Datenbank Spektrum
https://doi.org/10.1007/s13222-021-00372-0

SCHWERPUNKTBEITRAG

# SQLValidator – An Online Student Playground to Learn SQL

**Victor Obionwu[1] · David Broneske[1] · Anja Hawlitschek[1] · Veit Köppen[1] · Gunter Saake[1]**

Received: 1 October 2020 / Accepted: 11 February 2021


**Abstract**
The Structured Query Language (SQL) is the most widely-used language in database-related courses. As a consequence, writing SQL queries is a fundamental expectation from any university course in database systems. Practical exercises are an essential part of the SQL learning experience. These exercises enable participants to practice and acquire experience in the use of the different SQL concepts, such as clauses, predicates, and expressions. To this end, we developed the tool *SQLValidator* as a web-based interactive tool for learning and practicing SQL. Apart from using it for teaching, we also use it to administer questionnaires and practice tests to improve students' learning experience. In this paper, we present the architecture and functions of SQLValidator. In order to assess the usefulness of SQLValidator, we monitor the performance of our students based on the semester activities and examinations. Our evaluation shows that SQLValidator is an effective tool that improves a student's learning experience when learning SQL.

**Keywords** SQL · Web-based learning · Databases · Didactic technology · Tools · Learning Environment

## 1 Introduction

In the world today, most electronic interactions are supported by databases. A database is an organized collection of data that consists of tables, queries, views, reports, and other objects. To be able to access and manage a relational database, we mostly use the *Structured Query Language (SQL)* which is the de-facto standard for accessing and manipulating data. As such, SQL querying skills are in high demand in the computing industry. This makes the teaching of SQL in tertiary institutions very important.

Introducing students to a new programming language comes with several challenges for both instructors and students. Students are known to experience several challenges such as understanding the fundamental theory that drives SQL [5] or grasping the effect of a seemingly straightforward syntax [11]. For instructors, the challenge comes in form of having to correct a huge amount of handwritten SQL assignments, assist students who are unfamiliar with software installation processes, provisioning of meaningful feedback for the errors that occur during exercises and software compatibility challenges that occur as a result of the different operating system environments.

To solve these SQL-specific educational challenges, we developed the SQLValidator, a web-based interactive tool for learning and practicing SQL. In the SQLValidator environment, students form and test their queries against a database and receive immediate feedback. As we show in our evaluation, the effort is rewarding for the students who fully participated in the course. It also accommodates both English-speaking and German-speaking students, since we have an international community and provide both languages at our courses.

The paper is organized as follows. In the next section, we discuss some related approaches that assist in SQL learning. In Sect. 3, we present various use cases of the SQLValidator. In Sect. 4, we present the architecture of SQLValidator and briefly survey its components. In Sect. 5, we look at the error classes that a student may encounter while using the SQLValidator. The educational outcomes as a result of using the SQLValidator are discussed in Sect. 6. Finally, in Sect. 7, we give directions for future research and our plans for the SQLValidator.

✉ Victor Obionwu
chukwuka.obionwu@ovgu.de

[1] Otto von Guericke University, Magdeburg, Germany

Published online: 02 March 2021

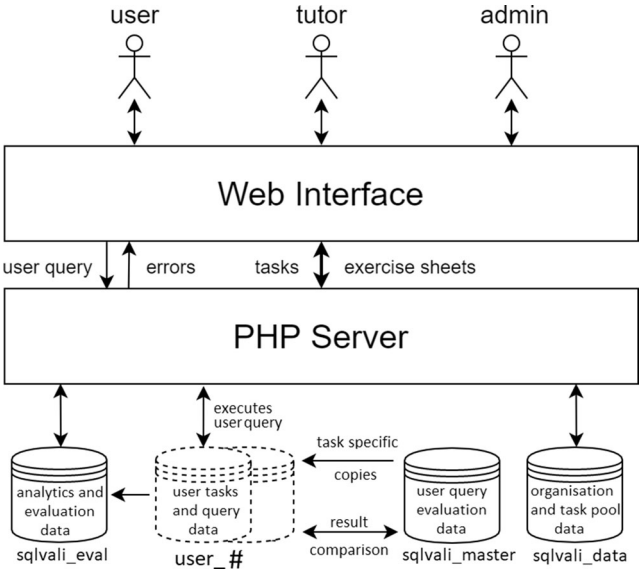


**Fig. 1** Architecture of SQLValidator

## 2 Related Work

In the past, there have been several systems developed to enhance SQL learning. Previous systems either focused on query execution and provided little feedback [3] or focused on SQL queries that mainly train the data manipulation language (DML). This is because compared to the data definition language (DDL), DML is more frequently used. In the subsequent paragraphs, we discuss the essential features of some of these systems that are similar to the SQLValidator.

