A Novel System for Automatic, Configurable and Partial Assessment of Student SQL Queries

Mario Fabijanić*, Goran Đambić* and Bojan Fulanović*
* Algebra University College/Software Engineering, Zagreb, Croatia
mario.fabijanic@algebra.hr, goran.dambic@algebra.hr, bojan.fulanovic@algebra.hr

Abstract - At some universities, students are required to write SQL queries to pass database-related courses. When it comes to assessing students’ queries, three options are available: manual assessment (all work is done by teachers), semi-automated assessment (computer system provides parts of the final score, teachers give final mark), and fully-automated assessment (all work is done by a computer system). There are many currently available computer systems that can be used for both semi-automated and fully-automated assessments. While evaluating them, it became clear that their focus was not primarily on providing partial marks for substandard students’ solutions and also not primarily on teachers’ influence on the assessment process. Since these two requirements are very important for some universities, this paper proposes a novel fully-automated assessment system that is focused on satisfying these requirements. The early version of the system is built and evaluated against manual assessments of student work, resulting in the conclusion that the system is indeed very promising. Furthermore, the system discovered some inconsistencies in marks awarded manually by teachers, giving another perspective on the usefulness of the proposed system.

Keywords – SQL; assessment; fully-automated; partial marks; configurable assessment process

I. INTRODUCTION

These years, there are more and more expectations for database technologies in general. For instance, the appearance of social network systems brought new demands for databases and systems that we knew earlier, which lead to the emergence of new database systems, like Not Only SQL Databases (NoSQL) [1]. For a while, traditional systems like RDBMS seemed to be losing their great market share, but the majority of business applications still use them, and RDBMS vendors are simultaneously finding a way to adapt for more and more data generated these years. An example is the availability of big data in educational contexts, which changes the way how the educational process can be monitored and evaluated [2]. Similar situations happened also earlier, in the 1990s when Object-Oriented Database Systems appeared with an idea to be a substitute for Relational Database Systems, but Object-Oriented databases still haven’t found widespread acceptance like relational databases [3]. It is obvious that Relational Databases as technology survived challenges from the past, and not just that, Relational Databases can even perform as good as other proposed systems, if used properly [4]. When using up to the medium amounts of data that could be organized in row and column format, relational databases perform better [5]. So, each system could be the solution for particular use cases [6]. Business applications built for business operations will often be the best served by relational databases and may even be served exclusively by relational databases [6]. Also, even though the relational database idea is many decades old, innovative methods of teaching relational database normalization are invented these years [7].

Today’s workplace is certainly dominated by enterprise-level DBMSs from vendors such as Oracle, IBM, and Microsoft. Having skill and experience with one of these enterprise-class DBMSs can greatly enhance a student’s chances of a successful entry into the world of IT careers [8].

At the Algebra University College, database-related courses are called Introduction to Databases (obligatory for Software Engineering students and System Engineering students) and Database Development (obligatory only for Software Engineering students). The objectives of Introduction to Databases course are to learn how to design simple relational databases and to learn how to use SQL language to retrieve, insert, update, and delete data from a relational database. It means that students learn about relational databases from scratch, which turned out that it is very important since a lot of students are not even familiar with a need to organize data in a way to, for instance, avoid data inconsistency and redundancy. The objectives of Database Development course are to broaden knowledge of relational databases and SQL language and to learn how to prepare a database for access from programming code. More precise learning objectives are divided onto learning outcomes defined for both courses so that students are aware of what exactly they need to know to accomplish a positive evaluation of their knowledge per each learning outcome. So, knowledge evaluation tests are organized in a manner that students must provide their own SQL code for each test assignment. No single or multi-choice questions are used (or any other type of questions that would leave a possibility that students don’t actually know how to write appropriate SQL code) because the idea is that students should build their solutions, test and debug them on sample database, which is the way of work that waits for them in the industry. Being said, it is the best possible way of knowledge evaluation.
Despite the quality of that kind of knowledge evaluation, there are also some challenges. First of all, around 240 students are currently enrolled in the course Introduction to Databases. Since teaching is held by two teachers, who also mark all exams, students are inevitably waiting for their results for a couple of days. After the process of evaluating, insights into the exams are held for anyone who wants to know where he or she made a mistake. From the experience, around 15% of students use this opportunity, which also prolongs the time when students’ grades are finalized. All of that causes dissatisfaction for students and also their teachers, which are under time pressure, and more importantly, they become prone to make mistakes during assessments.

Instead, there are many papers that explore the use of computers for semi-automated or fully-automated assessment [10][11][12][13][14][15][16][17][18]. The semi-automated assessment provides parts of the final score, and teachers do the final grading. In the fully automated assessment, teachers do not have to grade each student’s solution individually, but they need to prepare for setting up the grading process [9]. Also, semi-automated assessment gives students grades with immediate feedback about imperfections, mistakes, and errors in their solutions, to help them understand both the mistakes and the given grade, and in that way they support the learning process [9].

