Enhancing Interactivity for Self-Evaluation in XML-Based Courseware

Florin Bota
Dip. Automatica e Informatica - Politecnico di Torino
Corso Duca degli Abruzzi 24
10129 Torino, Italy
bota@ol-tutor.polito.it

Fulvio Corno
Dip. Automatica e Informatica - Politecnico di Torino
Corso Duca degli Abruzzi 24
10129 Torino, Italy
corno@polito.it

Laura Farinetti
Ce.Te.M. - Politecnico di Torino
Corso Duca degli Abruzzi 24
10129 Torino, Italy
farinetti@polito.it

Abstract: The paper describes an educational application, oriented to university students, to help them to learn the SQL language in the ambit of a course on database systems and applications. This tool is integrated with a XML-based framework for courseware authoring, whose main characteristic is flexibility, since contents are presented to the students in different ways, depending on their previous knowledge, on the kind of hardware used to access the content, and on some preferences of the user. The implemented application is interactive, and allows the students to practice their understanding and their skills about the SQL language, following the approach of "leaning by doing".

Introduction

Internet can represent a very effective platform for learning, providing many of the tools needed in the educational process, from the definition of curricula to the collection of educational material, from the delivery of this material to the interaction between the involved actors (Boyle, 1997, Farinetti et al., 1997). A large number of educational institutions are now involved in developing on-line courses used for distance learning or for supporting traditional face-to-face learning, usually providing a high level of interaction between users and authors.
The design of on-line courses involves different actors, that have different requirements: the *teacher-author* needs a tool easy to use in order to create the educational material; *the student-user* needs something more than a mere translation of a book in electronic format, that is some sort of guidance, a high level of interactivity and a tool for assessing the learning process; the *site administrator*, finally, needs an easy-to-maintain system for updating the content and the information about users.
Implementation of effective educational platforms needs to carefully take into account all of the previous requirements. For this reason we designed and implemented an application framework able to allow teachers with no specific computer science skills to create courses composed of different content formats: simple text and pictures, but also audio/video streams, Java applets and animations. The main characteristic of the tool is flexibility, in the sense that contents are presented to the students in different ways, depending on their previous knowledge, on the kind of hardware used to access the content, and on some preferences of the user (due to special educational needs, for example). In order to acquire new knowledge, students are supposed to study, to practice new concepts, to solve problems and, very importantly, to assess their knowledge level thank to self-evaluation mechanisms. Besides that, the system is designed to be simple enough for the site administrator, since it maintains all the course information in

one file only, and most of the processing is done on the student's computer; in this way one small server can provide content relative to a large number of courses to hundreds of students.
The developed application uses *XML* (*Extensible Markup Language*) for the organization of the contents, assuring a simple way to modify the presentation of the modules. The contents can be created either by using a simple text editor, or by using an XML editor, or by using a dedicated content-creation application at present under development. The use of the XML standard, which uses text files to maintain database-alike content, was the key for obtaining the desired flexibility.
One of the main advantages of hypermedia with respect to traditional educational aids is the possibility of interactive learning, self-assessment and self-evaluation. The developed application allows the insertion of animations, simulations, quizzes, video and audio streams and other form of interaction. The present paper is focused on interactivity, and aims at describing advanced possibilities for self-evaluation.
The paper describes a case-study for the XML-based framework, consisting in the design and implementation of courseware on database systems and applications. In courses about databases, usually the part regarding SQL language is the most critical: SQL can be difficult to learn, it is quite hard to get access to SQL servers, most of the tools are specifically made for designing application, and quizzes on SQL cannot be used to obtain an accurate information about the level of acquired knowledge. The proposed solution consists in the implementation of an on-line module for SQL training and self-evaluation.

The application framework

The XML framework aims at providing an educational-oriented authoring tool, capable of producing on-line courses with a structure and organisation of the contents already defined according to an educational methodology. The on-line courses developed with that application are flexible enough to adapt to different user profiles, with different entry levels in terms of already acquired concepts and skills, different learning goals, different hardware resources. For this purpose each course was considered as a collection of modules, part of a homogenous structure; each module has clearly stated pre-requisites for the fruition and well defined learning objectives. Entrance and exit levels of knowledge associated to each module permit the dynamical building of the educational path according to the student profile. Furthermore, each module allows different views of the content, since it consists mainly of a collection of sub-modules offering different presentation formats:

- the *introduction* sub-module, where the main goal of the module is stated, together with the list of the contents and the pre-requisites;
- the *main content* sub-module, which is the real body of the course, containing the explanation of all the concepts and procedures;
- the *conclusion* sub-module, which consists in the summary of the module, i. e. a list of the fundamental concepts the student should have acquired after studying the main content;
- the *example* sub-module, which is a collection of examples that help the student understand the concepts better;
- the *exercise* sub-module, which contains a collection of exercises with their solutions so that the student can check the knowledge of the concepts presented;
- the *test* sub-module, which provides self-evaluation under the form of a test with automatic feed-back; it consists in a set of questions with predefined answers, and the result obtained by the students influences the presentation of the subsequent modules.

