, 2001 would be named "01032001-001" and the 18th CD created would be named "01032001-018".

The *resgen* table is responsible for storing the laboratory's gene libraries data. It is important to note that while the table is vendor specific, it is not organism specific. That is, the table stores data pertaining to gene libraries obtained from Research Genetics[31], but can store data pertaining to any organism Research Genetics provides. The philosophy is that if a laboratory obtains its gene libraries from another provider, the data given to it concerning the libraries will be laid out differently. Thus, a table specific to that data can be created and used in place of, or alongside the *resgen* table. By doing this the relationships within the schema are preserved. In order to access data from various tables, one need only create a view within the warehouse that links the two tables together making them appear as one.

The remaining tables are for the purpose of proof of concept, and are not fully implemented. The *control_points* and *control_point_locations* tables were intended to allow viewing targets within a microarray that were to be used as control points during the normalization process. These control points are targets on the microarray that are known to be expressed at a constant level. Because there can be

problems with the hybridization process, it is necessary to review the control points so that one can choose which points on the array are to be used in the normalization process. Due to problems in the actual dependability of the expression level of the control points across microarrays, this was not fully implemented.

The *webuser* and *webuser_cookies* tables could be used to support secure connections to the data warehouse through the Web interface. These tables allow the storage of user names and passwords, along with cookies. Thus, a user can log into the web UI and be issued a cookie. This cookie can then be used throughout the session to verify that the user trying to retrieve data is actually the person that they say they are. By fully implementing this it would allow the warehouse to be accessible over the Internet in a controlled and somewhat secure manner.

The *gene_accession_number* table provides the link between the gene library data and the Public Database Integration project portion of BARRY under development. It is important to note that while these tables are part of the BARRY schema, they were not laid out in the original proposal, and were simply proof of concept ideas developed during implementation.

Because we decided that the warehouse itself should focus on expression data the warehouse stores no LIMS data. For data referring to how a specific experiment was carried

out, one needs to have access to either the laboratory's LIMS or the technician's notebooks.

Server downtime thus far has been due to maintenance and one incident in which an administrative error caused the need for rebuilding of the warehouse. This rebuilding, however, was successful and verified that current back-up schemes were sufficient. Thus, by using Oracle 8i, and having the schema laid out as described, the warehouse solves the problems of archive data storage, gene libraries information storage, and expression data storage as discussed above, along with fulfilling the goals of scalability, platform independence, and reliability[32].

To populate each of the three sections of the warehouse with data, scripts specific to each section of the warehouse are used. The gene expression portion of the warehouse has a set of scripts that are run on an hourly basis. These scripts traverse a directory tree searching for new gene expression data files deposited by the laboratory technicians. If a new file is found, the data within is parsed, cleaned and inserted into the warehouse. Cleaning includes such things as changing NULL or infinity to numbers or values representable within the database. To populate the archive location portion of the warehouse when a new CD is created, a script is manually run that inserts the CD's name and contents into the proper tables. To populate the gene libraries portion of the warehouse, a script is

manually run that parses data files provided by Research Genetics and inserts the appropriate data. All of the scripts used in population of the warehouse are written in the Perl scripting language[33].

Perl was chosen because if one has any programming experience, it is a relatively easy language to learn and because it has a wide range of modules available that allow one to do anything from web scripting, to database manipulation, to file system traversal, to mathematic calculations. Not only is Perl versatile, but it is itself relatively platform independent. As long as a Perl interpreter is available, the scripts can be moved from machine to machine and run without the need for recompilation or re-coding.

In order to browse and export the data stored within the warehouse a Web-based User Interface (UI) is used. A Web-based UI allows access to the warehouse from any platform whether one is using Netscape's Navigator, Microsoft's Internet Explorer, the Lynx text browser or any other standard web browser. Both HTML and Perl are used in generation of the web pages. However, in using HTML to create Web pages it is necessary to avoid certain HTML tags and the Java scripting/programming language, since they can be browser specific and/or implemented differently from browser to browser. Because of this, the UI has a fairly simplistic look (figures 6-9); however, it is able to

perform its task while maintaining platform independence. With this combination of Web and Perl platform independence, the user and the administrator are free to use the client and server of choice for implementation, thus fulfilling the platform independence requirement.

The UI allows one to browse and export the data previously described in various formats, as well as do other miscellaneous tasks. To ensure a standard look throughout the UI, the web pages contain a standard header allowing the user to quickly jump to various pages within the UI, and a standard footer which provides a link describing the server's hardware, the last update of the page, and an e-mail address to contact the person in charge of maintaining the site (figure 6).

When browsing the expression data, one can either browse the entire microarray collection from beginning to end or search the collection for specific microarray names. Once the microarray, or group of arrays, is found, the data is exportable in various formats. One can export complete data sets for a single array in tab delimited, comma delimited, or XML format. For each microarray listed, one can also search the file location archive for the data file(s) related to the microarray (figure 7). If a group of arrays is selected, one can export a subset of data from each array in either tab or comma delimited form. For example, if array M1 is the only microarray of interest, one

can export all of the data for array M1 in XML format. However, if one wishes to export data for arrays M3, M4 and M5, one must choose from a subset of data to export including, but not limited to, mean, median, mode and position. This is useful, for example, when you wish to compare the mean intensity for a certain gene across a set of arrays. In either case, the data is then saved to a text file with rows representing targets and columns representing the variables associated with each target. Note, however, that to export arrays in groups as described above, a check is performed to assure that arrays M3, M4 and M5 are of the same size, and therefore have the same number of targets to ensure a proper layout of the data file produced.

XML is used as one of the data exportation options because it is a more scalable, non-static meta-language that gives a more powerful and robust way to describe documents and dynamic content, such as microarray data. XML's power and generalizability comes from the fact that it is a simplified form of the Standard Generalized Markup Language (SGML), an international documentation standard that has been used since the 1980's. XML was developed by the World Wide Web Consortium (W3C).[34,35]

In order to export the data in an XML format, we needed a standard for describing microarray data in XML. After searching for various standards, we found that there were three competing standards in development: 1) Paul

Spellman's XML array specification[36], 2) the Gene Expression Markup Language (GeneXML)[37], and 3) the Gene Expression Markup Language (GEML) from Rossetta Inpharmatics[38]. After analysis of these three standards it was determined that Paul Spellman's XML specification and GeneXML focus too much on LIMS data. Since it was previously determined that BARRY should not incorporate LIMS data, such XML definitions could not be implemented because of lack of data within the warehouse itself. Rossetta's GEML, on the other hand, is broken into two pieces, one that focuses on LIMS data (GEMLPattern.dtd), and one that focuses on expression data (GEMLProfile.dtd). Because of this, and because Rossetta Inpharmatics has a large corporate influence in the world of the microarray, their XML definition was used as one of the forms in which gene expression data can be exported.

The information pertaining to the gene libraries is also searchable and exportable. After a search is performed the data is displayed in table format, and then if desired, exported in either tab or comma delimited format (figure 8). If the initial search returns numerous matches, it is also possible to click on specific entries within the table and have these entries displayed by themselves, thus making it easier for the user to concentrate on the desired data and not accidentally read data in the column above or below.

The UI also allows the searching of the archived file

location database. To search, the user simply enters a search string such as "array-name", and a list of matches is returned. If the file has been archived, the table of information displayed includes the CD name, the path on that CD to the file, and the file name (figure 9).

Throughout the UI, links are provided that take the user to help documentation. This help documentation is opened in a separate browser window when possible and is presented in two forms. One form is the entire help documentation in a single window, and the other is help specific to the page that the user is working with. For example, if a user chooses to view the help for the archived data location page the help presented would include information about the header, the archive page itself, and the footer.

The UI also allows one to perform other miscellaneous tasks. Current statistics on the warehouse can be viewed, including the number of microarrays stored in the warehouse, the number of targets associated with the arrays, the number of entries in the gene libraries, the number of storage devices (e.g. CDs) in the archive including an estimate of the archive size in megabytes, and the number of data files within the archive. In addition to the statistics page, there is also a document page that provides warehouse specific documentation and links, XML documentation and links, and miscellaneous links for the system administrator

or user.

Thus, the Web-based UI allows one to easily, and with little training, browse data within the warehouse in a platform independent and easy-to-develop-for fashion. The data is exportable in various formats, one of which is an XML format which is as close to an industry standard as is currently available, thus allowing the expression data to be easily shared among collaborating laboratories.