Some of the existing solutions are:

SQL Tester [10] is a tool that gives full points to the student only if the student solution exactly matches the reference answer. That’s not optimal because a student doesn’t get any comments regarding his or her mistakes made in the solution. Instead, a student gets only an error message from the RDBMS.

SQLFE System [11] is an automatic grading system and system which implement peer review. The main reason for including peer review is to offer the students richer learning experience, but that makes it more complex since it employs more reviewers.

ASQLAG System [12] consists of two main parts: assignment management and automated SQL grader. The automated SQL grader is designed to evaluate only the results of the SQL statement.

SQLg [13], or SQL-Statement Grader, has a high-quality evaluation process that gives student feedback concerning syntax, helping them to identify their mistakes, but the tool only focuses on helping student practice the SQL statements, it cannot be used as for exam assessment.

SQL-KnoT [14], or Knowledge Tester, is a tool that also provides feedback to the student, but it compares the solution of the student with the correct result stored earlier. The student solution must be the same as the correct solution stored earlier.

SQLify [15] is evaluating the queries by the determining value of Conjunctive queries. Different reviewers make possible solutions, possible with different criteria. No feedback is provided.

ActiveSQL [16] supports online assessment and gives students immediate feedback. It also gives student grade defined as a percentage, comparing their results with the correct results, as the proportion of correct cells against the higher of either the total cell count of the correct solution of the total cell count of the student answer. It does not identify the student’s understanding level and feedback given to the student is poor. There is also a lack of additional grading criteria.

SQLator [17] is a web-based tool that provides immediate binary feedback to the learners, so there is no useful feedback to students. SQLator does not check or analyze the SQL syntax at all, it provides the estimation: either a correct or incorrect statement.

AsseSQL [18] provides immediate feedback for the student solutions in a way to guide them to what corrections they need to apply compared to the results of the correct solution. If the submitted answer is syntactically incorrect, an SQL error message is displayed, without further explanation.

To summarize, none of those tools implement all of the following criteria for the student solutions:

- Is there any syntax error in the student solution?
- Is there any prohibited elements in the student solution?
- A tool should match student solution with the correct solution.
- A tool should give the result of the evaluation.

Also, it seems that all those solutions don’t give the teachers the needed flexibility to define his or her own criteria for evaluating students’ solutions.

Therefore, the motivation for this paper is to define a model and propose a tool for teachers to give them the ability to define exam question, to define one of the possible correct solutions, and to define the criteria for evaluation, for instance:

- Does the solution compile? If not, the percentage of point reduction might be 100%
- Is the FROM clause valid? If not, the percentage of point reduction might be 100%
- Is the WHERE clause valid?
  - If not, the percentage of point reduction is 50%
  - If yes, but with wrong columns included, the percentage of point reduction is 25% per the wrong column
  - If yes, with correct columns, but wrong values included, the percentage of point reduction is 25% per wrong value
- Is the SELECT clause valid?
  - If not, the percentage of point reduction is 10% per column that is missing

Also, the implementation of the model should be web-based and easy to use for students and teachers. It should provide feedback for students in the form of PDF report,
generated immediately after students upload their solutions, being able to download and analyze it. The report should use the above evaluation criteria, pointing at the parts where mistakes are being made, that way helping students to understand given points.

It needs to be said that the proposed model, in its initial version will be able to evaluate fundamental clauses of the SELECT statement (SELECT, FROM, WHERE, ORDER BY), fundamental clauses of the INSERT statement (VALUES), and fundamental clauses of UPDATE and DELETE statements. In future versions, additional clauses of SELECT statements will be added (GROUP BY, HAVING, all types of JOIN), but also SUBQUERIES as a querying method will be supported, additional INSERT possibilities will be possible to evaluate (like the combination of INSERT and SELECT statements), the composition of statements like when SELECT, INSERT, UPDATE and/or DELETE will also be possible to evaluate.

II. METHODOLOGY

The main focus when proposing our model for automatic assessment of SQL statements was the need of students and teachers; students’ goals are to receive results as soon as possible, together with correct and helpful feedback about mistakes they made. Teachers’ goals are to minimize efforts in marking student solutions and providing detailed feedback and also to maintain a consistent marking process for all students.

In the proposed model, when it comes to defining an exam, the teacher is required to define three inputs for every question they want to define: free form text of the question, one possible correct SQL answer (called a model) and which marking rules they want to apply to the process of marking student’s solution (called a student answer, or answer for short). The system is designed to be extensible: currently, there are 15+ defined and implemented marking rules, but it is very easy to define additional. The main idea is that authors plan to receive teachers’ feedback on their needs and then implement additional marking rules so each marking style can be accommodated in this system.