The self-evaluation mechanism allows the student to test his or her level of knowledge and understanding of the concepts, permitting as well the dynamical building of the educational paths among the modules of the courses. Two types of tests are proposed: a preliminary test, used for determining the student's level of knowledge when he is making the first contact with the course, and a final test for each module, used to determine whether or not the student has acquired the required exit level of knowledge for the module, and to choose the level of presentation of the following module. According to the results of the final test, the profile of the student is modified, and the choice of the following module is based on this profile. The main idea is that the better the student performs, the less content she needs. In an extreme situation, the student that did not acquire the desired level of knowledge could be forced to study once more the same module, in order to acquire the minimum level of knowledge required to proceed to the following module. A course structure like this has two degrees of flexibility in building the educational paths: on one hand it allows the combination of different modules, and on the other the amount and the type of information presented in each module vary.

The application uses XML for the organization of the contents, yelding a simple way to modify the presentation of the modules. At this purposes, we defined a grammar for all the courses, which is contained in a DTD (*Document Type Definition*) file (Megginson, 1998). The grammar lists the rules the course should satisfy, that is the division in different modules and the different views of each module (the sub-modules described above), together with information about the course and the course authors. For each sub-module we defined the possible contents and the minimum information that should be present.
As an example, each test contains a list of questions and their possible answers, a code for the identification of the test (either preliminary or final), the name of the test, and the location of the CGI program that evaluates the answers (that is part of the application as well). Most of the other sub-modules contain a title and some paragraphs. A paragraph can contain text, other media – images, audio, video, animations – or some sort of interactive unit. Some sub-modules are not mandatory (*example* and *exercise),* their presence depending on the way the authors structure the module.
The appearance of XML document on the users' screen depends on the associated XSL (Extensible Stylesheet Language) document, that controls the fonts, the colors, the size, and the amount of content to be displayed. The XSL files need to be different according to the profile of the user, and are based on a general XSL template and on some specific hardware configurations. The hardware characteristics of the user's computer, the speed of the connection, the previous preferences of the user influence the display of the multimedia content. For example, if the student uses a low-speed connection no audio/video stream is presented; on the contrary, in case of high-speed connection most of the multimedia content are displayed. For a person suffering of hearing impairment no audio content is presented.
In order to have this degree of flexibility for the display of the contents, the implementation uses two CGI programs, one to create the XSL file on-the-fly, and the other to generate the XML processing instruction required for processing the XML file and for displaying it on the screen. The XSL file is specifically generated for each user, depending on his or her profile (Farinetti et al., 2000).
Most of the courses provide interactive features for self-evaluation, through the use of tests with multiple answer questions. Some course may need a higher level of interactivity, so that the student can practice and test his or her knowledge in a more sophisticated way with respect to the one provided by the tests. Even if the *exercise* sub-module shows some problems with the possible solutions, in some cases these are not enough.
A course for learning database systems and applications, that uses this XML framework, is now under implementation and testing, and we observed that the students need some advanced form of interaction that allows a "learn by practice" approach. We decided to implement an interactive feature able to allow the students to practice and test SQL queries using a specific database – the one used in all the examples of the course (Date, 1990). In this way the students can learn directly from their own experiments, trying different queries and seeing the result immediately, or looking at the possible errors their query contains. The advantages of such an approach are obvious: the interaction offered by the application represents an increase in the quality of the educational process, allowing a fast and accurate method for the students to test their knowledge. The SQL application presented in the paper will be integrated in the *exercises* sub-module of the on-line course on databases, and we are evaluating the possibility of integrating it in the *test* sub-module too.