SQL Tester [6] is an online assessment tool based on a 50-minute test that comprises ten questions. These questions are based on a table with different categories of SQL concepts. There is no limit to the number of trials that a student can attempt the test. It also saves the last session for the user. The system also allows students to group questions based on levels of difficulty. SQLValidator, unlike the SQLTester, has an assessment feature in the form of self-checks but is mainly used to administer course exercises. SQLTester only returns error messages and no feedback during practice tests.

SQLator [10] is an online learning workbench. SQLator's engine evaluates whether a proposed solution in SQL is correct. It also has several databases and allows learners to choose which of the databases they want to study. The query requirements for these databases are also described and classified based on their complexity. SQLValidator unlike SQLator does not allow students to choose which database to work with. While the query requirements for each database are not defined, the language feature required to solve each exercise is taught in the lectures prior to the exercise.

SQL Tutor [9] is an intelligent teaching system employed in teaching SQL to university students. The system is customized to the specific needs of each student. It employs constraints in the feedback process to deliver the semantic and syntactic descriptions of the errors. The learning experience can further be customized based on the granularity of the feedback messages and the type of question posed to a student, i.e., the student is given questions relative to the type of errors encountered in their last session. SQLValidator does not have this level of customization, which opens up a future work for us. While both systems allow students to practice outside lecture periods, the SQL Tutor unlike the SQLValidator allows students to practice SQL independent of the class exercises.

Learn-SQL [1, 8], is a software tool for E-assessment of relational database skills. It evaluates a student's progress among other metrics. The Learn-SQL (Learning Environment for Automatic Rating Notions of SQL) system is composed of three tools: an authoring tool which allows the teacher to manage the repository of questions or exercises, the remote questionnaire tool that allows students to access remote questionnaires, to answer the questions that a teacher has assigned them and a grader tool, which evaluates the student's solutions. This architecture is different from that of SQLValidator. Furthermore, SQLValidator unlike SQLTester has an assessment feature in the form of self-checks but is mainly used to administer course exercises.

In the next section, we discuss various ways users can interact with SQLValidator.

## 3 Use Cases of SQLValidator

SQLValidator is a web-based interactive tool for learning and practicing SQL. In the SQLValidator environment, students can, among other activities, form and test their queries against a database and receive immediate feedback. The project was mainly initiated to:

- provide an environment where students can easily practice the SQL features they learned in class and
- reduce the administration workload on instructors during the exercises as a result of the high number of enrolled students in our respective course.

Like most web-based systems, students require credentials for access. Also, each student is assigned to a specific exercise group and has access to their task profile and activity statistics. Current use cases for SQLValidator fall into the following categories:

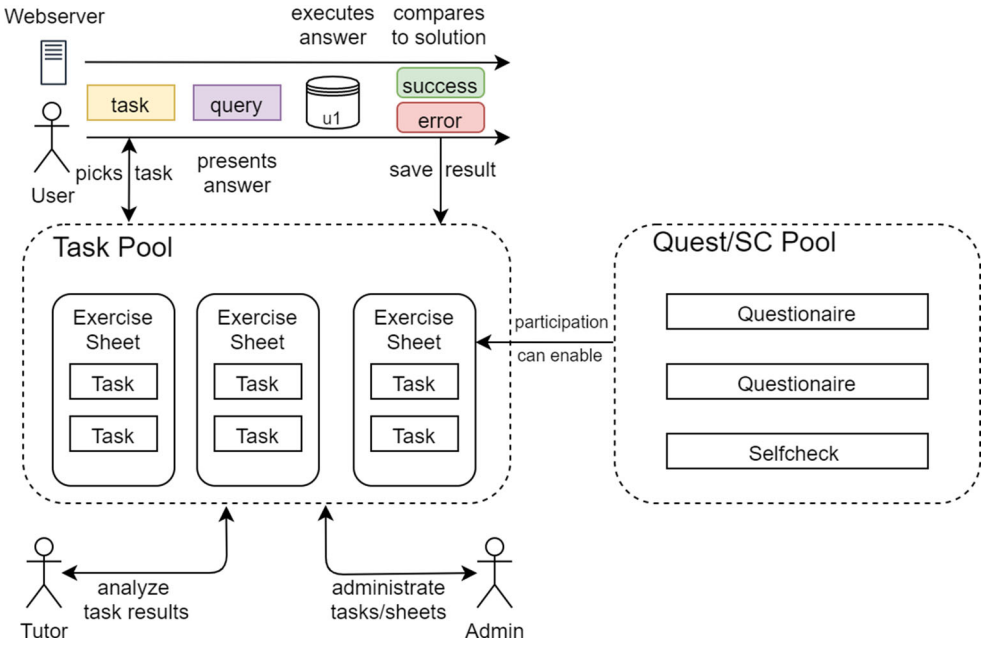


**Fig. 2** Granular architecture of the SQLValidator

- **Personal Study:** Students with access to the SQLValidator can freely explore the inbuilt database with tasks from the exercises to prepare for exams or for other purposes.
- **Course Exercises:** The database lecture in Magdeburg comes with several exercise tasks. SQLValidator is used to administer these exercises. In this use case, students are allowed to continue correcting errors in their queries until they get the expected result. During the trial and submission, they are provided with the expected schema and informative feedback.
- **Self Checks:** These tests are crafted to help students assess their level of understanding of the already covered topics in the database concept course. Students can repeatedly take the self-check test until they are satisfied. The questions used in these self-checks are mainly standardized and multiple choice. Additionally, we incorporate tests for SQL query skills relevant for the learned course skills.
- **Questionnaires:** At particular periods in the course of the semester, students are encouraged to evaluate either their learning experience during the course or their user experience with some of the tools we employ in administering the course exercises, for example, the SQLValidator system. To administer these evaluations, we directly use our SQLValidator.

