QUERY: Interactive SQL Learning and Assessment

Michal Barla, Matúš Kislan, Matej Víťaz

Abstract: *The paper presents a web-based application QUERY, which provides and interactive SQL learning environment through automatic evaluation SQL queries. We describe its service-oriented architecture, which foster re-usability and modularity and provide details on important aspects of the solution. We also share our experience from deployment of the system to support SQL learning and assessment in introductory course of database systems for undergraduate students.*

Key words: *SQL, Automatic evaluation, Interactive learning, RESTful Web APIs, Case study.*

INTRODUCTION & RELATED WORKS

Relational databases, with their ACID properties, became a de-facto standard for data persistence of software applications. NoSQL databases did not replace traditional relational databases, but rather serve as a supplement to provide more convenient and faster access to some part of the data, following principles of “Polyglot Persistence”, a term coined by Scott Leberknight[8]. Relational databases stay in their role of primary storage and retain crucial transactional data, which need to withstand all kinds of errors, originating either in user, software or even hardware.

This is the reason why every computer science student should master the way how we interact with relational databases – using structured query language (SQL). The only meaningful way how to achieve this is via intensive “hands-on” sessions, when students are practicing on given datasets and discovering effects of various SQL constructs on the result set. This brings in two key problems:

- High *time-to-first-query* – our experience shows that when students are provided with a dataset (database dump), which they are expected to import into their own DBMS, they spend way too much time on creating the database and importing the data – and thus not practicing SQL.
- Complicated feedback delivery – students need to know whether their solutions is correct or not. A teacher can either provide a correct SQL statement or list the correct (expected) result set. The former approach allows students to check and compare their solutions with the teacher’s one, but does not reflect the fact that SQL usually provides several different ways how to achieve the same result. At the same time, when a teacher makes the solution public, there will always be a group of lazy students, who will directly pick the teacher’s solution without giving any effort from their side. Employing the second approach (providing the expected result set) means that students will struggle comparing their results with the expected one, which is not only time-consuming but also very error prone.

There are several third-party web-based services, which allow anyone to practice SQL by solving exercises, such as SQLZoo[9], LeetCode[10,] SQLBolt[11] or SQLTutor[12.] Obvious drawback from a teacher’s perspective is a lack of pedagogical features of these systems – lack of control over exercises, lack of feedback on students’ progress and no way how to leverage such services for automatic assessment of students during exams.

In order to overcome aforementioned challenges and drawbacks of third-party services, we develop QUERY, a web-based system for interactive SQL learning and assessment. In this paper, we describe its architecture and functionality and share our experience with its deployment into introductory course of database systems at our school.

8 http://www.sleberknight.com/blog/sleberkn/entry/polyglot_persistence
9 http://sqlzoo.net/
10 https://leetcode.com/problemset/database/
11 http://sqlbolt.com/
12 http://sqltutor.fsv.cvut.cz/cgi-bin/sqltutor

The idea of computer-assisted learning and assessment of SQL is not completely novel, especially when we consider the age and maturity of relational databases, whose foundations were laid already in 1970 by Edgar F. Codd [1].

Apart from already mentioned publicly available web applications, there are many solutions from academics, who use them to support their classes. They are usually focusing on various challenges, faced when deploying such a system in a course: *(i) Students trying to "fool" the system*, by providing a hardcoded query, which gives exactly (and only) the requested correct output. This is usually solved by evaluating submitted queries on two distinct datasets, where the second one is never shown to the students. *(ii) Problem of binary scoring*, which is perceived very negatively by students. There are approaches relying on peer reviews [2], approaches giving partial marks for syntactically correct attempts returning at least correct schema of the output [3] or approaches which analyse and compare queries by means of abstract syntax trees [4]. *(iii) Problem of feedback* – students require more detailed feedback on their attempts than "correct/wrong". Some approaches are trying to provide hints guiding to correct solutions [4].

OVERVIEW OF QUERY APPLICATION

Our QUERY application consists of two completely separated services:

1. *QUERY backend* – RESTful web API implemented in Sails.js[13] and communicating using JSONs. It acts as a wrapper on top of Postgres relational database and handles everything related to datasets, exercises and evaluation of solution attempts.
2. *QUERY frontend* – web application written in Laravel[14], which acts as a client of QUERY *backend* service and adds pedagogical aspects of the problem into whole solution with concepts such as students access management, terms, weeks, tests, solution attempts history or feedback from students.

