A Decision Tree Approach for the Classification of Mistakes of Students Learning SQL, a case study about SELECT statements

Heide Faeskorn-Woyke[1], Birgit.Bertelsmeier [2] and Jan Strohschein[3]

Abstract: TH Köln provides a web-based e-learning platform edb[4], where novices can do their first steps in SQL. The goal of this paper is to build a decision tree (manually) that classifies the novice's errors. To do so we logged data containing tasks, solutions, and wrong statements over seven months and got a table with 7533 rows as a training set. Each leaf node of the decision tree is a class of errors of similar type and generates an error message with feedback to help the user to solve the task. Interesting and surprising are the mistakes that SQL novices make. The result improves the first steps of learning SQL in a simple and personalized way and gives the teachers hints to improve their learning outputs.

Keywords: Databases, SQL, web-based learning, e-learning, educational data mining, learning analytics, decision trees.

1 Introduction

A lot of different approaches to include e-learning components to a standard database lecture have been used in the last years. A short overview is given in Section 2 of this paper. Here, we report on our development, a decision tree approach to classify the mistakes that SQL novices make to generate hints helping them to avoid these mistakes. We include these hints in the SQL-Trainer of our e-learning platform. We collected 7353 wrong statements over seven months in 2019/2020 and compared them with the correct statement to answer SQL SELECT questions. We use our edb web application mainly for database courses at the Computer Science Institute of TH Köln in the third semester consisting of roughly 300 students. However, the platform is open for other universities too. Five of the total two hundred fifty different SQL tasks are randomly selected in one session, executed, and logged anonymously when the user starts the SQL trainer in our database e-learning platform.

[1]TH Köln/Fakultät 10/Institut für Informatik, Gustav-Heinemann-Ufer 54, 50968 Köln, heide.faeskorn-woyke@th-koeln.de
[2] TH Köln/Fakultät 10/Institut für Informatik, Gustav-Heinemann-Ufer 54, 50968 Köln, birgit.bertelsmeier@th-koeln.de
[3]TH Köln/Fakultät 10/Institut für Informatik, Gustav-Heinemann-Ufer 54, 50968 Köln, jan.strohschein@th-koeln.de
[4] See https://edb2.gm.th-koeln.de, visited: 25.03.20

The following research questions are covered in this paper:

Q1: How can we construct a decision tree to classify SQL errors that students make in their first SQL steps?

Q2: How can we derive useful hints and practice to avoid these errors?

Q3: What are the main mistakes that students make?

Q4: What are the main tips for teachers in SQL that they can apply to improve their learning outputs and decrease dropouts?

2 Related Works & History

Automatic SQL e-learning applications are already in use over a long time. A good recent overview of related works can be found in Kleerekoper et al. [KS18]. They treated the automatic assessment of SQL and compared the results that students can reach by this method. Ahadi et al. [Ah15, Ah16a, Ah16b] categorized SQL statements and examined the errors the students make in these particular categories. They used the AsseSQL-Tool of Prior [Pr03], similar to Sadiq [Sa04]. Mitrovic et al. implemented the first constraint-based-tutor in 1998 and published a lot of similar works over time described in [MO16]. Brusilovsky et al. [Br10] combined several e-learning tools to an open architecture. Brass and Goldberg [BG05] made a classification of logical SQL errors with the aspect of logical and semantic error type. Taipalus et al. [TSV18] completed the approach of [BG05], considered an extensive system of logical errors and also complications. Cembalo et al. [CDU11] visualized the intermediate steps of a SQL query up to the final result. Saatz [Sa17] used an NLP approach to compare stored solutions and student solutions by the Levenshtein-Distance. The RelaX tool of the University of Innsbruck [KTS19] transformed SQL Queries into interactive operator trees, and vice versa, operator trees can be transformed into SQL queries. The e-learning databases portal edb[5] exists since 2008, see [Fa13]. The new version has a SQL trainer, a trainer for normalization of relations, a trainer for SQL triggers, procedures and functions, and multiple-choice questions [RF19, Ra09]. Technically the Java Programming Language was replaced by the JavaScript library node.js, additionally using PHP and an Oracle Database in the background. It is a public web application, and registration is possible with any valid email address.