## 4 Architecture of SQLValidator

In this section, we discuss the general architecture of SQLValidator and the individual sub-components. Web-based educational systems compared to standalone systems are more advantageous because they:

- remove the challenge of software compatibility problems when distributing the installation package,
- remove platform restriction challenges since browsers are standardized environments, and
- the challenge of insufficient workstations in the PC-pools is resolved as students can use their mobile phones in solving tasks [2].

### 4.1 Components of the Architecture

In the implementation of web-based systems, three main features are important, which are centralization, replication, and distribution [9]. SQLValidator uses a centralized client-server architecture. In Fig. 1, we show the general architecture of SQLValidator.

In this architecture, users access the application via a web interface and submit requests, task submissions. These task submissions contain all information necessary for its processing. The PHP server mediates between user requests, the databases and the relational database management system. In SQLValidator, there are four main databases:

- `sqlvali_data` contains all relevant data to maintain the organization of the SQLValidator itself, such as user management and task definitions.
- `sqlvali_master` contains all standard tables and data used to perform an evaluation of the user query on the database.

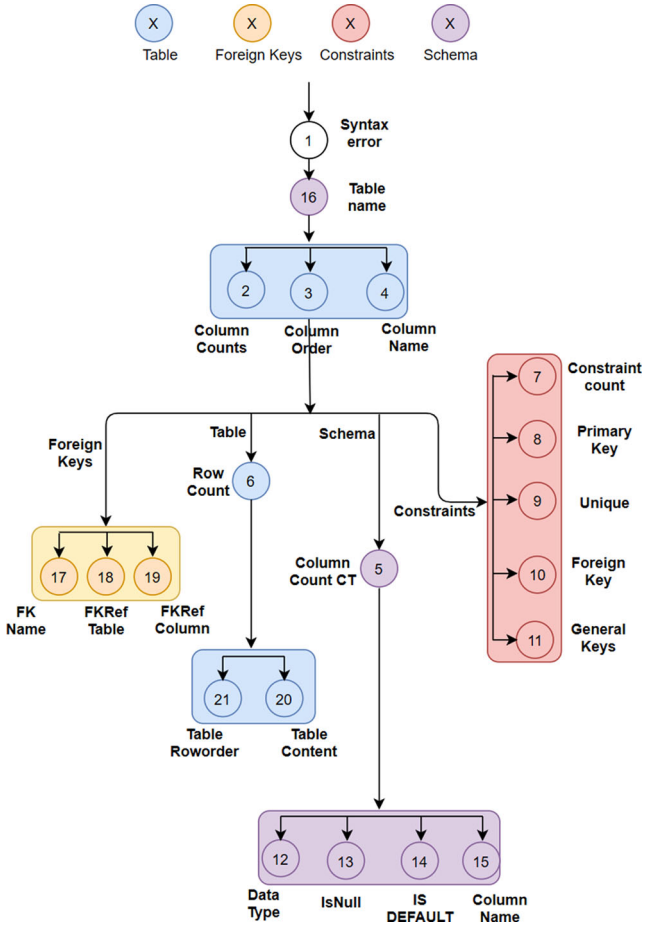


**Fig. 3** Error classes

- `user_#` contains physical user-specific slave databases related to the sqlvali_master where the users can perform their own queries.
- `sqlvali_eval` contains all relevant evaluation data to provide a proper basis for analytics.

Now given that a student with access already picked a task and submitted an answer for the task, as shown in Fig. 2, SQLValidator validates the query for syntactic errors and then compares the resulting schema with a pre-defined solution schema for the particular task. If there is a mismatch between the two schemas, further evaluation is carried out and the cause of the error is returned to the student. The student can then correct and resubmit the query again until the query statement is successfully validated. A record of all last attempts for each individual task is saved on the user's page for that task. For analytic purposes, all submission attempts are saved in the sqlvali_eval database.

