SQL-trainer

Harri Laine
Department of Computer Science, P.O.Box 26, FIN-00014 University of Helsinki

Harri.Laine@cs.Helsinki.FI

1 Introduction

SQL database language is a basic component in the first database course. Every programmer who writes programs for database applications must know how to use SQL. Learning SQL requires practice. If the practice is with pen and paper only, the students don't know whether their solutions are correct of not. They should be able to work with a live database. SQL is the standard interface in all relational database management systems. Thus, a practice environment may be build up by installing whatever relational database management system and establishing an example database. This is a straightforward solution, but is not very good for educational use. The user interfaces that may be used without installing the client software for the database management system are typically poor and assume that students have rights to access the database server. Our experience with this kind of practice environment indicates that some students do not get their tasks done at all, and most students are satisfied with the first attempt that is able to produce a real result instead of error messages.

In the University of Helsinki the first database course is a first year course in the curriculum of Computer science. About 700 students take the course yearly, and less than half of them are Computer science major students. The course is lectured in each term. Our goals for arranging the SQL practice in this course were

- to make it possible to practice with a live database anywhere without any special client software and user accounts
- to provide a user friendly multi-lingual (Finnish, English) user interface
- to allow the use of any database as the example database
- to provide students feedback about their solutions
- to automate the bookkeeping and checking of solutions and thus to reduce the teaching resources needed for the course.

SQL-trainer is our solution to these goals. SQL-trainer is a web-based practice environment for SQL. Chapter 2 of this paper discusses the tasks, the correctness of solutions and the feedback provided. Chapter 3 outlines the architecture of SQL-trainer. Chapter 4 discusses the analysis of the solutions. Chapter 5 presents how SQL-trainer is used in our courses. Chapter 6 contains some remarks about the further development of SQL-trainer.

2 Tasks and their solutions

SQL is a database language. It may be used for defining and re-organising the database, for making queries on the database, and for maintaining the database. SQL-trainer may be used in the practice of queries and maintenance operations. The tasks to be solved as SQL-queries are expressed as natural language definitions of information needs. The goal of the practice is that the students learn to build up queries that correctly produce results that satisfy the needs specified. We could make the tasks so specific that there is only one correct result to satisfy the need given in the task. Making queries is, however, more than just technically transforming a very detailed request into SQL. In practice, one must usually interpret the request, decide what data to include in the result and how to organise the result. When

this is the case, then there are usually more than one correct results. SQL-trainer allows many correct results. A result is considered to be correct if it fulfils the task specific result constraints. These include that

- the number of rows is correct
- at least the given columns are included in the result
- the order of the rows is the required one
- checksums over certain columns are correct (currently only one column is checked)
- there are not too many extra columns in the result, and
- columns with equal contents are not included unless required.

As compared to a single correct result this approach allows the columns of a result table to be ordered in alternate ways, some additional columns to be included and the freedom to choose the format of some non-essential columns.

In database maintenance the result is the subject table after the maintenance operation has been done. The correctness of a maintenance result assumes that

- the number of rows is correct, and
- the checksums over certain columns are correct.

Although the result of a query given as the solution for a task is correct, the query itself may not be correct. It may use tables that are not necessary or it may not use all the necessary tables. The selection of the rows may use information not given in the task definition. SQL-trainer uses query constrains to place requirements on the queries. These include the requirements that

- the given tables must be used
- the given tables should not be used
- certain character strings must be included, and
- certain character strings should not be present (this has been used to reject the fake queries)

SQL-trainer considers a task to be correctly solved, if both the constraints on the result and the constraints on the query are satisfied. Our constraints for correctly solved tasks are not complete. It might be possible to produce a query that is not correct but is accepted as a correct solution. Careful design of the example database and the tasks reduces this risk but does not eliminate it. Especially aggregate queries are problematic in this respect. The amount of constraints checked in SQL-trainer is considerably less that the collection of 199 constraints in the SQL-tutor learning tool [2, 3]. SQL-trainer is still able to provide useful feedback about the errors in the solutions.

