Database Courseware:
Animating SQL queries

Mario Guimaraes, mguimara@kennesaw.edu
Martha Myers, mmyers@kennesaw.edu
Tulin Kilinc, tkilinc@bellsouth.net
Steven Setzer, ssetzer@kennesaw.edu
Kennesaw State University

Abstract
The goal of this paper is to describe the development of animations and other automated courseware to support database curricula, as well as to elicit ideas and suggestions from other educators in this area. Part of an NSF funded project (NSF grant number 0089412), current work has focused on the design and implementation of animations concerning SQL concepts. The first section describes the project and its impetus. The next section describes prior software systems that communicate computer science concepts through animation. The fourth section describes the prototypes that have been implemented thus far. The last section describes preliminary assessment of the prototypes and the next phases of this project.

1 Objective and Motivation
Database concepts are a required and important component in many undergraduate computer science programs. It is difficult to communicate breadth and depth of concepts in a single course, yet most undergraduate programs do not have space to require, or even offer a second course on the subject. During ACM-SIGCSE 2000 a group of seasoned database professors presented several key issues in effective database instruction [1]. Participants report little consensus on content due to the number of important topics to cover in one semester. These experts also reported that the content and methods of teaching of changed substantially in recent years. Participants agreed that there was not enough time to teach all the fundamental database topics in one course.

Recently, Arizona State has recognized the need for a second database course in the undergraduate curriculum [2]. Their research confirms that most universities have room for a single undergraduate database course in their programs of study. This data supports a need to incorporate newer technologies into a database course, providing students with a variety of channels for knowledge acquisition. Students retain only 15% of material presented in lecture and readings [3]. Variety in teaching pedagogy, especially those using technologies of interest to CS students, can increase the likelihood of long-term mastery of this important subject. Animated, interactive courseware has the potential to increase knowledge retention by 50% [3].

The primary goal is to design and develop animated courseware that will deepen and enrich standard presentations of database topics to majors in computer science and related fields. The current project concentrates on the creation of animated courseware that may accompany and complement popular database textbooks such as those by Date, Elmasri and Navathe, O'Neil, and McFadden [4], [5], [6], [7]. The Elmasri and Navathe book provides the foundation for this preliminary work [4].

This paper describes specific prototypes implemented to teach certain aspects of SQL and Relational Algebra. Additional prototypes are now being designed and prepared for implementation. These include animations of the transformation from Entity Relationship Diagrams (ERD’s) to relational design specifications, as well as transformations from object-oriented models using class diagrams from the Unified Modeling Language (UML). Other prototypes include animations of issues related to concurrency, as well as database indexing.

2 Background and Related Literature
Animation has been developed and used to communicate other difficult concepts within computer science curricula. It is believed that animations enhance the student’s learning experience by providing an alternative presentation of abstract concepts. Students who struggle with abstract concepts may do so because they are unaccustomed to generating their own internal, pictorial representations of concepts communicated in narrative form. It has been said that a picture is worth a thousand words, and that an animation is worth thousands of pictures [8]. Prior systems have been developed to illustrate such concepts as programming principles and data structures [9-29].

A review of current literature indicates that little has been done previously to illustrate database concepts. One tool to teach databases is WinRDBI [29]. WinRDBI allows students to write their own code (in Relational Algebra, SQL, Tuple Calculus) and execute pre-defined examples (Company, University and Cars). However, most prior work has focused on programming knowledge and learning.

3 Prototypes Developed
Thusfar, the researchers have implemented prototypes in three areas: SQL, database design, and concurrency concepts. All related work is available at this web site:
The investigators have engaged several undergraduate and graduate student teams to address various aspects of the overall project. In order to make use of current student skills, all prototypes thus far have been developed in either Visual Basic (VB), or as Java applets. Since considerable modifications normally occur to any instructional software after testing it in a classroom environment and receiving feedback from teachers and students, VB seemed an appropriate RAD tool to use to generate rapid prototypes. Visual Basic and Java are two developing platforms taught at this university; thus it was easier to find students with some experience and it is available in the computer labs throughout campus.

The VB prototypes may be easily downloaded from the web site, but require the run time version of VB for execution. Interested readers may download a free run-time version at several web sites including http://www.windrivers.com/company/vb and http://support.microsoft.com/support/kb/articles/Q235/4/20.asp. The java applets will run within most web browsers without additional software. These have been tested with IE 5.5 and Netscape 4.7.

