University of Southampton
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

A project report submitted for the award of B.Sc. Computer Science

Project Supervisor: Hugh Glaser
Second Examiner: Michael Luck

Management of Research Project Websites

By Paul Richards

10 May 2001

Abstract

The use of the Internet to transfer and display information from anywhere on earth to anywhere else is becoming part of everyday life. A number of research projects are performed each year, in many cases the Internet is used to facilitate the transfer of results to the wider community. However the websites relating to these are often out of date or incomplete. University Researchers may not have sufficient time to manually generate a web site therefore it is proposed to provide a tool to generate a site from underlying data. Existing tools and websites were investigated to identify the structure of a ‘typical research project’ and evaluate existing techniques. A tool was then created to generate a web site from the underlying data of a typical research project. It was found that it would be possible to generate a tool, however it is likely that to provide advanced features, either customisation or clever use of HTML would be required. It is probable that our tool would be used to facilitate the modification of a database, and to provide a starting framework, from which a more complex web site could be generated.

Table of Contents

Acknowledgements

Thanks go to Hugh Glaser for his guidance and support in the creation of this project. Finally the author would like to thank his family and friends for their support and patience during the creation of this project.

1 Introduction

The use of the Internet to transfer and display information from anywhere on earth to anywhere else is becoming part of everyday life. The aim of universities is to "Further Knowledge". Every University around the world performs a number of research projects each year as part of this aim. The Internet is used to facilitate the transfer of this information to the wider community.

The accessibility of information about university research projects is limited by web pages which contain a limited amount of information or are out of date. University Researchers display their information on the web depending on their own ability to design a website and their time constraints. This project aims to develop a tool which will attempt to make it easier and quicker for a Researcher to implement a project website, and to make the website more accessible to both the researcher and the visitor.

The initial project brief set out the primary objectives to develop a tool which will help a researcher to build a website for a typical research project. A study was carried out to identify a model for the web presence of a typical research project, which provides a schema for the information contained in a research project website. The secondary objective of the project was to extend this to allow multiple projects and sites to be managed where multiple users may be involved.

In section 2, we outline some of the problems web developers face and identify existing systems that have been developed. We also discuss some of the choices behind the technology used in this project.

Section 3 and 4 introduces the architecture of the system, and of a 'typical' research project web site. The modules implemented in the development of the system are discussed.

Section 5 concentrates on the testing of the system and discussing the updates that resulted from this.

The experiences from the project are summarised in section 6, where the report is concluded. We also discuss the limitations and usefulness of attempting to build a website from relying on the data and data structure. We also discuss on the extendibility of using scripts, for example PHP scripts in this case, to allow extra features to be implemented or modification of our scripts to easily generate a different styled site.

A table of acronyms used within this report is provided in Appendix A.

2 Background

2.1 Existing Sites

The core site of most universities will contain links, which minimally give details of current research projects taking place. However the information provided is dependent on individual departments and researchers. As a research project progresses it is not always the case that the website is updated, or all information is displayed. The aim of universities is to research new ideas and to then share this information, the Internet is an ideal way to share this gathered knowledge. If a project web site does not contain current, or only contains brief information, then the Internet is not being used to its full potential. This project aims to provide a toolkit which will allow researchers, whatever their ability, to be able to generate a detailed website for a research project.

By analysing a large number of university web sites it has been possible to produce a model of a research project website by identifying entities within existing websites. This information has been abstracted from university websites in both the United Kingdom and around the world [Appendix B].

From analysing the design and layout of existing sites, it has been possible to identify a number of distinct entities that might be included within the structure of a research project website, for example people (in different roles; support staff, researchers), projects, events and publications. Some of these entities will be related to each other, for example a researcher may be related to a project, but will obviously not be related to every research project. Therefore our tool is designed to attempt to relate an individual entity, rather then a group of entities, wherever possible.

2.2 Existing tools Research

The management of a web site has always been a difficult task. The WWW is an open-ended system, with multiple standards available. It is distributed over a vast number of computers and has no inherent structure [20]. The analysis of the web by the ARANEUS team points out [17] that although the web itself has no apparent structure, individual sites are often fairly well structured and therefore ideal for database techniques. A number of research projects exist which look into database management systems. Existing commercial tools tend to be aimed at managing files, rather then managing data. They facilitate graphical presentation and represent a move away from ‘hand-crafted’ pages, where the author inputs every symbol and character.

Previous work [16] has reviewed and classified 33 exiting commercial products based on their impact on the development process. They group tools into five basic categories, from visual html editors and site managers to Model-driven application generators, which includes only 1 product. A number of solutions are available for generating database based web sites, although most of these are quite complex, and require development work and coding by the user to generate pages. A number of the existing solutions propose integrating HTML with a database programming language

(DBPL), which then requires significant developer effort before results can be produced.

A significant amount of the existing research has related to the problems of managing and querying information on the web with database techniques. Other technologies including Artificial Intelligence are important in this process. A survey of existing techniques identifies three classes of data techniques for the web. For the purposes of this report, web site construction and restructuring is the important technique. The other two techniques, querying the web and information extraction and integration are not so important in the context of designing web pages. [15]

The ARANEUS project [2,12,17,17] web-base management system developed at the Università degli studi di Roma tre was created a manage database data and HTML documents. A web-base is referred to as a collection of highly structured data e.g. typically stored in a database system or semi-structured data in the web style.
This system implements several languages and models for the definition and manipulation of data; a data model called ADM, languages called PDL and PML. The system is implemented in the java programming language, and therefore is able to run on a number of platforms. The interface is written in HTML, and so can be accessed from a number of different clients around a network. A database, for example a relational or object-orientated database, is used to store objects from the ADM data model. PDL is then used to specify how pages are to be mapped onto the underlying database, including complex features e.g. image maps/forms. PML is then used to generate or remove a site or individual pages.
A further tool is used to specify the graphical presentation. Further development being performed is related to the project and XML, and to dealing with adding services and application support in further tools.
The ARANEUS projects aims to reduce the complexity of managing a site by emphasising a design methodology, with distinctive levels.
So far, it appears that ARANEUS has been used for generating a description of a site through reverse engineering, rather then building new sites, which is the aim of this project.

A similar system called STRUDEL [14] is presented by AT&T Labs [7], for specifying and generating data-intensive web sites. This also separates the tasks of generating a web site into accessing data sources, building its structure and generating its HTML representation. A language called StruQL is implemented to support the first tasks, and a template language for the HTML presentation. The authors do acknowledge that one common criticism, and other specific languages, is that they require a user to learn a new language to solve a problem. They argue that the long-term benefits will outweigh the short-term cost of learning the language. For a large website, this may be true, however for a single researcher implementing a website for his research projects, it is unlikely he will see a large benefit. When developing a web site with STRUDEL, a designed would start with the raw data, build a data graph, a model of the raw data, and declaratively describe the content and structure of the site [13].

A third similar system, TIRAMISU developed by the university of Washington [10], takes a slightly different approach. They acknowledge that users may already be familiar with using web site implementation tools. After the designer completes the

high-level definition of the site, they are then able to choose which commercial implementation tool to use with different parts of the project. A site schema editor represents the site structure. A data management component interfaces with any data sources. This is similar to the systems already discussed. The unique feature of this system is that it does not itself create the web content. Instead an implementation manager coordinates through an API, a set of external tools that perform the task, e.g. Microsoft FrontPage.