As the main forte of this proposed system is to allow awarding partial marks to student solution, we believe this approach of fine-grained marking rules will allow exactly that. Each marking rule consists of a condition, percentage of points to add or subtract if the condition is (not) met and whether marking can continue if the condition is not met or not. Fig. 1 shows marking rules for marking SELECT statement and a similar approach is taken for marking INSERT, UPDATE, and DELETE statements.

The proposed system comes with a few templates for marking each statement. The teacher can start with a template that is the closest match for his marking style, adapt it to fit her style perfectly, and then save it as a template for future use. Fig. 1 shows a template that is used at our home university for marking a basic SELECT statement. The first condition is that an RDBMS engine can compile the student’s query, which is checked by sending the query to the engine and asking it to be compiled. If it does not compile, student feedback contains actual RDBMS error message, all points are subtracted and the process of marking is finished. FROM correctness is check next and if the student used the wrong data source, the process is also finished. Then, validation rules are checked against the main parts of SELECT statements. For each missing column in the SELECT list, 10% of points are subtracted. Optionally, the teacher can choose to subtract a certain percentage of points if the SELECT list is not exactly correct, without checking each individual column. Currently, there are two marking conditions that mark the WHERE part. The first one is used to check for a correct result of the WHERE filter, and the second one is used to check if correct columns were used for filtering. It is possible that the result of WHERE is wrong, but that the student used correct columns just with wrong operators, function, or literals. In this case, partial points are awarded. ORDER BY part is marked in a similar way: if the rows are not ordered as expected, a check is made to see if they are correctly ordered based on the first specified column in ORDER BY so partial points can be awarded.

To check marking rules, two protocols were defined, dynamic, and static. In the first, dynamic protocol, the marking rule check is done by sending the original or modified query to the RDBMS engine and deducing the marking result from RDBMS response. The second protocol is based on insight gotten from analyzing student query. The query is tokenized, converted into canonical form, semantics is given to sequences of tokens and then used to decide on the marking rule results. For example, if a model answer to a question is this:

```
SELECT Naziv, Boja
FROM Proizvod
WHERE Boja = ‘Crvena’
```

And student answer is this:

```
FROM dbo.Proizvod AS p
WHERE p.Boja = ‘Crveni’
```

Marking rules 4 and 5 from Fig. 1 will be assessed by combining dynamic and static approaches. First, student answer will be transformed into this:

```
SELECT ProizvodId
FROM Proizvod
WHERE p.Boja = ‘Crveni’
ORDER BY ProizvodId
```
After canonicalization and transformation, model and student queries are sent to the RDBMS engine. Returned result sets are checked for the exact match – if any value in any row is different in these two result sets, marking rule 4 is considered as failed and points are subtracted. Of course, by looking at model and student queries it is obvious that their results will not be equal, so marking rule 5 is the teacher’s attempt to give partial points. This is done by static analysis in which a set of model filters is found to be \{ Column: “Boja”, Operator: Equals, Value: “Crvena” \} and a set of answer filters is \{ Column: “Boja”, Operator: Equals, Value: “Crveni” \}. The second protocol observes that both Column and Operators are the same and that Levenshtein’s distance \[19\] between two Values is 1, so marking rule 5 is considered as passed and the student’s answer is awarded partial points.

Some additional characteristics if the proposed model will be presented next. In order for this model to work, a previously prepared database should be made available. It should be the same database student used at the exam when writing her queries. It is also worth noting that marking SELECT statements, the proposed system will deduce the correct columns, operators, and values based on the model query, leaving the teacher free of any additional configurations for each question. When it comes to marking INSERT, UPDATE, and DELETE statements, the system uses transactions at serializable isolation level to make sure that the only state changes made to the database are the ones made by student queries. At the end of each question marking process, the transaction is rolled back, leaving the database at the original state and ready for the next marking. Certain questions are asked by the proposed system to be able to understand the context of both the model and answer queries, two examples being: what is the primary key of the FROM table and what is the complete set of columns available for the FROM table. Although the proposed model was developed for SQL Server, these questions and all other where possible were asked in the form of queries based on INFORMATION_SCHEMA schema, making it possible to work the same in most currently available RDBMSes on the market.

### III. RESULTS

System for automatic assessment of SQL statements contains import feature which uses predefined well-formatted CSV file. The teacher fulfills CSV file with student solutions and imports them into the system. The teacher then needs to define three inputs for every question: free form text of the question, one possible correct SQL answer (called a model), and which marking rules they want to apply to the process of marking student’s solution. After that, the system process the data and generate the report, also in the CSV file.

