Automatic Checking of SQL: Computerised Grading

Dr Gordon Russell, g.russell@napier.ac.uk
Andrew Cumming, a.cumming@napier.ac.uk

Napier University, 10 Colinton Road, Edinburgh, EH10 5DT

Abstract

Internet-based learning environments have significant advantages for students. Students can manage their own time, and prioritise their studies. However, a common view of such learning environments is little more than a repository of documents, slides, and tutorial sheets. ActiveSQL takes this a stage further, offering interactive practical sessions for learning SQL. These sessions provide tutorial questions which the student answers by typing in SQL statements. These statements are executed, and the results of the SQL shown to the user. The system also evaluates the SQL, grading it as a percentage. This grading system is also used to automatically assess SQL coursework, which is itself integrated into the learning environment. This paper considers this automatic grading system as it has evolved over a number of years, and evaluates the impact it has had on student learning and behaviour.

Introduction

The growth of computer-based learning over the last decade has helped to support students who previously struggled to learn amid their workload and lifestyle commitments. It is also suggested that such systems are also 20%-30% more cost effective than traditional approaches [1] . It can be argued that online resources, such as electronic slides, books, forums, and email discussions, could replace the concept of lectures but the replacement of practical sessions poses significant challenges.

Practical sessions revolve around the students receiving timely and relevant feedback on their progress. Without this, students can become confused, misdirected, and disillusioned. A good example of this is shown when the authors teach SQL in a practical session. Students read the tutorials, and write some SQL statements to produce the data requested. Typically a student's first attempt at SQL will fail to parse. After several attempts the student arrives at a syntactically sound SQL statement that gives the wrong answer. Having received the only feedback available (that the SQL statement now runs) student confidence is high that the question is now completed and that they can proceed to the next question. However, the reality of this situation is that frequently their SQL statements, although syntactically correct, simply do not match the question asked. Immediate feedback is needed to inform the student that their answer is not complete, and needs more work.

Constructive feedback is important for students. The idea of TRUE/FALSE feedback does not encourage the student to continue to work towards a solution. What instead is needed is a grading of the solution thus far, and therefore an idea as to whether the student is heading in the correct direction or not with their current approach.

In ActiveSQL[2] , the authors have constructed a managed learning environment which provides online practical sessions to the students learning SQL. It also provides automatic grading of SQL queries. It also contains all the coursework assessment material, which is presented incrementally to students as they progress through the tutorial material. This assessment material is also automatically graded without staff intervention, giving the students instant feedback on their progression through the material and their current level of achievement. The system is designed to minimise plagiarism, while encouraging progression through the material and student-based time-management.

This paper considers the experiences gained in constructing the ActiveSQL site, focusing on the student experience. It considers the evolution of the system, and the statistical data gathered from students who have used the system over the last 3 years, extending significantly the interim results presented in [3] . It also offers advice that should be considered in future automatic grading systems.

SQL Practicals

SQL is a query language for extracting data from databases. It has a strict language specification. However, a statement designed to extract a particular dataset from a database can be written in a wide variety of different ways, all of which will be more or less as good as any other way. Consider this tutorial question:

> *List the surname, forenames, and department name of all employees who have a surname starting with the letter 'R'.*

```
SELECT surname,forenames,depname
FROM employee JOIN department
          on (employee.depno = department.dno)
WHERE surname LIKE 'R%'
```

Variations include replacing "employee.depno" with "depno", replacing "department.dno" with "dno", swapping around "surname" and "depname" on the SELECT line, and changing "FROM" to "From" (*much* of SQL is not case-sensitive). With so many variations it is often easier to execute the SQL and see what it does, rather than try to evaluate the SQL directly by hand.

The variations of correct SQL makes automatic grading of student SQL submissions difficult.

Automatic Grading

The ActiveSQL system is a collection of questions that are presented to the students one at a time. Each question is defined so that it is clear what *column names* are required. Each question also asks for some data to be extracted and processed, which requires the student to write a series of SQL statements.

ActiveSQL takes the students SQL statements and then executes them, showing the result of the execution to the student. Invisible to the student ActiveSQL also executes a sample solution to the question, and compares the result executing the sample solution against the result of executing the student's solution.