### 4.2 The Validation Process

The validation module is the core of the SQLValidator system and contains all relevant operations to manage task evaluation, which is triggered when the student submits a task. For the subsequent validation, we use the natural representation of a database query, which is a relational table. This representation can be used to evaluate all respective language features from the DDL (Data Definition Language), the DML (Data Manipulation Language), the QL (Query language) which may be required by the question. For instance, a DDL statement creates a schema, which can be retrieved as a table and can be compared with the tabular representation of the schema of the solution. Similar to this, the resulting table of a student's QL query can be compared with the result of a stored reference query. Hence, the validation process works independently of what exact language features the student uses but requires that the solution gives the expected result. On an incorrect result, the student is given direct feedback depending on his or her mistakes. Of course, this could lead to students „cheating" in their submissions by not using the needed language constructs (e.g., joins), but selecting values fitting to the expected results (e.g., predicates on the primary key column in the WHERE clause). However, we have so far not acknowledged such semantically wrong submissions in an extensive manner and are sure that students are not tricking on SQLValidator.

### 4.3 Types of Feedback

Currently, there are four levels of feedback in the SQLValidator system: positive/negative feedback which is indicated with three color codes (green, yellow, and red), a hint, a reference solution, the student's partial or correct solution. Feedback for a DDL error is shown in Fig. 4. So, during task attempts and submissions, feedback containing an error class name is sent to the student's task result page. These error classes are represented in the system with numbers, i.e., error codes. In the next section, we discuss these error classes.

## 5 Error Classes in the SQLValidator

SQL consists of several language features and concepts such as predicates, joins, groupings, constraints, ordering. All these by design can lead to different types of errors. While some of these errors are easily understandable, others require some debugging for resolution. In this section, we look at various types of error classes that a student may encounter while solving their tasks using SQLValidator. Fig. 3 gives an overview of the error classes. For each error, we

