Abstract

“GIS i skolan” is a project at Umeå University with the purpose of supporting education in outdoor environments using Geographical Information Systems. This article describes a prototype implementation of a web application for the project, an application allowing students and teachers to share information gathered during excursions.

The prototype shows how a successful and sophisticated web application for use in learning environments can be designed and implemented by taking advantage of advanced database features. The prototype has been tested in practice and proved to work well with adequate performance.

1 Introduction

“GIS i skolan” is a project at Umeå University with the purpose of supporting education in outdoor environments, e.g. biology classes, using Geographical Information Systems, GIS. The project needed a web application to allow students and teachers to share information gathered during excursions. This article describes a prototype implementation of said web application.

The fundamental functionality of the application is to allow users to add objects to a digital map, and edit or remove objects which they have added. The objects represent observations with properties such as description, location (supplied by GPS), time, URLs, pictures or sound clips. Other users should be able to search for existing observations and annotate such observations with comments. Searches could be launched by selecting points on a map, entering time intervals or through supplying key-word based queries. To keep track of who added what and prevent abuse the application has to be able to keep track of multiple users and groups with different permissions, e.g. teachers and students. It is also desirable to be able to choose a map from a set of several pre-defined, or to allow the user to select one or more map layers. Finally, multilingual support is desired.

2 Approach

The work started by establishing the requirements, from which the overall design and major tools were selected. Thereafter the database was designed which

\(^1\)GIS in the school
resulted in an Extended Entity-Relationship (EER) diagram which was then translated into an SQL database schema. More detailed tools and techniques were evaluated for things such as multilingual support and generic database access. Finally the implementation was performed, starting with general framework components, user session management and so on, and then proceeding with the specific parts of the user interface. During the implementation some minor adjustments to the database design were performed. Testing and evaluation of the application was performed continuously.

The application is implemented in PHP 4 using extensions from PEAR\(^2\) with PostgreSQL as its database backend. The PHP language is widely used in web applications and was chosen based on its reputation and ease of use. PostgreSQL was chosen based on its rich set of features, reliability and free license.

The fundamental part of the user management in the application is session handling, which is implemented using the built in functionality in PHP together with a custom security manager. When logging in, a user is authenticated by verifying the supplied user name and password by using one of three authentication methods. The particular authentication method for a user is set by an administrator and can be plain text, crypt\(^3\) or domain authentication. The domain authentication is a flexible solution for authentication with external services. Each domain describes a certain authentication method consisting of protocol\(^4\), host and port. If domain authentication is selected for a user, the user name and password are verified with the external service rather than storing it in the local database. This feature is particularly useful in Swedish school environments since it enables the students and teachers to use their present user name and password in the common FirstClass system to authenticate, i.e. no requirement for the users to remember yet another user name/password pair.

A successful authentication will provide the user with a valid session containing the user ID. At each subsequent request the user ID is verified against the database to retrieve the user details and authorisation information, i.e. the password verification is only made once but the authorisation information is always up-to-date.

Users are given different access and modification permissions based on their type, e.g. teacher or student, and membership in different schools and school classes. For example, a student may only modify his or her own observations while a teacher may modify all observations made by students in the class he or she is teaching. All policy decisions are made by one component, the security manager. This design makes it easy to maintain a consistent security policy.

Another central piece of the application is the digital map. A map is composed of one or more layers and is created on the fly for each request by a user. Each layer consists of a description of its dimensions and name which is stored in the database, and an image file with transparency information. Markers for visible observations can also be added to the map. These markers are stored as image files as well.

To store the coordinates of observations and the dimensions of layers, the spatial data types \texttt{point} and \texttt{box} in PostgreSQL are used. The coordinates\(^2\)\(^3\)\(^4\)

\(^2\)PEAR - PHP Extension and Application Repository
\(^3\)UNIX-style password storage
\(^4\)Currently the POP3 protocol is supported but other protocols could easily be added
are in the Swedish RT90 system\(^5\), which uses a rectangular grid on a cylindrical projection. This makes it very easy to calculate distances between points, for example when searching for observations, by using the ordinary Cartesian distance formula which is supported by PostgreSQL.

The application includes functionality for searching for observations by specifying a date range. This is implemented by adding a date field to the observation table indicating when an observation was made. A search then places a condition on the value of this field. True temporal support, i.e., keeping track of multiple versions of records, were not implemented as it was deemed to be too complicated to use for ordinary users, and unnecessarily complex to implement.

Multimedia data, i.e., images and sound clips, are stored as binary large objects (BLOB:s) in the database. The application uses the PostgreSQL interface for sending and retrieving BLOB:s, which allow the application to handle BLOB:s without requiring them to be stored in main memory. The correct data type for a multimedia item is determined automatically from the contents of the uploaded files. A rule was created on the multimedia table to automatically delete the BLOB data if a multimedia item is deleted.