With such a separation of concerns, where all the functionality, which somehow depends on a task of SQL handling and evaluation is extracted into a separate web service, we gain high configurability and flexibility of the final solution. For instance, we can provide alternative clients on top of our backend service – this actually happened, when the backend was used to evaluate SQL part of solutions submitted by participants of *Game of Codes* competition held at our faculty (the competition was using its own client for all kinds of problems, not limited to relational databases). Next, this separation allowed us to build our frontend in a way that is fairly well abstracted from the domain of SQL. It can be used as an interface for solving various types of tasks, ranging from querying non-relational databases such as redis or elasticsearch, through scripting in languages such as bash, python or ruby up to programming assessments in compiled languages such as C and C++. All we need to do is to setup another backend service responsible for evaluating user input for that particular type of task.

QUERY backend

Our backend service provide a RESTful API, which can be divided into two parts: *(i) structure definition section*, providing endpoints for creating/updating/deleting schemas and exercises and *(ii) operational section,* providing endpoints for registration of clients, retrieval of exercises and submitting solution attempts.

An interesting feature of our solution is automated versioning of database schemas. A teacher can decide to add additional tables into the existing schema, change names or structure of existing ones or update the data in the tables. In order to prevent an unwanted effect of such change on exercises already present in the system, we create a new version

13 Sails MVC framework for Node.js, http://sailsjs.org/
14 Laravel PHP framework, https://laravel.com/

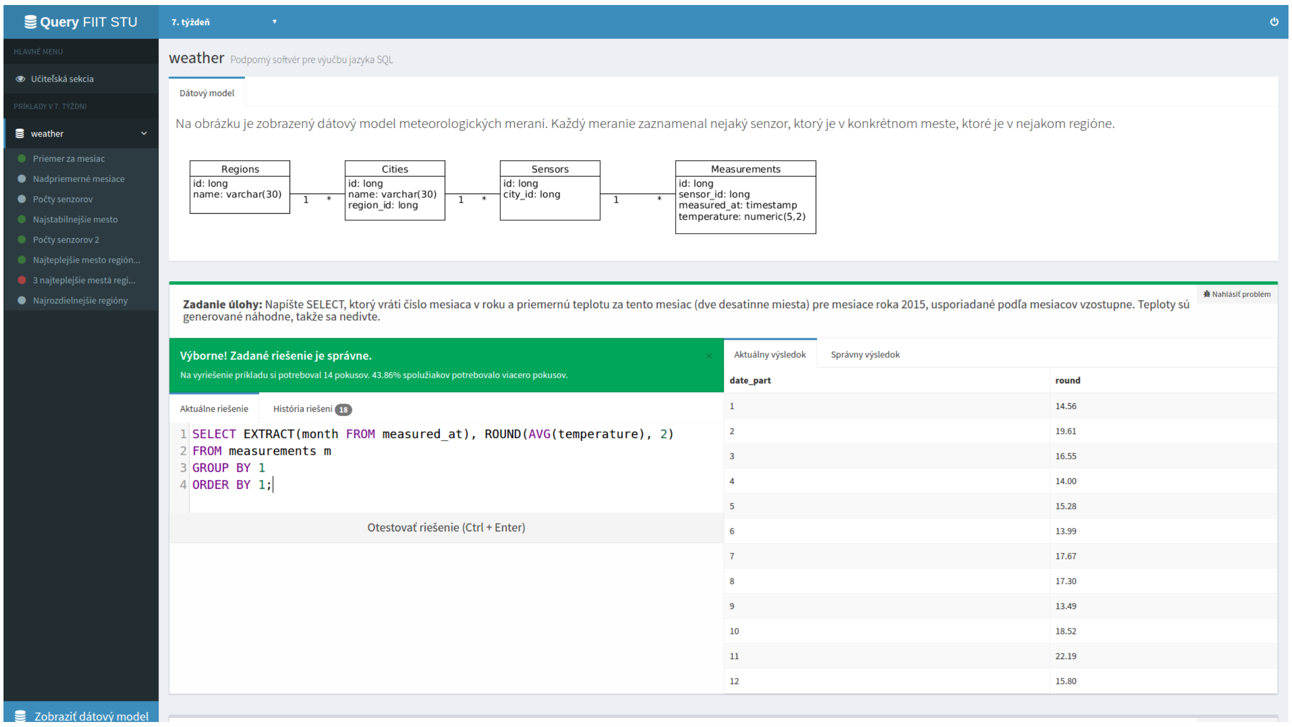
of the schema and link all subsequently created exercises into it – without breaking the functionality of previously created exercises. All these schema versions are realized using schemas of Postgres and are thus completely separated from the operational database.