2.3 Analysis of the problem

The core of a university website will generally fall under a specific style, or layout. As we move towards research projects, the style and layout is often that of the researcher. Similarly, depending on the researchers skill and time commitments, the project website will contain varying amounts of information. However each project generally displays the same information.

Searching any of the popular web index servers, such as AltaVista [1], it can be seen that a large number of sites, after being established are almost completely abandoned, that is, they present numerous errors and inconsistencies due to poor maintenance; their information is often obsolete.

In a number of cases this is due to the fact that managing large collections of files, HTML pages, does not scale well: due to the URL method of identification used, missing pages and inconsistencies in links often occur.

This project aims to provide a toolkit which enables this information to be stored in a database, which is often the case anyway, and to then generate a website from this, without the need of a researcher to spend hours on HTML development.

HTML does not define a structure to a document. This means that it is difficult to produce web pages that are consistent across a site, especially if different commercial programs are used to generate the files.

It is hard to maintain a site that has a poor organisation, and similarly it is difficult for the users to navigate. This is the justification for the flourishing and large number of tools for the automatic generation of HTML pages based on common specifications.

Browsing through a series of projects, even within the Computer Science Department at Southampton illustrates some of the problems mentioned above. For example the Communications group web site (http://www.comms.ecs.soton.ac.uk/) visited in May 2001 contains 3 areas of interest; one being a blank page, under development and last modified in March 1999. The latest page on there web site appears to have been modified on 26/7/2000, which is 10 months old. This illustrates why our toolkit would be useful. However the new Intelligence, Agents, Multimedia group, which was only recently established, has a design for its web site that has features that look as though it has been created through a custom script.

The following functional requirements were identified for the toolkit:

- It must be simple enough to use that a researcher that has not had time to build his own website, would be able to quickly develop a site.
- It should be possible to customise the underlying data format for different projects. It should therefore be possible to extend the toolkit from Research sites to other sites.
- The style of the output produced should be customisable and easy to modify if additional features are required.
- A graphical interface should be provided which allows users to accomplish these tasks.
- The HTML generation part of the toolkit should be modifiable so that advanced users can extend or modify the output produced.

2.4 Available technologies

There are a broad range of technologies available which could be used to implement this toolkit, ranging from desktop applications to web based environments. The tool we are proposing to design could be implemented in a vast number of languages, however some offer benefits for development, and ease of use. We are also potentially dealing with the storage and manipulation of a large quantity of data, which needs careful consideration.

2.4.1 Data Storage

The key element in this project is the data, from which a website will be generated. There are a number of possible methods that can be used for dealing with data ranging from text files to database solutions. Although a software solution could be developed for accessing any type of data source, some methods offer benefits. Ideally the data should be stored, such that it can be easily accessed by a researcher or a system administrator, similarly it should also be related to existing university systems (and software).

The simplest method for data access by a researcher would be using ordinary text files, e.g. Comma Separated Variable (CSV) files. A unique file would be created for storing group of entities. This method has the benefit that the files can be read or imported easily into other applications however the file is not structured; it is either the software's or researchers responsibility to ensure the format is correct and valid. This adds complexity to the software, and may also result in duplicate data, as each researcher maintains he's own list of files.

A database solution using relational database technology may be more suitable for a data-intensive website such as this. Two popular SQL databases exist; MySQL [3. A real validator- software by L. Quinn– http://www.arealvalidator.com

4] and PostgreSQL [6], both of these are available freely and are very popular. Many universities will support or use one of these two database servers. Therefore a tool that supports these could be easily implemented within a university without any additional load on support staff. Similarly it may be possible to generate web pages

from the data contained in multiple databases, therefore reducing inconsistency and wasted effort.

MySQL is regularly used due to the excellent performance that it returns, compared with PostgreSQL. The PHP scripting language discussed later supports both of these databases. The limitations of MySQL performance come when the server is under a large number of simultaneous connections, where the database software collapses. Similarly MySQL can only lock an entire table during operations. It is unlikely that these limitations would effect a university web site where the load is likely to be less, and the web pages could be generated or cached statically; the information on a project is likely to change on a daily or weekly basis at most.

PostgreSQL is also an excellent server. However it does have a couple of limitations. There is a limit of 8 – 32kB of data per row. For large text entries, this is a problem; the PostgreSQL development team have fixed this bug in the recently released version 7.1 on 13/04/01. Auto increment fields in PostgreSQL, could confuse users when re-creating tables, and importing data, as the sequence numbers are not reset, causing confusion to new users and inserts to fail. PostgreSQL does however have the advantage that it supports foreign keys, for validation checks, and also transaction support, which are being developed for MySQL.

As we are looking at potentially storing large amounts of data, a research paper for example, the limits of PostgreSQL become more important. An article comparing these two databases suggests that the developer can overcome the limitations of MySQL, and also that PHP, discussed that, support for MySQL has a couple of extra useful functions.

XML was also considered a possible data storage method. However as we are trying to generate a web site just from the data source, we may need to impose some restrictions on the data. XML is referred to as semi-structured data. Semi-structured data is characterized as having few type constraints, irregular and an evolving or loosely defined schema [14]. For example XML elements may have missing attributes. Although we do not want to impose that an attribute is defined, a relational structure does allow us to prompt the user for information, and to specify the type of the information we want.

Overall, MySQL is the data storage method that we have chosen to follow in this project. However, as the database interface could be implemented in a modular fashion, there is no reason why the code could not be modified to support PostgreSQL, XML or another SQL server. MySQL is supported within the department, and for the reasons discussed above is chosen over the PostgreSQL server.

2.4.2 Development Language

There are a large number of languages that are available for web development. These range from programming languages to scripting languages. Perl is a fairly standard language for web development. It is a very powerful language, supporting regular expressions, and with a large number of modules available to extend itself.

PHP is a nice language designed specifically for web and database development. Compared to Perl it is quite user-friendly. There is some support for classes, which would be useful in developing our tool kit. PHP can be included within a standard HTML page, so unlike c or perl, once does not need to generate a complete page, but can just ‘program’ a specific part of a web page. Active Server Pages are similar to PHP, however they are promoted by a commercial company, Microsoft Corporation, they are only supported on one platform and are not cross platform like PHP.

We have already suggested that we would like to use a multi-platform SQL server, and it is for this reason that PHP is chosen over ASP for a development language. PHP is preferred over Perl and C, because although it may lack advanced features, it has code support for databases, and it’s ability to incorporate itself in a standard HTML page, means that we do not need code to deal with file I/O, but just the core functions of out tool.

2.4.3 Other tools

So that the initial database schema could be developed, it was chosen to use Select Enterprise from Princeton Softech, Inc. This product provides a series of tools for designing software applications, and has been used to define the layout of the sample database. The Select SQL generator has been used to generate the initial tables, which were then modified manually for use with MySQL.