To validate the capabilities of the proposed model for automatic assessment of SQL statements, we used 135 grades given by teachers that manually evaluated those SELECT statement solutions at some earlier exam period and compared them with the grades that were given by the system for automatic assessment of SQL statements. Analyzed SQL statement assessments are restricted to simple SELECT statements, without JOINs, GROUP BY and HAVING clauses, aggregate functions, and subqueries. The Maximum for each grade is 1 point, and Fig. 2 shows that total points (for 135 evaluated assignments) given by the proposed system are close to total points given by teachers.
Also if we analyze an average of given points, we can see that are similar values of average points given by teachers and average points given by the proposed system (Fig. 3), which is an early indication that the proposed system could be appropriate for its purpose.

When analyzing the number of assignments and their distribution by given points (Fig. 4), we can see that number of assignments evaluated with a minimum (0) and maximum (1) points is bigger when the teacher evaluated assignments. When the proposed system evaluated assignments, more assignments are evaluated with partial points. It means mean that teachers more often evaluate on a binary basis, and the proposed system more often evaluate with partial points, which can be explained as more precise, therefore fairer.

Further on, we conducted statistical hypothesis testing using the Chi-square test (χ² test) based on a sample of 135 grades. Our null hypothesis is that there is no significant difference between the grades given by the system and grades given by teachers. Table 1 shows observed frequencies of grades, given by the teacher and proposed system, and Table 2 shows expected frequencies of grades, needed to calculate degrees of freedom and χ² value.

The number of degrees of freedom is:
\[
(Number\ of\ rows\ -\ 1)\ \times\ (Number\ of\ columns\ -\ 1) = (2 - 1) \times (5 - 1) = 1 \times 4 = 4
\]

\[
\chi^2 = \sum_{n=0}^{f} \frac{(f_0 - f_t)^2}{f_t} = 1.210
\]

For a significance level of 5% and 4 degrees of freedom (1), critical value χ² is 9.488. Compared with the contingency table, χ² value is 1.210 (2), so we concluded there is no statistical significance between grades given by the system compared with grades given by teachers, and our null hypothesis is not rejected.

### Table 1. Observed Frequencies of Grades

<table>
<thead>
<tr>
<th>Grades</th>
<th>0</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proportion of grades assigned by teachers</td>
<td>8</td>
<td>3</td>
<td>19</td>
<td>36</td>
<td>69</td>
<td>135</td>
</tr>
<tr>
<td>Proportion of grades assigned by system</td>
<td>6</td>
<td>5</td>
<td>20</td>
<td>40</td>
<td>64</td>
<td>135</td>
</tr>
<tr>
<td>Total</td>
<td>14</td>
<td>8</td>
<td>39</td>
<td>76</td>
<td>133</td>
<td>270</td>
</tr>
</tbody>
</table>

### Table 2. Expected Frequencies of Grades

<table>
<thead>
<tr>
<th>Grades</th>
<th>0</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proportion of grades assigned by teachers</td>
<td>7.0</td>
<td>4.0</td>
<td>19.5</td>
<td>38.0</td>
<td>66.5</td>
<td>135</td>
</tr>
<tr>
<td>Proportion of grades assigned by system</td>
<td>7.0</td>
<td>4.0</td>
<td>19.5</td>
<td>38.0</td>
<td>66.5</td>
<td>135</td>
</tr>
<tr>
<td>Total</td>
<td>14</td>
<td>8</td>
<td>39</td>
<td>76</td>
<td>133</td>
<td>270</td>
</tr>
</tbody>
</table>

### IV. Conclusion

This paper proposed a novel model for automatic assessment of SQL statements which should help students and teachers; students will receive their grades quicker, but they will also get helpful feedback about mistakes they made, which is important for their learning process. As opposed to most existing solutions, this model is strongly focused on the partial marking of student solutions and giving teachers a tool to configure the exact marking procedures.

The system was configured by a teacher who also conducts manual marking of students’ solutions and the system was used to mark 135 student solutions. Conducted statistical hypothesis testing using Chi-square showed that there is no statistical significance between grades given by the system compared with grades given by teachers, so the possibility of using this system in the university environment looks very promising. To do so, current limitations of the system need to be addressed: the system is currently restricted to simple SELECT statements, without JOINs, GROUP BY and HAVING clauses, aggregate functions, and subqueries.

Additionally, the system discovered cases with minimal, but existing nonconsistency mistakes made by teachers during manual grading resulting in different
numbers on awarded points for the same student solutions. This situation does not happen with our proposed system, resulting in additional positive effects on the grading process. From the work done in this paper, it would seem that this proposed system might be a good starting point in building a more comprehensive solution, with additional features like this (but not limited to): implementation remaining clauses in the SELECT statement, INSERT, UPDATE and DELETE statements, complex tasks that should be resolved with the combination of statements, etc.

REFERENCES


