Ajax Enabled Query Tool – The Capstone Experience

Robert Dollinger
rdolling@uwsp.edu
CNMT Department, University of Wisconsin Stevens Point
Stevens Point, WI 54481, USA

Rachel Ford
Rachel.A.Ford@uwsp.edu
CNMT Department, University of Wisconsin Stevens Point
Stevens Point, WI 54481, USA

Brad Helf
Brad.R.Helf@uwsp.edu
CNMT Department, University of Wisconsin Stevens Point
Stevens Point, WI 54481, USA

Kevin Reimer
Kevin.R.Reimer@uwsp.edu
CNMT Department, University of Wisconsin Stevens Point
Stevens Point, WI 54481, USA

Abstract

The Web Based AJAX Enabled Query Tool (AEQ tool) is an educational software that uses technologies specific to Rich Internet Applications to support activities related to database classes. Years of teaching experience revealed multiple areas of the instructional process that would benefit of a tool like AEQ: (1) **Remote, on-line access to database servers via the Internet** – traditionally, students do their work in the on-campus labs. Available remoting approaches to get into the campus network are slow and often unreliable. The AEQ tool allows direct, on-line, authenticated access to the databases. (2) **Unified query tool** - the tool acts as an SQL client for various types of on-campus databases (Microsoft SQL Server, Oracle, MySQL and others) by using one single, specifically tailored, Web based interface. (3) **Course content provider and development tool** - students will be able to test the pre-loaded classroom query examples against the databases used in the classroom, as well as edit and test their own queries. (4) **Collaborative learning tool** - when using the AEQ tool, students will be able to login either into a private individual space or into a shared working environment, where they can engage in group activities, exchange messages, view live the queries issued by other users, as well as the response returned from the database. The new AJAX technology is the key to this project since it allows the development of responsive and flexible Web applications that are very much like fat client Windows applications.

Keywords: Educational software, Query tool, Rich Internet Applications, AJAX

1. INTRODUCTION

The emergence of Rich Internet Applications and the technologies supporting them, like AJAX (Asynchronous JavaScript And XML), provide new opportunities in areas that, until recently, were in the realm of Windows applications. Web applications have the advantage of wide availability and platform independence to the extent where any browser can be used to run them. The AJAX technology adds other valuable features that in the past were available only in Windows applications: responsiveness, interactivity, flexibility, increased capability to manage data in a variety of forms, etc. Combining the advantages of Web applications with those provided by AJAX opens generous opportunities in the area of educational software and classroom support tools.

The Web Based AJAX Enabled Query Tool (AEQ tool) is an educational software and learning support tool that uses the technologies specific to Rich Internet Applications in order to improve both students' and instructors' experience in classes that focus on database related topics. This paper presents the experience of initiating a development project for the AEQ Tool as an undergraduate student capstone project. The remainder of the paper is organized as follows: section 2 explains the difficulties with the current practices and establishes the motivation for initiating the AEQ project, section 3 identifies the main features of the application and the requirements it must satisfy, while section 4 presents the application architecture. Section 5 presents the main functional components of the AEQ application (the SQL Launcher, the Content Viewer and Manager, the Administration Tool, the Chatroom and Shared Mode Support) with details of their most important features. Section 6 outlines the strategy for assessing and evaluating the AEQ application, as well as the approach for further refinement and integration into the existing course activities. Finally, section 7, concludes with some thoughts about the experience of engaging undergraduate students in the development of this project, as well as their role in evaluating the impact of the AEQ tool on the current teaching practices. The potential for further developments is also briefly analyzed.

2. MOTIVATION

Many courses in the computing programs, like the ones having database as the main topic, as well as many software development classes involving some sort of database interaction (Production Programming, Client/Server Computing, Web Application Development, etc.), require students to be able to access one or more databases supported by various types of Database Management Systems (DBMS) like Oracle, Microsoft SQL Server, MySQL and other. These databases are typically located on campus servers, and specific client software tools are used for common activities like editing, debugging, and running SQL queries and then observing the returned results. These activities are performed both by instructors (while preparing classes or during live classroom demos) and by students (in hands-on lab activities or while preparing their assignments). The currently available client tools are either vendor provided, like: MSSQL Management Studio, Oracle SQLPlus Worksheet, or provided by third party developers like Toad, Oracle SqlDeveloper and other.

Over the years, the practice of using such tools for classroom activities revealed several issues of concern:

1) **Lack of remote access and limited availability** – one limiting aspect is that the currently available client tools can access the database servers only from within the campus network. Students who want work at home found this to be problematic in most cases, as it requires the installation of the database client tools on the students' home computers and the availability of a remote connection like VPN, remote desktop, or other technical solutions.
2) **Lack of uniformity in the design and functionality of the available database client tools** – a variety of database client tools are available from the DBMS vendors (MSSQL Management Studio, Oracle SQLPlus

Worksheet, Oracle SQLPlus) or from third parties (Toad, Oracle SQL Developer, EMS SQL Manager and others). Although all these tools are essentially used for the same kinds of tasks, they come with quite different user interfaces that require an unnecessary effort added to the learning curve associated with database environments.

3) **Non-user friendly database client tools** – some vendor tools, like the Oracle SQLPlus client, have become outdated and are not particularly easy to use due to the lack of the interactive features expected from what is considered a user friendly application.

4) **Maintenance** – last, but not least, the maintenance of multiple database clients, which includes installations, upgrades, patching and servicing, is an effort multiplied by the number of different clients in use.

All these problems, encountered in the current common practice of database classes, provided sufficient motivation to initiate the development project of a software tool that aims to eliminate these deficiencies and provides a better learning experience both for students and instructors.

3. MAIN FEATURES AND BASIC REQUIREMENTS

From the very beginning, the AEQ project had as objective the development of a Web Based AJAX Enabled Query Tool (AEQ tool) as an educational software application to support the teaching activities of database specific classes. Meanwhile, the issues and challenges of implementing the new tool in the classroom environment, as well as the assessment of the impact on the current teaching practices emerged as equally important. This application was meant as an on-line remoting complement to the existing database client query tools, as well as a specialized content provider for the database classes and as a domain specific environment for student group work and collaborative activities in the database area.

The AEQ tool is a hybrid software, part educational software, and part development tool. While for educational software requirements and quality assessment there are studies available in the specialized literature (Blease, 1986; Brogan, 2001; Gold, 1984; Goyne, 2000; Johnson-Glenberg, 2006; Lauterbach, 1989), for the database query and development features we relied on our own experience with the various commercial DBMS client tools. AEQ attempts to provide features similar to these Windows forms based applications by leveraging the capabilities of the new AJAX technology.

The AJAX technology (Gibbs, 2007; McClure, 2006; Moore, 2007; Woolston, 2006; Zakas, 2006) plays a key role in the development of the AEQ tool due to the requirement to closely mimic the behavior of standard database client tools, traditionally developed as Windows applications: high level of interactivity, responsiveness, flexible manipulation of data etc. The following main characteristics and core features have been identified as part of the AEQ Tool specification:

- **Provide remote, on-line access to database servers via Internet** – the AEQ tool is a Web based application available with the appropriate credentials from anywhere on the Internet. As an AJAX-enabled Web application, it will provide all the benefits that come with this technology: platform independence, ease of deployment and maintenance, responsiveness and flexibility, all this at a level comparable with traditional Windows applications. These features simplify the deployment of the AEQ tool (e.g. one single Web Server installation could cover the needs of several university campuses), and make it an ideal support tool for on-line and hybrid courses.
- **Content provider, interactive query tool and test bed** – the AEQ tool provides access to course content and materials through a hierarchical content provider. Students can navigate across the course materials (text, slides, queries etc.), copy pieces of code (queries) and paste them into an interactive query component. Here the users will be able to edit, run and test the pre-loaded samples, live, against a target database chosen from among those created by the class instructor. Re-

sult sets and/or database server messages are returned into the same Web page.