Consider the example question and query shown above. Supposing a student wrote the following as an answer:

```
SELECT surname,forenames,depname,dno
FROM employee JOIN department
                on (employee.depno = department.dno)
```

The student output would be compared to the sample solution output. Where the student's submission was not present in the sample solution a colour is used to indicate the erroneous data, and a lack of colour indicates correctness. The student might see:

surname	forenames	depname	dno
Russell	Gordon	Computing	1
Cumming	Andrew	Computing	1
Smith	Jim	Maths	2
Else	Someone	Somewhere	5

The student would receive an Accuracy measure, which is the proportion of correct cells (data not header cells) against the higher of either the total cell count of the sample solution of the total cell count of the student answer. In this case the accuracy is 3/(4*4) or 19%. The advantage of the percentage approach is that as more filtering is added to the student query, the number in general will rise. This gives a motivational signal to the students when they are heading in the "right direction".

Actually the sample solution and the student's solution is executed twice each. The first time they are executed using a known and visible dataset, and the second time on a modified *hidden* dataset. The results from execution on the second dataset is also compared and evaluated, and is used to catch out students who have hard-coded parts of the answer into their queries. For instance, a query to find out how many employees are over 50 could result in the number 10 being displayed. Rather than write a query to calculate this number, the student could work it out by hand and then write a query to simply print the number 10. This would be 100% accurate for the first dataset, but providing that there were more or less than 10 employees over 50 in the hidden dataset their query would then give a wrong answer. This difference in output is detected and the student penalised for failing the *hidden database check.*

In addition to the accuracy measure, and the hidden database check, some hand-coded rules are also used to scan the text of the student's SQL statement. These rules penalise the student for queries with a number of easy to detect weaknesses. One of the rules penalises students for having a query twice as long as the sample solution query, another penalises the use of LIKE when the operand does not contain wildcard characters. These rules partially are there to give a quality measure to the SQL statements, but also to give an impression of intelligence to the feedback. This feedback was based on common issues reported on feedback sheets when the marking was originally done by hand.

Experimentation

The ActiveSQL system was constructed with post-analysis of usage and performance in mind. It tracks a wide range of interactions, allowing the authors to investigate for example question difficulty, marks received, time to pass, total time per question, student attendance, and patterns of usage. This information is used at the end of each year to better understand student behaviour, and to suggest system modifications which can be made to improve some factor of the student experience. This change can then be evaluated in the following year.

ActiveSQL has been running for three years. During this time it has looked after 16 student cohorts of size ranging from 10 to over 300. In total it has over 1100 real students in its records. This has given the authors a sizeable population to extract statistically significant information.

Modifications

The key behavioural changes which are considered in this paper are as follows:

- Improving coursework performance
- Increasing the number of tutorial questions attempted by students
- Increasing the number of tutorial questions which when attempted were answered correctly
- Increasing the number of assessment questions attempted
- Promoting good time management skills in the students

Table 1: Cohort tutorial performance

Cohort	Questions attempted	Questions Completed	Questions Incorrect	Questions not Attempted
2002	29.38%	27.75%	8.37%	72.25%
2003	63.68%	59.51%	4.16%	36.32%
2004	78.06%	74.28%	3.78%	21.94%

Consider the results shown in Table 1 of three comparable cohorts over the period 2002-2004. In 2002, this cohort did not have the benefit of the automatic grader. In other respects the site operated as normal. Students did not know when they were right in their answers, and often thought they were right when they were wrong. They did not see any worth in completing more than 30% of the tutorials. Student satisfaction feedback was low.

The automatic grader was introduced in 2003. At the same time an incremental assessment strategy was introduced. This split the tutorials into 4 groups, and split the assessment into 4 assessments, one related to each tutorial group. Only when a student completed 75% of a tutorial group was the related assessment made available to that student. This change was only possible with the automatic grader. Now the scheduling of the assessments was at the control of the student, with the only limiting factor being that the lecturer set a final deadline by which time all assessment work must end. This forced increase coverage of the tutorial material. Even so significantly less questions were left incomplete. Student satisfaction feedback was significantly better, but a new issue was raised.