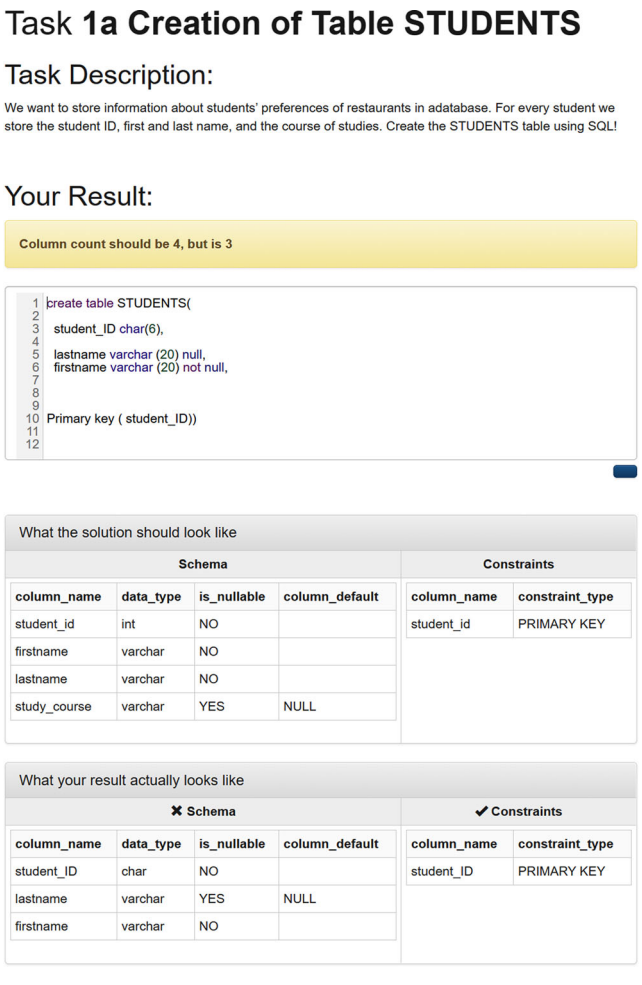


**Fig. 4** DDL error notice

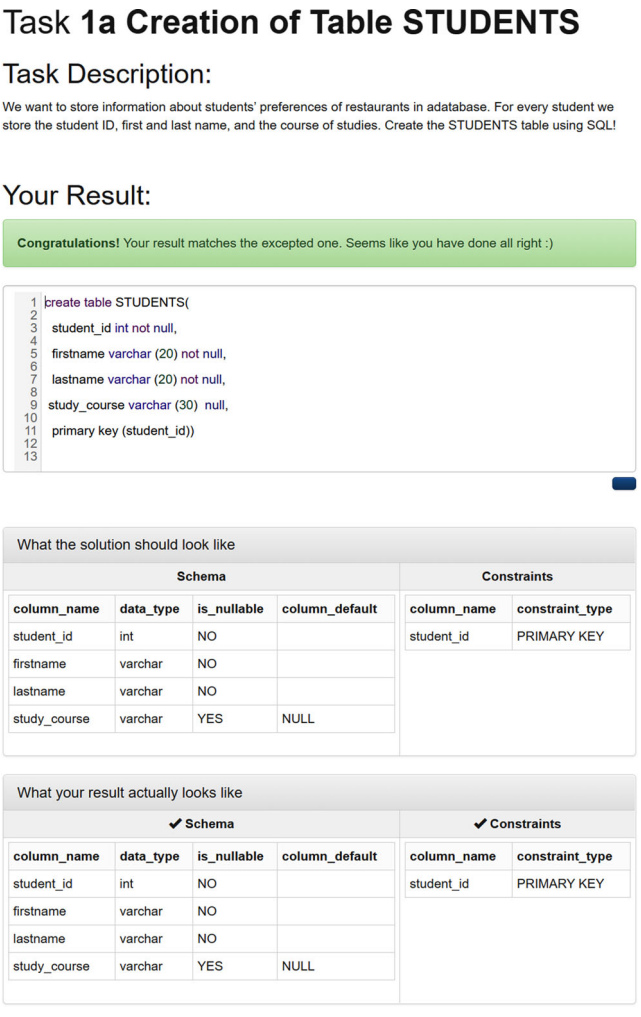


**Fig. 5** Success feedback page

display the corresponding error code that we use in our system and throughout the paper. Furthermore, as shown in Fig. 3, we grouped these error into specific classes. These groups of errors can be reported together if they occur together. The classes displayed individually dominate other error classes down the hierarchy. For example, if a syntax error (e.g, a missing bracket) occurs together with a column counts error (e.g, the user selected a wrong amount of columns from the database with a SELECT statement) the column counts error is not displayed until the syntax error is corrected. This means that a syntax error dominates all other errors. The same applies to the table name error, which dominates all other errors after the syntax error, while the column counts error and column name error can occur together. For example, given the following DDL statements to create a table `'student'` as shown below, syntactical errors such as a missing semicolon, unbalanced parenthesis, are first checked. Then the table name, column name, the constraints, etc., are validated. A syntax error shows a red color, while a semantic error is shown in a yellow color.

```
CREATE TABLE student(

StudentID char(6) NOT NULL,
Firstname varchar(20) NOT NULL,
Lastname varchar(20) NOT NULL,
Course varchar(20),
PRIMARY KEY (StudentID));
```

If there are still semantic errors, the student receives further feedback, which compares the result of the student's input with the expected output. Based on the SQL statement below, this type of error falls into the class of data definition language error. The feedback in the case of a DDL error is shown in Fig. 4 above.

Once all errors are corrected, the student gets the positive feedback of Fig. 5. As mentioned above, the main use of SQLValidator is to administer course exercises. In the next section, we discuss the educational outcomes of using SQLValidator in our introductory database course of the summer semester 2020.

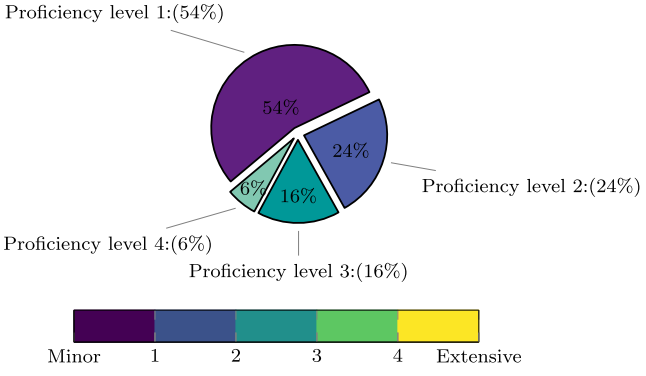


**Fig. 6** Initial SQL competency

## 6 Educational Outcomes

In this section, we present our quantitative results. We had 160+ enrolments in the Moodle course website for the Database Concepts course that was conducted during the 2020 summer term. We coordinated weekly exercises, which consist of several tasks that test the students' understanding of the weekly lectures. Of course not all task sheets relate to SQL and thus, we focus mainly on Exercises 2 and 7 in our evaluation. All tasks of the week have to be submitted within Moodle, which are then graded.

For the exam allowance, students have to reach a 50% threshold of successful tasks i.e., 50% score in the exercises is an essential prerequisite to take the exam. Although the usage of SQLValidator is optional for the course, 116 students created an account in our SQLValidator system. From the task submissions, we observed that more than 72 students made consistent use of the SQLValidator.

### 6.1 Prior Skills of Students

At the beginning of the exercise classes, we carried out a survey in which we inquired about the students' general programming and SQL competency level. The results of these two surveys are shown in Figs. 6 and 7. The results suggest that: 54% had only minor SQL skills, 40% already had a brief introduction to the SQL language, while only 6% had broader experience with the language. The students' general programming knowledge level was similar to that of SQL. As 55% had no programming skill, 41% already had a previous introduction to programming, and 4% already had noticeable previous experiences with programming languages.

Overall, the majority of the students in this population are beginners in SQL and even do not have skills in other programming languages. This means that we taught them the basics. Considering that the course during this semester was administered via Zoom, the normal group work was not feasible since the students attended the lectures online. Hence, novice students had to rely on the feedback feature of SQLValidator. They relied on correcting their previous submission and then resubmitting it again until the correct answer is achieved. Giving this possibility of trial and resubmission, we were able to observe the engagement dynamics, the different types of error, and the average time spent on various exercises. In the next section, we show improvement and learning from the repeated errors encountered by students during their weekly use of SQLValidator in solving the course exercises.