To fulfill the final aspect of the scalability requirement, an operating system, a web server, and the proper hardware had to be chosen. During most of the warehouse development Microsoft's Windows NT operating system was used, because, at the time, Windows NT was the only operating system available to the laboratory capable of 1) running Oracle and 2) being implemented on reasonably affordable hardware. As mentioned, Oracle is available on many platforms, but the hardware behind such platforms is very expensive. For example, the Intel Pentium based server purchased cost approximately half of what a comparable Sun Microsystems Sparc based server would have cost.

The open source web server Apache[39] was chosen as the Web server to implement the UI because of its scalability, security and reputation of reliability. As the Web-based UI was developed, it became clear that some flavor of Unix would be best suited to run the web server and Perl scripts behind the UI because development and maintenance of Apache

and Perl for Windows NT lagged behind that of Unix. Because of this, warehouse development occurred under Windows NT, and UI development occurred under Slackware Linux, a distribution of the open source Unix operating system Linux[40].

Along with the problem of BARRY being implemented on two differing platforms, it was determined that as the lab grew the current server hardware configuration would not be suitable. The server had 74 GB of space for experiment data, and this was being consumed nearly every two months. Since larger and more time consuming projects lay on the horizon, the server needed more room to grow. At the same time, the warehouse was run from a 9 GB hard drive, which itself was nearing capacity. In order to solve these problems, it was decided that a massive upgrade would need to occur, at which time the server OS would be changed.

In order to meet the hardware and software scalability requirements, BARRY currently runs under Slackware Linux[40] on a dual Pentium III 600 MHz system with 768 MB of RAM. The system has three 9 GB 7200 rpm 160/LVD SCSI drives, three 18 GB 7200 rpm 160/LVD SCSI drives and one 74 GB 10K rpm 160/LVD SCSI drive attached to an Adaptec 2940 U2W controller for system and warehouse use. In order to store the large amounts of data generated, and ensure their safety, a RAID-5 array[41] consisting of five 74 GB 10K rpm 160/LVD SCSI drives (totalling 273 GB) using an Adaptec dual

channel 3200S RAID controller is being used.

Evaluation

In this section an evaluation of BArri's capabilities will be discussed along with some current data warehouse access statistics. A description of areas in which BArri could be further developed will also be presented. It must be noted, however, that while this evaluation is not totally quantitative, it is meant to show that the basic requirements outlined above have been fulfilled, and that BArri does indeed meet the initial requirements of the laboratory.

To fulfill the warehouse design requirements BArri is designed not to be LIMS oriented, and is RDBMS independent because of its use of SQL. The warehouse focuses on storing expression data, the location of archived image and data files and the information pertaining to a laboratory's gene libraries as outlined. The Web-based UI is platform independent and is easy to develop for using HTML and Perl. The UI allows browsing of the data, and exportation of the expression data in various formats, one being the nearest to an industry standard that is currently available (Rosetta's GEML XML data definition[38]). For further development and upkeep, one need only be familiar with the existing system documentation, SQL, HTML, XML and Perl languages. The Linux operating system and the hardware being used currently

fulfill Barry's needs and should be scalable well into the future providing the speed, accessibility and maintainability required. In short, all of the requirements outlined have been met.

Requirement	How requirement was achieved
Usable Design	Through lab interaction and experience that remains ongoing.
Ease of development	Documentation, SQL, HTML, XML and Perl
Data warehouse schema	Archive, gene library, expression data storage. No LIMS data stored.
Simple Web User Interface	Little training, simple look, and platform independence
Data sharing	Tab and comma delimited format, as well as Rosetta Inpharmatics GEML XML format.
Scalability	Software (Linux and Oracle 8i), as well as suitable hardware.

Table 2: This is a list of the requirements for Barry, along with a brief description of how each requirement was fulfilled.

All of the requirements being fulfilled means that Barry is currently in a Beta stage of development. Beta means that the software is currently in a state in which no major features will be added, but great emphasis is placed on debugging. This is to ensure that when it is officially released, it will work seamlessly as described and intended. However, being in a beta stage also means that it not appropriate, as of yet, to perform a true formal evaluation.

Although Barry is in beta, there are many ways in which further development could be done to enhance the project. While the schema provides for experiment and experimental

condition grouping as described, this is as yet not implemented. The warehouse is capable of exporting expression data in Rossetta's GEML, but is as yet incapable of importing data in this fashion. At this point in time, it is unlikely that Rossetta's GEML will become the XML standard for microarray data exchange. According to sessions at MGED3[23], the standard will actually be a hybrid of all 3 XML specifications described (the Microarray Markup Language, or MAML[42]), and when this is released it will need to be implemented. While the warehouse is capable of exporting groups of microarrays and their data in either tab or comma delimited form, it is not capable of doing this in XML. Further, the warehouse currently has no means of normalizing the expression data stored within it, and the public database information pieces alluded to above are still being developed.

All of the above described features are but a sampling of what could be done in the future to make BARRY a more robust and usable data warehouse. Absence of these features does not reduce the usability or importance of BARRY, they simply are features that when implemented will increase the functionality and robustness of the warehouse.

Now, let us look at some actual statistics. As of March 13th, 2001 BARRY contained data on 599 different microarray slides with 7,789,056 associated data points. From this data one could retrieve a single array in less

than 15 seconds, or 5 variables from 10 arrays in under 60 seconds. The laboratory's gene libraries contained 47,424 unique genes or ESTs. When a search with the common term "kidney" was performed 1119 records were found, and 100 of these were printed out in approximately 20 seconds. When a simpler 2 term, 2 result search was performed it finished in under 3 seconds. The file archive contained 122 CDs with 13,193 files, approximately 73,200 MB (73.2 GB) of information, and a search that returned 850 matches completed in less than 4 seconds. As required, the data is accessible via the Web UI in what seems to be a timely manner. Further, considering that the warehouse has grown from 0 to 599 thus far, and has no significant signs of slowness, this demonstrates that BARRY is indeed scalable. Due to the nature of Oracle's RDBMS, and the hardware described above BARRY should have little problem scaling well into the future.

Summary

It was determined that three data storage problems were presented by the microarray: 1) the storage of microarray expression data, 2) the storage of information describing a laboratory's gene libraries, and 3) the storage of image and data archive locations. This data then needed to be available for browsing and exportation via an easy-to-use User Interface. Unlike the other data, the expression data

needed to be exportable in an industry standard format that would allow for easy sharing and collaboration between labs. The resulting warehouse and UI then needed to be implemented in a platform independent fashion, and easy-to-develop-for way that would not require collaborators to learn a large number of development languages. The software and hardware on platforms would also need to be scalable and reliable.