Previous to 2003 the assessment was 15 questions, arranged in difficulty groups, from which the students selected any 5. Many students did the minimum, and achieved minimum passing marks as a result. With the 4 assessments of 2003 students were encouraged to do harder questions, and the idea of just completing the first 2 assessments to reach a minimum passing grade was an approach which the students now saw as *failing*. The assessment statistics were still acceptable in this new model, but feedback suggested that by not making completing only half the assessments appear "acceptable" to the average student, the perception was now that the material was too hard. Students also did not want to progress into completing a tutorial group if their perception was that the related assessment would be undoable.

In 2004 the assessments were modified so that an assessment, rather than containing 2 questions with a difficulty level related to the tutorial just completed, it contained 1 question of that difficulty level and 1 question of the previous difficulty level. It was thought that this would mean students would be tempted to complete a tutorial group to reach an assessment if they new that at least one of the questions would be doable with the knowledge they gained from the previous tutorial group. As you can see in the statistics, more questions were indeed attempted, and more questions were done correctly. In addition student satisfaction feedback no longer complained of the difficulty level.

Performance

In this section the impact of the modification made through the years is considered against student performance. Students can attempt up to 8 questions as part of their coursework assessment. In 2002, with the best 5 questions from 15 approach using a single assessment, the results remarked using the automatic systems and normalised to match the approach used in subsequent years. The assessment performance statistics can be seen in Table 2.

Table 2: Assessment Performance

	Cohort		
	2002	**2003**	**2004**
Total Number of Students considered	312	301	263
Average Score	(60%)	51%	64%
Assessment questions attempted	(5.7)	4.6	5.8

From the results it is clear that the change to incremental assessments had a significant impact on assessment performance. Although it did encourage more tutorial questions to be completed, the perceived difficulty level increase caused a knock-on effect in the marks and the questions attempted. In 2004, with the rebalancing of the difficulty levels, marks are up and questions attempted have increased too.

It could be argued that, by giving an additional simple question to the students, the authors are *gifting* 12.5% to the students. This seems confirmed by the 13% swing in the marks. However, this is really only partially true. This would assume that all students get full marks for easy questions (false), all students attempt at least 1 assessment question (sadly false too), and that the standard deviation remains the same (it grows from 2003 to 2004). Even then, it must be said that giving away a few marks seems a good trade-off if the result is a significant improvement in the total material covered by the students, and an increase in student feedback. Lastly, these coursework marks make up only 40% of the overall module mark, with 60% coming from a formal exam.

Side Effects

One area of interest to the authors is the time management skills of the students. Time management is not something which is formally part of the database module, but if the design of the environment can encourage a good approach to time planning of tutorial and assessment time then there are many advantages to be gained. In particular, if students can be encouraged to progress through the material at a reasonable and consistent rate, perceived workload in the module is reduced (in comparison to trying to do it all at the last minute) and deep learning is encouraged.