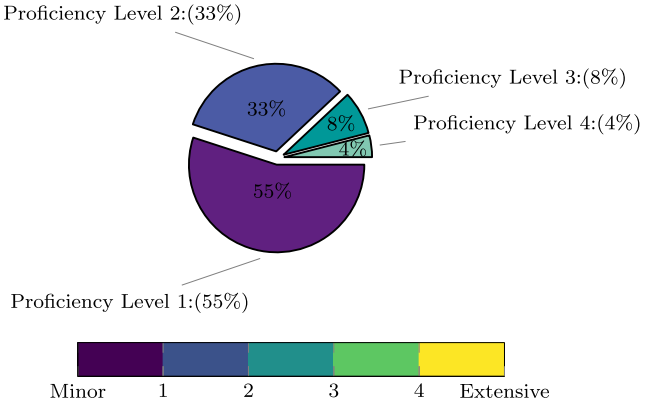


**Fig. 7** Initial programming competency

### 6.2 Error Analysis

Errors are part of learning processes and can improve learning [7]. However, errors can also have a negative effect on learning and motivation to learn if students are not able

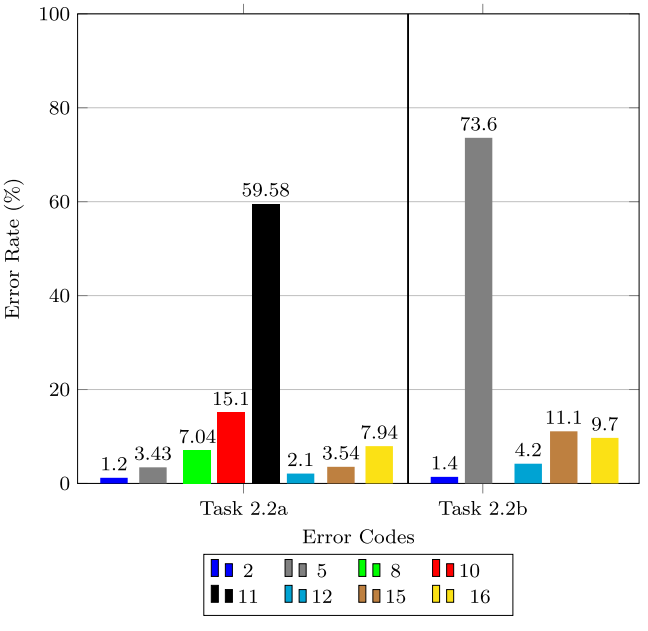


**Fig. 8** Error Analysis for DDL Task

to correct them [4]. In this section, we discuss the various errors that students encountered while solving their assignments in the database concepts exercises. It is important to note that the exercises tested more than one SQL concept. These concepts are first introduced and taught in lectures and after the students' submissions, the results are discussed in the exercise meetings. For brevity, we rely on the error classes that we have shown in Fig. 3 to compare the error rates.

Fig. 8 shows an indication of learning with respect to the percentage of errors made by the students while solving two similar data definition language tasks. The bars represent different error classes as described above. Task 2.2a submissions resulted in 544 errors, 1.2% of which were column count errors, 2.43% row count errors, 7.04% primary key errors, 15.1% foreign key errors which indicates that the foreign Key Constraints were incorrectly formed, 59.58% general key errors which indicates that a foreign or primary key is incorrectly formed, 2.1% data type error, 3.54% column name error. Task 2.2a is the first exercise task that required the submission of an SQL query so this error rate was expected. Task 2.2b questions are similar to Task 2.2a but only 72 errors occurred. Column count CT, which indicates that the submitted query contains a wrong

