Pathbase Technical Documentation

Introduction

Pathbase has been in development since April 2000 and is currently running as Pathbase2. Pathbase has been completely rewritten from January – May 2003 in order to use the various biological ontologies which have recently emerged and which should make the data interchange between different databases much more straight forward and much more accurate.

The data interchange and database connectivity will also be a major focus of future Pathbase development and the various technologies will be compared and evaluated later in this report.

The Pathbase Processes

Before looking at the technical details of Pathbase it is necessary to understand the logical processes of the system.

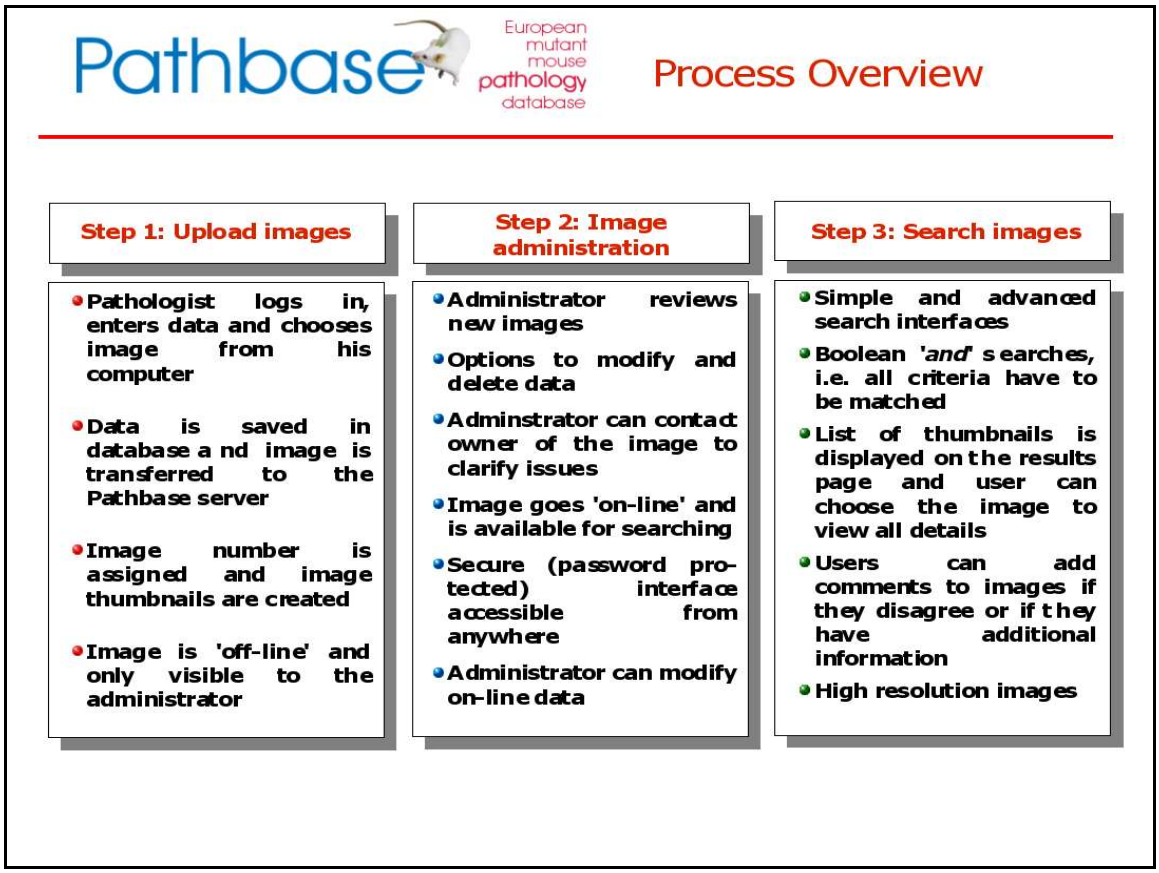

Figure 1: Pathbase Process Overview

There are 3 basic processes that make up Pathbase, all of which are web based:

Step 1: Upload images

The web site visitor / pathologist who wants to upload his data and images has to have an existing Pathbase user account before he can upload his data. This is necessary in order to display the originating organisation of the image for copyright purposes, but also to allow users to search for their own images. This way Pathbase administrators are also able to contact image submitters for further information. Users have to provide name, email and their organisation. If a user has an existing Pathbase account he can log in with his user-id and password.