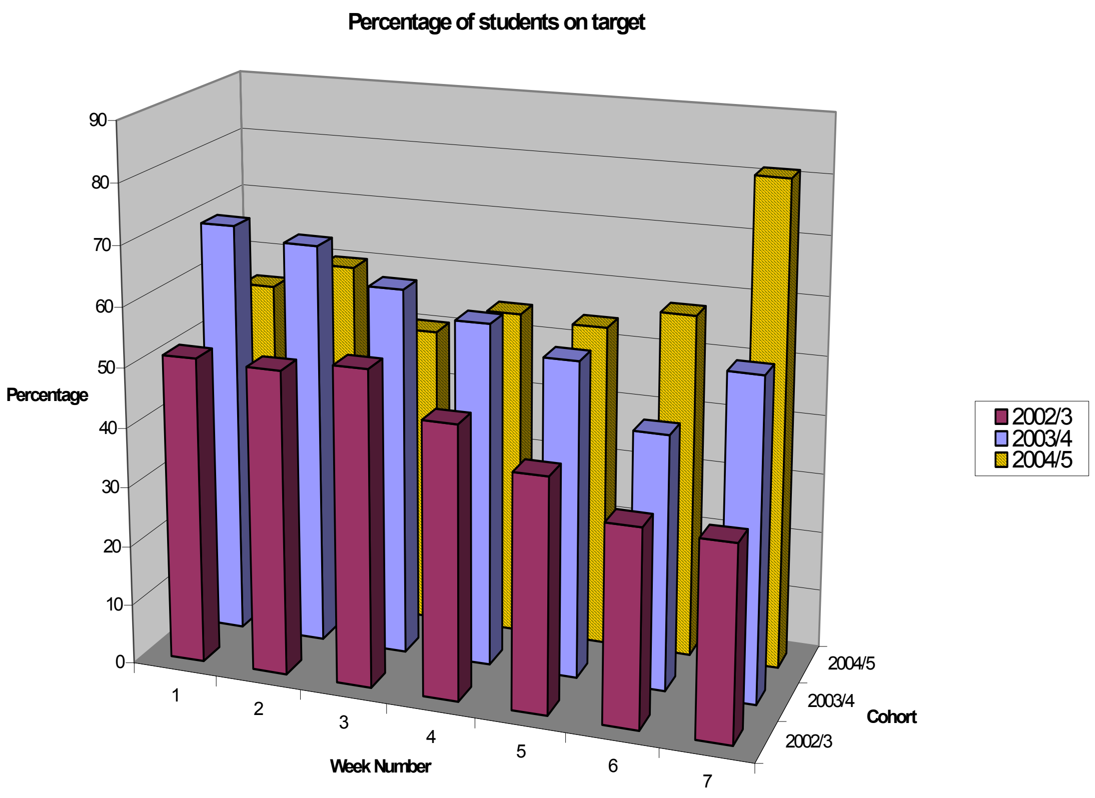

Figure 1: Students who match or exceed the tutorial timeplan.

Figure 1: Students who match or exceed the tutorial timeplan.Figure 1 shows the results of an analysis into student tutorial progression against week numbers. The analysis compares as a percentage how many students have completed enough

questions to meet the target number of questions for that week. The target number of questions in the case is based on a timeplan given to the students, which when followed should give them a good chance of achieving the normal average mark for this module (55%-60%).

From the graph it can be seen that in 2002 and 2003 the tendency of students was to start off with good intentions, but later in the module to slow down significantly. In 2004 there is a significant change to this pattern, as here the students on target stay relatively constant throughout the study period and then near the end more students actually put more work in to achieve the recommended targets. This seems to confirm that the approach of making tutorial groups more attractive by making the associated assessments more attractive has had resulted in significant improvements to the study practices of the average student.