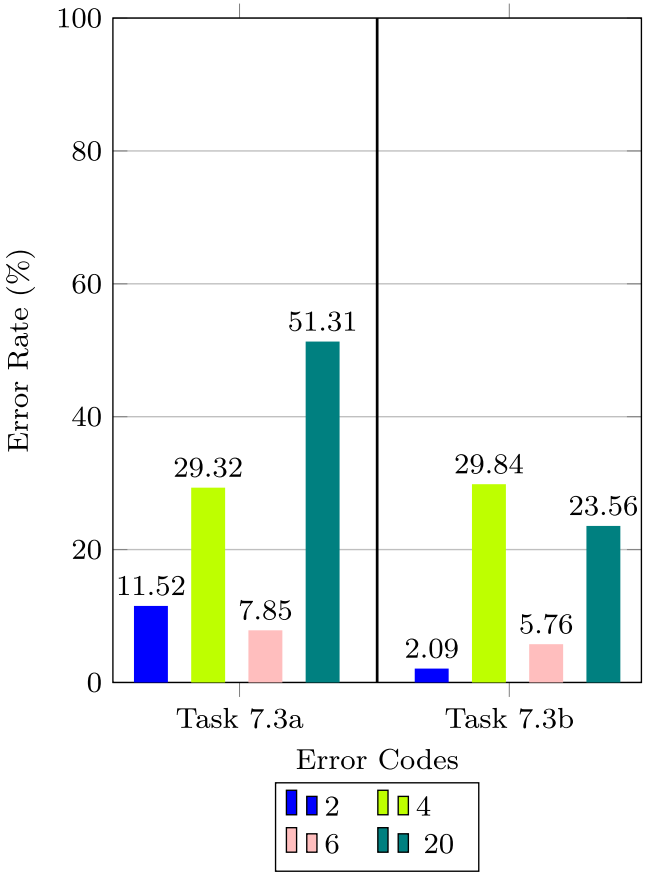


**Fig. 9** Error Analysis for QL Task

number of columns in a CT statement was the most frequent error followed by column name error which means that the name of a column is wrong or missing. This result shows that learning occurred. The error rate for Task 7.3, which tests the understanding of the query language part of SQL is shown in Fig. 9. Task 7.3a submissions resulted in 191 errors, 11.52% of which were column count errors, 29.32% column name errors, 7.85% row count errors, and 51.31% table content errors while submissions in Task 7.3b resulted in 117 errors. In general, the feedback received from iteratively solving the first part of both tasks 2 and 7, resulted in a lower error rate in the second part of both tasks. This is evidence of learning. The same pattern was also observed in other tasks. As a result, the gained information from this analysis helps us to know in which area we have to place more emphasis when conducting future exercises.

### 6.3 Student Performance

Having discussed the learning improvements with respect to the different types of errors encountered by students while using SQLValidator, we provide in Fig. 10 the analysis of their performance in the SQL task for the database concepts exam. For this analysis, we group students according to two criteria: their effort and motivation regarding the course (i.e., their prerequisite percentage of submitted exercise task as we stated above) and their usage of SQLValidator. The 50–70% prerequisite group consists of 11 participants, the 70–80% prerequisite group consists of 15 participants, 80–90% prerequisite group consists of 10 participants and the 90–100% prerequisite group consists of 12 participants. The total achievable points for this task is 16. Each of the represented points is an average of the total group point. The red bars represent the individuals that did not use SQLValidator in solving their exercises while the blue bars represent students that used SQLValidator. SQLValidator has a feature that allows administrators to analyze user activities. These activities are generated whenever a user submits a query for every given task. These users that did not use the SQLValidator had no activity or never registered for the SQLValidator.

As discussed before, students need at least 50% as a passing score for the course exercise submission. Their actual final passing score can however be way higher than 50%. Hence, we created four different prerequisite groups for the performance analysis: 50–70% prerequisite group (as rather unmotivated students that do just as much as necessary to pass the assignment), 70–80% prerequisite group, 80–90% prerequisite group, and 90–100% prerequisite group (which are the most motivated students with some reaching 100%). For these groups, we compared the final exam points that they achieved in the SQL task. Please mind we excluded students that failed the exam. We assume that these stu-

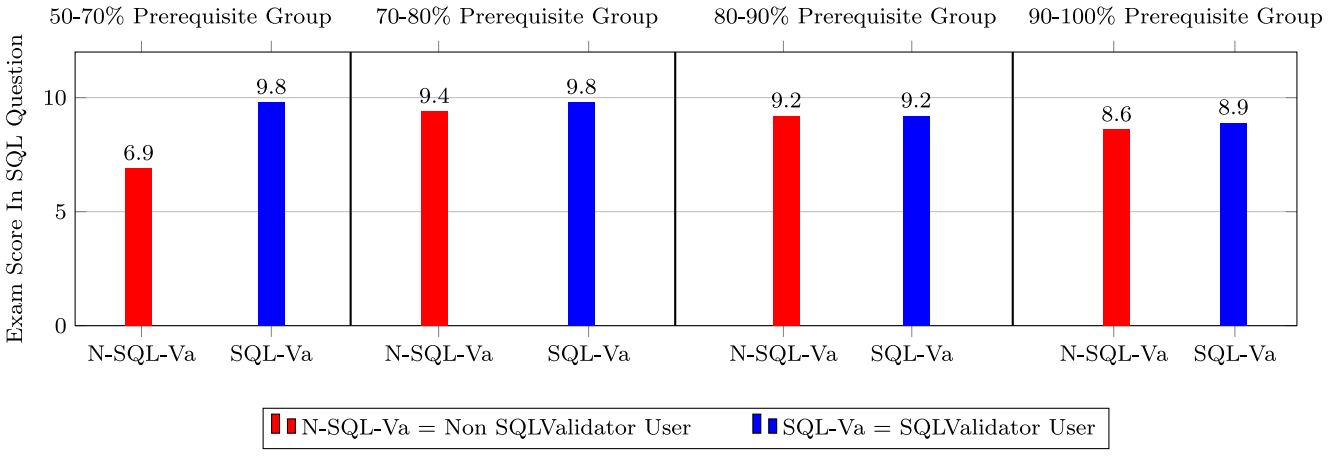


**Fig. 10** Average Exam Points for SQL Exam Task

dents took the exam to see the style of the questions. Also, a failed exam does not count this semester due to the Corona situation.