- **Unified client interface and development tool** – the query component allows users to edit their own SQL or any other extension code (like PL SQL or Transact SQL), run it against a target database, and view the systems response (result table, error messages or other). The AEQ is intended as a query tool with features specifically tailored for use in course related activities, thus abstracting out the differences among various DBMSs and the complex functionality related to database administration. The same query may be tested against different types of DBMSs (Oracle, MSSQL, MySql or other) by simply selecting a different target database.
- **Domain specific collaborative learning tool –** the AEQ tool can function in two different modes: **private** and **shared**. A user in private mode has his or her own private space in isolation from other concurrent users. The users in shared mode share their input and responses from the database with all other concurrent users. A history of the activity during the last working session will be available for every user, and can be made public in shared mode.
- **Content manager** – The AEQ tool provides an upload and edit feature available to instructors in order to prepare their course materials and make them available through the application. An interactive tree-like control allows editing of the course structure: add/delete chapters and sub-chapters, move around course chapters etc. The tool will also be able to act as a course content integrator as instructors will have the option of combining their own course materials with that available from other campuses, based on mutual agreements.
- **Course and class management tool –** the AEQ tool also provides support for activities the instructors usually have to perform at the beginning/end of semester for setting/cleaning up the classroom environment: generate dedicated database/schema for every enrolled student on every target database server, automatically generate database users and passwords with appropriate access privileges, sent email notifications to students with their login and password information, destroy databases/schemas/users and revoke access privileges at the end of the semester.

4. APPLICATION ARCHITECTURE

The architecture of the AEQ tool is typical for AJAX applications. It consists of interacting components that are, potentially located on three different computers (Figure 1.).

1. **The AJAX Client** – consists of the JavaScript code executed by the browser on the end user machine. The code is downloaded as a result of the page requests from the AEQ Web Site, or even as a response to specific AJAX requests. The AJAX client takes care of the immediate user interaction that does not require the Web Server's participation, issues background AJAX requests upon specific user actions, receives and processes the data returned by the Web Server. E.g. a user click on the "Run" button in the SQL Launcher sends the SQL query in the query window as a parameter of an AJAX request to the Web Server to be forwarded to the selected target database. The query result is returned as part of the response to the AJAX request and is processed accordingly by the AJAX client, i.e. display the result set in case of success or the error message if the query failed.
2. **The AEQ Web Site** – is located on a campus Web Server and consists of the middle-tier logic of the application. It receives regular Web page or AJAX requests from the AJAX Client, interacts with the backend and/or target databases, and responds to the client requests. E.g. when an AJAX request to execute an SQL query/command is received, the AEQ

Web Site component, connects to the selected target database and submits the SQL query/command for execution. The results from the database are then forwarded back to the AJAX client.

3. **The AEQ Backend Database** – is located on a campus database server and stores data that is vital to the functioning of the application: course lists, course content, student/user lists, passwords and permissions, information about the registered target databases and other.

In addition to the three components above, the AEQ application also interacts with a number of target databases towards which the user's queries are directed. These databases can rely on different types of Database Management Systems, and are created by the class instructors for the specific teaching needs of their classes.

5. FUNCTIONAL MODULES

The AEQ tool is composed of several interacting modules with complex, but distinct functionalities:

- SQL Launcher
- Content Viewer and Manager
- Administration Tool
- Chatroom and Shared Mode Support

5.1. The SQL Launcher

The SQL Launcher provides a Web based, unified interface to execute SQL queries and commands against a variety of DBMSs'. This module is a good example of embodiment for one of the most important changes the AJAX technology is expected to bring in our lives, that is "free the computer users from the constraints of desktop applications and from the dependence on a specific software provider" (McClure, 2006). From any Web browser, based on a simple login procedure, the SQL Launcher provides an easy access, potentially, to any type of database system. Currently, plug-ins are provided for the Oracle, MySQL and MSSQL databases, but any number of DBMS can be easily accommodated. As such the SQL Launcher can be used as an alternative to all those DBMS specific, Windows based, client interfaces (see Figure 2).

As part of the class setup process at the beginning of each semester, every student would have his/her own database created on each of the DBMSs' he/she is interested to train with. The AEQ tool will automatically create and store the corresponding connection strings and retrieve them when the user logs in. Switching from one database to the other is as easy as selecting the appropriate database in a group of radio buttons, which makes running the same SQL code against the various systems a very easy task.

One of the goals for this project was to make the user experience as seamless as possible. In order to achieve this, the SQL Launcher was designed to behave as if it were a typical client side application. This meant a traditional Web development strategy using the well known post-backs was out of the question. Instead, post-backs are avoided entirely by using AJAX calls. Also several JavaScript controls have been implemented to emulate the functionality of typical database client side applications. The end result is an application that is both useable, and fairly powerful. With a few exceptions almost any SQL code can be executed against any database a user has access to. The entire user activity is, of course, free from the flickers of the page postbacks.

There are two main areas in the SQL Launcher: the Query pane and the result pane (Figure 3).

The size of both areas is dynamically adjustable so that the two trade the available space in the browser window. In the query area one can edit and/or paste any amount of SQL code, then highlight a selection from it and run the highlighted code. If nothing is selected the entire code in the window will be executed. The result pane will display either the results returned by the SQL queries or the messages returned by the target server, each in its own tab.

To run a query, the SQL Launcher submits the SQL code via an AJAX request to an HttpHandler on the Web server. The server receives this code and begins processing it: the appropriate database connections are opened and configured, and the code is forwarded for execution to the corresponding database server. The Web server then captures the results, converts them to an XML format, stores a truncated version of it in the AEQ Backend database for logging pur-

poses, and returns the full XML result set to the client. Upon receiving the XML result set the client formats it as HTML and the user is able to view their results as if they were run locally. Noticeably, non-relational returns, like the XML data returned by queries with the FOR XML clause in MSSQL are also correctly displayed in the interface. Since AJAX is used to pass the data between the client and server the page is not forced to reload, and the only noticeable result that the user sees is that his/her query has been executed, and the system is ready for the next query.

The truncated version of the query results and the associated SQL code are automatically logged during each query cycle. Users can view this information for their passed queries at any time by opening the Query History tab. This feature allows the user to see the history of its own queries, or of any other user working in shared mode (see section Chatroom and Shared Mode).

5.2. Content Viewer and Manager

The Content Viewer and Manager is the most complex part of the AEQ application. Its role is to accompany the SQL Launcher in an enhanced the educational experience, integrating in one single tool course content and interactive working features. It provides a structured and intuitive interface to serve the needs of both students, who are able to view the available course content, and instructors, who have the possibility to add/edit/delete and manage the content of their courses in a flexible way. Two different views, a Student View and an Instructor View of the course content are provided. Instructors have access to both views, while students have access only to the Student View.

The Student View allows users to navigate through the course content in two ways: by using a context sensitive Table of Contents control where a click on an item brings forward the corresponding full content or by expanding and hiding the full content items as desired (Figure 4). Furthermore, implementation is provided for users to copy and paste SQL code into the SQL Launcher for testing and study.