Apart from linking exercises to their schema versions, we allow a teacher to tag exercises with any number of arbitrary created tags. Tags are the only metadata about exercises the QUERY backend is aware and responsible of.

Evaluation of solution attempts is performed using a transaction which is rolled-back at the end, to prevent any intentional or unintentional data changes from students’ queries. The evaluation itself is performed by comparing results sets of student’s and teacher’s queries on public and testing datasets. A teacher can flag whether an exercise requires a specific ordering of results or not during its creation. If results are not identical or student’s query did not executed correctly in the database engine, an appropriate error message is returned to the client.

QUERY frontend

Our frontend service is providing a user-friendly client access to QUERY backend. When logged in, a student sees an interface as presented on Figure 1, where she can see all exercises of the current week, grouped by their schemas. Top part of the screen shows a description of the current schema along with its data model. This data model can be displayed at any time in a modal dialog by clicking the icon in the bottom left corner. The central part show individual exercises, where the left part contains the input area and allows students to browse their previous attempts, while the right part shows database output for the most recently submitted query with a possibility to show the expected result.



weather Podporný softvér pre výučbu jazyka SQL

Na obrázku je zobrazený dátový model meteorologických meraní. Každý meranie zaznamenal nejaký senzor, ktorý je v konkrétnom meste, ktoré je v nejakom regióne.

```
Regions: id: long, name: varchar(30)
Cities: id: long, name: varchar(30), region_id: long
Sensors: id: long, city_id: long
Measurements: id: long, sensor_id: long, measured_at: timestamp, temperature: numeric(5,2)
```

Zadanie úlohy: Napíšte SELECT, ktorý vráti číslo mesiaca v roku a priemernú teplotu za tento mesiac (dve desatinne miesta) pre mesiace roka 2015, usporiadané podľa mesiacov vzostupne. Teploty sú generované náhodne, takže sa nedivte.

Výborne! Zadané riešenie je správne.
Na vyriešenie príkladu si potreboval 14 pokusov. 43.86% spolužiakov potrebovalo viacero pokusov.

```sql
SELECT EXTRACT(month FROM measured_at), ROUND(AVG(temperature), 2)
FROM measurements m
GROUP BY 1
ORDER BY 1;
```

date_part	round
1	14.56
2	19.61
3	16.55
4	14.00
5	15.28
6	13.99
7	17.67
8	17.30
9	13.49
10	18.52
11	22.19
12	15.80

Figure 1. Screenshot of QUERY student interface (in Slovak)

Left sidebar depicts shortcuts to all available exercises of this schema, with a color-coded status of solution (green – with correct attempt, red – incorrect, grey – not attempted yet). The same color-coding is also used in the central part, on top of each exercise. In the case of a correct attempt, a student is also informed about her performance among her classmates – i.e., percentage of students who needed more trials to get to the correct solution.

Each exercise contains a button to give feedback about the exercise to the teacher. This can be a notification about a typo, error in solution, an unclear assignment etc.. The feedback is visible in the teacher's section and application can notify about it to email or Slack. A teacher can mark any feedback as resolved or completely remove it.

We observed that many students had difficulties solving some harder examples, but did not initiate any discussion about it via email or through faculty-supported CQA system [6]. Some of them tried to communicate via feedback functionality, but this communication is not public to their peers nor it is what feedback is meant to be for. We hope that by lowering the barrier of communication we will be able to incent students to communicate more publicly. Therefore, we plan to extend student section with tools supporting social learning, promoting public discussion about particular exercises. This can be done, for instance, by embedding social widgets.