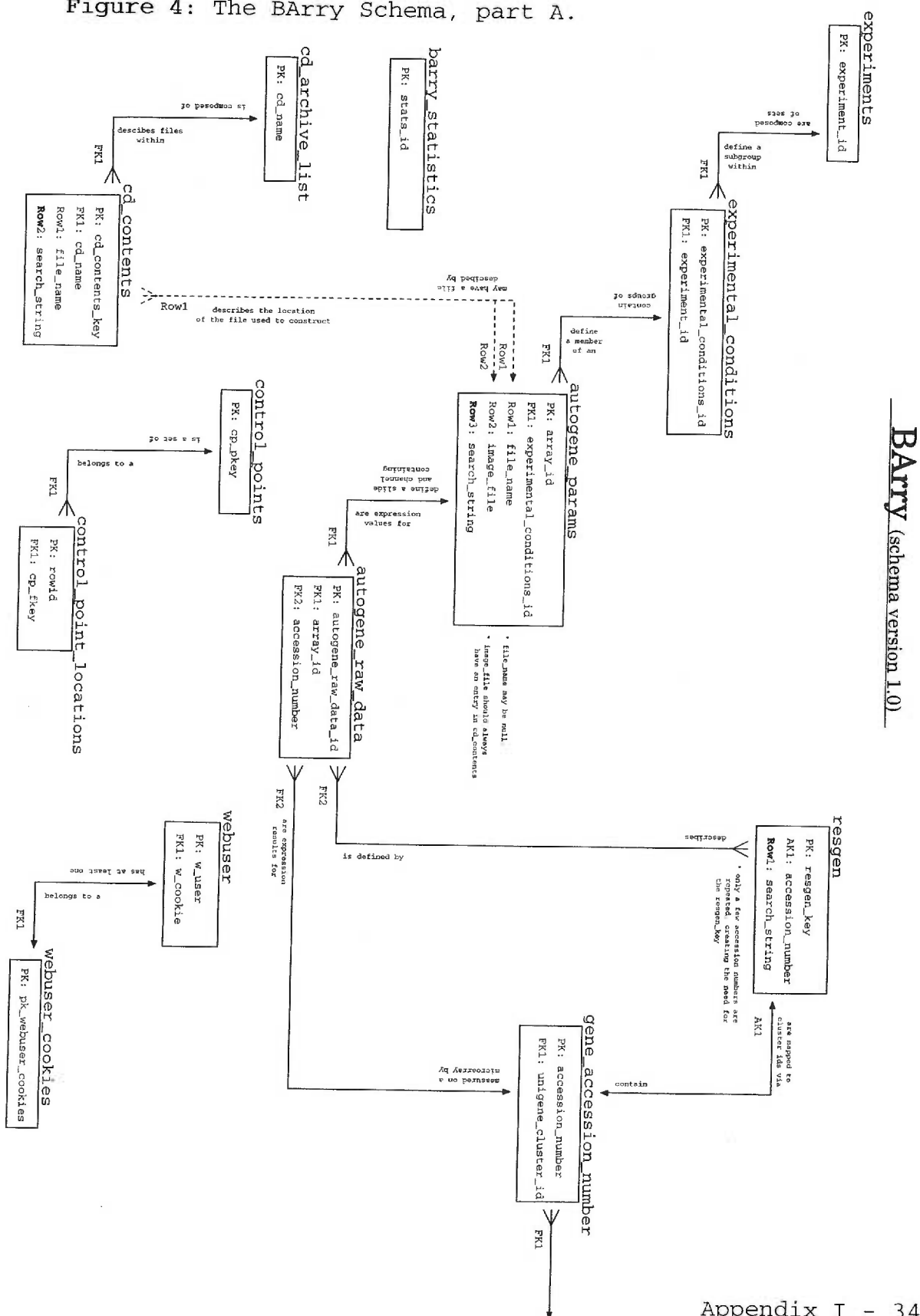
To meet these requirements, BArry was developed. The warehouse portion is implemented using Oracle Corporation's Oracle 8i RDBMS. The UI is implemented as an easy-to-use web based UI using the Apache web server. To avoid the need for collaborators to learn and/or use many different development languages the Perl scripting language is used almost exclusively. This includes scripts to populate the data warehouse, and generate dynamic web pages. The gene expression data is exportable in 3 formats, one of which being Rosetta Inpharmatics' GEML XML format.

In order to ensure scalability and reliability into the future the above software was chosen due to its reputation in the Information Technology community, and is currently running on hardware deemed capable of meeting the current storage needs, as well as those in the future.

Due to its software and hardware scalability along with BArry's generic design, it has the opportunity to become a general tool used in Functional Genomics research at OHSU. With continued development it could grow from an in-house

project to one that could easily become a standard used in microarray labs throughout the United States or even the world. Even if BArri were to remain an in-house project, its ability to export data in varying formats now, and in the future, will allow it to easily cooperate with other microarray warehouses around the world, thus allowing it to remain a vital tool in the microarray suite at OHSU.

Figure 4: The Barry Schema, part A.



Page 2 of 3

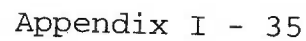


Figure 4: The Barry Schema, part C. The Public Database Integration Project continued.

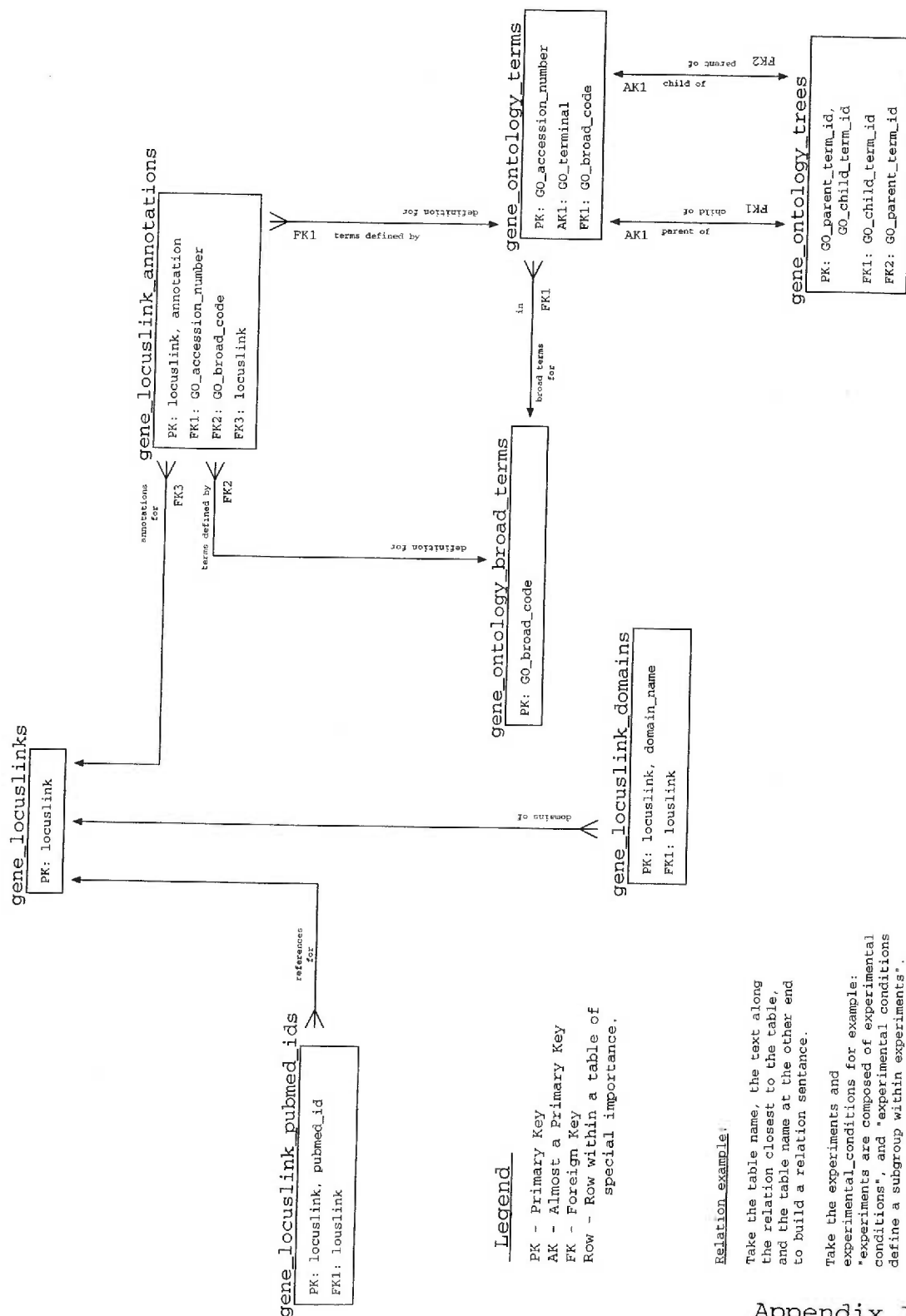


Figure 5: Barry Schema Example

This is an example of Barry's Schema layout using an experiment named "Cancer Experiment". A cancer was treated with an experimental drug and then was analyzed at day 1, day 2, day 7 and day 14. A microarray with 10,000 targets was used. The experiment was then repeated 3 times.