3 Design

There are two general classes of tasks when building data intensive web sites: one in which websites are built from one or more underlying data sources and another where they are created by restructuring existing sites. To perform either of these tasks a web designer must consider the following

- Choosing and accessing the data displayed on the site
- Designing the structure of the site i.e. data on pages and links between pages.
- Designing the graphical presentation of the pages.

Unlike some of the research projects already discussed in section 2 of this report, here the main step is to build a website by writing a series of expressions that declaratively specify the structure of the site. The system that is implemented, attempts to allow someone to generate a website through a graphical interface, and without the use of a declarative language. Similarly it is not expected that a researcher should define a set of expressions to deal with the graphical presentation of the site.

The first step of the design process was to perform an analysis of the layout of a number of existing project websites. From the resulting analysis a series of database tables could be produced to specify the structure of the site.

The second step was to develop an interface for modifying the tables within the database, and finally for generating html pages.

Although many ideas which are discussed could be applied more generally, we mainly focus on the case in which data regarding research projects is stored in a relational database to be published on a website.

As discussed in the previous chapter, it was chosen to develop the target application using PHP, which was configured as a module running on the apache web server. The web server had access to an SQL database running the MySQL server.

3.1 Research project structure

The starting point for the design process is a database conceptual design phase, where we have generated an Entity-Relationship (ER) model. This has been produced through the analysis of a number of research project web sites. Figure 1 shows a diagram representing the model of a typical research project website.

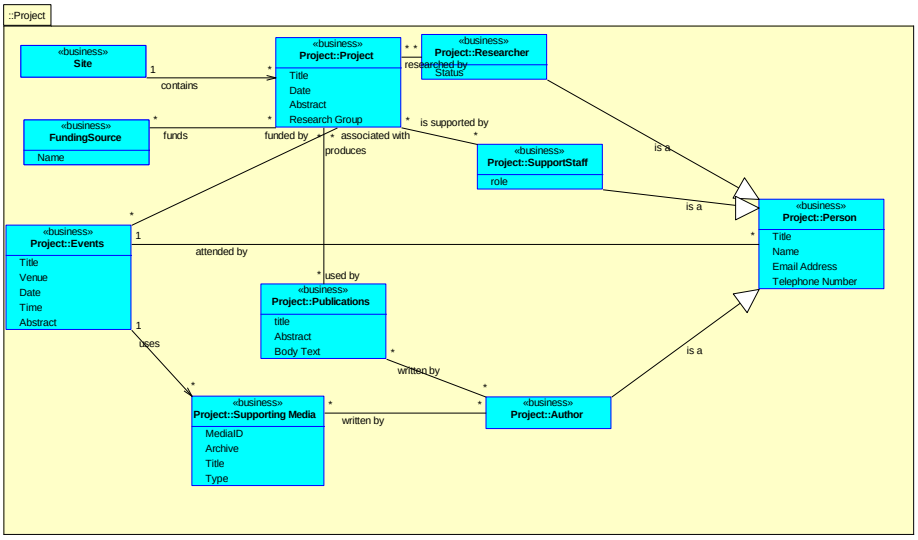


Figure 1: Entity Relationship model showing entities in a 'typical' research project. Full size version included in appendix.

3.2 SQL Schema

To generate an SQL schema to deal with the many-to-many relationships that generally exist within our model of a research project, we have generated a set of tables that consist of a composite primary key, and reference both tables. For example, in Figure 2 it can be seen that a table called Project_ResearcherLnk is defined which references the primary key in Project and Researcher. This is our method for the tool kit to be able to create hyperlinks in many-to-many relationships. Any table that contains a composite primary key of 2 keys, will be scanned and used to link tables rather then display them.

A table with a single primary key e.g. in Figure 2, the Project table, will be used to generate either a single page or a series of pages.

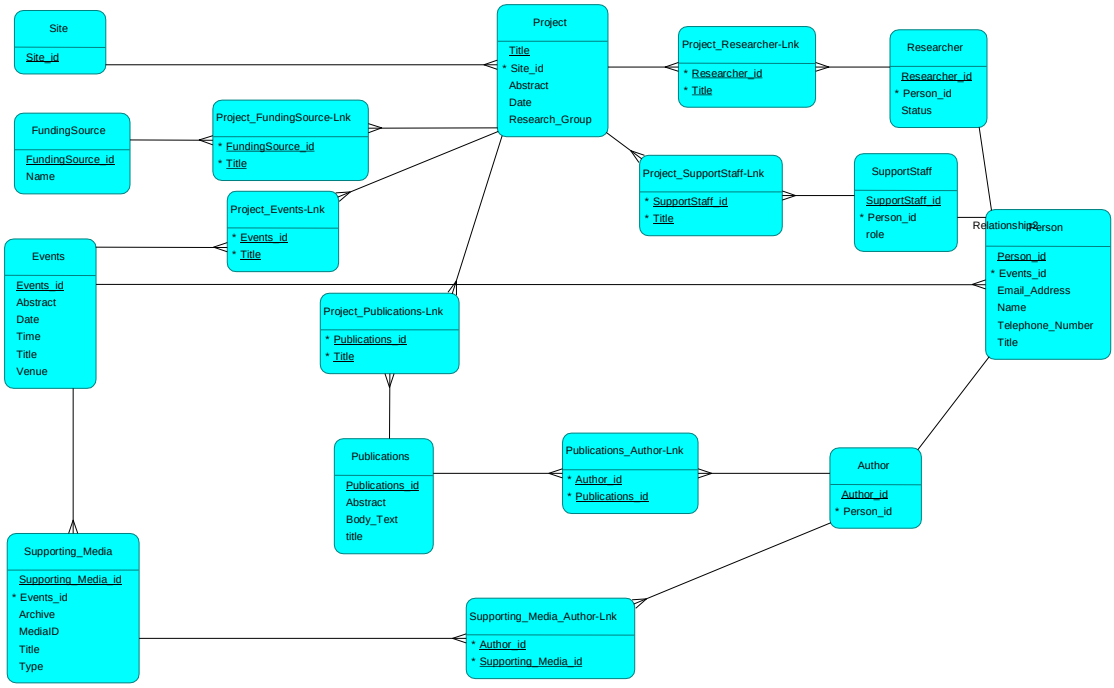


Figure 2 : Table relationships diagram of a typical research project. Full size version included in appendix.

The significant parts of figure 2 above are as follows. Tables are defined consisting of one primary key and a number of attributes. These tables define the data which will be generated into web pages. The second type of table has two primary keys and contains details of relationships between two different tables. It is from this table that the hyperlinks between pages in the website will be generated. The other key part is the relationship between a site and a project. The project table contains a foreign key to the site table. This allows us to generate a hyperlink between these two entities, while ensuring that a project can only be taking place at one site.

In this structure, the site table would typically represent the ‘index’ page, containing a list to the various projects under development. If we were developing a multi-site project, the database would either have to be restructured, if another table would make a suitable index page. Or, as would be more likely, the Projects table would be used as the index page. It is likely that the table would be restructured slightly, e.g. merge support staff and researchers together so there is one hyperlink to persons. Otherwise, the initial page would start getting reasonably crowded.