There are two types of errors in the solutions: syntactic and semantic. Syntactic errors are violations against SQL syntax. Each database management system (DBMS) carries out a syntax check during query compilation and reports about the syntax errors. The quality of error messages varies according to the DBMS, but that is the facility the users have to work with, in practice. SQL-trainer lets the DBMS to take care of the syntax checking. Database management systems are not able to check the semantics of database operations. The constraints in SQL-trainer deal with the semantics. They are checked when the operation is found to be syntactically correct. Checking the constraints produces feedback. The feedback is provided according to the constraints that are violated. The feedback tries to point out the problems in the solution and to assist the student in finding the correct solution. The feedback might be, for example,

- *You should use table X*
- *You need to provide the columns Y and Z*
- *The result should be ordered according to Z*
- *The result contains too many rows. The correct amount is P. The result probably has duplicate rows*
- *The result contains too many rows. The correct amount is P. There may be a missing join condition*
- *You are not supposed to know the identifier W*
- *The content of column Q is not correct.*

3 The architecture

SQL-trainer is designed to be used connected with a database course. It provides services to keep book on the students and their solutions. It uses a centralised system database for bookkeeping. The definitions of the tasks are also stored in the system database. One SQL-trainer instance is able to handle many concurrent courses, for example a lecture course and an e-learning course. Also the example databases used in queries and in maintenance operations are centralised.

SQL-trainer consists of three tools: a training tool for the students, and a task manager and statistics tools for the teachers.

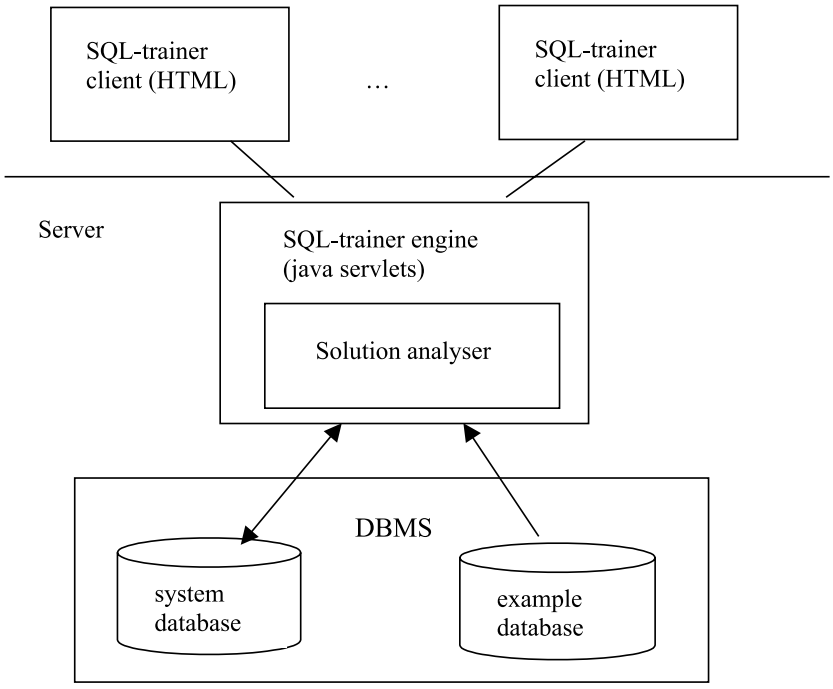

Figure 1: Components of SQL-trainer

The training tool is basically a user interface for working with the example databases. Its components are outlined in Fig. 1. Operations may be entered and the results of the operations are presented to the user. SQL-trainer uses a real database management system, currently Oracle, for compiling and executing the SQL-operations and for handling the system database. Using a real DBMS has both advantages and disadvantages as compared to self-made SQL-interpreters used in WinRDBI [1] and query analysers used in SQL-tutor [2, 3]

learning tools for SQL. A real DBMS supports multiple simultaneous users, checks the syntax of operations and executes them efficiently, which makes it possible to use the tool in large courses. However, all the error messages may not be as good as what can be provided by a self-made interpreters or analysers.

SQL-trainer stores all the solutions students provide and the results of their analysis in the system database. Statistics about the solutions may be produced on both task and student basis. SQL-trainer's statistic tools are currently implemented as SQL-scripts. The other tools are implemented as Java-servlets. Standard JDBC-API is used in communicating with the database management system. Thus, in theory, it would be possible to change the DBMS just by changing the configuration data. In practice, we have changed the DBMS once and it took us a couple of days to find out and fix the problems due to the differences in the behaviour of the JDBC API-functions.

The user interface of the SQL-trainer is implemented in standard HTML and may be run on any web browser. The user interface of the training tool contains the pages for

- *login* - Students make their own user identifiers and passwords
- *student status* - This page shows how many times the student has tried to solve each task, and which tasks he/she has solved correctly. The number of tries is unlimited. This page also acts as an index for the task pages.
- *previous solutions* - Students have access on their own solutions and may use them in building new solutions,
- *task* - This page shows the task and provides a field to write the solution. After posting the solution this page is shown again furnished with the result and the feedback on the operation. The task page provides also buttons for opening the database definition and the SQL-syntax assistant in separate windows.