This paper describes the prototypes to aid in teaching SQL. SQL exercises require that the user transform English requests or queries into SQL syntax, include relational selections, union, and a variety of joins. Details of these exercises follow:

1) **Problem**: List all customers who live in Dallas.  
   **Solution**: Users are to develop a restriction/selection (Figure 1a), and implement the query  
   “Select * From Customers Where city = ‘Dallas’;”

2) **Problem**: List all students and list all alumni.  
   **Solution**: Users are to develop a union (Figure 1b), and implement the query  
   “Select * From Students Union Select * From Alumni;”

   Note that this implementation of the union allows duplicates. Another part illustrates solutions not allowing duplicates.

3) **Problem**: List all department names and corresponding employee names.  
   **Solution**: The user is to develop a join (Figure 1c), and implement the query  
   “Select d.dname, e.ename From Department d, Employee e  
   Where d.dnumber = e.enumber;”

4) **Problem**: List customers that ordered all products.  
   **Solution**: The user is to develop a Division (Figure 1d), implementing the query  
   “Select cid, cname from Customers c where not exists  
   (Select * from Products p where not exists  
   (Select * from Orders o where c.cid = o.cid and o.pid = p.pid));”

5) **Problem**: List all employees and their corresponding bosses.  
   **Solution**: The user is to develop a join within a single table (Figure 2), and implement the query  
   “Select eName, superName From employee as e, employee as s  
   Where e.superSSN = s.SSN;”

6) **Problem**: List all customers that did not order a product (Figure 3).  
   **Solution**: The user is to develop and implement the following query  
   “Select cid, cname From Customers c Where  
   cid not in  
   (Select cid From orders Where pid = ‘p01’);”

Animations of these exercises are composed of (1) a description window that describes the problem and the objectives of the program; (2) an SQL query window containing the SQL query that solves the problem; (3) one or more data windows containing the tables; (4) a procedure window containing the simplest form of algorithm that implements the SQL query; (5) and an output window containing the results of the query. They also contain at least two additional buttons: a “clear” button that returns the query to the initial stage, and a “step” button that executes the highlighted line of the procedure window and highlights the next line every time that it is clicked. Depending on the query, there will also be a Relational Algebra window, containing the corresponding Relational Algebra solution to the exercise. These animations allows students to associate code that students already know (procedural or relational algebra) with the concepts they are learning (SQL). It also allows them to examine the execution of a query in steps.

Figures 1a through 1d show animations of SQL statements that implement basic relational algebra operations: selection, union, join, and division. Figure 2 shows a join example and Figure 3 shows a “Not In” example.

Additional prototypes have been implemented though they are not discussed in this paper. The reader is encouraged to review additional prototypes including relational projection, multiplication, intersection and minus. Additionally there are prototypes examining the use of functions, as well as “group by” and “having” clauses.
4 Preliminary Results and Future Work

The first step in evaluation involved tests by local instructors and a small group of students. Following this early feedback, prototypes were then provided to one ongoing database class. Students were shown one example in class and then asked to review the remaining SQL examples outside of class. Results from these assessments led to several improvements in the software, described below. The next step in assessment is to test it and receive feedback on a larger scale.

Some improvements were needed in order to provide students with a clear context and problem narrative. As with other educational software, such as animations of data structures and programming concepts, developers assumed some knowledge that was not confirmed. This led to the addition of a description window, describing the problem and the objectives. Other improvements have focused on user interface issues such as color and highlighting. Helpful suggestions led to placing more emphasis on the SQL Window and the problem description window. Another enhancement is allowing the user to simply watch the full animation or to step through it using a button. However, for this option to be truly useful, the end-user must be able to control the velocity and be able to interrupt the execution. Future work will incorporate this capability.

We have also created an environment where students can construct their own queries. This gives the end-user the additional choice of hands-on query construction, in addition to the above described visualization exercise. Figure 4 shows an example of the Construction exercise using the GROUP BY and HAVING clause. The construction module allows student to break SQL queries into parts and quickly access the results. Notice that the student can manipulate the variable part through drop down menus and immediately see the results.
5 Presentation Focus

The researchers will briefly present background literature and demonstrate some key prototypes in order to elicit suggestions from the audience. Audience members will also be encouraged to review the remaining prototypes and provide feedback via an on-line form at http://coffee.kennesaw.edu.

References


Computer Science


[20] Sorting Algorithms, 


[22] The JAL animator, 

[23] Java-Based Collaborative Active Textbooks, 
www.research.digital.com/SRC/JCAT.

[24] The Lawrence Algorithm Visualization Web Server, 
http://gaigs.cmsc.lawrence.edu/.

[25] Welcome To Algorithmia, 


[28] ACM Student Research Competition 
http://www.acm.org/spost/.

http://www.eas.asu.edu/~cse412/.