3.3 PHP Development

The PHP implementation attempts to abstract the various operations of the toolkit into different classes. In this manner, it would be easy, for example, to extend the script to communicate with different supported database programs and to re-use components. The other key thing with scripting languages for the web is that it would be quite easy

to modify the code to generate a different style of page if required. We now discuss a number of features that the source code provides.

3.3.1 HTTP authentication

There are two methods of authenticating users; either the web server may perform the task, or PHP. HTTP authentication is only supported in PHP when it is configured and running as an Apache module. Therefore for the HTTP authentication to work the code is limited to the Apache Server. However, it is still possible to use authentication through the server.

If PHP detects that external authentication is enabled for that page, then the PHP_AUTH variables will not be set, instead the $REMOTE_USER variable should be used to identify the authenticated user.

3.3.2 Database modification

Once a user is successfully connected, they are presented with a list of tables included in the database which are available for modification. The system only displays those tables that are allowed for user modification. Similarly the system identifies which tables can be built. It is down to the individual administrator to specify these options.

A series of HTML forms are generated from a single script, to allow a user to add, modify or delete entries from a table. These provide our interface for modifying the database.

4 Implementation

4.1 Overall Structure of System

As has already been stated, our tool kit is developed using the PHP scripting language. To allow for reuse of components a number of modules have been implemented.

4.1.1 Sqldb class

This class provides a skeleton interface that can then be extended for use by multiple databases. The class implements a constructor and functions which either set, or provide access to the hostname, username, password and database name to be used when connection to a database server.

The interface to be used when writing extended classes is also defined. For example there are functions to return the number of fields and number of rows returned by the previous database query. Similarly information is stored related to a current database connection, but it is unlikely that would ever be used from outside the class. The aim is to provide a separate interface for the definition of the database server. This simplifies the implementation of our project, so that it could easily be customised by a different user to provide a more sophisticated view for use in their website.

4.1.1.1 Mysqldb class

The mysqldb classs provides the support for the MySQL server, which is used within this project. It inherits the functions defined by the sqldb class and provides support for them in the MySQL environment.

The class does not implement persistent database connections. It is likely that any SQL links will be local to the developer. Persistent connections do not implement any additional functionality, however they are good if the overhead to create a SQL link is high. However as the database is used to generate a site statically, it is not likely that multiple people will use connections at once. The apache web server is designed such that it has multiple child processes, each one would create it's own persistent connection if necessary. In summary, it should always be possible to swap persistent and non-persistent connections without changing the behaviour of the script. The efficiency of the script will probably change by doing this. Whether to use persistent connections or not will depend on the configuration of PHP, apache and MySQL.

4.1.1.2 Other SQL servers

It should be quite easy to use the above interface with alternative SQL servers that PHP supports. The article[19] comparing MySQL and PostgreSQL, suggests that this would be fairly simple, achieved by replacing any mysql_query() functions with calls to pg_exec(). However by implementing a class for data access, it should be possible

to extend the application to other data sources, whether that is to a Microsoft SQL server or to a Microsoft access database through the use of ODBC.

4.2 Description of Algorithms

4.2.1 Page generation

It is necessary to decide how pages generated will be organised; we provide two distinct methods of page generation, single or multiple pages per table. We may think of generating a page for each instance of a RESEARCHER entity, whereas a single page could contain a list of all instances of the PROJECTS entity.

Meta data [9] can be used to provide information for use by search engines. However this will be dependant on each individual page. For some common Meta tags, these could be placed in a custom header. For more complex tags, for example description or keywords, it would be necessary to specify keywords individually for each page.

4.2.2 Generation of hyperlinks

As we are using the database as a basis for the generation of web pages, we need to generate hyperlinks from the underlying database. We make an assumption at this point that all primary key/ foreign keys have unique names in the database. This is to work around the current limitation of MySQL not supporting foreign keys. When choosing field names, unique fieldnames have been chosen for simplicity.

To generate a hyperlinks in a one-to-one or one-to-many relationship, the following pseudo code describes the functionality. The code described is the initial algorithms, which were used.

This piece of code deals with checking whether or not a non-primary key i.e. an ordinary key, is a foreign key in another table. If it is, table information is returned.

```
For each field in table being generated, do
{
        For each table except table being generated
        {
                for each field
                {
                        If primary key & field names match
                                Then get table.
                }
                if we have one primary key in table, return got table.
        }
}
```

Similarly we need to check whether or not a primary key is referenced as a foreign key in any other table. For example, a site_id may be ‘referenced’ by a number of projects.

```
For primary key in table being generated, do
{
        for each table
        {
                for each field
                {
                        If field names match, then return table.
                }
        }
}
```

It was then noted that there would be the cases where we would need to return a number of tables. Also it was noted that it returning the data would be useful, so that we can link to an exact entry on a page. This algorithm was therefore modified so that when the field names matched, an array of tables and data was created. This array was then returned by the function.

This routine was also updated to only return if there was a single primary key. Otherwise the function could return a table that consists of a composite primary key, and will never be generated by our html generator.

For a many-to-many relationship, we have defined that a separate table will be created. Therefore for each many-to-many relationship that exists, we need to look at the data and link, if necessary to the foreign table.

```
For each field in table being generated, do
{
        for each table
        {
                for each field
                {
                        if primary key then store field name
                }
                if 2 primary keys and 1 matches table being generated then
                        return table of other primary key, and associated data.
        }
}
```

It is possible to improve the algorithm described above my only checking the primary key in the table being generated. From our definition of a many-to-many relationship, we state there will be a table containing a composite primary key, where the fields are the same as a primary key in another table. Therefore, the algorithm does not need to be run on every field, which should improve execution time.

The main concern with this design is it is necessary to loop through the database to check for a number of conditions. Therefore there is a trade off between the ability to customise the code, and performance. For optimal performance, we would really want to merge the algorithms described above together, to minimise the number of calls to the SQL server. However, each type of link may want to be displayed in a specific manner, or the routines may like to be customised for a specific purpose. For this to be as easy as possible, the algorithms have been implemented separately.

5 Program Testing

Software testing forms an important part of the development of any software product. This is to ensure that the highest possible software quality is achieved.

Test plans were designed with the intent of finding any errors within the software. As we have developed a web-based system with the PHP scripting language, traditional errors, such as overwriting memory are not applicable. However, validation of links and HTML has been conducted to try to ensure compatibility with World Wide Web Consortium (W3C) Standards [8].
The next few sections describe the testing of the tool which was carried out.

5.1.1 Authenticaion and Session testing

Some testing was done of the PHP authentication code. The only problem with the authentication code is with regards to headers. For some reason after PHP was initialised the mysql class, php was writing headers back to the web server, so after querying a database for a connection, we could no longer send a 401 Authorisation required or similar HTTP error.

Code to handle individual sessions was not implemented specifically. The only reason for implementing session handling was to deal with someone clicking ‘back’ on their browser or submitting data to the web server twice. The problem with people using back, can be dealt with through using Meta refresh tags, with a zero time, to force users to a different page. Similarly through modifying headers to inform web browsers/proxy’s not to cache data, the need for session management is reduced.