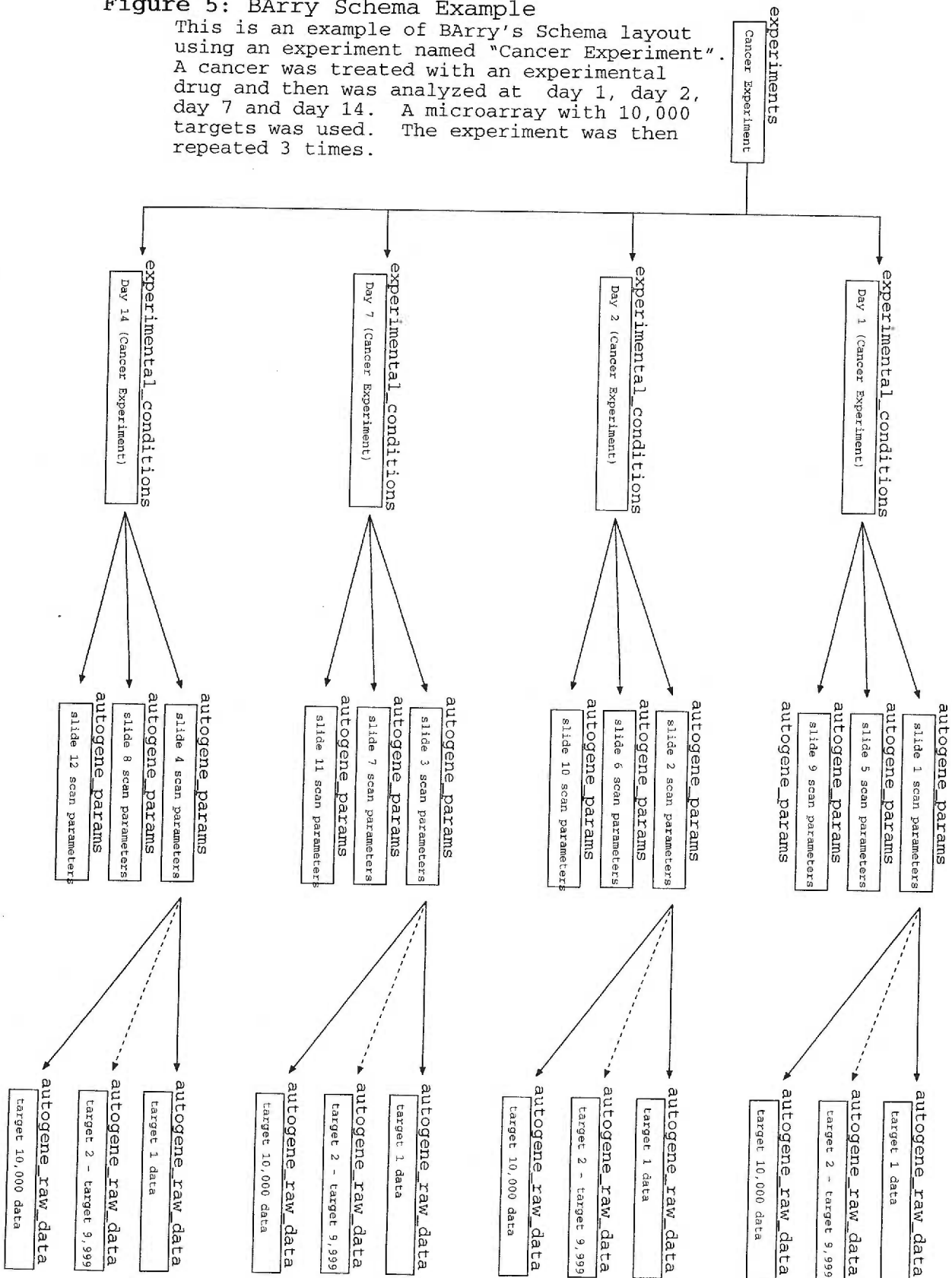


Figure 6: Barry microarray Search Page

To search for a microarray you can either browse through the entire set of slides in the warehouse, or perform a search. Note the standard header and footer that is used on each page of the UI.

Retrieval Data from the Warehouse

[Home](#)
[Research Genetics Data](#)
[Warehouse Data](#)
[Warehouse statistics](#)
[login](#)

List the arrays currently in the warehouse:
[List All](#)

Search for arrays currently in the warehouse:

To do a more complicated search:

1- Boolean "AND": enter "keyword1 keyword2" or "keyword1 and keyword2" to find records with both keywords.

2- Boolean "OR": enter "keyword1 or keyword2 or keyword3" to find records that have any one of the listed keywords.

note - the retrieved data files will be deleted nightly from the server.

[help?all](#)

[server specs](#)
last html update: 2/20/2001
comments, questions, concerns?
you can reach us at olmstead@ohsu.edu

Figure 7: Microarrays returned by a search.

Array ID Search Results

[Home](#) [Research Genetics Data](#) [Warehouse Data](#) [Warehouse statistics](#) [login](#)

☐ 1: "H15-04_Control-1_80_60_Cy5"

completed at 18:28 on 23-Feb-2001.
Dump all of the raw data: [xml](#), [tab](#) or [comma](#) delimited format.
Normalize the data: use [all](#) the standards, or [choose](#) the standards.
[Whereis](#) the file "H15-04_Control-1_80_60_Cy5_data.txt"?

☐ 2: "H15-05_Control-3_80_60_Cy5"

completed at 19:50 on 23-Feb-2001.
Dump all of the raw data: [xml](#), [tab](#) or [comma](#) delimited format.
Normalize the data: use [all](#) the standards, or [choose](#) the standards.
[Whereis](#) the file "H15-05_Control-3_80_60_Cy5_data.txt"?

Figure 8: Results from a gene library search.

resgen Search Results					
Home	Research Genetics Data	Warehouse Data	Warehouse statistics	login	
Version	Plate	Row	Column	Cluster ID	
gf205	299	c	6	Hs.301032	unig
gf205	291	b	2	Hs.153685	unig
gf202	109	f	1	Hs.298033	unig
gf200	6	c	8	Hs.153685	unig
gf200	6	b	5	Hs.7006	unig
gf200	19	f	1	Hs.25515	unig
gf201	60	d	1	Hs.301833	unig
gf200	6	b	6	Hs.4791	unig
gf200	6	c	10	Hs.89666	unig
gf204	260	a	2	Hs.197114	unig

* (S) - fetch this record only

dump to a file: [tab](#) or [comma](#) delimited.

10 entries selected by query "brain protein KIAA03"

[help/all](#)

server [specs](#)

last html update: 2/20/2001

comments, questions, concerns?

you can reach us at olmstead@ohsu.edu

Figure 9: Results from an archived file location search.

Find File Results

[Home](#)
[Research Genetics Data](#)
[Warehouse Data](#)
[Warehouse statistics](#)
[login](#)

CD Name	Path	File Name
01122001_003	patrick_1/Asnoff	m17-18_90-80_8-23-00Cy3.Tif
01122001_003	patrick_1/Asnoff	m17-16_CNITL_MBR_TotalCy3.info
01122001_003	patrick_1/Asnoff	m17-19_90-80_8-23-00Cy3.info
01122001_003	patrick_1/Asnoff	m17-16_CNITL_MBR_TotalCy3.Tif
01122001_003	patrick_1/Asnoff	m17-16_90-80_08-23-00.Tif
01122001_003	patrick_1/Asnoff	m17-19_90-80_8-23-00Cy3.Tif
01122001_003	patrick_1/Asnoff	m17-17_90-70_KO_MBR_8-23-00Cy3.info
01122001_003	patrick_1/Asnoff	m17-17_90-70_KO_MBR_8-23-00Cy3.Tif
01122001_003	patrick_1/Asnoff	m17-17_90-80_08-23-00Cy3.info
01122001_003	patrick_1/Asnoff	m17-17_90-80_08-23-00Cy3.Tif
06142000-003	cd-8/Stacey	4-7-00 M17.Tif
01122001_003	patrick_1/Asnoff	m17-18_mRNACNTL_90-70_8-23-00Cy3.info
01122001_003	patrick_1/Asnoff	m17-18_mRNACNTL_90-70_8-23-00Cy3.Tif
01122001_003	patrick_1/Asnoff	m17-18_90-80_8-23-00Cy3.info

Results returned: 14

Search file archive for:

Submit

Barry Administrative Documentation

ver. 0.0.1
2/1/2001

Data Warehouse

expr

The expr database contains the entire BArny schema, and is responsible for holding all of the data related to microarray analysis, the cd-rom archive, the WWW user interface, as well as the data for the Public Database Integration project worked on by Mark Turner.

User Name	Description
expradmin	The administrator account. This account has nearly all of the privileges of dba, or sysdba.
expruser	A general user account. This account can add and modify tables, but should not be allowed to grant privileges, or add/drop tablespaces.
html	The web account. This is an account that has only select options from specified tables so that information can be selected for web page rendering.

repo

The repo database was created for general database administration purposes. When running such DBA applications as the Oracle Enterprise Manager, or rman (a backup and recovery tool) it is necessary to have a separate database created for internal oracle use while running such applications.

User Name	Description
repoadmin	The user designated for use with Oracle Enterprise Manager.