Apart from showing students' feedback, teacher's section of the application allows to setup basic metadata such as list of students or creation of exercise groups. A group represents a week of term, i.e., it groups exercises, which should be solved together, either during the labs or on the exam. For this purpose, a teacher can decide to hide a week with its exercises from students (e.g., exercises prepared for an exam) or lock the system to a specific week (e.g., only exam week is visible).

We are currently working on an advanced reporting, which will be added to this section of the system. It will allow a teacher to see the overall activity and performance of the class (or particular group of students), to identify exercises, where students do struggle to find the correct answer. Currently we are doing such analysis outside of the system, using the database dump and external tools.

CASE STUDY

We were using our QUERY system in introductory course of database systems for undergraduate students (270 in total at the beginning of the course) at Faculty of Informatics and Information Technologies of Slovak University of Technology in Bratislava. The system was available from the week #5 when we started the topic of SQL and students were receiving new exercises in weeks #6, #7 and #11. Students were not forced to use the system, but they were motivated to do so and to practice their SQL skills, as there were small, paper-based tests of SQL at the end of each session.

Apart from exercises for the labs, the system was also used during final exam (and its resit) where students were allowed to freely trial their solutions in the QUERY.

Figure 2 depicts the number of attempts registered by the system for each day. We can see that the system was used regularly during the weeks when the labs were scheduled. The highest peak is, as we expected, on the eve of exam, where many students were reviewing and redoing their solutions from the labs in the system. However, we were surprised to find substantial activity also during the days right after the exam, where students did not know yet the official results of the exam. Obviously, many of them started to prepare themselves already for the resit.

We were interested to find out how many failed attempts students submitted before they got to a (hopefully) correct solution. Figure 3 on the left shows a boxplot showing distribution of attempts for all students, where each student got an average count of her attempts over all exercises. On the right, we show how this shifts if we take 99 percentile into account instead of an average. We can see that there were students who needed a fairly high number of attempts to get the correct result. We believe that such high numbers are not really desirable, it seems that those students were just desperately guessing the solution, but did not make any effort to contact the teacher and ask for some additional hints or explanations. As we already mentioned, we would like to overcome this problem in the future by incorporating social supporting tools directly into the system. We received a very positive feedback from students, regarding the QUERY system.