5.1.2 Generated HTML validation

There are a number of tools available which provide both validation of HTML code and link verification. One of these tools, a real validator [3] was used to test the generated code to HTML 4 specification.
The first time the validator was run, a number of errors were returned, due to a missing DOCTYPE definition. This affected the validator however not web browsers which tend to work around missing elements in HTML. Before continuing any further tests, a DOCTYPE definition was added.

6 Conclusion and Future Work

6.1 Future Work

Future extensions of this project could involve allowing links to be in one direction only. The project could be extended so that paths are designed; transforming ER undirected relationships into directed relationships; for example, one could decide to allow the navigation from PROJECT to RESEARCHER and not vice versa.

6.2 Conclusion

This paper has presented our approach to build a research project website. However the ideas that have been developed and implemented could be considered of general applicability. It should be possible to generate a website from any database, with only a few modifications if any to the underlying database structure or scripts.

We have shown that it is possible to generate and make assumptions about web sites from underlying data source, however further work would be required to make the tool kit general purpose enough to design sites in different layout styles. Extending the tool could allow this, however this could introduce new problems. We have identified a number of ways in which the tool could be extended, however as these are implemented, they will add new tables and requirements to the database structure, which when we would like to be able to use an underlying database, then these modifications may not be acceptable.

As a stand-alone tool, unless additional features were added, although a suitable web site could be developed, it is unlikely that a researcher would want to further modify the generated HTML. The methods discussed earlier, implement declarative languages for the simple reason of allowing complete flexibility over site design. What we have however developed is a starting framework, which could be taking by a researcher or department, and with a few minor modifications, allows them to generate a web site in the style they want. For example, the generation code has been implemented such that any data elements are placed within a table. This could be easily modified to place the data in a list.

PHP, by its nature, allows a small amount of code embedded in a web page, to perform a complex task. Therefore it is likely that any individual who wishes to develop a website could use this tool as a starting point, and perform some small additions to the code to generate the features they would like to implement.

References

1. Altavista Technology Home Page – http://www.altavista.com

2. The ARANEUS System Home page. http://www.dia.uniroma3.it/Araneus/

3. A real validator- software by L. Quinn– http://www.arealvalidator.com

4. MySQL AB home page - http://www.mysql.com/

5. The PHP Group home page - http://www.php.org/

6. PostgreSQL Inc Home page - http://www.postgresql.org/

7. Strudel web site management system home page.
http://www.research.att.com/~mff/strudel/

8. World Wide Web Consortium (W3C) – http://www.w3.org

9. A. Anders. The why's, what's and how's of metadata.
http://nwi.dtv.dk/MD/whyMD.html (last updated: 22/07/1999)

10. C. Anderson, A. Levy, D. Weld. Declarative web-site management with Tiramisu. In Proceedings of the International Workshop on The Web and Databases (WebDB). 1999. http://www.cs.washington.edu/ai/tiramisu/anderson-webdb99.ps.gz.

11. P. Atzeni, G. Mecca, P. Merialdo Design and Maintenance of Data-Intensive Web Sites - Technical Report n. 25-1997 - Dipartimento di Informatica e Automazione, Universita' di Roma Tre, June 1997.
http://www.dia.uniroma3.it/Araneus/publications/rt25-97.ps.gz

12. P. Atzeni, G. Mecca, P. Merialdo. To Weave the Web - In Proceedings of the 23rd International Conference on Very Large Databases (VLDB'97), 1997.
http://www.dia.uniroma3.it/Araneus/publications/vldb97.zip

13. M. Fernandez, D. Florescu, A. Levy and D. Suciu. Reasoning about web-site structure. 1998. http://www.research.att.com/~suciu/strudel/external/files/from-alon.ps

14. M. Fernández, D. Suciu and I. Tatarinov. Declarative Specification of Data-intensive web sites.
http://www.usenix.org/events/dsl99/full_papers/fernandez/fernandez.pdf

15. D. Florescu, A. Levy and A. Mendelzon. Database Techniques for the World-Wide Web: A survey. 1998.
http://www.cs.washington.edu/homes/alon/site/files/sigrec98.ps

16. P. Fraternali. Web application Development: Tools and Approaches.
ftp://ftp.elet.polimi.it/pub/Piero.Fraternali/www_docs/www7/webtools.html

17. G. Mecca, P. Atzeni, A. Masci, P. Merialdo, G. Sindoni From Databases to Web-Bases: The ARANEUS Experience - Technical Report n. 34-1998 - Dipartimento di Informatica e Automazione, Universita' di Roma Tre, May, 1998. http://www.dia.uniroma3.it/Araneus/publications/rt34-98.zip

18. G. Mecca, P. Merialdo, P. Atzeni, V. Crescenzi. The Araneus Guide to Web-Site Development - Araneus Project Working Report, AWR-1-99 (version 1.0 - March, 9, 1999). http://www.dia.uniroma3.it/Araneus/publications/AWR-1-99.zip

19. T. Perdue. MySQL and PostgreSQL compared. http://www.phpbuilder.com/columns/tim20000705.php3

20. C. Work, Web Tools 2 – Administration & Management. http://www.ucisa.ac.uk/TLIG/conf/tlig98/ckw (viewed 12/04/2001)

Appendices

Appendix A

Table of Acronyms

Acronym	**Meaning**
HTML	Hypertext Mark-Up Language
DBPL	Database Programming Language
SQL	Structured Query Language
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Mark-up Language
API	Application Programming Interface
URL	Uniform Resource Identifier
CSV	Comma Separated Variables
ER	Entity-Relationship
HTTP	Hypertext Transfer Protocol
ODBC	Open database connectivity