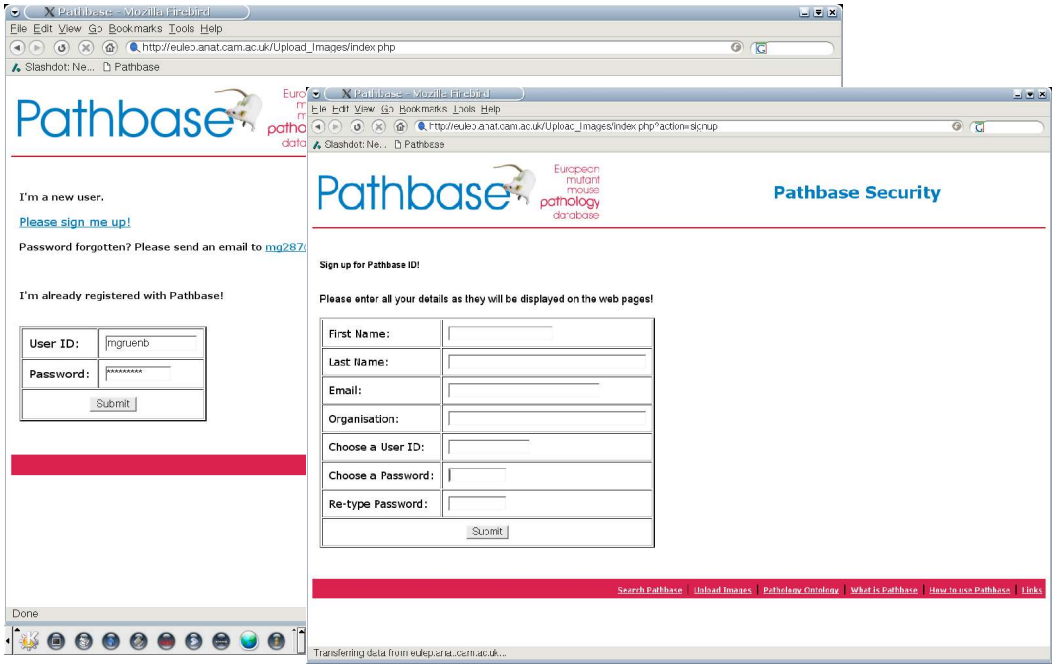

Figure 2: User sign-up / sign-in screens

Now the user can upload his data. Currently the user has to fill in the 8 mandatory fields and has a choice of which of the 11 optional fields he wants to fill in. User are also encouraged to use the ontologies for the various fields, however there is no user-friendly way to force the user to use the ontologies and most of the fields basically allow free text entries. If (mandatory) data is submitted that can't be processed an error message is displayed and the user is asked to use the ontologies.

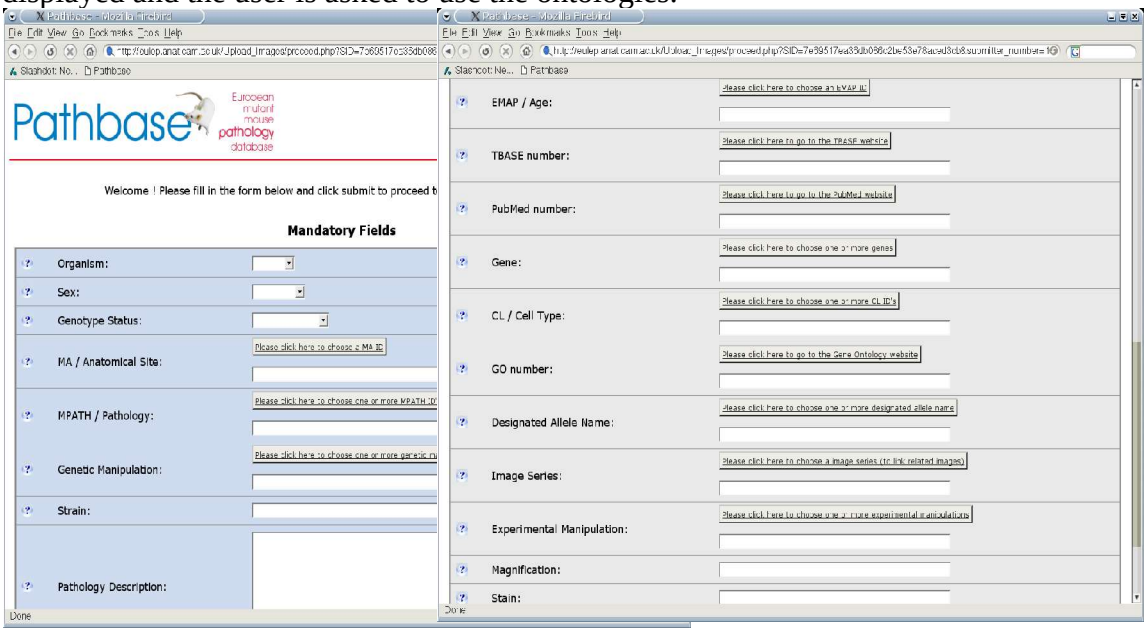

Figure 3: Data upload form

After the data has been submitted, the user is presented with an overview of his submitted data and asked to submit his image file. The data is inserted in one database transaction in which first all the data in the image_data table is inserted in order to create a new image_number, then this image_number is used to insert the data in all other tables.