Much more functionality is available to instructors in order to not only view, but also to manage their course content. The main tool is a Table of Content Manager where instructors can dynamically rearrange the chapters of their course in a tree structured pattern. There are two types of nodes in this tree: **intermediate nodes**, which are essentially wrapper containers, representing for example course sections, and **terminal nodes**, which contain the actual course materials. The structure of this tree can be modified by dynamically moving around the nodes and changing their relative positions through a drag-and–drop function. The only restriction is that terminal nodes cannot have or acquire child nodes. In addition, specific functions, which can be accessed by a right click on the node, are available for each type of node. Intermediate nodes can be deleted or renamed, or one can add a new child that can be either an intermediate or a terminal node. Terminal nodes can be deleted or renamed, and one can also edit their content. No changes are committed unless the user chooses to save them. Until saved one can undo or redo, one by one, all the changes since the last save operation.

A new child can be added to an intermediate node at any time. The new node has all the functions intermediate and terminal nodes have as it can potentially become either, depending on the user actions. By adding content to it, it becomes a terminal node, while by adding a child it becomes an intermediate node (Figure 5).

Content can be added to a new node through a text editor and upload tool in several ways: simple typing, copy-and-paste from a different source, as well as by uploading a file. In addition an instructor may choose to designate parts of the content as SQL (ensuring that SQL formatting is provided in Student View and making it easier to transfer this content into the SQL Launcher); otherwise, all content is read as text. Content is designated as SQL by selecting that portion of the content and pressing the SQL button; likewise, SQL formatting can be removed in the same manner (Figure 6).

All the above functionality is provided through an AJAX implementation consisting of a combination of server side C# and client JavaScript code. Custom AJAX controls (such as the tree view, used both for the Instructor View content tree and the Student View Table of Contents) are combined with existing ASP.NET controls (such as the FileUp-

load) to provide as seamless and convenient implementation as possible.

5.3. The Administration Tool

The AEQ tool is a complex and powerful application which, through some of its functions, is managing resources, like campus databases, that need to be protected from unauthorized access. For example, students are not supposed to modify a course content or delete a database, and instructors can edit the content of their own courses, but not other. As a result, it is critical for the application to have a well defined user management and security system to properly restrict access to various resources only to authorized users. In addition to this, specific functionality is available to the instructors for management of different kinds of class resources, like: student lists and accounts, and student working databases. All these features are integrated into the Administration Tool.

In order to provide user account support, the AEQ application takes advantage of several of the new Membership, Roles, and Profile features built in to ASP.NET 2.0 (Mitchell 2006). These built in features are used for authenticating, adding, creating, deleting, and modifying user account information, which is stored in a specific set of required tables. These tables are created automatically, and have been placed in the AEQ back-end database. The infrastructure created by the ASP.NET 2.0 Membership, Roles, and Profile features is programmatically used to provide the user account management functionality within the AEQ application. The same infrastructure can also be accessed and used interactively through the Web Site Administration Tool provided with ASP.NET to manage some of the application's security, application, and provider settings. Through the security section of this tool you can create and manage users, roles, and access rules. This includes creating new users and roles as well as assigning existing users to roles.

Three different roles have been defined for the AEQ application: (1) Administrator, (2) Instructor and (3) Student.

(1) Administrator – this role is the only one that provides access to the Administrator tab. Typically, there would be one single administrator for a given installation of the AEQ application. The administrator is the only user account that cannot be created through the available functions of the AEQ application. One would create an administrator through the ASP.NET Web Site Administration Tool at application installation.

The main task of the administrator is to add and remove Instructor users (Figure 7).

A new instructor is created based on three pieces of information: first name, last name and an email address. Based on these, the application generates a username for the instructor consisting of the first initial of the first name and the last name plus an appended number in case of collision with an already existing username. A random password is also generated, and both the username and password are sent into an e-mail to the address that was entered as the third piece of information. In addition to creating the user account, procedures are called to create the working databases for the new user in the MSSQL, MySQL, and Oracle databases. The corresponding database connection strings are generated and linked to the new user account in the AEQ database.

The administrator tab also provides the functionality for dropping an instructor. The first thing the application does, at a request to drop an instructor, is to check that the instructor is not associated with any classes. If the instructor is still related to a class, an error message will be displayed instructing the user that the course must be dropped first prior to removing the instructor. If the user does not have a class, then the procedure removes all data related to the user from the database. The associated working MSSQL, MySQL, and Oracle databases will also be dropped.

(2) Instructor - only instructors and the administrator have access to the User/Class Management tab which allows them to create classes, import students to each class and create corresponding student accounts. This set-up makes it possible to start with the single administrator role when installing the application and create and remove all other needed accounts from within the application itself.

The goal of the User/Class Management tab is to support instructors with some of the routine management tasks they perform at the start and end of each semester: create a

new class, create accounts for the students in that class, and remove a class with all dependent information associated to it; i.e. students and accounts as well as working databases.

On the User/Class Management tab the instructor can choose to modify an existing class or create a new class (Figure 8). A JavaScript procedure is called to hide/show the appropriate fields based on whether an existing class or the option to create a new class is selected. When a new class option is selected, an AJAX request is sent to the Web server to create a new class. The application will first verify that a class with that name does not already exist, and if not, it will create a new class and populate the necessary tables in the AEQ database, then it will link the current user to that class as the instructor.

Once a class has been created, the instructor has two options. First, the instructor can import a list of students to add to the class. The list must exist in a .csv file and include a Student_ID, First Name, Last Name, and E-mail address for every student. For each student that is imported, a new Student user is created with an automatically generated username and password. The username consists of the class name, the first letter of the first name, and the last name of the student. If the username already exists in the system and it belongs to a student with a different Student_ID, then the system will append a number to the end of it to produce a unique username. The system also uses a built-in method to generate a random password, which currently consists of 10 characters with at least three non-alphanumeric characters. Next, the procedures are called to create the working MSSQL, MySQL, and Oracle databases for the user. The corresponding connection strings are linked to the user account, and an e-mail is sent to each new user providing them with their username and password for the application.

When a class with students enrolled in it is selected in the dropdown list, the list of students currently assigned to that class will be listed below for verification. Further students can be added to an existing class from a different import file.

The instructor also has the ability to drop a class he/she selects. When the drop button is clicked, a confirm dialog appears making sure the instructor wants to drop the course. If it is confirmed, an AJAX call is sent to the server and calls a procedure to drop the class. The process of dropping a class starts by removing all dependent databases associated to the students in the class, followed by deleting the students' accounts. Upon completion, if any error messages are received, they are displayed to the user in a JavaScript alert.

Both the Administrator and the Instructor roles have access to the functions of the User/Class Management tab. However, while an administrator can manage all the existing classes in the application, instructors can only see the classes created by themselves.

(3) Student – this is the most restrictive role in the application. None of the functionality related to the Administrator and Instructor role are available to student users. Students are limited to access the course content viewer, the SQL Launcher and the features in the chatroom.

5.4. The Chatroom and Shared Mode Support

One of the main objectives in the development of the AEQ application was to promote collaborative work among students, and between students and instructors. The AEQ application allows users to communicate and to work together from wherever they may be accessing the site. This concept is supported by three different features that complement each other: (1) The Chatroom, (2) The Active Users Window and (3) Shared Mode Support.