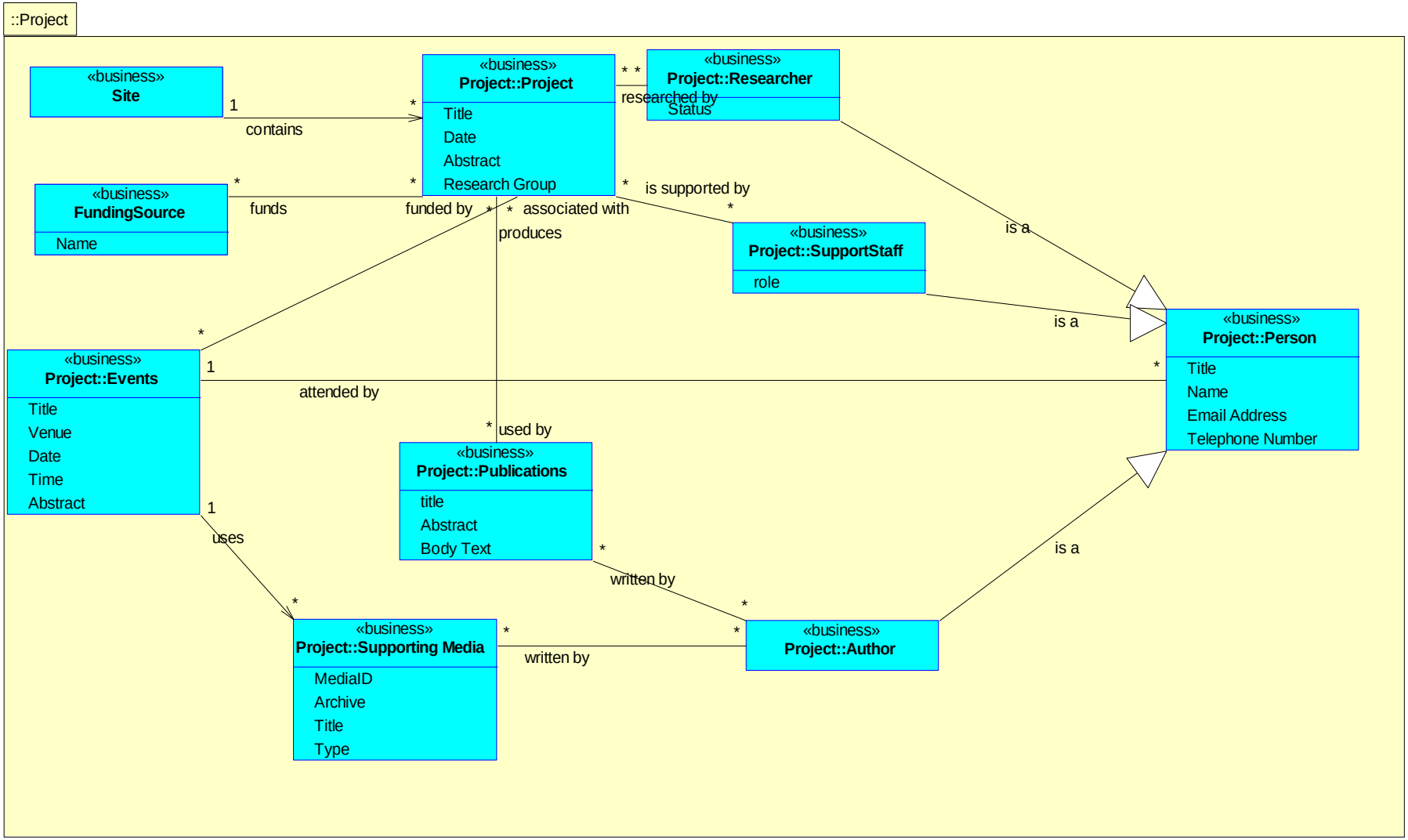
Appendix B

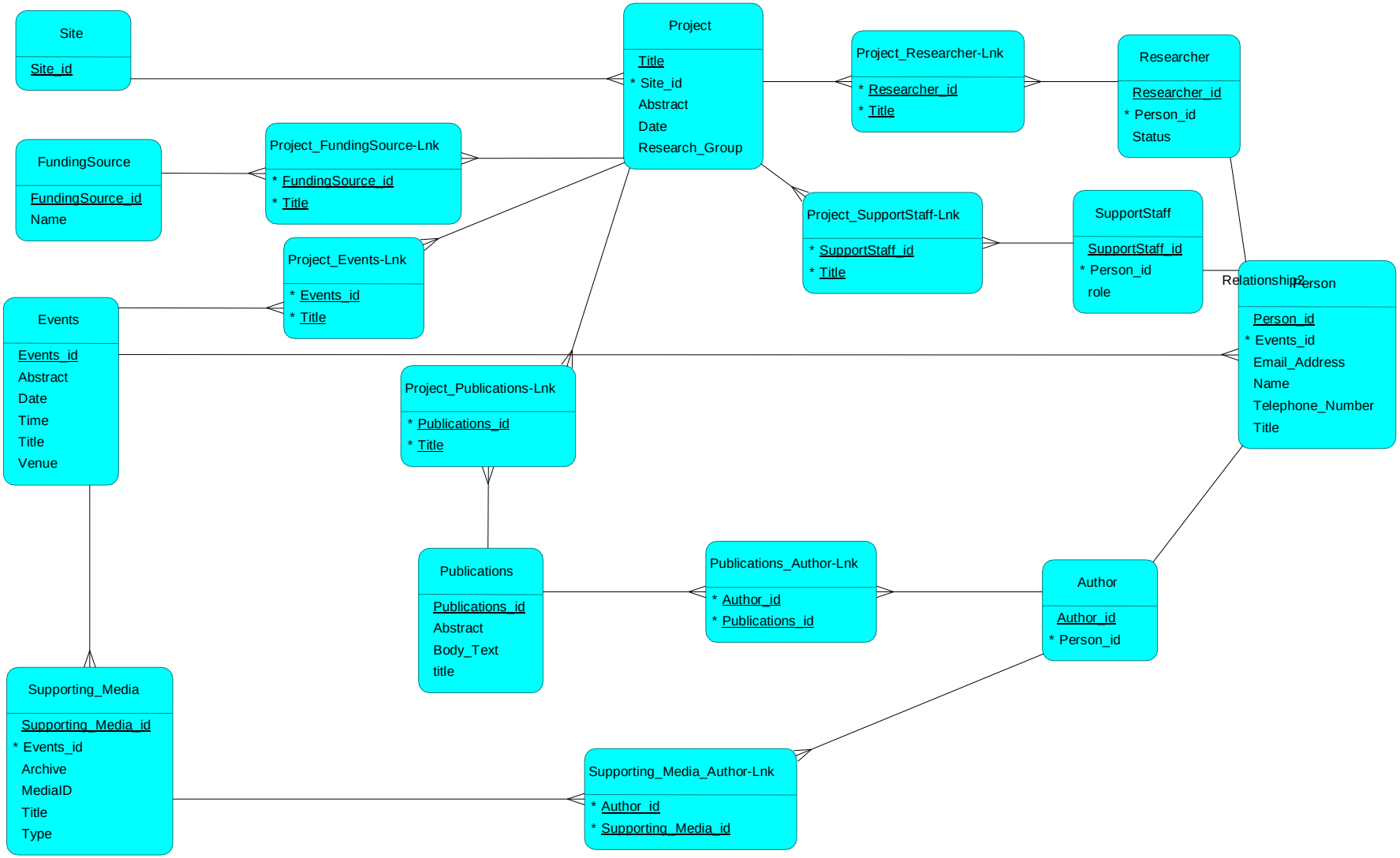
Information was extracted from the following websites:

http://www.epsrc.ac.uk
http://iam.ecs.soton.ac.uk
http://www.aktors.org
http://www.csis.org
http://www.cs.unibo.it/projects/projects.html
http://www.cs.arizona.edu/indexes/research.html

Other universities were visited to see if they could provide extended information or ideas.

Full size diagrams of figure 1 and figure 2 are included on next two pages.
Followed by our definition of the SQL schema.