To allow the application to handle several languages, it must be possible to translate the descriptions for several objects in the database as well as the user interface.

Translated descriptions of database objects were implemented by replacing fields which contain descriptions that must be translated with a reference to a pair of generic string tables that hold the string in each language. This solution allows new languages to be added without modifying the table definitions. When retrieving data from a table that has multilingual strings several joins have to be performed to get the string for the desired language, or from the default language if a translation is missing. To ease programming and direct database manipulation a view was created for each table that contains translatable strings. Each view adds one language column to the table and performs all joining behind the scenes. By using the rule system of PostgreSQL it was also possible to make the views updateable, so that if a record is inserted or updated in a translation view, the base tables are automatically modified instead.

To make the user interface multilingual the application logic was separated from the presentation layer, i.e., the generation of the user interface. Both the application logic and presentation layer are implemented as ordinary PHP files. The most suitable presentation layer file is automatically selected depending on the user’s desired language. If no file exists in the desired language, the application looks for a file in the default language, and then for a language-neutral file.

3 Results

The “GIS i skolan” concept together with the web application has been tested in a live excursion with teachers and students from Grubbeskolan in Umeå. All feedback received so far has been positive, and no reports of malfunctions have appeared. Several suggestions for future development have been noted.

A screenshot of the finished application is shown in Figure 1.

---

The performance of the application has been measured, as it is vital for a web application to be able to handle the users’ requests quickly. A typical request of the application, e.g. the page seen in the screenshot, took on average approximately 230 ms to serve. Most of the time was spent in the webserver parsing and executing the PHP code, only about 2-5% of the required CPU time was used by the database.

Generation of the map, which is a separate request, took between 300 and 600 ms, depending on the type of map, colour depth and number of observation markers. The simple map, 9.5KB in size, had an average request time of 340 ms while the detailed map, 65KB in size, had an average request time of 460 ms. Generating the detailed map with an additional layer with altitude information took 550 ms. Rendering of ten observation markers increased the request time with 35-40 ms. The above timings all used truecolour for rendering, disabling truecolour saved from 90 ms to 180 ms in request times, depending on the type of map. The simple map, which has low colour detail, suffered the most from using truecolour rendering.

The benchmarks were performed on a Intel PIII, 866MHz, running Debian GNU/Linux and kernel 2.4.25. This is a low-end system by today’s standards, and in a production environment a more modern system should be used.

### 4 Discussion

The database design was essential in this project since the functionality of the database is highly dependent on the application logic. The initial database de-
sign was therefore made thoroughly and only needed slight modifications during the application development.

The use of special database features such as update rules for views were of substantial benefit to the application by effectively reducing the implementation time. The translation views also have the advantage of allowing developers and system administrators to examine and manipulate the database directly, i.e. without needing to manually join the base tables with the translation tables. The current usage of the database depends on a few features specific to PostgreSQL which may be somewhat difficult to port to other database systems. However, PostgreSQL is quite capable and should be sufficient for most application usage scenarios, especially considering its favourable license.

The programming language, PHP 4, gave a mixed impression. The main advantages of PHP is the rich library of useful function, e.g. for managing users’ sessions and dynamic generation of images. The main disadvantages of PHP, experienced in this project, are the use of weak typing and the limitation of run-time only correctness checking. This can be convenient for prototype development but appears less suitable for larger projects.

Another fundamental limitation of PHP and similar tools, which became apparent during the development, is the large amount of manual work required to create an application user interface presented in HTML. A middle layer allowing for graphical design of user interfaces and event driven programming would reduce the development time significantly.

The application performance is adequate for moderate scale usage. A typical request requires more than ten thousand lines of PHP code to be parsed. Considering this, the request time of about 250 ms on low end hardware is quite good. The scalability of the application should also be sufficient as the number of database queries issued per page request is independent of the database contents, i.e. $\Theta(1)$ queries per page request.

The performance of the application could be increased by using a PHP accelerator that saves the parsed code so subsequent requests don’t have to parse it again. The map image generation could also be made faster by use of suitable server side caching.

5 Conclusions

We have shown how to create a sophisticated web application that allow students and teachers to easily share and discuss observations from outdoor excursions.

The application has a security model designed to easily enforce a consistent policy while being convenient to users and allowing the use of external authentication services. The support for several languages is easy to extend to more languages and the application has proven to work well in practise with adequate performance.

The project has proven the benefits of using advanced database features in real-world applications and the importance of solid database design in web applications.