(1) The Chatroom – is an AJAX driven chat feature that is implemented within the main design of the application, so that it is available at any time no matter the kind of activity the user is performing. The feature consists of a chat window and a text box used to send messages, which are located on the left side of the application window. The area occupied by this feature can be adjusted horizontally or can be minimized to leave almost the entire window space to other activities. Users can send public messages and view all the messages sent by other users. Any messages received while the user is logged in are displayed along with the timestamp of the message and the first and last name of the user who sent it. The list of messages is refreshed automatically every two seconds

in all active user browsers via AJAX requests issued by the clients. New users logging into the application will receive only the messages issued after the moment their working session started. Figure 9 shows a sample communication session among three users.

(2) The Active Users Window – complements the functionality of the Chatroom by showing the list of users currently online and their session mode, i.e. shared or not. This list is refreshed every two seconds through a mechanism similar to that used for the Chatroom. The users list is displayed in a separate browser window that can be dragged anywhere, resized, closed, and reopened by clicking the 'View Online Users' link. By convention a user is considered on-line for 2 minutes after his/her Last Activity Date. A user with no action for more than 2 minutes will be automatically removed from the list.

(3) Shared Mode Support – every user can view his/her activity history by clicking on the icon in the upper-right corner of the SQL Launcher. The history consists of the past executed queries and their truncated outcome as logged in the AEQ database. Only the first 10 rows returned by a query are included in the history. Whenever a user enters the shared mode he/she makes his/her own history available to all other users who then can view, copy and test on their own the queries just produced by their peers (Figure 10). When a user enters the shared mode his/her name is inserted automatically into the drop-down list located by the "History View" icon. This makes the history of that particular user history available for selection by other users.

This feature is an important part of the collaborative work support in the AEQ application, since it allows the students to truly share their work and experience, while through the Chatroom feature they can share comments and suggestions related to the executed queries and their outcome. The two features nicely complement each other to provide a powerful support for various types of collaborative activity.

6. ASSESMENT, EVALUATION, EXPECTED IMPACT ON CURRENT PRACTICES

The nice thing about the AJAX technology is that features that traditionally were available only in Windows applications can now be incorporated in Web applications as well. The AEQ project taps into this opportunity and combines it with other specific features that are expected to have an impact on both student and faculty experience from several points of views. By adopting the AEQ tool in the database classes, changes in every day activity can be anticipated, such that both the student's learning and instructor's teaching experience will be improved:

1) The AEQ tool acts as a Web complement, and in some situation, even as a replacement of various DBMS client query tools, that will be much easier available and will likely allow and encourage students to put in extra hours in their class preparation.

2) As a combination of both content viewer and query tool, AEQ provides the students with the illusion of "live" course content; the students will be able to read and study the course materials, but also to test and run the sample queries directly from the lecture notes by using the same software interface.

   The above two features make it faster to achieve the objective of increasing student proficiency in using SQL syntax, as the whole user experience is much simpler and more natural.

3) The unified interface query tool feature allows students to run the same query under several DBMSs (Oracle, MSSQL, etc) after a simple option selection in a radio button. One of the important tasks when teaching SQL syntax is to make students understand the differences between various SQL dialects, and emphasize on the concept of portable SQL. Traditionally, this requires working with different DBMS clients at the same time, which may be a confusing and frustrating experience, due to differences and inconsistencies between various database client tools. AEQ provides a unique and simplified interface to several DBMSs at the same time, thus allowing students to test their queries in all these systems with much less distraction, making it easier to focus on the in-

trinsic task of understanding syntax differences between various SQL dialects, as well as on portable SQL syntax.

4) The chatroom feature and shared mode capability are expected to stimulate the collaborative learning experience among students as they will be able to share their impressions, findings and experience on-line and watch live each other's attempts to solve difficult problems.

The AEQ tool is likely to impact not only on students' experience, but also on instructors' activity. The nature and benefits of this impact may not even be fully understood at this point, and may turn out to be more profound than expected:

1) The dual nature of query tool and content viewer of AEQ is expected to encourage instructors to develop their course materials with interaction in mind, thus opening a variety of opportunities for innovation in course development. The versatile features of the content management component (text editing, file upload, dynamic content tree operations) are expected to make AEQ a preferred tool for instructors to prepare the on-line presentation of their courses.

2) As a Web based content management tool, AEQ provides the opportunity for joint class development, co-authoring, and even co-teaching. In this process the instructors themselves can benefit from the advantages of the chatroom feature and shared mode capability.

The features of the AEQ tool, as currently envisioned, are the result of introspection and reflection on several years of teaching experience. However, these features are expected to be refined, extended or replaced by other features in a phased process of alternating development, implementation, and assessment over several cycles. In this sense, the AEQ experience is only partially a software development project, as the incremental process of implementation in real classroom environments, followed by thorough assessment from several perspectives are equally important.

In an initial phase, the pilot implementation of the AEQ tool is to be made available to the faculty participating in the project for the following purposes: functional evaluation, bug hunting and code review. The findings in this phase are to be incorporated into a prototype that will be used experimentally in selected classes. Instructors will prepare and upload course materials for their database classes into the prototype implementation of the AEQ tool. The course material will be organized in a manner that would support the concept of "live content" within the AEQ tool. A group of selected students in these classes will be required to use the AEQ tool for their class activities. An assessment process will follow and the findings will be further incorporated in the AEQ tool implementation.

The assessment process is viewed as a cyclic activity in time that is meant to provide the development team with the feedback needed to improve and refine the AEQ tool. The information in this process will be collected from students, directly and indirectly, as well as from their instructors. The assessment process is also expected to provide feedback that instructors themselves may benefit in their effort to better use the AEQ tool.

The information from students will be collected both directly and indirectly through a multiphase activity distributed in time:

1) A detailed student questionnaire will be given to the students using the AEQ tool at the end of the semester. Students will be asked to respond a set of questions that focus on several key aspects of the AEQ tool and its use: rank the usefulness of the various features of the AEQ tool, make suggestions to add/change/refine features, evaluate the total number of hours they spent working with the tool and other.

2) The statistical results of the questionnaires collected at different points in time will be compared to assess the dynamic of the tool development and refinement, as well as the instructors' learning curve in how to better use the tool.

3) Standardized SQL query tests will be given to the students exposed to the AEQ tool as well as to students using

the traditional learning techniques. The test results of the sample groups will be comparatively analyzed.

The second important source of information in the assessment process is faculty feedback. Faculty feedback will be used from two different perspectives:

1) Their direct experience with the AEQ tool: ease of use and effectiveness of various features, overall critical evaluation of the tool.

2) Indirect experience, via their students, by observation and subjective evaluation of how students relate and react to the AEQ working experience.

7. IMPLEMENTING THE AEQ TOOL - THE CAPSTONE PROJECT

The implementation of the AEQ tool, with the specifications outlined in section 3, was proposed to a group of graduating students as their capstone project. The result is a functional pilot application as described in section 5, based on an architecture distributed on three levels of coding: JavaScript on the user machine, ASP.NET and C# on the Web Server, and SQL stored procedures on the backend database server (see section 4).

The choice on the software technologies used in the development: AJAX+ASP.NET+MSSQL, was conditioned by the previous experience of the students and their instructor. All students have formal training in ASP.NET and one student attended a formal AJAX class. To this, one can also add the favorable experience all participants had with these technologies in the past.

If the choice of technology clearly implies the given architecture, one could have quite some debate as far as the functional specifications of the application are concerned.

The specifications aimed to provide a complete set of features, which illustrate the initial vision of the AEQ tool and address the main problems that motivated the entire initiative, but have been deliberately kept at a minimum for two reasons:

- the main purpose of the capstone project was to validate the concept of the AEQ tool by finalizing a minimal, but functionally complete implementation, rather than an in-depth approach;

- the limited time-frame (less than 15 weeks) available for the entire cycle of analysis, design and implementation.