::Project
«business»
Site
1
contains
*
«business»
Project::Project
Title
Date
Abstract
Research Group
* *
researched by
«business»
Project::Researcher
Status
«business»
FundingSource
Name
*
funds
*
funded by
* * associated with
produces
*
is supported by
*
«business»
Project::SupportStaff
role
is a
is a
«business»
Project::Person
Title
Name
Email Address
Telephone Number
*
«business»
Project::Events
Title
Venue
Date
Time
Abstract
1
attended by
*
1
uses
* used by
«business»
Project::Publications
title
Abstract
Body Text
*
written by
*
«business»
Project::Supporting Media
MediaID
Archive
Title
Type
*
written by
*
«business»
Project::Author
is a


Site
Site_id
Project
Title
* Site_id
Abstract
Date
Research_Group
Project_Researcher-Lnk
* Researcher_id
* Title
Researcher
Researcher_id
* Person_id
Status
FundingSource
FundingSource_id
Name
Project_FundingSource-Lnk
* FundingSource_id
* Title
Project_SupportStaff-Lnk
* SupportStaff_id
* Title
SupportStaff
SupportStaff_id
* Person_id
role
Relationship2
Person
Person_id
* Events_id
Email_Address
Name
Telephone_Number
Title
Project_Events-Lnk
* Events_id
* Title
Events
Events_id
Abstract
Date
Time
Title
Venue
Project_Publications-Lnk
* Publications_id
* Title
Publications
Publications_id
Abstract
Body_Text
title
Publications_Author-Lnk
* Author_id
* Publications_id
Author
Author_id
* Person_id
Supporting_Media
Supporting_Media_id
* Events_id
Archive
MediaID
Title
Type
Supporting_Media_Author-Lnk
* Author_id
* Supporting_Media_id

```
CREATE TABLE admin(
 TableName CHAR(255),
 Generate INT,
 Modify INT
);
INSERT INTO admin  VALUES ("Admin",0,0);

INSERT INTO admin VALUES ("Author",1,1);
INSERT INTO admin VALUES ("Events",1,1);
INSERT INTO admin VALUES ("FundingSource",1,1);
INSERT INTO admin VALUES ("Person",1,1);
INSERT INTO admin VALUES ("Project",1,1);
INSERT INTO admin VALUES ("Project_EventsLnk",0,1);
INSERT INTO admin VALUES ("Project_FundingSourceLnk",0,1);
INSERT INTO admin VALUES ("Project_PublicationsLnk",0,1);
INSERT INTO admin VALUES ("Project_ResearcherLnk",0,1);
INSERT INTO admin VALUES ("Project_SupportStaffLnk",0,1);
INSERT INTO admin VALUES ("Publications",1,1);
INSERT INTO admin VALUES ("Publications_AuthorLnk",0,1);
INSERT INTO admin VALUES ("Researcher",1,1);
INSERT INTO admin VALUES ("Site",1,1);
INSERT INTO admin VALUES ("SupportStaff",1,1);
INSERT INTO admin VALUES ("Supporting_Media",1,1);
INSERT INTO admin VALUES ("Supporting_Media_AuthorLnk",0,1);


CREATE TABLE Author (
	Author_id INTEGER AUTO_INCREMENT,
	Person_id INTEGER NOT NULL,
	PRIMARY KEY ( Author_id ),
	UNIQUE ( Author_id )
);

CREATE TABLE Events (
	Events_id INTEGER AUTO_INCREMENT,
	EvtAbstract TINYTEXT ,
	Date Date ,
	Time time ,
	EventTitle CHAR(50) ,
	Venue CHAR(50) ,
	PRIMARY KEY ( Events_id ),
	UNIQUE ( Events_id )
);

CREATE TABLE FundingSource (
	FundingSource_id INTEGER AUTO_INCREMENT,
	Name CHAR(50),
	PRIMARY KEY ( FundingSource_id ),
	UNIQUE ( FundingSource_id )
);

CREATE TABLE Person (
	Person_id INTEGER AUTO_INCREMENT,
	Events_id INTEGER,
	Email_Address CHAR(50),
	Name CHAR(50),
	Telephone_Number CHAR(50),
	PersTitle CHAR(50),
	PRIMARY KEY ( Person_id ),
	UNIQUE ( Person_id )
```

```
);
CREATE TABLE Project (
	Title CHAR(50) NOT NULL,
	Site_id INTEGER,
	ProAbstract TINYTEXT,
	Date Date,
	Research_Group CHAR(50),
	PRIMARY KEY ( Title ),
	UNIQUE ( Title )
);

CREATE TABLE Project_EventsLnk (
	Events_id INTEGER NOT NULL,
	Title CHAR(50) NOT NULL,
	PRIMARY KEY ( Events_id, Title )
);

CREATE TABLE Project_FundingSourceLnk (
	FundingSource_id INTEGER NOT NULL,
	Title CHAR(50) NOT NULL,
	PRIMARY KEY ( FundingSource_id, Title )
);

CREATE TABLE Project_PublicationsLnk (
	Publications_id INTEGER NOT NULL,
	Title CHAR(50) NOT NULL,
	PRIMARY KEY ( Publications_id, Title )
);

CREATE TABLE Project_ResearcherLnk (
	Researcher_id INTEGER NOT NULL,
	Title CHAR(50) NOT NULL,
	PRIMARY KEY ( Researcher_id, Title )
);
CREATE TABLE Project_SupportStaffLnk (
	SupportStaff_id INTEGER NOT NULL,
	Title CHAR(50) NOT NULL,
	PRIMARY KEY ( SupportStaff_id, Title )
);

CREATE TABLE Publications (
	Publications_id INTEGER AUTO_INCREMENT,
	Abstract TINYTEXT,
	Body_Text LONGTEXT,
	Publication_title CHAR(255),
	PRIMARY KEY ( Publications_id ),
	UNIQUE ( Publications_id )
);

CREATE TABLE Publications_AuthorLnk (
	Author_id INTEGER NOT NULL,
	Publications_id INTEGER NOT NULL,
	PRIMARY KEY ( Author_id, Publications_id )
);

CREATE TABLE Researcher (
	Researcher_id INTEGER AUTO_INCREMENT,
	Person_id INTEGER NOT NULL,
	Status CHAR(50),
	PRIMARY KEY ( Researcher_id ),
```

```
    UNIQUE ( Researcher_id )
);

CREATE TABLE Site (
    Site_id INTEGER AUTO_INCREMENT,
    PRIMARY KEY ( Site_id ),
    UNIQUE ( Site_id )
);

CREATE TABLE Supporting_Media (
    Supporting_Media_id INTEGER AUTO_INCREMENT,
    Events_id INTEGER,
    Archive CHAR(50),
    MediaID int,
    MediaTitle CHAR(50),
    Type CHAR(50),
    PRIMARY KEY ( Supporting_Media_id ),
    UNIQUE ( Supporting_Media_id, MediaID )
);

CREATE TABLE Supporting_Media_AuthorLnk (
    Author_id INTEGER NOT NULL,
    Supporting_Media_id INTEGER NOT NULL,
    PRIMARY KEY ( Author_id, Supporting_Media_id )
);

CREATE TABLE SupportStaff (
    SupportStaff_id INTEGER AUTO_INCREMENT,
    Person_id INTEGER NOT NULL,
    role CHAR(50),
    PRIMARY KEY ( SupportStaff_id ),
    UNIQUE ( SupportStaff_id )
);
```

Appendix C

PHP classes

Class sqldb

Direct Subclasses

mysqldb

Description

class for defining functions used for sql transactions. Generic class which can be extended to provide functionality for each individual sql server.

Constructor

sqldb()

Description

Constructor for sqldb class. Sets host/user/database and password fields, using global variables: $dbhost $dbuser $dbname $dbpass.