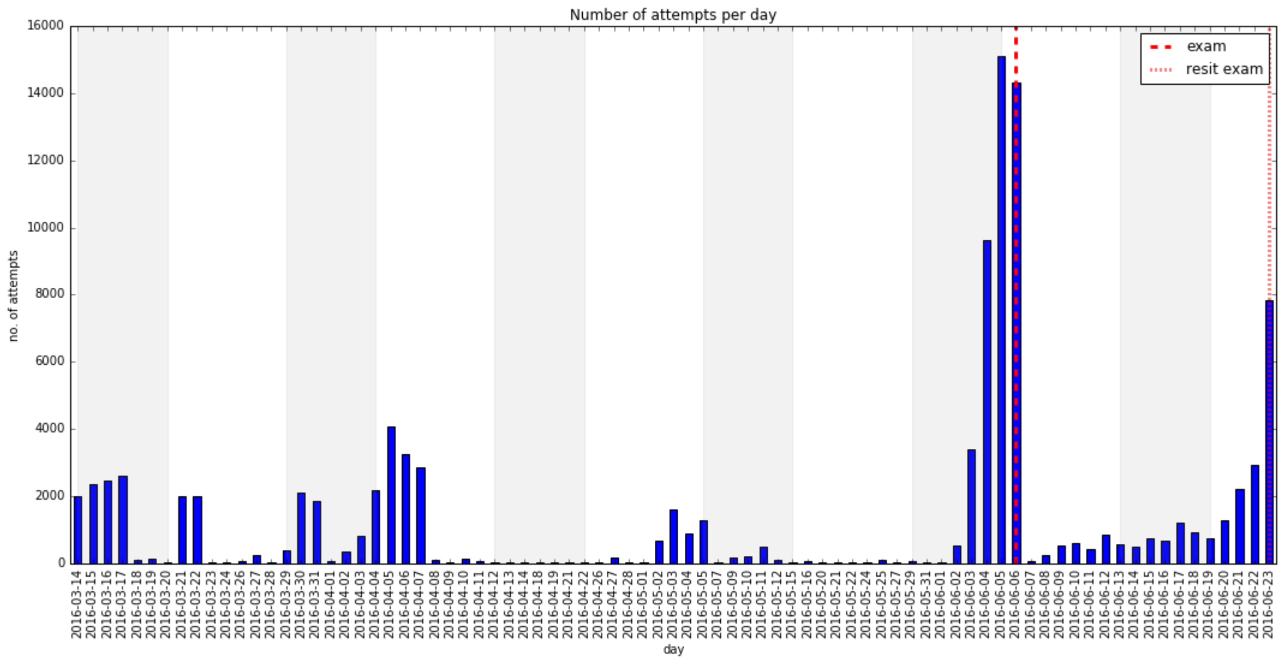


Figure 2. Visualization of total amounts of attempts per day, from the first day when exercises were made public in the system until the resit exam of the course.

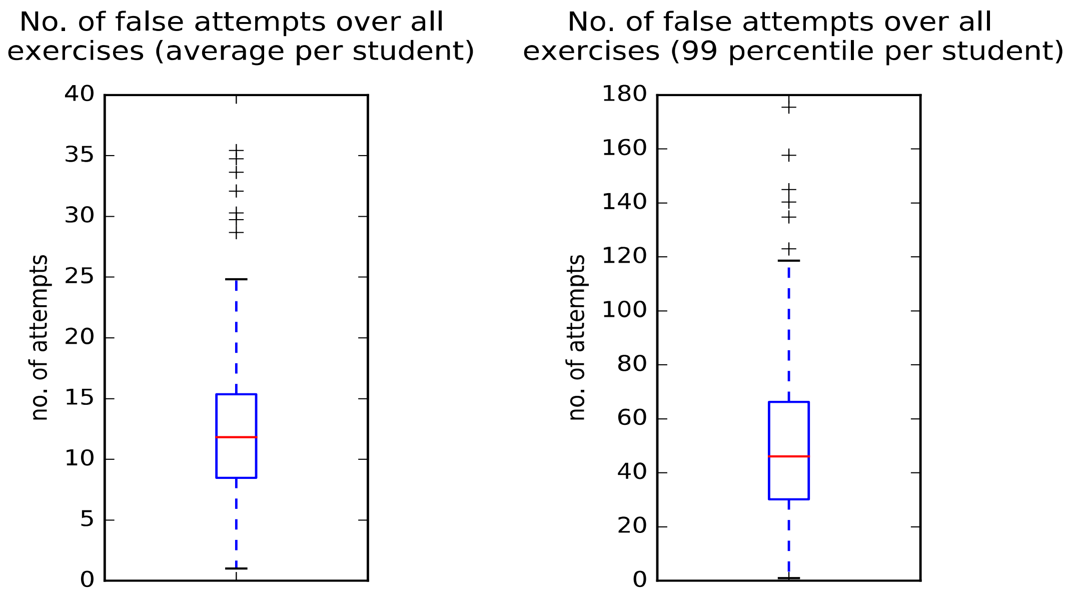


Figure 3. Boxplots showing the number of false attempts of all students over all exercises.

Figure 4 shows how well students performed by taking into account number of successfully solved exercises in the system prior the exam and plotting this against their success rate on the exam. There were 4 rather challenging tasks on the exam, covering SQL practice part. Possible success rates were thus 0%, 25%, 50%, 75% and 100%. The figure indicates that there is indeed a correlation and students who solved more exercises prior the exam were more likely to score well on the exam. This was furthermore confirmed by Spearman correlation coefficient of **0.502** (p-value 7.47e-15), which indicates moderate correlation.

CONCLUSIONS AND FUTURE WORK

Concept of automatic evaluation with instant feedback, along with possibility for students to trial their solutions proves to be very efficient way how technology can really enhance learning and is also very well perceived by students.