Tablespaces

expr tablespaces

Tablespace Name	Description
cd_archive	Holds the tables associated with maintaining information about the cd-rom archive of expression data.
fgcore	A temporary tablespace used when making massive changes to any table, or tablespace.
main	The default user tablespace. It currently holds all of the proteomics tables.
quantitation	The main tablespace for the expression warehouse. It holds all of the tables for expression data, web login, and warehouse statistics.
resgen	Holds the tables responsible for storing the Research Genetics gene library information.

note - The tables expr_rollback, indx, temporary, usr and system are used by the system, and not directly modified by any user.

repo tablespaces

Tablespace Name	Description
cd_archive	currently unused
expr_rollback	currently unused
repo_rollback	currently unused
repository_table	Holds all of the information generated by the Oracle Enterprise Manager and its tools.

note - The tables indx, rbs, rcat, system, temporary, and user are used by the system, and not directly modified by any user.

autogene_params

Name	Null?	Type
ARRAY_ID	NOT NULL	VARCHAR2(256)
RAW_ARRAY_ID	NOT NULL	VARCHAR2(256)
SLIDE_ID	NOT NULL	VARCHAR2(32)
RELATED_EXPERIMENT_ID		VARCHAR2(256)
EXPERIMENTAL_CONDITIONS_ID		NUMBER
CONTROL_POINTS		NUMBER
SLIDE_WEIGHTING_FACTOR		NUMBER
DATE_ADDED	NOT NULL	DATE
DATE_MODIFIED	NOT NULL	DATE
SOFTWARE		VARCHAR2(255)
SOFTWARE_ID		VARCHAR2(32)
SOFTWARE_USER		VARCHAR2(256)
SOFTWARE_COMPANY		VARCHAR2(256)
DATE_GENERATED	NOT NULL	VARCHAR2(11)
TIME_GENERATED	NOT NULL	VARCHAR2(5)
IMAGE_FILE	NOT NULL	VARCHAR2(256)
INTENSITY_SCALING		NUMBER
ROTATION		NUMBER
HIGH_PIXEL_COLOR		VARCHAR2(16)
PREFILTERING		VARCHAR2(16)
META_ROWS	NOT NULL	NUMBER
META_COLS	NOT NULL	NUMBER
ROWS_PER_SUBGRID	NOT NULL	NUMBER
COLS_PER_SUBGRID	NOT NULL	NUMBER
NULL_SPOT		VARCHAR2(256)
DIAMETER		NUMBER
INTENSITY_SPOT_FINDING		VARCHAR2(256)
MIND		NUMBER
MAXD		NUMBER
CIRCLE_SEGMENTER		VARCHAR2(256)
BACKGROUND_BUFFER		NUMBER
BACKGROUND_WIDTH		NUMBER
SIGNAL_LOW		NUMBER
SIGNAL_HIGH		NUMBER
BACKGROUND_LOW		NUMBER
BACKGROUND_HIGH		NUMBER
QQ_MEAN		VARCHAR2(256)
QQ_MEDIAN		VARCHAR2(256)
QQ_MODE		VARCHAR2(256)
QQ_TOTAL		VARCHAR2(256)
QQ_STANDARD		VARCHAR2(256)
QQ_SPOT_AREA		VARCHAR2(256)
QQ_MAJOR_AXIS		VARCHAR2(256)
QQ_MINOR_AXIS		VARCHAR2(256)
QQ_CIRCULAR_PA_RATIO		VARCHAR2(256)
QQ_AREA_IGNORED		VARCHAR2(256)
QQ_MEDIAN_IGNORED		VARCHAR2(256)
QQ_STANDARD_IGNORED		VARCHAR2(256)
QQ_POSITION_OFFSET		VARCHAR2(256)
QQ_GOODNESS_FIT		VARCHAR2(256)
QQ_SIGNAL_AREA		VARCHAR2(256)
QQ_ORIENTATION		VARCHAR2(256)
FILE_NAME		VARCHAR2(256)
SEARCH_STRING		VARCHAR2(4000)

autogene_params cont.

Primary key: array_id
Foreign key(s): experimental_conditions_id

Tablespace: quantitation

Trigger(s): autogene_params_trigger1

Description: This table holds configuration data pertaining to an individual array. It describes a number of things, including who scanned and extracted the data, how the data was extracted and what data was extracted. See the AutoGene manual for a description of the columns. Populated with the script prdi-x.x.x.pl.

autogene_raw_data

Name	Null?	Type
AUTOGENE_RAW_DATA_ID	NOT NULL	VARCHAR2(270)
POSITION		VARCHAR2(12)
ARRAY_ID	NOT NULL	VARCHAR2(256)
META_ROW	NOT NULL	NUMBER(3)
META_COL	NOT NULL	NUMBER(3)
SUB_ROW	NOT NULL	NUMBER(3)
SUB_COL	NOT NULL	NUMBER(3)
ACCESSION_NUMBER	NOT NULL	VARCHAR2(64)
SELECTED		NUMBER(2)
XCOORDINATE	NOT NULL	NUMBER
YCOORDINATE	NOT NULL	NUMBER
MEAN_SIGNAL		NUMBER
MEDIAN_SIGNAL		NUMBER
MODE_SIGNAL		NUMBER
TOTAL_SIGNAL		NUMBER
STANDARD_SIGNAL		NUMBER
MEAN_BACKGROUND		NUMBER
MEDIAN_BACKGROUND		NUMBER
MODE_BACKGROUND		NUMBER
TOTAL_BACKGROUND		NUMBER
STANDARD_BACKGROUND		NUMBER
SPOT_AREA		NUMBER
CIRCULAR_PA_RATIO		NUMBER
AREA_IGNORED		NUMBER
MEDIAN_IGNORED		NUMBER
STANDARD_OF_IGNORED		NUMBER
POSITION_OFFSET		NUMBER
GOODNESS_FIT		NUMBER
SIGNAL_AREA		NUMBER
NORMALIZED		NUMBER
MAJOR_AXIS		NUMBER
MINOR_AXIS		NUMBER

Primary key: autogene_raw_data_id
 Foreign key(s): array_id
 accession_number

Tablespace: quantitation

Trigger(s): none

Description: This is the table that actually holds all of the raw data associated with each spot on the array. Populated with the script prdi-x.x.x.pl.

resgen

Name	Null?	Type
RESGEN_KEY	NOT NULL	NUMBER
RESGEN_VERSION		VARCHAR2(10)
RESGEN_PLATE	NOT NULL	NUMBER
RESGEN_ROW	NOT NULL	VARCHAR2(2)
RESGEN_COLUMN	NOT NULL	NUMBER
UNIGENE_CLUSTER_ID		VARCHAR2(32)
UNIGENE_BUILD		VARCHAR2(128)
ORGANISM		VARCHAR2(4000)
RESGEN_CLONE_ID	NOT NULL	VARCHAR2(64)
ACCESSION_NUMBER	NOT NULL	VARCHAR2(64)
RESGEN_NID		VARCHAR2(64)
GENE_NAME		VARCHAR2(4000)
GENE_SYMBOL		VARCHAR2(64)
CHROMOSOME		VARCHAR2(2)
RESGEN_LIBRARY		VARCHAR2(128)
RESGEN_VECTOR		VARCHAR2(128)
RESGEN_PID		VARCHAR2(64)
CLONE_END		NUMBER
VALID_3P		NUMBER
VALID_5P		NUMBER
INSERT_SIZE		VARCHAR2(64)
TISSUE		VARCHAR2(4000)
CYTOGENETIC_POSITION		VARCHAR2(128)
MARKER		VARCHAR2(128)
RESGEN_BARCODE		VARCHAR2(128)
ANTIBIOTIC		VARCHAR2(16)
RESGEN_CHIP_NUMBER		NUMBER
RESGEN_FILE_DATE	NOT NULL	VARCHAR2(9)
ORGANISM_CODE	NOT NULL	NUMBER
SEARCH_STRING		VARCHAR2(4000)

Primary key: resgen_key

Foreign key(s): none

Tablespace: resgen

Trigger(s): resgen_trigger1

Description: This table holds all the data regarding the laboratory gene library as provided by Research Genetics. (see ftp://ftp.resgen.com/pub/sv_libraries/ for the raw files). Populated with three different scripts. prdh-vx.x.x.pl is used to parse human data files, whereas prdm-vx.x.x.pl is used to parse the mouse data files. Note - read the comments at the top of each script, as different scripts are used for varying files, even within the same organism.