Function Summary	
non-void	getdatabase()
non-void	getdbconnection()
non-void	getdbpassword()
non-void	getdbuser()
non-void	gethost()
non-void	getnumberfields()
non-void	getnumberrows()
void	setdatabase($req_db)
non-void	setdbconnection($req_dbconnection)
non-void	setdbpassword($req_password)
void	setdbuser($db_user)
non-void	sethost($req_host)

Function Detail

---

getdatabase()

Description

Return name of current database in use

Returns

String - database name

---

getdbconnection()

Description

.

Returns

dbconnection

---

getdbpassword()

Description

Return password of current database user

Returns

String - user password

---

getdbuser()

Description

Return name of current database user

Returns

String - user name

---

gethost()

Description

Returns the current database server.

Returns

String: server hostname.

---

getnumberfields()

Description

This function returns the number of fields columns) from the previous query, initiated by this class.

Returns

Integer Variable - number of fields.

---

getnumberrows()

Description

This function returns the number of rows from the previous query, initiated by this class.

Returns

Integer variable - number of rows

---

setdatabase($req_db)

Parameters

$req_db

Description

Sets the name of the current database

---

setdbconnection($req_dbconnection)

Parameters

$req_dbconnection

Description

.

Returns

N/A

---

setdbpassword($req_password)

Parameters

$req_password

Description

Sets the password for the user accessing the current database

Returns

N/A

---

setdbuser($db_user)

Parameters

$db_user

Description

Sets the username accessing the current database

---

sethost($req_host)

Parameters

$req_host

String value representing database server host

Description

Set the current database server

Returns

N/A

Class mysqldb

Superclasses

sqldb
|
+--**mysqldb**

Description

Extension of sqldb class to provide support for the mysql sql server.

Constructor

mysqldb()

Description

Construction for mysql class, this just calls sqldb's constructor

Function Summary	
non-void	deletequery($req_sql)
non-void	getfieldflags($fieldid)
non-void	getfieldname($fieldid)
non-void	getfieldtype($fieldid)
non-void	listfields($table)
non-void	listtables()
non-void	opendbconnection()
non-void	query($req_sql)

Functions inherited from **sqldb**
getdatabase , getdbconnection , getdbpassword , getdbuser , gethost , getnumberfields , getnumberrows , setdatabase , setdbconnection , setdbpassword , setdbuser , sethost

Function Detail

---

deletequery($req_sql)

Parameters

$req_sql

query string containing operation to carry out, i.e. DELETE.

Description

This is function for DELETE and other operations where 'rows' will not be returned.

Returns

Boolean - return success

---

getfieldflags($fieldid)

Parameters

$fieldid

Description

This function gets any field flags from the given field from the last mysql query.

Returns

String - Any flags i.e. primary, not null, assosciated with the field.

---

getfieldname($fieldid)

Parameters

$fieldid

Description

This function gets the field name of the given field from the last mysql query.

Returns

String - representing name of field.

---

getfieldtype($fieldid)

Parameters

$fieldid

Description

This function gets the field type of the given field from the last mysql query.

Returns

String - representing type of field.

---

listfields($table)

Parameters

$table

String of table name to list fields for.

Description

This function a list of fields in a table in the give database.

Returns

... - list of fields.

---

listtables()

Description

This function lists the tables in the given database

Returns

... - List of tables in the database.

---

opendbconnection()

Description

Function to open a new database connection and set connected state.

Returns

Boolean - returns success

---

query($req_sql)

Parameters

$req_sql

SQL to be parsed in query.

Description

Query function supporting valid sql queries. i.e. SELECT,INSERT

Returns

BOOL - whether this appears to be successful.

Appendix D

6.2.1 Authentication / Session Handling

Test Procedure	Pass/Fail Criteria	Result
No username / password is entered when requested	PASS: The system will refuse access, and return a 401 error Fail: Entry is allowed.	PASS: Access Refused
Incorrect username/ password is entered	PASS: The System will refuse access, and return a 401 error. Fail: Entry is allowed.	Access is refused, however a 401 error is not returned.
Correct Password and username combination	PASS: Entry Granted FAIL: Entry is not allowed.	Pass: entry is granted

6.2.2 Validation of generated HTML

Test Procedure	Pass/Fail Criteria	Result
Use a HTML validation program to check generated files for errors	PASS: The HTML validator accept the HTML as valid. FAIL: The HTML validator decides an amendment should be made.	Fail: 1. No Doctype tag defined. 2. No <HEAD> entry in generated pages.
Use HTML validation against a table with only a primary key.	PASS: The HTML validator accept the HTML as valid. FAIL: The HTML	Fail: (once doctype was tag is defined) 1. <TABLE> element is included. This is not

	validator decides an amendment should be made.	needed.
Use HTML validation against a empty table: i.e. no fields, not even a primary key.	PASS: The HTML validator accept the HTML as valid. FAIL: The HTML validator decides an amendment should be made.	Pass: (once doctype was tag is defined.)
Use a HTML link validation tool to check for any broken links from the generated database.	PASS: All links should exist. FAIL: The tool detects that a broken link exists	PASS

6.2.3 Modification of database

Test Procedure	Pass/Fail Criteria	Result
Remove a table from database	PASS: The table will not be displayed anywhere on web page. FAIL: Table information still exists	FAIL: Dropping a table manually from the database, does not remove the entry for that table's description from the website.
Inserting a new record into the database	PASS: A new record is added to specified table FAIL: New record does not appear	PASS: Record appears in database. FAIL: On incorrect record, limited error handling – user has to debug.
Edit an existing record	PASS: Record is modified FAIL: No change to existing record	PASS: Record is updated.

Appendix E - User Guide

Installation / Configuration.

To install, place the scripts in a public_html or .WWW directory which apache uses as a web site. Before accessing the web site, you need to create tables in MySQL.

The following tables are used by the system:

This is a table of username password mappings, to be used by the authentication system.

CREATE TABLE users(

```
 Username CHAR(8),
 Password CHAR(8)
);

INSERT INTO users VALUES ("Admin","test");
```
(initial Admin account, with password test.)

The admin table consists of a list of tables. Generate and modify values are used to set whether the table should be generated as html, or available for modifcation in the system.

```
CREATE TABLE admin(
 TableName CHAR(255),
 Generate INT,
 Modify INT
);

CREATE TABLE descript(
 TableName CHAR(255),
 ColumnName CHAR(255),
 Description CHAR(255)
);
```

This table consists of a number of more detailed descriptions for table/column names. We are using data from an existing database, so we may not want to change field name, however on generated sites, different information may be required.

It is down to the administrator to import an SQL schema for the web site under development. And to allow PHP permissions to create the html documents. PHP will probably want to create the documents as the same user running apache, so therefore you may need to grant adequate file permissions.

The other step which needs to be perform is modify the configuration files as necessary to include details of the MySQL server being used, database name, username, password etc.

Using the System

After the system is configured, you can log in. When you log in, you are presented with a set of php forms, which allow you to modify the database.

By selecting a set of tables to produce and clicking on the generate button on the left hand frame, a set of HTML pages will be produced.