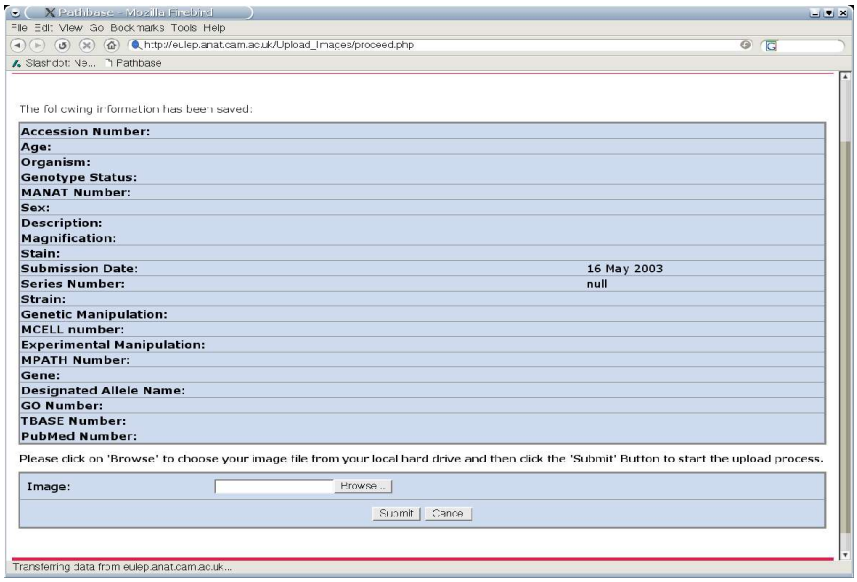

Figure 4: Submitted data and image file upload screen

Most common graphics format are supported for the uploaded images and automatically converted by a PHP script which calls the ImageMagick libraries for a) converting images to the JPEG format and b) creating a 300x200 pixel thumbnail version of the image. After submitting the image file, the user is taken back to the data upload screen with a thumbnail of his uploaded image and the upload form loaded with his previously submitted data. This allows for quick uploads of multiple images with similar data.

The image is not yet accessible via the search interface. It is marked as 'not checked' in the database and has to be reviewed by a Pathbase administrator.

Step 2 Image administration

The Pathbase administrator interface works similar to the search interface (see below) with some additional search fields:

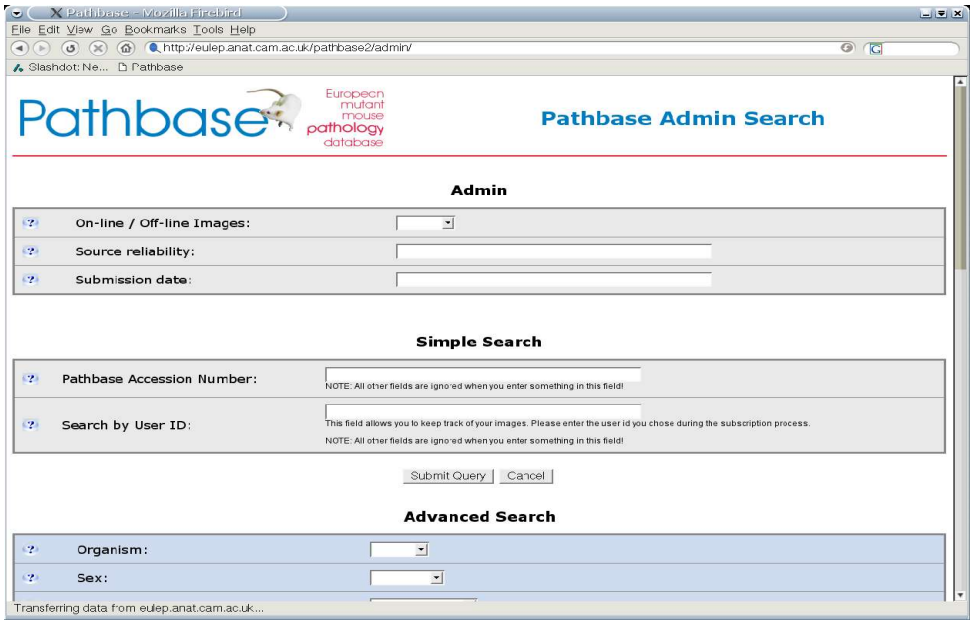

Figure 5: Pathbase administrator interface

The additional fields let the administrator search for on- /off-line images, search by source reliability and by date. The administrator search results are displayed the same way as the normal search results. However, instead of displaying a factsheet after clicking on a thumbnail, an administrator interface is displayed (similar to the upload form).

The administrator can change data here, upload a different image or delete the image data. When image data is modified, the data is first delete and then re-inserted except for the data in the image_data table. This step is necessary because of the possibility of multiple entries in some tables.

Step 3 Image Search

The image search interface is similar to the administrator interface.

The search has some advanced features:

When searching for a term in the ontologies, the user can either use the pop-up windows and browse through the ontologies or he can enter a GO-id or a term name. The search algorithm tries to find as many results as possible and also includes all lower levels of the hierarchies. Only on-line images are displayed, except when the user searches by 'submitter id', then off-line images are displayed as well. This way users can check their submitted data.

The results page shows some basic information and an image thumbnail:

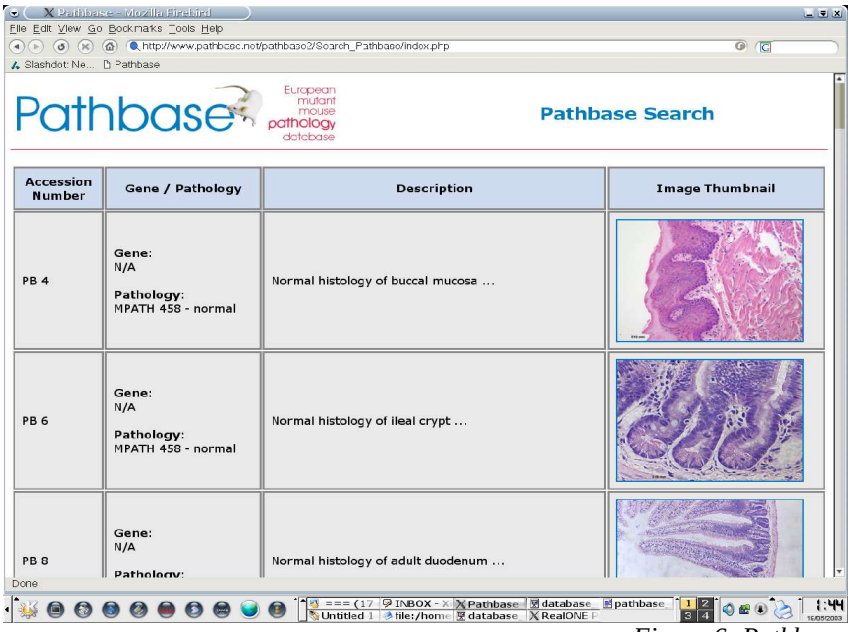

Figure 6: Pathbase search results

The user can now click on the thumbnail image to view all the details:

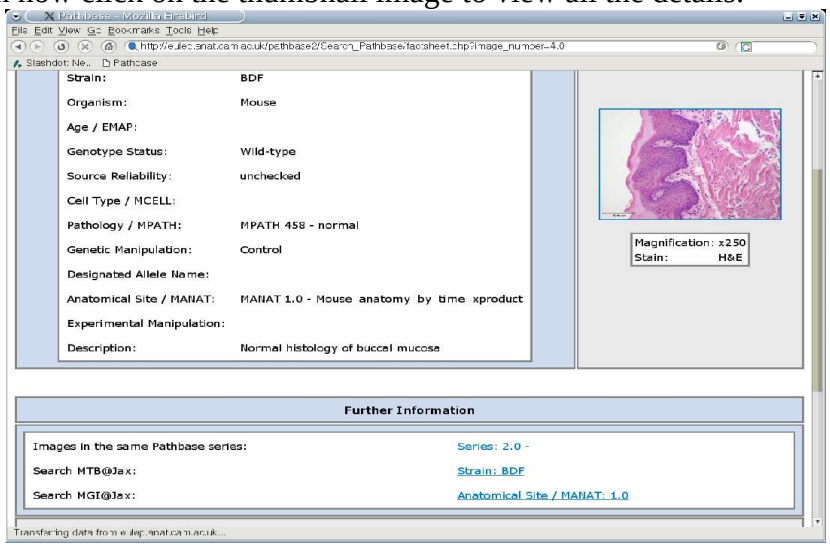

Figure 8: Pathbase factsheet

From the factsheet the user can now either view the full-size image, add a comment to the image or click on a link to MTB, Tbase, or PubMed. The links to external databases are created from the stored information and the search queries to those databases are submitted via data in the URL. While this is working satisfactory, it will be necessary in the future to implement some way for the various biological database to connect and to query each other. The disadvantage of linking via HTTP URL's is mainly that when the URL changes all databases that use this URL will be broken. This is also true for the data

which is submitted via the URL. A short discussion with some suggestions for various available technologies follows in the second part of this report.

Technical Details

The server architecture structured in the following way:

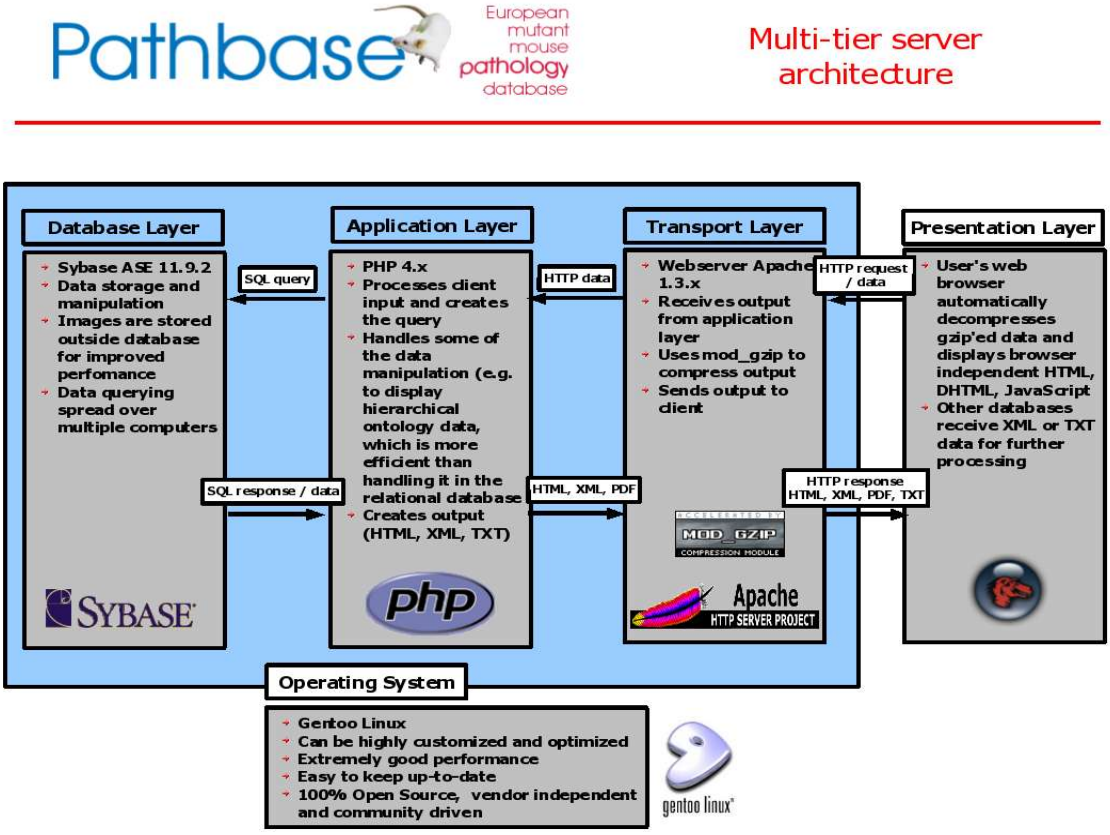

Figure 9: Pathbase server overview

Database Layer

Pathbase is currently using Sybase ASE 11.9.2 as database backend. Because of features like transactions and joins Sybase seemed to be the most mature and advanced database server for Linux when we first set up Pathbase in 2000. However, Sybase does not offer any up-to-date database servers for Linux for free (for educational use) and Pathbase 3 will probably use a different database engine as back end. PostgreSQL and MySQL are the two open-source database servers which are now mature enough and which are still in active development and make use of the advance d features of the Linux operating system.

The Pathbase database schema can roughly be divided into two sections:

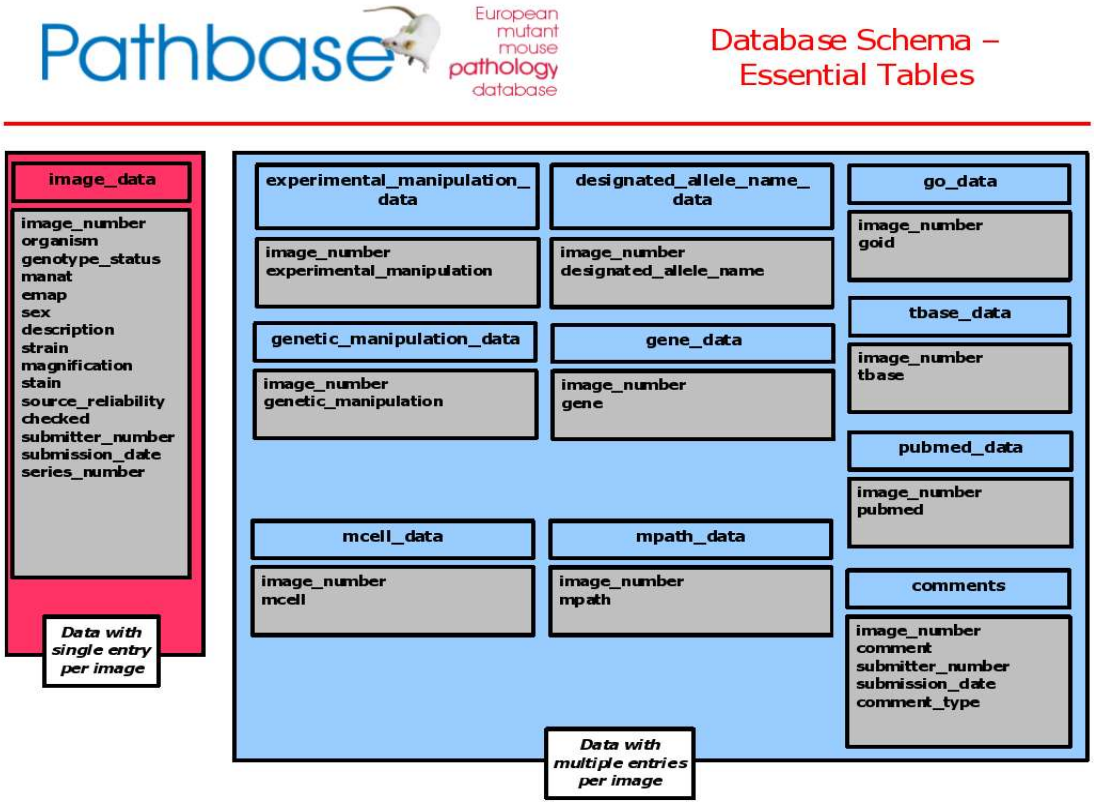

Figure 10: Database schema - Essential tables

The above diagram shows how the data is stored which is directly related to images. The image_data table stores all the data which has single values per image, for example each image can only have one organism associated with it. The image_data table is also the 'master' table: All other tables link to this table via the image_number foreign key. All of the other above tables can have multiple entries per image, for example an image can be associated with many Tbase id's.

The tables mcell_data and mpath_data as well as the emap and manat fields in the image_data table only store the ontology id's and provide the link to the ontology tables which are the second part of the database schema:

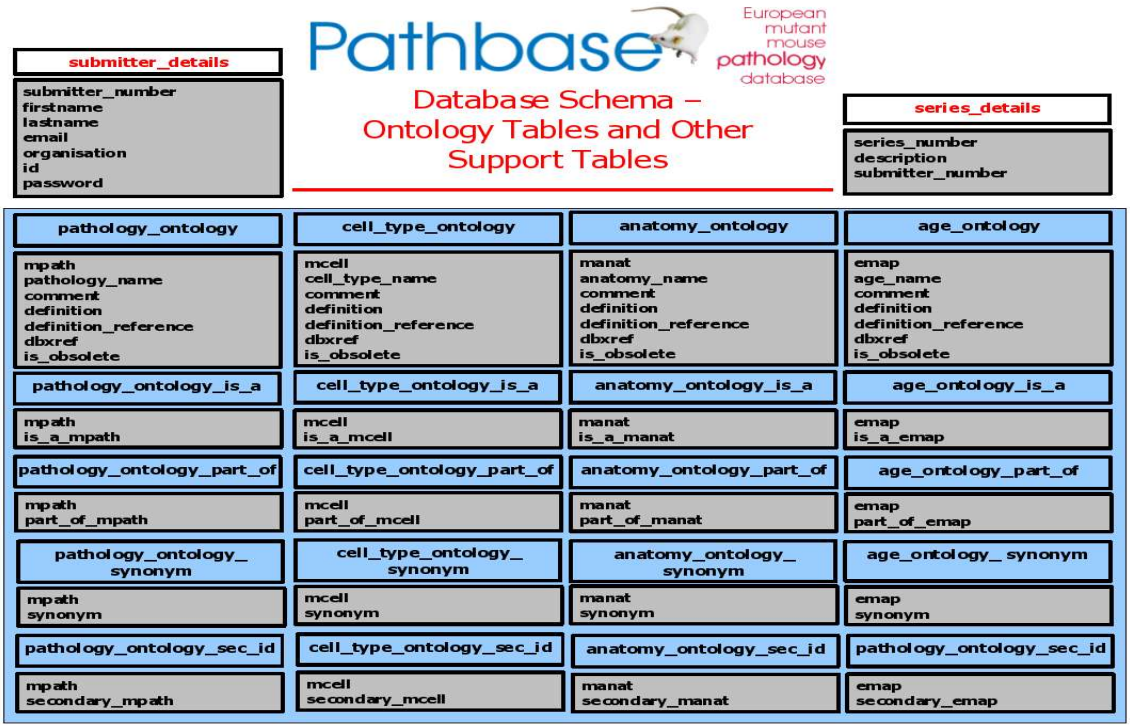

Figure 11: Database schema – Ontology tables

The second part of the Pathbase database schema shows support tables with submitter_details and (image) series_details as well as tables for the four ontologies: pathology, cell type, anatomical site and age. Each ontology needs five tables to store it's data efficiently (only explained for pathology):

- pathology_ontology (stores general information about a term)
- pathology_ontology_is_a (stores parent-child relationships, can have multiple entries)
- pathology_ontology_part_of (stores parent-child relationships, can have multiple entries)
- pathology_ontology_synonym (stores synonyms to a term, can have multiple entries)
- pathology_ontology_sec_id (stores secondary id's, can have multiple entries)

  This set-up allows for child nodes to have multiple parents with different relationships (DAG's).

As databases aren't very efficient in reading BLOB's Pathbase doesn't store it's images within Sybase but it stores it as files on the file system which the web server accesses directly, i.e. without querying the database.

Application Layer

The application layer is written in PHP and works as the 'glue' between the database and the web server.