Interactive tests

The application has been specifically designed to help students to practice SQL language, a task not easy to do using common tools. Usually it is quite difficult for the students to practice SQL, since an SQL server is not easy to configure without already having quite a good knowledge of the SQL language, and access to an existing one might be hard as well. There are only a few client tools that could be used to practice SQL by those having low-level knowledge of the language, and the interface provided by most applications used for database courses does not allow the student to use SQL in an efficient way. Most of them are not intended for learning purposes, but for the design of applications, and they allow the use of a limited subset of the SQL syntax. An application for educational purposes should allow the students to check their knowledge of SQL, with an automatic feedback, but going further than proposing a test with questions and predefined answers. An open questions test is the best tool for self-evaluation, and the application should propose different problems to the student, get the answer and evaluate it correctly.
The interactive SQL *exercise* sub-module provides three important functions. The first one consists in offering the user the possibility to practice any SQL query and to see the results almost instantly on the screen. The second functionality is proposing problems to the students and allowing them either to test a solution, or to request an evaluation of the solution. The third functionality is a variation of the second one and will probably be integrated in

the *test* sub-module, using different levels of difficulty for the problems, based on the level of knowledge of the student (obtained thanks to the XML framework).
The first one is important from the student's point of view, which is interested in practicing the concepts and the skills that he is supposed to acquire during the course, by experiencing directly how small variations in a query could lead to different results. The second one provides the student an accurate self-evaluation of the acquired knowledge, by solving real problems, and not just by checking some predefined answers.
The third one is important from the teacher's point of view, which is interested in evaluating the level of knowledge acquired by the students. Based on this the teacher can choose to modify the course, to offer more examples or more exercises, and so on. In addition to this, the XML framework can choose the best educational path for the student using this third functionality.
The first thing that the student has to do is to choose a problem. This leads to a screen that contains the statement of a problem, and allows the student either to practice SQL queries, or to propose a solution for the problem. The student writes the query and press a button to send all the data to the server specifying whether he is just testing the query or he is submitting it for evaluation. For the first functionality, the student gets a screen with the results of the query, while for the second one the display contains the evaluation of the proposed solution as well. In both cases, if the query contains an error, the student gets the description and the position of the error in the query.

The implementation

In the application we used an Apache web server running on a UNIX machine to provide a simple interface for the students. The students have to choose the functionality they prefer, and fill in the query. For both functionalities the same CGI program will present them a page containing the results of their actions; in the first case the CGI program performs only this task, while in the second case it also checks whether the query inserted by the student is a possible solution for the proposed problem. The CGI programs interacts with a SQL server; the structure of the application is shown in Figure 1.

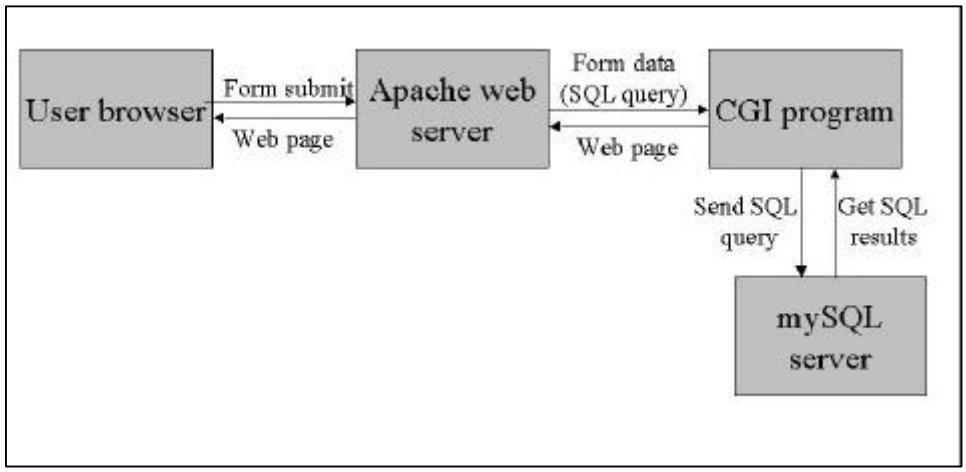

Fig. 1 – Structure of the application

The application has to distinguish between the read-only queries and the ones that perform write or update operations. The simpler read-only queries (SELECT, SHOW, DESCRIBE, EXPLAIN) are dealt with directly, running the query on a mySQL server located on the same UNIX machine as the web server, while for the queries that require modifications of the database (UPDATE, INSERT, DELETE), a new temporary database is created, by copying the original one, and all the changes are made only in the temporary database. In this way, each time someone is using the application, he can be sure that data are those presented in the examples of the course, no matter how many other students used the application before. After displaying the results, the temporary database is removed from the server, so that the application is stateless. At present students are practicing using a single-query model, and all the proposed problems are supposed to be solved by using a unique query. It would be quite easy to pass from a stateless application to a stateful one, by implementing some kind of user identification, so that the