3 METHOD AND DATA

The decision tree was constructed manually by the error examples in the training set based on a student project work [HKM19]. The leaf nodes are placed regarding the

[5] See https://edb2.gm.th-koeln.de/, visited: 25.03.2020

execution order of the clauses of a select statement (i.e. FROM clause with table list before SELECT list of columns and WHERE clause), the difficulty of the corresponding SELECT type or a significant amount of errors in the training data set. The decision tree is implemented as an Oracle PL/SQL function using regular expressions intensively and was tested on our training set. In the leaves, we have error messages and hints for the user to improve his statement. The leaf nodes, which are nearby the root, are executed first. These error messages and resulting hints are implemented in the SQL trainer of our database e-learning platform edb. The first step was to distinguish semantic (43%) and syntax errors (57%). In the background, we used an Oracle database, and so we decided to use the Oracle error messages for syntax errors.

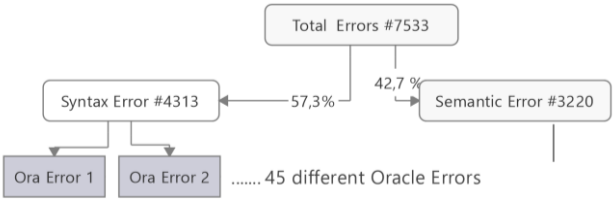

Fig. 1: Syntax errors versus Semantic errors

Compared with Ahadi [Ah16b], we found that the ORACLE syntax errors were more specific than the Postgres errors they used. Most of the errors were misunderstandings of the data model or the correct structure of a simple SQL SELECT statement. Some of the mistakes result from the misunderstanding of the character of the relational database when a set is, for example, compared with a single value (ID 920 and 979).

ID	%	HINT
904	17,8	A column of your query does not exist in the referenced table.
936	11,3	Fundamental syntax error of a select-statement
942	5,7	One of the specified tables or views does not exist.
918	5,5	Please define your columns uniquely by using a table alias!
920	3,7	Incorrect use of comparison operators (<,> =, is) or logical operators (or, and, not)
923	2,9	Missing from-clause
979	2,2	All columns in the select list that do not contain group functions must also be listed in the group by clause.
908	1,1	Operator "is" can only be used with "null"
103	0,1	SQL injection: Other SQL-statements than select are not allowed here

Table 1 Oracle Errors and their frequency of occurring

To distinguish semantic errors, we have two different syntactically correct query results of the wrong statement and the correct statement. Both were executed in the database itself. A user statement is evaluated wrong if the two results differ. As Saatz [Sa17] mentioned, this is only a heuristic method but with 95% accuracy. First, we made a classification of the wrong statements regarding the SELECT-type of the question. Here we expanded the seven SELECT-types of Ahadi [Ah15] to 17 SELECT-types.

Compared with Ahadi [Ah16a], we found similar results. Correlated subqueries, divisions, set operators, recursive SELECTS, WHERE clauses with date column, natural joins, GROUP BY, and ORDER BY clauses have an error quote of larger than 80%. We decided to use the SELECT types with high error quotes and the difficulty of finding errors with text analysis as the first step in our decision tree.

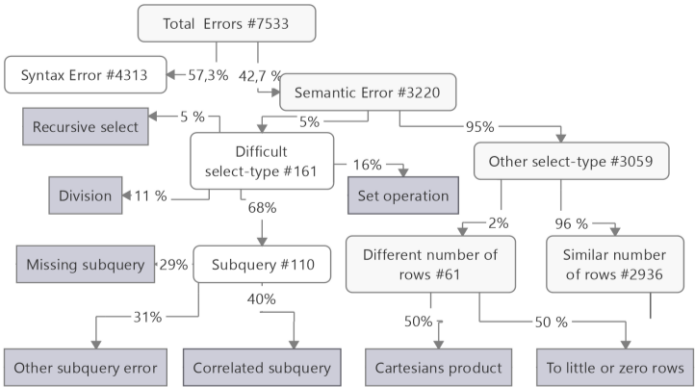

Fig. 2: Decision tree with SELECT types, left side of semantic errors

The next step to construct our decision tree was to make a text analysis with regular expressions to find the errors in comparison to the wrong statement and the correct statement. This is the part under the leaf node "Similar number of rows" in figure 2.