cd_archive_list

Name	Null?	Type
CD_NAME	NOT NULL	VARCHAR2(255)
CD_UPPER_NAME		VARCHAR2(255)
CD_CREATION_DATE		VARCHAR2(8)

Primary key: cd_name

Foreign key(s): none

Tablespace: cd_archive

Trigger(s): none

Description: This table holds the names and associated information of individual cd-roms in the archive. Populated with the script catalog-vx.x.x.pl.

cd_contents

Name	Null?	Type
CD_CONTENTS_KEY	NOT NULL	NUMBER
FILE_NAME	NOT NULL	VARCHAR2(255)
FILE_PATH		VARCHAR2(4000)
SEARCH_STRING		VARCHAR2(4000)
CD_NAME	NOT NULL	VARCHAR2(255)

Primary key: cd_contents_key

Foreign key(s): cd_name

Tablespace: cd_archive

Trigger(s): cd_contents_trigger1

Description: This table holds the actual path and file name information for individual files found on the cd-rom. Populated with the script catalog-vx.x.x.pl.

barry_statistics

Name	Null?	Type
STATS_ID	NOT NULL	NUMBER
AUTOGENE_PARAMS_COUNT		NUMBER
AUTOGENE_PARAMS_NOTE		VARCHAR2(4000)
AUTOGENE_RAW_DATA_COUNT		NUMBER
AUTOGENE_RAW_DATA_NOTE		VARCHAR2(4000)
CD_ARCHIVE_LIST_COUNT		NUMBER
CD_ARCHIVE_LIST_NOTE		VARCHAR2(4000)
CD_CONTENTS_COUNT		NUMBER
CD_CONTENTS_NOTE		VARCHAR2(4000)
STATS_DATE		DATE

Primary key: stats_id

Foreign key(s): none

Tablespace: quantitation

Trigger(s): none

Description: This table holds statistical information concerning the expression warehouse that is generated by the perl script stat-gen-0.0.x.pl for display on a web page via the script statistics-0.0.x.pl.

control_points

Name	Null?	Type
CP_PKEY	NOT NULL	NUMBER
CREATION_DATE	NOT NULL	DATE

Primary key: cp_pkey
Foreign key(s): none

Tablespace: quantitation

Trigger(s): none

Description: This table holds information about a group of control points meant to be used by normalization scripts throughout the normalization process. It has not been used in a long time, and will be deleted in the near future. Currently not populated.

control_point_locations

Name	Null?	Type
META_ROW		NUMBER
META_COL		NUMBER
SUB_ROW		NUMBER
SUB_COL		NUMBER
ACCESSION	NOT NULL	VARCHAR2(64)
CP_FKEY	NOT NULL	NUMBER

Primary key: (the Oracle) rowid
Foreign key(s): cp_fkey

Tablespace: quantitaion

Trigger(s): none

Description: This table holds the actual location, and accession number to a single point within a control point group. It too has not been used in a long time, and will be deleted in the near future. Currently not populated.

webuser

Name	Null?	Type
W_USER	NOT NULL	VARCHAR2(8)
W_PASSWD	NOT NULL	VARCHAR2(8)
W_COOKIE	NOT NULL	NUMBER

Primary key: w_user
Foreign key(s): w_cookie

Tablespace: quantitation

Trigger(s): none

Description: This table is used to hold user information for the login feature of the web interface. It was an initial "first try" implementation, and is not currently used, except for proof of concept.

webuser_cookies

Name	Null?	Type
PK_WEBUSER_COOKIES	NOT NULL	NUMBER
COOKIE		VARCHAR2(32)

Primary key: pk_webuser_cookies
Foreign key(s): none

Tablespace: quantitation

Trigger(s): none

Description: This table is meant to hold all web cookies associated with a user defined in webuser. Currently, however, it is just a one to one relationship, and needs to be worked on before it is fully implemented.

Views

The following is a list of important views used within the warehouse. As with the list of tables above, this list is not meant to be comprehensive. It is only meant to contain views used by the expression warehouse.

m_gen

Name	Null?	Type
PKEY		NUMBER
RELEASE_VER		VARCHAR2(10)
RELEASE_PLATE	NOT NULL	NUMBER
RELEASE_ROW	NOT NULL	VARCHAR2(2)
RELEASE_COL	NOT NULL	NUMBER
CLUSTER_ID		VARCHAR2(32)
BUILD_VER		VARCHAR2(128)
ORGANISM		VARCHAR2(4000)
CDNA_VER	NOT NULL	VARCHAR2(64)
ACC	NOT NULL	VARCHAR2(64)
NID		VARCHAR2(64)
PID		VARCHAR2(64)
CLONE_END		NUMBER
TITLE		VARCHAR2(4000)
GENE		VARCHAR2(64)
CHROM		VARCHAR2(2)
ANTIBIOTIC		VARCHAR2(16)
VALID_3P		NUMBER
VALID_5P		NUMBER
LIBRARY_NAME		VARCHAR2(128)
VECTPR_NAME		VARCHAR2(128)
CHIP_NAME		NUMBER
TDATE	NOT NULL	VARCHAR2(9)

Description: This view is used by the perl scripts that pull mouse genome data from the resgen table (specifically mhresgen.pl). It was added for backwards compatibility when the original mouse_genome (m_gen) and human_genome (h_gen) tables were merged to form the resgen table.

h_gen

Name	Null?	Type
PKEY		NUMBER
RELEASE_NUM		VARCHAR2(10)
RG_PLATE	NOT NULL	NUMBER
RG_ROW	NOT NULL	VARCHAR2(2)
RG_COL	NOT NULL	NUMBER
INSERT_SIZE		VARCHAR2(64)
CLUSTER_ID		VARCHAR2(32)
UG_BUILD		VARCHAR2(128)
CLONE_ID	NOT NULL	VARCHAR2(64)
VECTOR		VARCHAR2(128)
TISSUE		VARCHAR2(4000)
LIBRARY		VARCHAR2(128)
ACC	NOT NULL	VARCHAR2(64)
NID		VARCHAR2(64)
GENE_NAME		VARCHAR2(4000)
GENE_SYMBOL		VARCHAR2(64)
CHROMOSOME		VARCHAR2(2)
BAND		VARCHAR2(128)
MARKERS		VARCHAR2(128)
BARCODE		VARCHAR2(128)
ANTIBIOTIC		VARCHAR2(16)
CHIP_NUM		NUMBER
TDATE	NOT NULL	VARCHAR2(9)

Description: This view is used by the perl scripts that pull human genome data from the resgen table (specifically mhresgen.pl). It was added for backwards compatibility when the original mouse_genome (m_gen) and human_genome (h_gen) tables were merged to form the resgen table.

Indices

The following is a list of indices used throughout the expression warehouse. These indices are necessary to keep search time to a minimum when using the web interface.

Index Name	Table	Column(s)
autogene_raw_data_index0	autogene_raw_data	array_id
autogene_raw_data_index1	autogene_raw_data	meta_row, meta_col, sub_row, sub_col
autogene_raw_data_index3	autogene_raw_data	mean_signal, mean_background
autogene_raw_data_index4	autogene_raw_data	position, array_id
resgen_n1	resgen	accession_number

Bitmap Indices: Currently, none of the indices were created as an bitmap index.

Triggers

`autogene_params_trigger1`

Action: This trigger builds the `search_string` column as each record is being inserted into the `autogene_params` table. Currently, it only concatenates an uppercase version of the `array_id`.

`cd_contents_trigger1`

Action: This trigger builds the `search_string` column as each record is being inserted into the `cd_contents` table. Currently, it concatenates the `file_name`, and `file_path` column. Once this is done, it converts `search_string` to uppercase.

`resgen_trigger1`

Action: This trigger builds the `search_string` column as each record is being inserted into the `resgen` table. Currently, it concatenates columns from various tables and then converts `search_string` to uppercase.

Note - see `quant-triggers.txt`, `cd-triggers.txt` and `resgen-triggers.txt` for a detailed description of each trigger.

Synonyms

The synonyms listed below serve the sole purpose of giving the html user read access to various tables throughout the data warehouse. The html user is the generic user used to provide global access to the data warehouse. By not allowing the html user anything but read access, and read access to only those tables that have a public synonym, we are tightening security on the warehouse. This is by no means all that needs to be done in order to secure the warehouse, but it is a step in the right direction.