The task manager makes it possible to define new tasks and to modify the existing ones. A task definition contains descriptors for the constraints introduced in Chapter 2. In addition there are descriptors needed for detecting the cause for erroneous results, for example, the number of rows in the result when the DISTINCT-clause is not included. Teachers may assign values for each descriptor separately. They may also specify an example solution that is used in determining many descriptors at a time, automatically. Typically a task is defined by using the example solution first and then modifying some of the descriptors to increase freedom in the solution.

SQL-trainer makes it possible to practice both queries and maintenance operations. Maintenance operations do not operate directly on the tables of the example database but on student specific copy rows that are made as the first step of executing a maintenance operation and deleted as the last step.

4 Answer analysis

The solutions are subjects to both pre- and post-execution analysis. The pre-execution analysis is currently very simple. Checks are made that the users are not trying to do any operations on the system database and that they are not trying to alter or re-define the example database. The pre-execution analysis may also modify the solution, for example, change the maintenance operations to work with the student specific copy rows. Students are able to produce very large results even with small example databases. Pre-execution analysis of missing join conditions in multi-table queries would be useful in preventing queries with very large results of being executed. This analysis is to be included in the next version of the trainer.

We use the DBMS to check the syntax of the operations and to execute them. Our current Oracle DBMS provides poor error messages in some common syntax error situations,

for example, if column names are misspelled. We are considering post execution analysis or changing the DBMS to get rid of this problem.

Post execution analysis takes place if the query is found to be syntactically correct and has been executed without errors. This analysis evaluates the semantic constraints placed on the result and on the solution. If the operation is a query, the result of this query is analysed. If the operation is a maintenance operation, the result to be analysed is produced by a query that finds out the state of the student specific copy of the target table after the maintenance.

5 Teaching arrangements and experiences

SQL-trainer has been used since autumn 1999 in 8 lecture courses and in two e-learning courses. The task collections for the courses have contained 30-40 tasks in 2-3 groups with different deadlines. An average course with 300 students produces about 50000 answers, i.e. about 5 tries per task for each student. Our task collections contain both easy and hard tasks. Students get credit according to the number of tasks they have solved successfully. About 80 percent of tasks are needed for the maximum credit and about half of the students in lecture courses achieve it.

We have had one teaching assistant available to assist the students in solving the tasks. She has been on duty a couple of hours twice a week in a computer class during the most active practice periods just before the deadlines. She has also provided assistance by e-mail. All the classroom exercises have been substituted by practice with SQL-trainer. After each deadline there has been a lecture where the tasks that have proven to be most difficult are explained. According to the course evaluations, students prefer this way of practice to the standard classroom exercises.

We have used the SQL-trainer now for two years in large database courses. It has been found to be an easy to use interface for entering database operations. The feedback based on the semantic constraints assists the students in finding the correct solutions. The error messages produced by the DBMS are not always informative. We have tried two data base management systems and both had problems in their reporting of syntax errors. The problems were not, however, the same. SQL-trainer seems to activate the students. The exam results are at least as good as with the classroom exercises. Students that have been able to solve many trainer tasks manage well in exam.

6 Further development

Our first version of SQL-trainer supported only queries. Maintenance operations were included in 2001. We have considered new task types for SQL-trainer, for example, relational algebra tasks and simple multi-choice tasks that would be useful in a database course. We have made a prototype where relational algebra operations are transformed into SQL queries and view definitions. This would make it possible to use the same analyser for relational algebra expressions as we use for the SQL-queries. Multi-choice queries however need a different kind of analyser and different kinds of task descriptors. Our next goal is to redesign the system database and the software architecture to such that we may easily add new task types just by registering new analysers and task presenter modules. We are also analysing the solutions stored in our database to find out typical cases requiring better feedback than what we are able to provide currently.

References

[1] Dietrich S. W., Eckert E. and Piscator K., A Windows-based Relational Database Educational Tool, *Proc of the 28th SIGCSE technical symposium on Computer science education*, 126–130 (1997).

[2] Mitrovic A.,Learning SQL with a Computerized Tutor, *Proc. of the 29th SIGCSE technical symposium on Computer science education*, 307–311 (1998).

[3] Mitrovic A, Hauser K., Porting SQL-Tutor on the Web, *Proc. ITS'2000 workshop on Adaptive and Intelligent Web-based Education Systems*, 37–44 (2000).