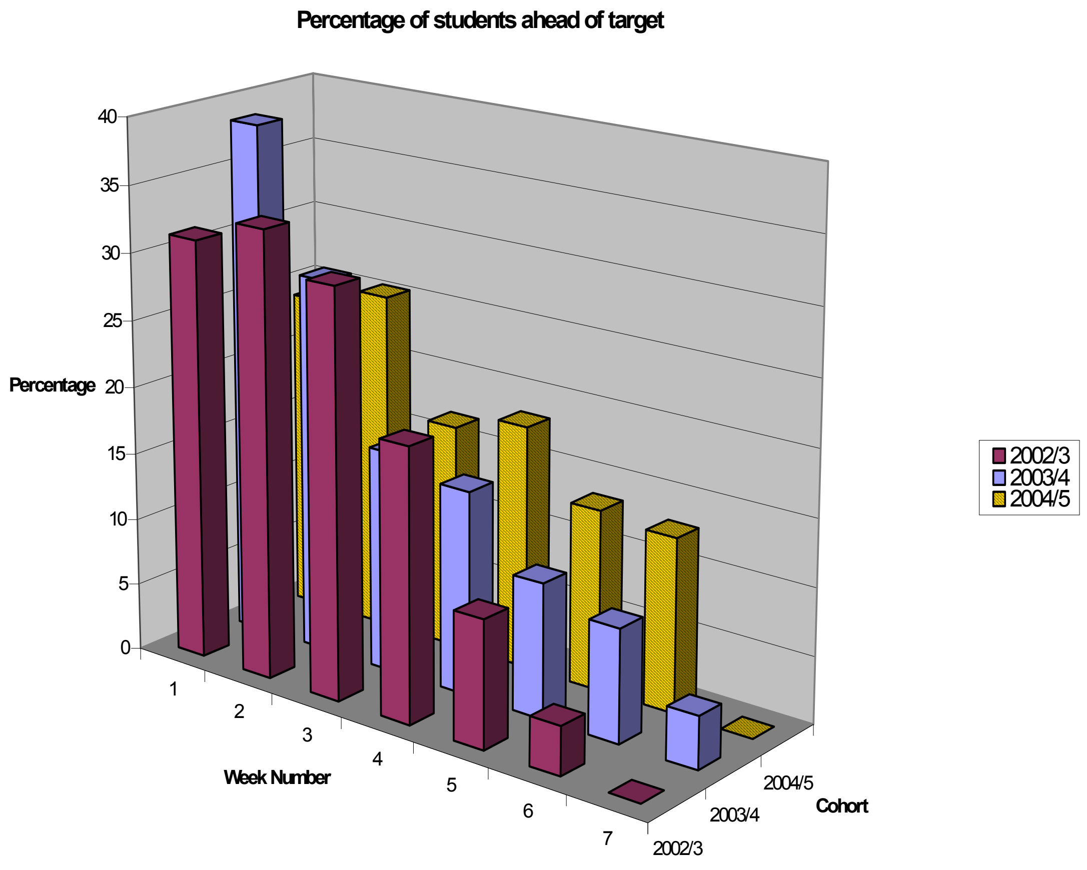

Figure 2: Students who match or exceed the timeplan for top quartile students.

Figure 1 concentrated on the average student reaching the average mark. However, some students wish to obtain high marks in this module. An analysis of the students who followed a time plan that would allow them to reach a top 90% class position was also carried out. The results of this are shown in Figure 2. Here the results are less attractive.

In 2002 many students tended to work hard right from the start of the module, slowing down only after about a month of time (presumably due to increased difficulty level holding them back). In 2003 students started out well, but immediately fell behind the targets reached by the 2002 students. It is thought that this is largely due to a slightly increased workload during that time (they had to do some of the assessments earlier than in 2002, when the assessment was done in the last week). It was also thought that by encouraging time management suitable for average students, that this had interfered with the natural time management approaches used by better students. Lastly, with the perceived difficulty level of the 2002 assessments, progression onto tutorials 3 and 4 was depressed.

Finally in 2004, a small but significant improvement in progress can be seen for those aiming to be top students. Indeed, from week 3 the number of students following a plan that should result in top marks stays relatively constant. These results would seem to confirm that the 2004 approach has indirectly encouraged good time management skills amongst the average and good students alike.

Conclusions

A number of key lessons can be drawn from this study that may prove invaluable for future managed learning environments:

- The motivation of students to progress through the material can be easily damaged if they feel there is no advantage to the marks they will receive at the end of the module.
- Incremental assessments provide feedback that is well liked by students, and such assessments are excellent motivational tools.
- Changing from one assessment to 4 incremental assessments needs careful thought. Students view a single assessment with 8 questions, and 4 incremental assessments of 2 questions each, entirely differently. Doing half of either may seem like 50%. However, passing 4 questions from 8 is seen as acceptable by students, and only completing 2 assessments from 4 seems like failure.
- Automatic grading of tutorial questions results in significant improvements to tutorial progression, and should in turn significantly improve subject understanding.

The authors are currently investigating the impact of showing the time that a student has actually spent on a tutorial or a question has on their overall performance. They are also analysing the results of using automatic grading in teaching programming and in teaching Linux system administration.

References

[1] Committee of the Principals of Scottish Universities (1992). *Teaching and Learning in an Expanding Higher Education System*, SCFC, Edinburgh.

[2] ActiveSQL: http://db.grussell.org .

[3] Russell, Gordon and Cumming, Andrew (2004). Improving the Student Learning Experience for SQL using Automatic Marking. In Demetrios Kinshuk and Pedro Isaias (Eds.), Cognition and Expolaratory Learning in Digital Age (CELDA 2004) pp 281-288. Lisbon: IADIS Press. ISBN 972-98947-7-9.