As a result, many new features, as well as refinements to the existing ones, can be added in various parts of the application, and will be considered in further development phases:

1) The SQL Launcher could benefit a function that would allow the interactive exploration of the target database content. At a minimum, one should be able to view the list of available tables, but more sophistication could be added to see the structure of the tables (columns, keys, etc). Also, interactive visualization of other database objects like views, constraints, stored procedures and other may be of interest. However, the AEQ tool does not intend to compete with the available windows based database clients, some of which have a quite sophisticated set of such features. Besides, all that information can be currently obtained without too much effort by any clever user simply by running the right queries against the system tables in the target databases.

2) Several additions could be considered for the Content Viewer and Manager:

- connect the "Table of content" navigation function with the full view expansion feature in the same window, such that navigation in the table of content will bring the corresponding full content in the view;

- allow direct execution of queries the full content text;

- enable the content manager to accept various content formats and sources, e.g. Power Point.

3) The Administrator tool should add functionality to the Instructor role for more refined control on the classroom working environment:

- the ability to manage individual students: accounts, databases, etc.

- instructors could be allowed to define and manage the access privileges students have on the target databases. For example students in an advanced database class may be given more extensive rights than those in an introductory database class.

- other features like monitoring activity time, managing grades etc. Many such features are common to more general products like D2L and it is uncertain whether and which of these should be included in the AEQ tool.

4) Addition of many new features can be imagined for the Chatroom and Shared Mode Support:

- true group collaboration, such that messages of students from different classes or groups do not interfere;

- availability of various levels of message broadcasting for instructors such that messages can be sent to a class, a section or a group of students;

- shared query window where students, or students and instructor can co-edit SQL queries and execute them.

Some of these additional features, and other not anticipated at this point, may prove to be useful, while other may turn out to be unrealistic or useless. In order to find out, it is important to approach the entire development process incrementally, adding new features based on the assessment and user feedback from previous cycles as outlined in section 6.

8. CONCLUSIONS

For several decades there has been quite some anticipation with respect to the role information technology would play in the realization of the "dream of universal access to high–quality, personalized educational content (…) available both synchronously and asynchronously" (van Dam et. all 2005). So far, reality is behind expectations, in spite of the fact that there is no shortage of visions about how information technology in general and educational software in particular, may change the way we teach and the way our students learn. The complexity and multidisciplinary nature of the factors conditioning progress in this area needs to be correctly recognized and understood. Successful and effective teaching is largely about interaction, and the on-line teaching experience demonstrates how important it is to properly address the shift from the traditional forms of instructor-student or student-student interaction towards the new forms of interaction involving various technologies used in distance education (Sherry 1996; Ekhaml, Roblyer 2005; Feenberg, Xin 2005). Another important factor in the equation is availability, as without easy and widespread access, even the most wonderful tool will have limited or no impact. Van Dam (van Dam et. all 2005) also points out that achieving the ambitious goals of modern education has been conditioned by the pace of advances in hardware and software technology. The implication of this is that any incremental progress in technology would be followed by educational tools that "fill the gap" by putting the new technology to work. In this perspective, the AEQ tool presented in this paper, may have the chance to be on the right track by its potential widespread availability due to the ubiquitous presence of the Internet, as well as through the way it approaches the issue of interaction by leveraging the AJAX technology.

9. ACKNOWLEDGEMENTS

This paper has been supported by a University of Wisconsin System LTDC (Learning Technology Development Council) Grant (2008-2009) as well as by the University of Wisconsin Stevens Point L&S UEI 2009 Program.

10. REFERENCES

Blease, Derek (1986) Evaluating Educational Software, Chapter 5, pp.96, Published by Rutledge, 1986, ISBN 0709939159

Brogan, Patricia (2001) "The Good, The Bad, and the Useless – Recognizing the signs of quality in educational software", http://www.electronicschool.com/2001/03/0301f3.html

Van Dam, Andries and Shasha Becker, Rosemary Simpson (2005) "Next-Generation Educational Software: Why We Need It and a Research Agenda for

Getting It", EDUCAUSE Review, vol. 40, no.2, (March/April 2005):26-43.

Ekhaml, Leticia and M.D. Roblyer (2005) "How Interactive are YOUR Distance Courses? A Rubric for Assessing Interaction in Distance Learning", Online Journal of Distance Learning Administration, Volume III, (Number II).

Feenberg, Andrew and Cindy Xin (2005) "A Teacher's Guide to Moderating Online Discussion Forums: From Theory to Practice."

Gibbs, Matt and Dan Wahlin (2007) Professional ASP.NET 2.0 AJAX, Wiley Publishing, Inc.

Gold, Patricia Cohen (1984) "Educational Software--New Guidelines for Development", AEDS Journal, Vol.18, no.1, pp. 41-50, Fall 1984

Goyne, June S., Sharon K.; McDonough, Dara D. Padgett (2000) "Practical Guidelines for Evaluating Educational Software", The Clearing House, July 1, 2000, http://www.allbusiness.com/technology/computing-information-technology-overview/7358245-1.html

Johnson-Glenberg, Mina C. (2006) "Judging Educational Software", WTN – Wisconsin Technology News, July 18, 2006

Lauterback, Roland (1989) "In Search of Quality – Educational Software", Zeitschrift fur Padagogik, Vol.35, no.5, pp.699-710, September 1989

McClure, Wallace B. and Scott Cate, Paul Glavich, Craig Shoemaker (2006) Beginning AJAX with ASP.NET, Wiley Publishing, Inc.

Mitchell Scott (2006) Examining ASP.NET 2.0's Membership, Roles, and Profile - Part 3**,** http://aspnet.4guysfromrolla.com/articles/040506-1.aspx

Moore, Dana and Raymond Budd, Edward Benson (2007) Professional Rich Internet Applications: AJAX and Beyond, Wiley Publishing, Inc.

Sherry, Lorraine (1996) "Issues in Distance Learning", International Journal of Educational Telecommunications, 1 (4), 337-365 Issues in Distance Learning

Woolston, Daniel (2006) Pro AJAX and the .NET 2.0 Platform, Apress.

Zakas, Nicholas C. and Jeremy McPeak, Joe Fawcett (2006) Professional AJAX, Wiley Publishing, Inc.