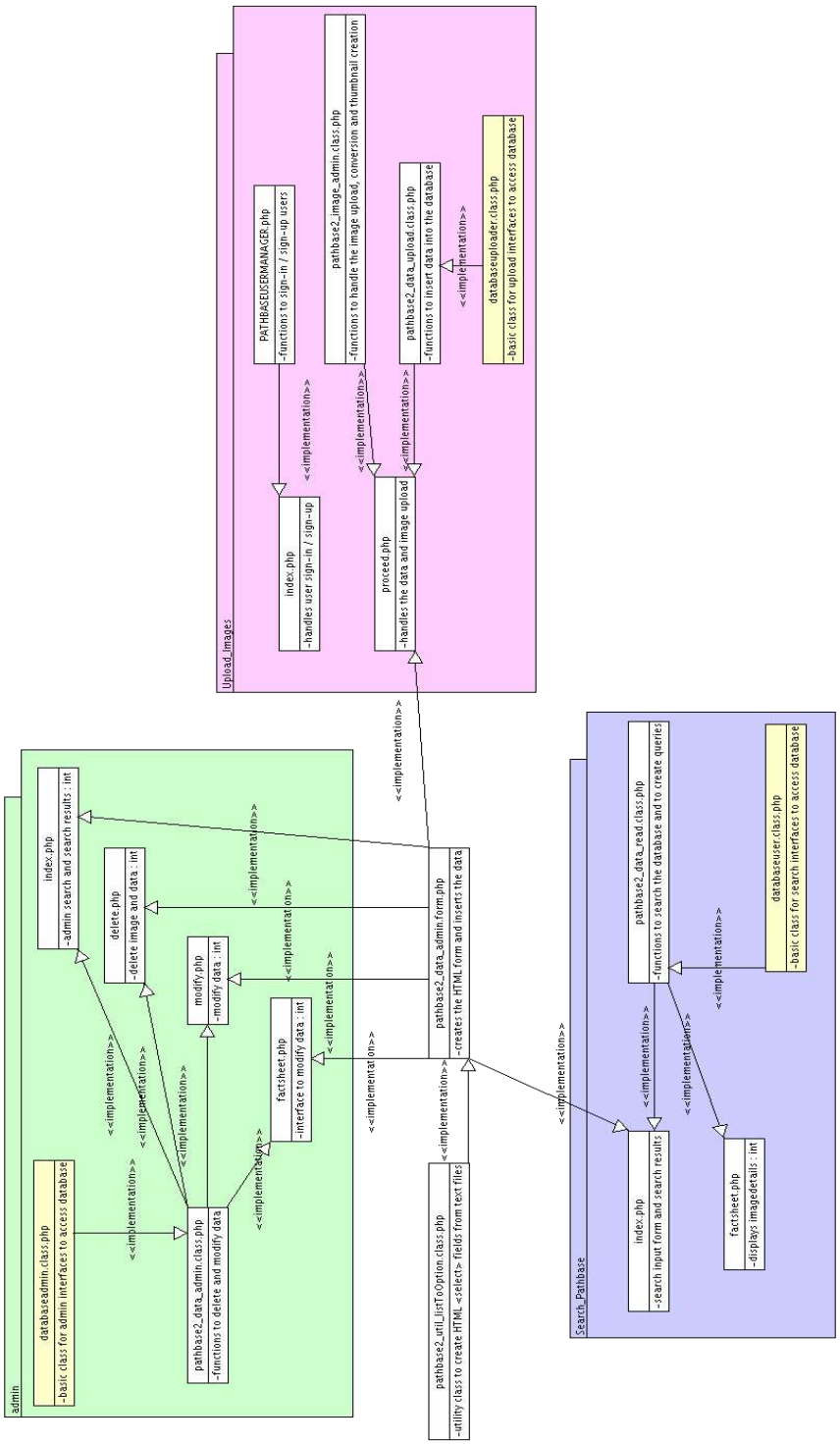

Figure 12: Pathbase class diagram

The PHP files and classes are similarly ordered as the Pathbase process: Image upload (pink), administration (green) and search (purple). Each area uses a database class (yellow) that handles the connection to Sybase (Login, sending query). This set-up should allow for a fairly seamless transition to another database engine in future such as MySQL or PostgreSQL. However, it wasn't possible to use this set-up for all connections, because of a bug in PHP that doesn't allow Sybase connections in a script that opens a file handler before the connection is opened.

Most PHP scripts also use the pathbase2_data_admin.form.php script(at the centre of the diagram) which is the script that creates the forms for uploading, searching and administering image data. It also handles the automatic filling-in of the form, e.g. after uploading each image or for the administrator interface. The data however has to be provided by the implementing script.

Each area has a main class with functions to process data and to create database queries. These classes are then implemented by various scripts in each area. These scripts also use the web site wide HTMLBASE.class.php script which creates the layout and look-and-feel of www.pathbase.net

The scripts and classes handling the ontologies aren't shown in the above diagram, because they aren't directly implemented by any of the above scripts. The ontology scripts are only called as pop-up windows from links in the upload, admin and search forms. There is also a admin interface for the pathology ontology, which is maintained by the Pathbase team. The administrator has full control over the ontology and can make terms obsolete, move them in the hierarchy and add or change term names, definitions and synonyms. All ontology scripts use a central pathbase2_ontology.class.php class. All ontology scripts are fairly generic and also the database tables for all the ontologies used in Pathbase have the same structure, so it is now very easy to create interfaces for all ontologies.

Except for the pathology ontology all other ontologies have to be updated regularly with files from www.geneontology.org These files have to be loaded into a customized version of DAG-Edit, a Java program that allows for working with ontology files. The files can then be exported from DAG-Edit as RDF-XML files. These RDF files can now be handled in PHP with OpenSource PHP RDF parsers and the data can be loaded into the database. The advantage is clearly that no custom parser for the ontology files had to be developed. The disadvantage is that the standard DAG-Edit version doesn't export synonyms and descriptions in the RDF output and that we had to add some Java code to DAG-Edit to handle this functionality. As this is a fairly complex process and doesn't always work as intended (due to corrupt files or bugs in DAG-Edit or the PHP-RDF parser), there is no administrator interface and the loading of the ontologies has to be handled by a Pathbase developer (first on a development machine and after verification on the server).

Transport Layer

The transport layer is handled by a fairly standard installation of Apache 1.3.27 with mod_gzip. mod_gzip is especially beneficial for users with a slow internet connection and for overseas users as it compresses all data on the server before sending it to the client. Most modern browsers are able to handle and decompress the received data on-the-fly. The web server also handles some of the security of the web site, e.g. The password protection of the administration area.

Presentation Layer

The presentation layer in this case is the users web browser or another database server. Pathbase sends standard HTML and CSS which every modern browser should be able to process and display correctly. JavaScript is used for the pop-up ontology windows and has been tested successfully on most modern browsers. We are also currently implementing a DHTML help system that displays some instructions to users when they mouse-over input fields.

Operating System

Linux is the obvious choice for any web based database due to it's low cost and high security features. The Pathbase server used to be running SuSE Linux 7.2, however due to the short support period of 2 years we migrated the server to Gentoo Linux. Gentoo Linux is a unique new operating system, which is community built and community supported. This means it's not dependent on any company and updates and security patches will always be available. The operating system as well as most applications are compiled from source (where the source is available) which allows for a very high customisability and extreme high performance. Gentoo also has an advanced package management system and updating the system is as easy as entering one command.

The disadvantage is the long time it takes to install packages, because some packages are quite large and take several hours to compile. This disadvantage becomes smaller the more processors are available for cross-compilation and should become neglectable once we migrate more development machines to Gentoo. Another disadvantage is the knowledge required to install Gentoo as even the Linux kernel has to be compiled from scratch and all services have to be customized, however once installed it is a very easy to maintain system.

Apart from web server and database server we are also running a FTP server and a SSH server. Both are not publicly available and are only accessible from some restricted machines.

Other Servers

Backup server

System backups are done regularly (weekly) by copying the whole system to tapes. However, to ensure the safety of data in case of a fire, a backup server which is physically located in a different building copy's the most important data every night. This includes all database files, images and web files (PHP, HTML, etc).