temporary database is maintained waiting for another query from the same user, and it is removed only after a timeout period, but at present this was not one of the goal of the application.
Both functionalities are based on the use of the mySQL C API. For the first functionality, the query is executed by the CGI program and the results are formatted in HTML code and sent to the student's browser.
To verify of the correctness of the solutions proposed by the student, the comparison of the queries was not a good solution, since the correct result can be obtained by many different queries. To evaluate the results, one of the correct queries is stored in the sub-module: the application runs it on the SQL server, and its result is compared with the one obtained by executing the query proposed by the student. In the case of a read-only query this is very simple (even if different approaches are needed, one for the case that the problem does not require ordered results and another one when the problem requires ordered results), while it is a little bit more complex for the queries that modify the database (in which the contents of two different databases are compared).
Another problem to consider is the possibility that for a specific database some incorrect queries could return correct results. This happens usually in databases where an injective functions exists between some subsets of the values of two fields, so that selecting the records with a given value in one of those fields is the same as selecting the records with a specific value in the other field. As an example, in a fruit database, that contains the name of the fruit and its color, in the case that the only red fruits are apples and all the apples are red, selecting all the apples is the same as selecting the red fruits. If a new record is added to the table, for example cherry which is red as well, this problem no longer exists. In the first case it is theoretically impossible to check the correctness of the solution proposed by the student (since incorrect queries could lead to correct results), so the solution consisted in choosing a database where this problem does not exists (we added new records to solve the problem and disambiguate the records). We used a database very familiar to students, since it is used in all the SQL courses (Date, 1990). We selected carefully the values to be used in the database so that the problem above mentioned does not occur for any of the possible queries.
In the case the query proposed by the student contains a syntax error, the CGI program shows the error and gives some form of advice about the source of the error and the actions the student could take to correct it. In this way the student can see immediately the mistakes, can go back and correct them, learning from them.
The problem of the security of the database has been considered as well: in case of read-only queries the connection to the mySQL server uses a username and a password that allows only read-only access to the course database, while for the modifying queries a different username and password allows read-only access to the course database and the right to create a new database (so that the creation of a new temporary database is possible, with full rights on it).
The modular implementation allows the creation of new functionalities for the students when the need for those arises. In fact, it is possible to implement easily new categories of queries besides the ones already provided: database and table creation and administration (CREATE, DROP), or SQL server administration. This however increases the security problems, and new solutions for the problems created by those types of queries should be found before implementing them in practice.

Conclusions

The designed application aims at providing the students a better way to learn the SQL language; it will be completely integrated in the on-line course on database systems and applications, implemented using the XML framework. The advantage of the designed framework consists in the flexibility of the produced course, which on one hand allows the combination of different modules, and on the other varies the amount and the type of information presented in each module.
Interaction is one of the main requirements of an on-line course, both for providing the students a self-evaluation mechanism, and for helping the author to maintain and update an effective application for learning, by collecting and analyzing the results of the students. The paper described an interaction mechanism that allows the students to practice their understanding and their skills about the SQL language, following the approach of "leaning by doing", and that goes beyond the usual multiple answer tests.
The student can insert and evaluate queries for solving the proposed problems, and have an automatic feedback on possible mistakes or misunderstandings. He or she can also use this facility only to experiment the results of queries, to practice what they learned. The proposed interface is much easier than the tools usually available to students, that have not been designed for educational purposes.
In the future we intend to develop a stateful version of the application, allowing the students to perform multiple queries in sequence and to see the results immediately after each one. This would permit the teacher to propose more complex problems for the students to solve, and would allow the students to experience more realistic situations,

where they will have to create a database, populate it, modify and update it, obtain data from it and so on, operations which of course cannot not be done using just a single SQL query.

References

Boyle, T. (1997). *Design for Multimedia Learning*. Hertfordshire, UK: Prentice Hall Europe.

Farinetti, L., Malnati, G. & Mezzalama, M. (1997). Le nuove esigenze didattiche nella società tecnologica. In P. Terna (Ed.), *La formazione e il lavoro al tempo delle reti telematiche* (pp.149-168). Torino, Italy: Rosenberg&Sellier.

Megginson, D. (1998). *Structuring XML documents*. Upper Saddle River, NJ: Prentice Hall.

Farinetti L., Bota, F. & Rarau, A. (2000). *An authoring tool for building flexible on-line courses using XML*. In XML Europe 2000 Conference Proceedings, Paris, 12-16 June.

Date, C. J. (1990). *An Introduction to Database Systems*. Volume I, Fifth Edition. Addison-Wesley Publishing Company.