Appendices and Annexures

Architecture of the AEQ Application

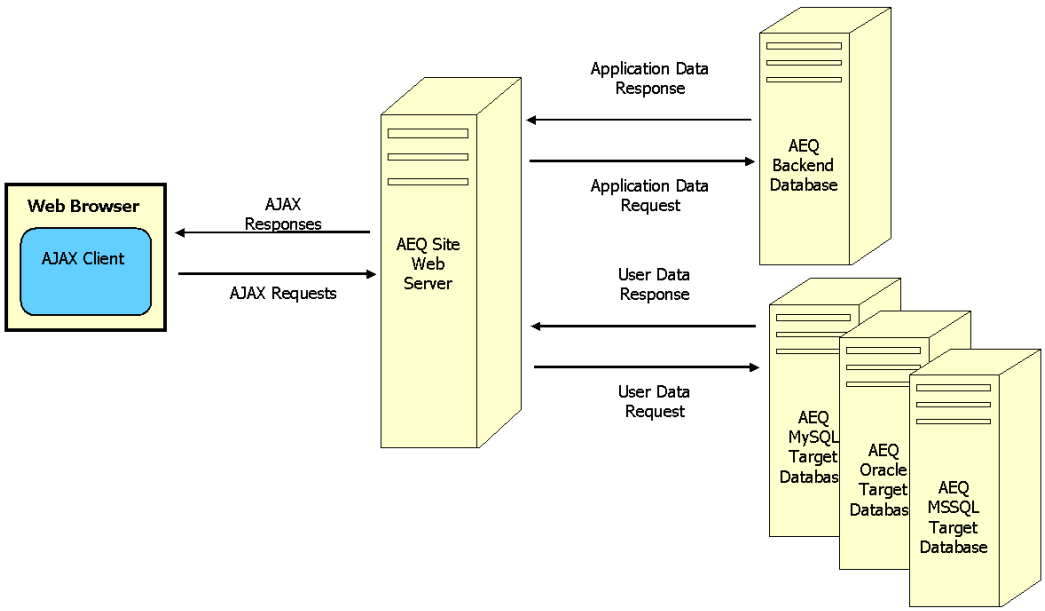

Figure 1. Architecture of the AEQ Application

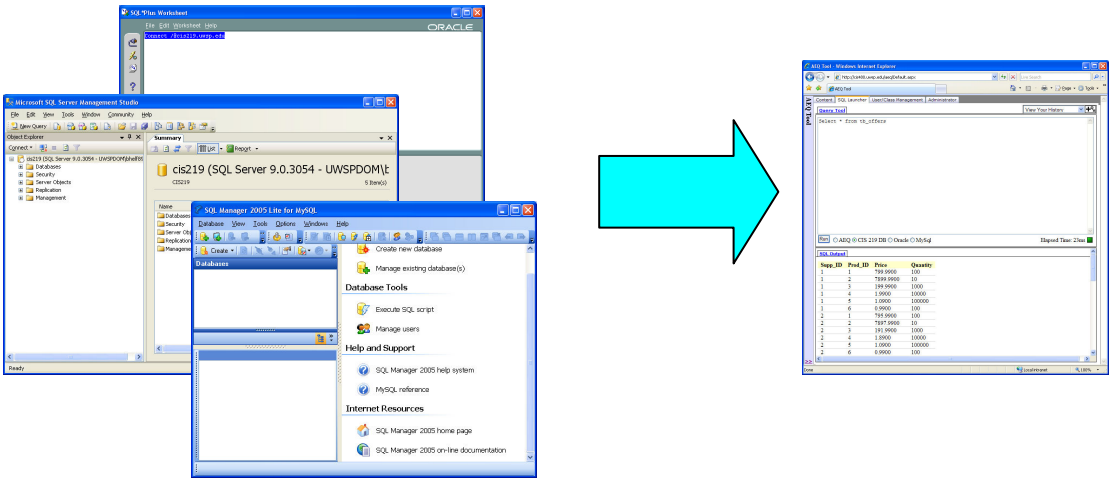

Figure 2. The SQL Launcher Equivalent Functionality

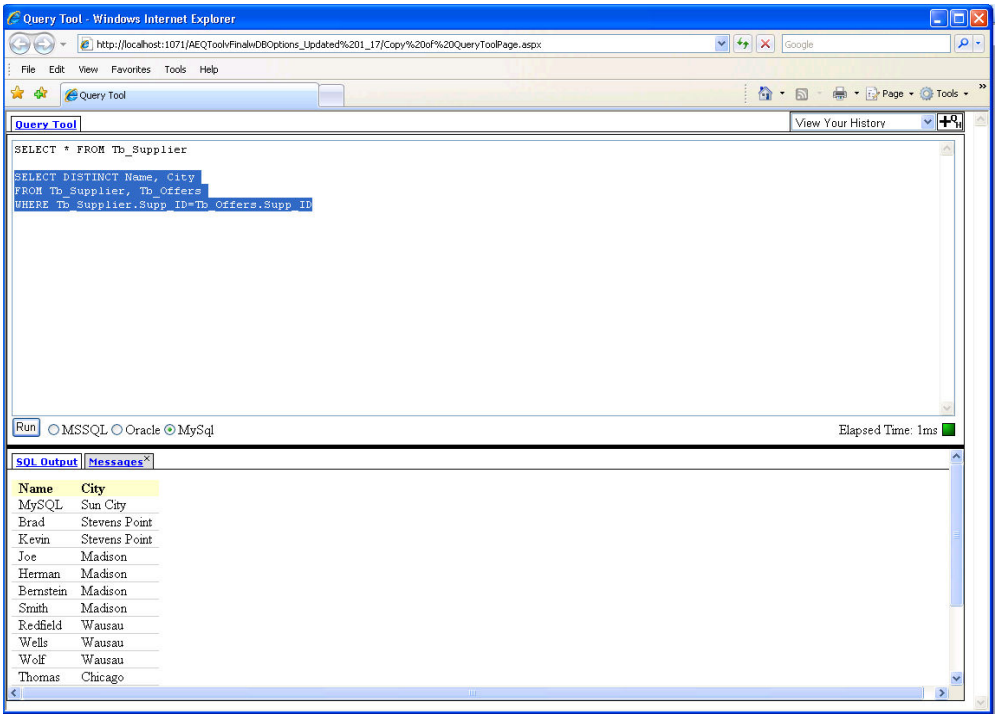

Figure 3. The SQL Launcher executing a query

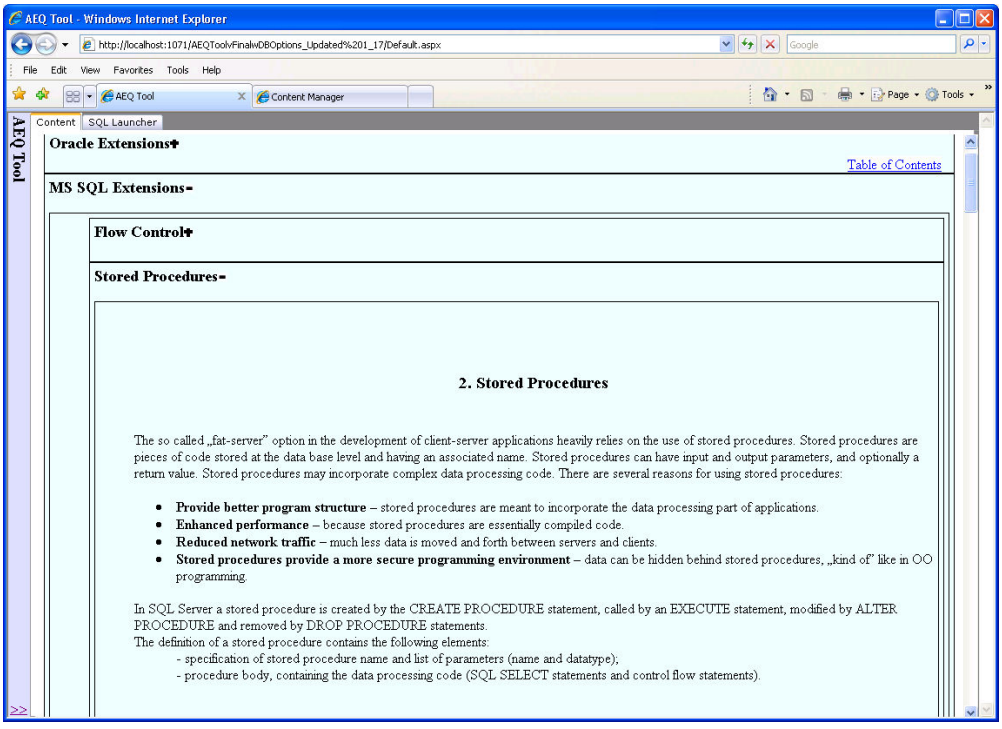

Figure 4. Student View of the full course content.