Our future work is focused on the automatic scoring of student solutions, which is binary at the moment. Not only we should give some score to those solutions which are "on a good track", but we would also like to penalize those solutions which are unnecessarily complex or inefficient. This can be achieved by analyzing and comparing query execution plans provided by the database.

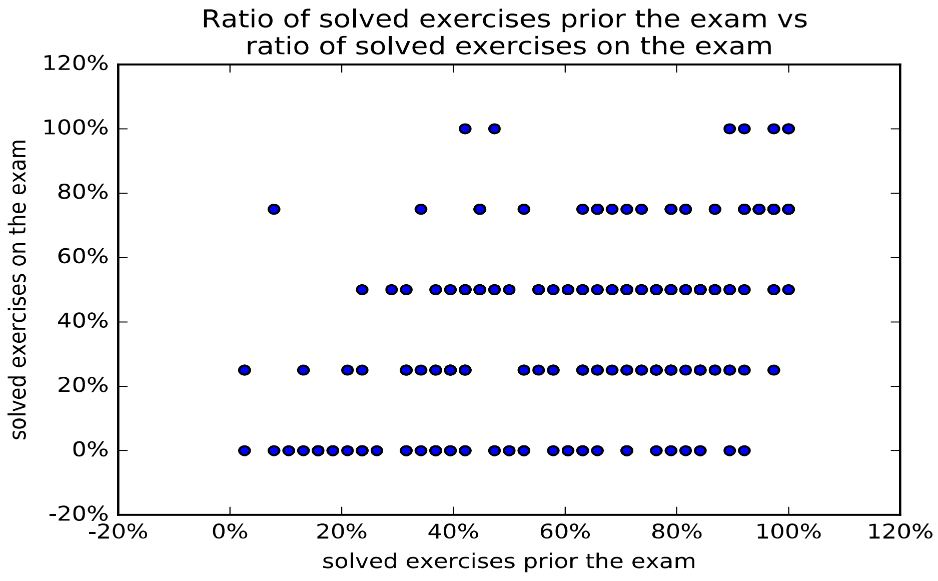


Figure 4. Correlation of meaningful activity in the system vs success in final exam.

ACKNOWLEDGEMENTS

This work was supported by the Cultural and Educational Grant Agency of the Slovak Republic, grant No. KEGA 009STU-4/2014 and is the partial result of the collaboration within the FETCH 539461-LLP-1-2013-1-BG-ERASMUS-ENW and SCOPES JRP/IP, No. 160480/2015.

REFERENCES

[1] Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. Commun. ACM 13, 6, 1970, pp. 377-387

[2] Dekeyser, S., de Raadt, M., Lee, T. Y.: Computer Assisted Assessment of SQL Query Skills. In Proc. of the 18th conference on Australasian database, Vol. 63, Australian Computer Society, Inc, 2007, pp. 53-62

[3] Kleiner, C., Tebbe Ch., Heine, F.: Automated Grading and Tutoring of SQL Statements to Improve Student Learning. In Koli Calling '13, ACM, 2013, pp.

[4] Mitrović, A.: An Intelligent SQL Tutor on the Web. International Journal of Artificial Intelligence in Education, 13, 2-4, 2003, pp. 173-197.

[5] Sadiq, S., Orlowska, M. Sadiq, W., Lin, J.: SQLator: an online SQL learning workbench. In Proc. of the 9th SIGCSE conference on Innovation and technology in computer science education (ITiCSE '04), ACM, New York, NY, USA, 2004, pp. 223-227.

[6] Srba, I., Bieliková, M.: Askalot: Community Question Answering as a Means for Knowledge Sharing in an Educational Organization. In CSCW'15 Companion Volume, ACM, pp. 179-182.

ABOUT THE AUTHOR

Dr. Michal Barla, Slovak University of Technology in Bratislava, Faculty of Informatics and Information Technologies, E-mail: michal.barla@stuba.sk.

Matúš Kislan, master degree student, Slovak University of Technology in Bratislava, Faculty of Informatics and Information Technologies, E-mail: xkislan@stuba.sk.

Matej Víťaz, bachelor degree student, Slovak University of Technology in Bratislava, Faculty of Informatics and Information Technologies, E-mail: xvitaz@stuba.sk.

The paper has been reviewed.