<u>Table Name</u>	<u>Synonym Name</u>	<u>Owner</u>
autogene_params	autogene_params	public
autogene_raw_data	autogene_raw_data	public
barry_statistics	barry_statistics	public
cd_archive_list	cd_archive_list	public
cd_contents	cd_contents	public
experiments	experiments	public
experimental_conditions	experimental_conditions	public
resgen	resgen	public
webuser	webuser	public
webuser_cookies	webuser_cookies	public

Maintenance Information

The warehouse requires little maintenance beyond that normally performed by the systems administrator/DBA. For a summary of directory structures, and data files that need to be backed up presented in a format to easily keep track of dates back-ups were performed, visit the on-line documentation at <http://mandrake.ohsu.edu/docs>.

Notes

Data Warehouse Creation

For SQL scripts used in creation of the warehouse, please see either the cd archive "BArry ver. 1.0", or visit the web page at <http://turgidson.ohsu.edu>.

File Naming Convention

Files are referenced as "foo-x.x.x.pl", where the x's represent numbers in the version. Thus, version 0.0.2 of the foo perl script would be named "foo-0.0.2.pl". The 0.2.3 version of the passwords text file would be "passwords-0.2.3.txt".

Oracle Documentation

For documentation provided by Oracle Corporation for the 8i Server, visit the URL <http://oradoc.photo.net/ora81/DOC/server.815/>.

The search_string Column

The "search_string" column, found in various tables, is used by perl scripts (see WebUtils.pm) in the web interface to conduct searches of the tables. The string held in this column is just a concatenation of fields from within the table so that a generic search interface could be built, and one would not have to be written for each individual table based on what columns one wanted to search.

```
<!-- GEMLProfile.dtd -->
<!-- Gene Expression Markup Language (GEML(tm)). Copyright (C) 2000-2001, -->
<!-- Rosetta Inpharmatics, Inc. All Rights Reserved. You may freely use, -->
<!-- publish, and redistribute the GEML DTDs, subject to the restrictions -->
<!-- in the Terms of Use or pursuant to a written agreement between you -->
<!-- and Rosetta Inpharmatics. This complete paragraph must be included -->
<!-- in all copies of this DTD. See the full Terms of Use section at -->
<!-- http://www.geml.org. GEML is trademarked to avoid proliferation of -->
<!-- incompatible variations. Please obtain the latest GEML DTDs and -->
<!-- documentation from http://www.geml.org. To contribute ideas toward -->
<!-- the development of the GEML format, email geml@rii.com. -->
<!-- <!-- -->
<!ELEMENT project (profile+,
    other*)>      <!--Project = group of one or more profiles -->
<!ATTLIST project
    name CDATA #IMPLIED
    id CDATA #IMPLIED
    date CDATA #IMPLIED
    by CDATA #IMPLIED
    company CDATA #IMPLIED >
<!ELEMENT profile (hyb?,
    image_file*,
    channel_info*,
    reporter+,
    comment*,
    other* )>
    <!--Profile = single hyb, optional channel info, -->
    <!-- data for one or more reporters -->
<!ATTLIST profile
    name CDATA #IMPLIED
    id CDATA #IMPLIED
    barcode CDATA #IMPLIED
    access CDATA #IMPLIED
    scanner CDATA #IMPLIED
    scan_number CDATA #IMPLIED
    scanned_date CDATA #IMPLIED
    owner CDATA #IMPLIED
    scanned_by CDATA #IMPLIED
    analyzed_date CDATA #IMPLIED
    analyzed_by CDATA #IMPLIED
    fail_type CDATA 'false'
    control_flag CDATA 'false'
    qc_by_user CDATA #IMPLIED
    profile_quality CDATA 'Pending QC'>
<!ELEMENT hyb
    (prep*,
    comment*,
    other* )>
    <!--Hyb = Zero or more sample preparations-->
<!ATTLIST hyb
    name CDATA #REQUIRED
    id CDATA #IMPLIED
    date CDATA #IMPLIED
    performed_by CDATA #IMPLIED
    labeled_by CDATA #IMPLIED
    fluor_reversal CDATA 'false'
    hyb_station CDATA 'manual'
    hyb_number CDATA #IMPLIED
    description CDATA #IMPLIED>
<!ELEMENT prep
    (treatment*,
    comment*,
    other* )>
    <!--Preparation has zero or more treatment steps-->
```

```

<!--ATTLIST prep
        name CDATA #IMPLIED
        code CDATA #IMPLIED
        id CDATA #IMPLIED
        date CDATA #IMPLIED
        prepared_by CDATA #IMPLIED
        type CDATA #IMPLIED
        method CDATA #IMPLIED
        description CDATA #IMPLIED>
<!--ELEMENT treatment (compound*,
        other* )> <!--Treatment uses zero or more compounds-->
<!--ATTLIST treatment
        name CDATA #REQUIRED
        id CDATA #IMPLIED
        step_number CDATA #REQUIRED
        media CDATA #IMPLIED
        volume CDATA #IMPLIED
        volume_units CDATA 'ml'
        temperature CDATA #IMPLIED
        temperature_units CDATA 'C'
        duration CDATA #IMPLIED
        wash_before CDATA 'false'
        description CDATA #IMPLIED>
<!--ELEMENT compound (other* )> <!--Compound requires at least a name-->
<!--ATTLIST compound
        name CDATA #REQUIRED
        id CDATA #IMPLIED
        molregno CDATA #IMPLIED
        description CDATA #IMPLIED
        source CDATA #IMPLIED
        lot_number CDATA #IMPLIED
        stock_concentration CDATA #IMPLIED
        stock_concentration_units CDATA #IMPLIED
        treatment_concentration CDATA #IMPLIED
        treatment_concentration_units CDATA 'mg'>
<!--ELEMENT image_file (other* )>
<!--ATTLIST image_file
        name CDATA #REQUIRED
        number CDATA #IMPLIED>
        <!--Image_File allows a file or files to accompany-->
        <!--a profile-->
<!--ELEMENT channel_info (other* )>
        <!--Channel_Info's channel_name should match-->
        <!--the name of a feature's channel-->
<!--ATTLIST channel_info
        channel_name CDATA #REQUIRED
        barcode CDATA #IMPLIED
        additive_error CDATA #IMPLIED
        multiplicative_error CDATA #IMPLIED
        mean_signal CDATA #IMPLIED
        raw_image_filename CDATA #IMPLIED>
<!--ELEMENT reporter (feature+,
        gene?,
        accession*,
        other* )>
        <!--Reporter is in one or more features, optionally -->
        <!--references the gene it reports on, and may be -->
        <!--be identified by an accession id in some database-->
<!--ATTLIST reporter
        name CDATA #IMPLIED
        systematic_name CDATA #REQUIRED
        gene_deletion CDATA 'false'
        sequence CDATA #IMPLIED
        control_type CDATA #IMPLIED>
<!--ELEMENT feature (channel*,
        (ratio | log_ratio | ln_ratio )?,
        position?,
        other* )>
<!--ATTLIST feature
        id CDATA #IMPLIED
        number CDATA #IMPLIED
        ctrl_for_feat_num CDATA #IMPLIED
        fail_type CDATA 'false'>