According to our questionnaire, the 50–70% group contains mostly new students that had little contact with other students. These students took the course while still in their hometowns as a result of the lockdowns. Hence, they had no access to previous solutions to the exercises and heavily relied on SQLValidator. To re-enforce the location observation, we analyzed their Moodle data. The analytics page of the Moodle learning management system used to administer the course records user activities, including their location information. So, instructors can determine a student's location, analyze study patterns, degree of participation, and other information. We observed that these novice students had good participation in lecture activities. Their usage of SQLValidator had a positive impact on their exam performance and we can state that they learned SQL better than without using our SQLValidator.

The performance observed for the 70–80% group is interesting. This grade group is also mostly made up of new students and we also observe that members of this group have a high activity on SQLValidator ranging from 210 activity hit points to 580+ hit points. This activity correlates to their performance as shown in Fig. 10. Despite not having access to past questions, their performance is still better compared to the 90–100% group. The performance for the 80–90% and 90–100% groups show only a small improvement when using SQLValidator. Apparently, their motivation towards the course led to an advanced knowledge state that cannot be improved substantially by SQLValidator anymore. However, especially the exercise submissions of the 90–100% group showed a significant amount of copied solutions. These submissions were exactly the same in both the SQL theory task answers and the answers submitted for the query questions. This suggests that their motivation towards the course was only pretended as they did not engage with the course. This insight accounts for an overall lower exam score for this group.

## 7 Summary and Future work

In this paper, we introduced SQLValidator, a web-based interactive tool for learning and practicing SQL. We evaluated students' exercise engagements and presented the errors they encountered. Our results show strong engagement from mostly new students who spent an average of 10+ hours in each exercise. We also provided an analysis of the possible errors they encountered in the process of using SQLValidator in solving their course exercises. Finally, we presented an analysis of the exam score they achieved in the SQL exam task. This enables us to evaluate the effectiveness of the SQLValidator system with respect to the goals of its creation.

SQLValidator is designed as a playground where students who are new to the SQL language can explore and understand SQL. It is possible that we make it open source but currently, it is not yet feature complete. We plan to extend its functionality by integrating a relational algebra interpreter. This allows users to phrase relational algebra queries and run it on a given database. We plan to add a personalized feedback system, which serves as a recommendation system with respect to frequent query mistakes made by a student. Exercise solution distribution among students cannot be stopped as it is also part of learning. Hence, in a future version of SQLValidator, a time tracking and query count feature will be implemented. Such that for all submissions to a given task, we know which student was first to submit a particular query. Additionally, we will know how often a particular query was used. Collaboration among students during course exercises and projects is a concept

we are very interested in investigating and nurturing. Hence, we will extend SQLValidator with pair programming features. This helps students to develop a sound collaborative programming culture, which is important in software-related projects. Furthermore, the process of query validation is not fully discussed here. A future paper will discuss this process.

## References


1. Abelló A, Burgués X, Casany M, Martín C, Quer C, Rodríguez M, Romero O, Urpí T (2016) A software tool for e-assessment of relational database skills. Int J Eng Educ 32(3(A)):1289–1312
2. Alpert SR, Singley MK, Fairweather PG (1999) Deploying intelligent tutors on the web: an architecture and an example. Int J Artif Intell Educ 10(2):183–197
3. Dietrich SW (1993) An educational tool for formal relational database query languages. Comput Sci Educ 4(2):157–184
4. Hawlitschek A, Köppen V, Dietrich A, Zug S (2019) Drop-out in programming courses–prediction and prevention. J Appl Res High Educ. https://doi.org/10.1108/JARHE-02-2019-0035
5. Kearns R, Shead S, Fekete A (1997) A teaching system for SQL. In: Proceedings of Australasian conference on Computer science education, pp 224–231
6. Kleerekoper A, Schofield A (2018) SQL tester: an online SQL assessment tool and its impact. In: Proceedings of the Annual ACM Conference on Innovation and Technology in Computer Science Education, pp 87–92
7. Metcalfe J (2017) Learning from errors. Annu Rev Psychol 68:465–489
8. Mitrovic A (1998) Learning SQL with a computerized tutor. In: Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education, pp 307–311
9. Mitrovic A (2003) An intelligent SQL tutor on the web. Int J Artif Intell Educ 13(2–4):173–197
10. Sadiq S, Orlowska M, Sadiq W, Lin J (2004) SQLator: an online SQL learning workbench. In: Proceedings of the annual SIGCSE conference on Innovation and technology in computer science education, pp 223–227
11. Taipalus T, Perälä P (2019) What to expect and what to focus on in SQL query teaching. In: Proceedings of ACM Technical Symposium on Computer Science Education, pp 198–203