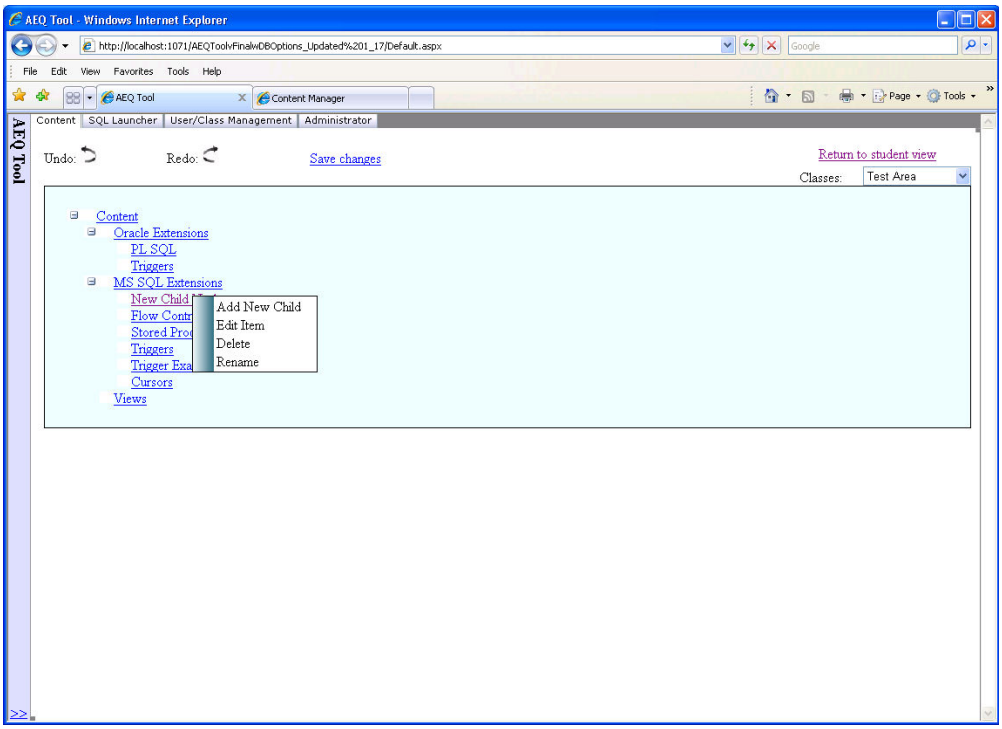

Figure 5. Content Manager with options for a new child node

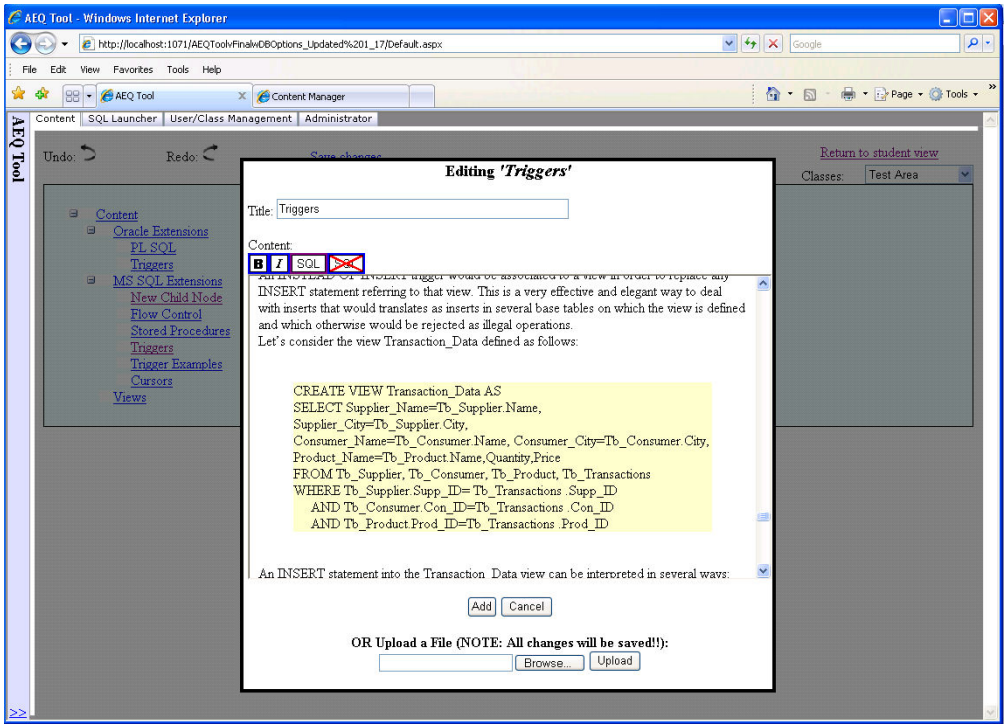

Figure 6. Content Editor with text and SQL content.