```

```

<!ELEMENT channel ( (signal | log_signal | ln_signal ),
                    (background | log_background | ln_background )?,
                    other* )>
<!ATTLIST channel
    name CDATA #REQUIRED
    fail_type CDATA 'false' >
    <!--Potentially there will be more than two -->
<!ELEMENT signal (other* )>
<!ATTLIST signal normalized_value CDATA #REQUIRED
    raw_value CDATA #REQUIRED
    stddev CDATA #IMPLIED
    pixels CDATA #IMPLIED >
<!ELEMENT log_signal (other* )>
<!ATTLIST log_signal normalized_value CDATA #REQUIRED
    raw_value CDATA #REQUIRED
    stddev CDATA #IMPLIED
    pixels CDATA #IMPLIED >
<!ELEMENT ln_signal (other* )>
<!ATTLIST ln_signal normalized_value CDATA #REQUIRED
    raw_value CDATA #REQUIRED
    stddev CDATA #IMPLIED
    pixels CDATA #IMPLIED >
<!ELEMENT background (other* )>
<!ATTLIST background
    value CDATA #REQUIRED
    stddev CDATA #IMPLIED
    pixels CDATA #IMPLIED >
<!ELEMENT log_background (other* )>
<!ATTLIST log_background value CDATA #REQUIRED
    stddev CDATA #IMPLIED
    pixels CDATA #IMPLIED >
<!ELEMENT ln_background (other* )>
<!ATTLIST ln_background value CDATA #REQUIRED
    stddev CDATA #IMPLIED
    pixels CDATA #IMPLIED >
<!ELEMENT ratio (other* )>
<!ATTLIST ratio
    value CDATA #REQUIRED
    pvalue CDATA #IMPLIED
    error CDATA #IMPLIED
    xdev CDATA #IMPLIED
    fail_type CDATA 'false' >
<!ELEMENT log_ratio (other* )>
<!ATTLIST log_ratio
    value CDATA #REQUIRED
    pvalue CDATA #IMPLIED
    log_error CDATA #IMPLIED
    xdev CDATA #IMPLIED
    fail_type CDATA 'false' >
<!ELEMENT ln_ratio (other* )>
<!ATTLIST ln_ratio
    value CDATA #REQUIRED
    pvalue CDATA #IMPLIED
    ln_error CDATA #IMPLIED
    xdev CDATA #IMPLIED
    fail_type CDATA 'false' >
<!ELEMENT position (other* )>
<!ATTLIST position
    x CDATA #REQUIRED
    y CDATA #REQUIRED
    units CDATA #REQUIRED >
<!ELEMENT gene (accession*,
    alias*,
    other*)>
<!ATTLIST gene
    primary_name CDATA #REQUIRED
    systematic_name CDATA #REQUIRED
    species CDATA #IMPLIED
    chromosome CDATA #IMPLIED
    description CDATA #IMPLIED>
<!ELEMENT accession (other*)>
<!ATTLIST accession
    database CDATA #REQUIRED
    id CDATA #REQUIRED>
<!ELEMENT alias (other*)>
<!ATTLIST alias
    name CDATA #REQUIRED>
<!ELEMENT comment (other*)>

```

```
<!ATTLIST comment      text CDATA #REQUIRED>
<!ELEMENT other        (other*)>
<!ATTLIST other         name CDATA #REQUIRED
                        value CDATA #REQUIRED>
```

Bibliography

- 1) Tamarin, R.H. "Principles of Genetics Fifth Edition". Wm. C. Brown Publishers. 1996. p. 3.
- 2) Schena, M. *Genome analysis with gene expression*. Bioessays. 1996. Vol. 18, #5. p. 427-431.
- 3) Dhad, R. *Nature Insight, Functional Genomics (Editor's Introduction)*. Nature. June 15, 2000. Vol. 405. p. 819.
- 4) Shalon, D. *DNA micro arrays: a new tool for genetic analysis*. PhD Thesis, Stanford University. 1995.
- 5) De Francesco, L. *Taking the Measure of the Message: From a Single Message to Thousands, Technology Exists to Quantitate mRNA's*. The Scientist. November 23, 1998. Vol. 12, #23. p. 20-21.
- 6) Brown, P.O. *Exploring the new world of the genome with DNA microarrays*. Nature Genetics Supplement. January 1999. Vol. 21. p. 33-37.
- 7) DeRisi, J., Penland, L. and Brown, P.O. (Group 1), Bittner, M.L., Meltzer, P.S., Ray, M., Chen, Y., Su, Y.A., and Trent, J.M. (Group 2). *Use of a cDNA microarray to analyze gene expression patterns in human cancer*. Nature Genetics. December 1996. Vol. 14. p. 457-460.
- 8) Schena, M., Shalon, D., Heller, R., Chai, A., Brown, P.O., Davis, R.W. *Parallel human genome analysis: Microarray-based expression monitoring of 1000 genes*. Proc. Natl. Acad. Sci. October 1996. Vol. 93. p. 10614-10619.
- 9) Clark, C. *Mapping Mankind's Future*.
<http://www.cnn.com/2000/fyi/news/08/01/genome/index.html>. June 26, 2000.
- 10) Ewing, B., Green, P. *Analysis of expressed sequence tags indicates 35,000 human genes*. Nature Genetics. June, 2000. Vol. 25. p. 232-234.
- 11) Crollius, H.R. et al. *Estimate of human gene number provided by genome-wide analysis using Tetraodon nigroviridis DNA sequence*. Nature Genetics. June, 2000. Vol. 25. p. 235-238.
- 12) Baltimore, David. *Our genome unveiled*. Nature. February 15, 2001. Vol. 409. p. 814-816.
- 13) Bork, P., Copley, R. *Filling in the gaps*. Nature. February 15, 2001. Vol. 409. p. 818-820.
- 14) Schultz, J., Tobias, D., Ponting, C.P., Copley, R.R, and Bork, P.. *More than 1,000 putative new human signalling proteins revealed by EST data mining*. Nature Genetics. June 2000, Vol. 25. p. 201-204.
- 15) Special Genomics Edition of Science. Science. October 15, 1999. Vol. 286. p. 447-457, 487-490, 531-537.
- 16) Nature Genetics (Supplemental). January 1999. Vol. 21.
- 17) Nature (Insight). June 15, 2000. Vol. 405. p. 819-865.
- 18) Lockhar, D.J., Winzeler E.A. *Genomics, gene expression and DNA*

- arrays. Nature. June 15, 2000. Vol. 405. p. 827-836.
- 19) Gwynne, P., Page, G. *Microarray Analysis: the next revolution in molecular biology*. Science. August 6, 1999.
<http://www.sciencemag.org/feature/e-market/benchtop/micro.shl>.
- 20) Schena, M., Shalon, D., Davis, R.W., Brown, P.O. *Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray*. Science. October 20, 1995. Vol. 270. p. 467-470.
- 21) Nagalla Laboratory Protocols. February 16, 2001*.
<http://medir.ohsu.edu/~geneview/pages/protocols.html>.
- 22) The Second International Meeting on Data Standards, Annotations, Ontologies and Databases. February 12, 2001*.
<http://www.ebi.ac.uk/microarray/MGED>.
- 23) Collected Abstracts as PDF (MGED3). April 4, 2001*.
http://www.dnachip.org/mged3/mged3_abstracts_v4.pdf
- 24) Rothenberg, J. Ensuring the Longevity of Digital Documents. Scientific American. January, 1995. Vol. 272. p. 42-47.
- 25) Gene Expression Omnibus. February 16, 2001*.
<http://www.ncbi.nlm.nih.gov/geo/>.
- 26) Division of Intramural Research: ArrayDB. February 16, 2001*.
<http://genome.nhgri.nih.gov/arraydb/schema.html>
- 27) GATC Expression Database. February 16, 2001*.
<http://www.gatconsortium.org/schema.pdf>.
- 28) Stanford Microarray Database (SMD) Specifications. February 16, 2001*. http://genome-www4.stanford.edu/MicroArray/SMD/doc/db_specifications.html.
- 29) Structure and Design of ArrayExpress Database. February 16, 2001*.
<http://www.ebi.ac.uk/arrayexpress/Design/design.html>.
- 30) The Oracle Corporation. February 22, 2001*. <http://www.oracle.com>.
- 31) Research Genetics. February 21, 2001*. <http://www.resgen.com>.
- 32) Data Sheet for Oracle 8i Enterprise Edition. March 13, 2001*.
http://www.oracle.com/collateral/o8i_enterprise_ds.pdf.
- 33) www.perl.com - Embperl. March 14, 2001*. <http://www.perl.com/pub>.
- 34) The World Wide Web Consortium. February 22, 2001*.
<http://www.w3c.org>.
- 35) Eckstein, Robert. XML Pocket Reference. O'Reilly and Associates, Inc. Sebastopol, CA. October 1999. p. 1.
- 36) XML Proposals (Paul Spellman). February 21, 2001*.
<http://beamish.lbl.gov/current.shtml>.
- 37) The Gene Expression Markup Language (GeneXML). February 21, 2001*.
<http://www.ncgr.org/research/genex/genexml.html>.
- 38) GEML DTDs. February 21, 2001*. <http://www.geml.org/dtds.html>.

- 39) Apache. February 21, 2001*. <http://www.apache.org>.
- 40) Slackware Linux. February 21, 2001*. <http://www.slackware.com>.
- 41) Twincom - Overview of RAID levels. March 13, 2001*. <http://www.twincom.com/raid.html#RAID5>.
- 42) SourceForge: Project Info - MGED. April 17th, 2001*. <http://www.slipups.com/items/9254.html>.

* This date represents the date that the web page was accessed, and was available.