The backup server has a fairly old processor with low memory and migrating to Gentoo Linux would not be advisable, so it is still running on SuSE Linux 7.2 . It is only running a SSH server for remote control and uses the FTP server on the main server machine to copy all backups. As our data amount grows it will probably become necessary to move to a different protocol such as RSYNC, which only copy's data which has changed since the last backup instead of copying everything.

Second server

The second server is currently being installed and will basically be a lower spec mirror of the main server. This means it can take over some queries from the main server during high load times or during maintenance / repair times when the main server has to be off-line. The final decission on how to connect the two machines hasn't been taken yet and we might use SOAP or XML-RPC to connect both machines. This would provide a good testing ground for connection to other database servers. This might add too much overhead and a simpler method might be used, e.g. to use one machine for the ontology queries and the other machine for the main search queries.

Future Development and Technology

Pathbase is at the moment at the forefront of development in bioinformatics databases and seems to be the only database to have successfully implemented ontologies to store it's data. It uses modern, easy to use web interfaces developed on the latest web technologies. Development in the next few months will focus on improving the existing technologies, testing and securing the data input and to increase performance both in the database layer and the application layer. Further development will the focus on the following three areas:

Distributed Sequence Annotation Server

"*The distributed annotation system (DAS) is a client-server system in which a single client integrates information from multiple servers. It allows a single machine to gather up genome annotation information from multiple distant web sites, collate the information, and display it to the user in a single view. " (www.biodas.org)*

The biodas.org website currently specifies the DAS/1 protocol and cites the DAS/2 protocol as 'in development'. According to the DAS/2 specification it will (probably) be

based on SOAP which would make the deployment for Pathbase much easier. Currently, the DAS/1 protocol requires a dedicated server written either in Perl or Java and it is optimized for a MySQL database. This would mean that we would probably need a dedicated server machine to run the Pathbase DAS server and it is probably advisable to wait with the DAS deployment until the final DAS/2 specification is available.

Data exchange with other bioinformatics databases

A large number of groups have expressed interest in a closer collaboration with Pathbase and a closer integration of each other's databases. In order to use the data stored in various database's and database systems a common agreement on which protocol should be used is necessary. There are already a number of standards available which would make sense for exchanging data in the bioinformatics field:

XML-RPC

XML-RPC is a protocol that allows applications to make remote procedure calls. This protocol allows an application written in any language running on any operating system to use a procedure written in another language on another operating system. XML-RPC encodes the RPC request into XML and sends it over a HTTP connection to a server. The server (or listener) decodes the XML, executes the requested procedure and the packages the results in XML and sends them back to the client. The client decodes the XML, converts the results into standard language data types and continues executing.

The advantage of XML-RPC over, for example Java RPC, is it's language independence and implementations in most modern languages (including PHP) exist. Another advantage is it's simplicity and ease of use.

CORBA

CORBA is the acronym for Common Object Request Broker Architecture, OMG's open, vendor-independent architecture and infrastructure that computer applications use to work together over networks. Using the standard protocol IIOP, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network. Corba is very similar to XML-RPC but much more complicated to implement. It does offer more powerful functionality, however, this is probably not needed in our case, as the focus will be on exchanging data and not on distributing computing processes.

SOAP

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to

process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. SOAP defines more closely than XML-RPC how the data is exchanged, but it also offers more flexibility as to which transport protocol is being used. XML-RPC is about simple, easy to understand, requests and responses. It is a lowest common denominator form of communication that allows you to get almost any job done with a minimum amount of complexity. SOAP, on the other hand, is designed for transferring far more complex sets of information. It requires profuse attribute specification tags, namespaces, and other complexities, to describe exactly what is being sent. This has its advantages and disadvantages. SOAP involves significantly more overhead but adds much more information about what is being sent. If you require complex user defined data types and the ability to have each message define how it should be processed then SOAP is a better solution than XML-RPC. In my view SOAP would be the ideal system for exchanging data between bioinformatics databases especially as DAS/2 will probably be based on SOAP.

Transition to a new database server system

Sybase 11.9.2 isn't very well supported in the latest versions of PHP and it lacks modern features for example in handling XML. Sybase doesn't provide any free-for-educational-use databases any more and therefore a move a different system will be necessary. An OpenSource database system would be ideal as they are vendor independent and there will always be up-to-date versions available. The two major database systems that might be used for Pathbase 3 are:

MySQL

MySQL was designed with performance in mind. It is probably the fastest database available especially when running mainly 'SELECT' statements, which is usually the case for web backend databases. MySQL currently doesn't support sub-selects, foreign keys and transactions which might be quite a problem for use for Pathbase as all three features are quite essential. Especially sub-selects are essential for searching the hierarchies in the ontology tables. All of the above issues are currently addressed by the MySQL developers and it's only a matter of time until they are implemented.

PostgreSQL

PostgreSQL supports all of the features which MySQL currently lacks, it is fully ACID compliant, however many benchmarks have proven that it is significantly slower than MySQL. Another advantage (proven in many benchmarks) is it's better stability over long periods and under high load.

Currently, it looks like PostgreSQL would be a better choice, especially as AmiGO is

now being ported from MySQL to PostgreSQL. However, DAS servers are designed for MySQL and we have to wait and see what our own benchmarks for Pathbase reveal and how development on both systems progresses.