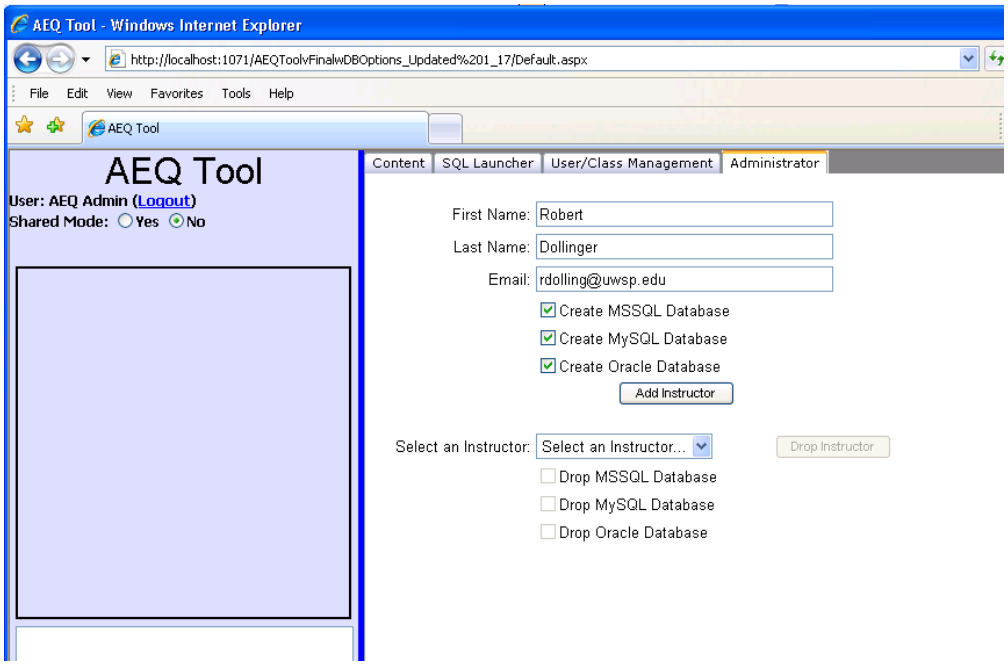

Figure 7. The Administrator tab

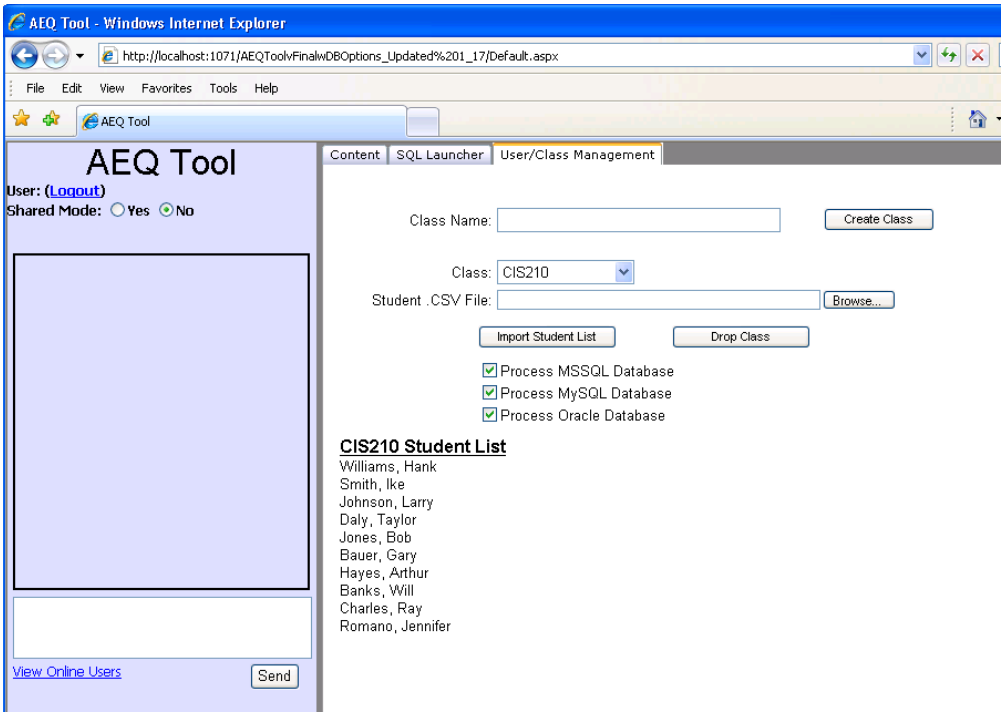

Figure 8. User/Class Management tab

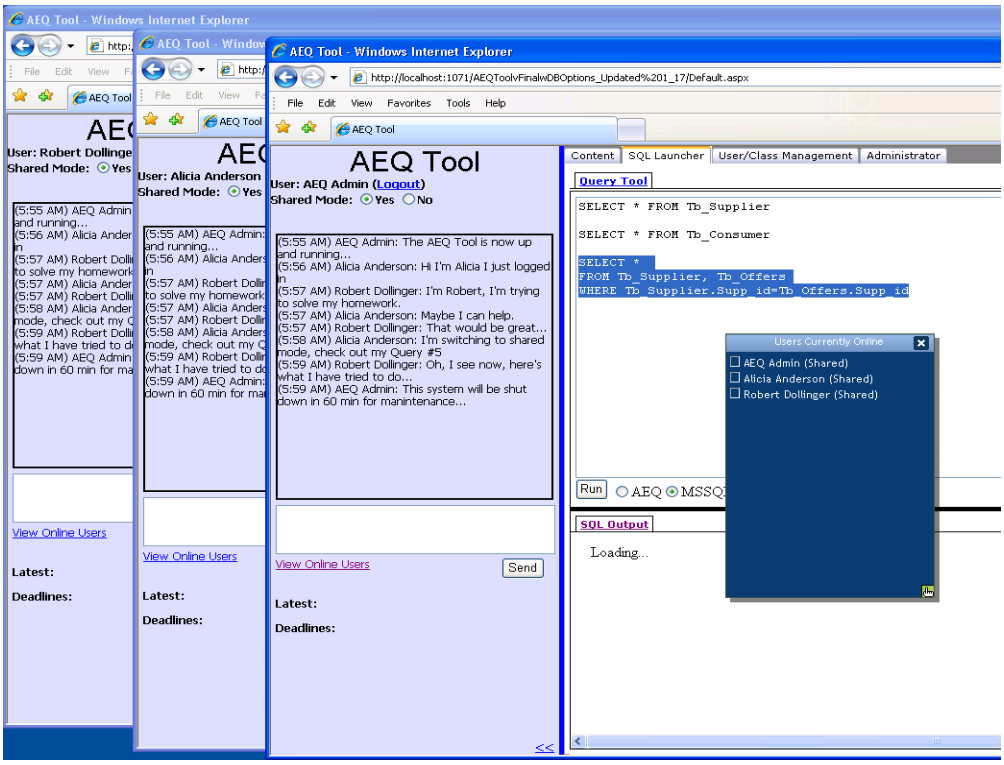

Figure 9. The Chatroom and the Active Users Window

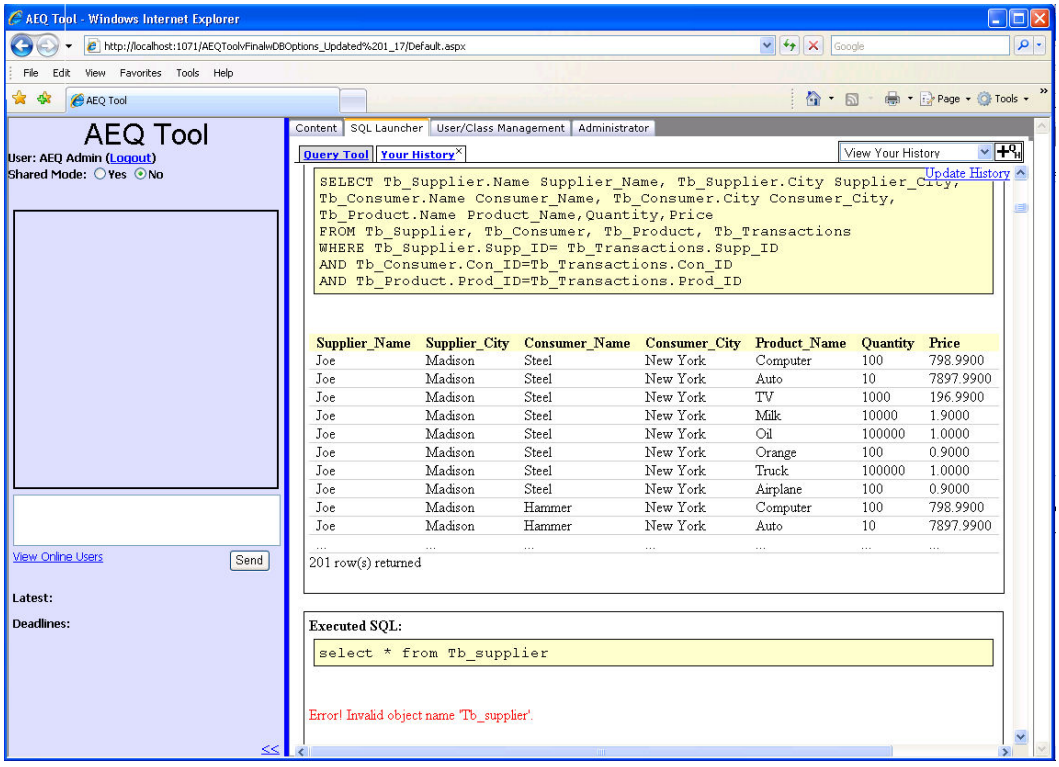

Figure 10. History with truncated query results