4 Results

Regarding our previous questions from section 1, we have the following results:

Q1: We used a heuristic and manual manner of constructing such a decision tree. We programmed an Oracle PL/SQL-function that implements the decision tree and classifies all errors of our training set for future use.

Q2: From each leaf node of the decision tree, we can derive automatically hints that allow the SQL novice to improve his/her SQL SELECT statements. This is now installed in the public databases e-learning platform edb of our university TH Köln.

Q3: We found that most of the errors were due to insufficient knowledge/studying of the data model regarding the table names, the required columns, and the fundamental statement of a SELECT-Statement.

Q4: The main problem was, that it is difficult for SQL novices to think in sets. No other programming language is using this kind of rather unusual but straightforward kind of mathematical thinking. Therefore, SQL teachers should spend more time on this type of errors in their lectures to enable student to get more acquainted with this type of structure.

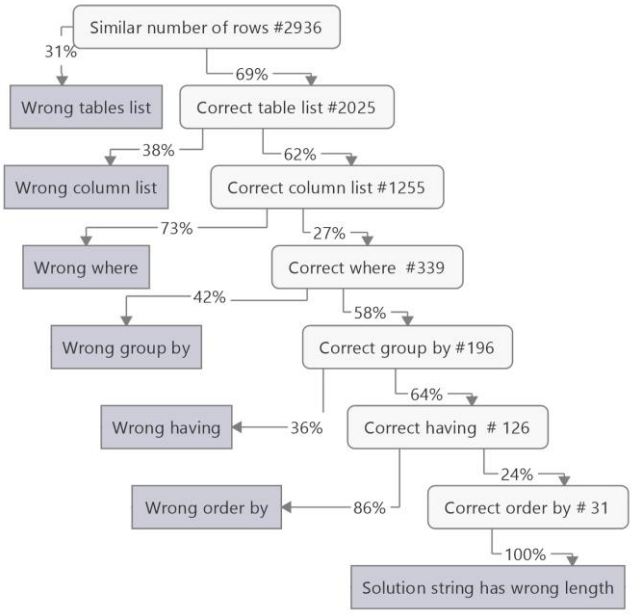

Fig. 3: Decision tree right side for semantic errors

Compared with other works like SQL Tutor of Mitrovic we have a rather simple approach with good results focused on the hints that are generated in our edb platform. In [MO16] about 700 unique constraints were used. In our approach, we needed only 106 leaf nodes and a rather simple program that quickly classifies the wrong solution to an error class and gives the end-user useful hints to avoid those errors.

5 Future Work

One possibility for future work is to sample all errors, not only one error, that are concerned with a specific wrong statement. Then the position of the error in the tree does not influence the final result. We will intensively test our algorithm that implements the decision tree, improving the algorithm, and adding more nodes if necessary. For some examples, the oracle errors are not specific enough. We could make a special consideration for those cases, including text analyses. Additionally, we plan to expand our SQL trainer to other SQL DML statements (INSERT, UPDATE and DELETE) as we have done with stored procedures, triggers, and functions and publish the result as an open-source project. A long version of this article and editorial materials like all errors and hints and the decision tree implementation are available on Researchgate[6].

[6] https://www.researchgate.net/project/edb-das-ELearning-Datenbankportal

Bibliography

[Ah15] Ahadi, A. et al.: A quantitative study of the relative difficulty for novices of writing seven different types of SQL queries. ITiCSE ’15. S. 201–206, 2015.

[Ah16a] Ahadi, A. et al: Students semantic mistakes in writing seven different types of SQL queries., ITiCSE’16, S. 272–277, 2016.

[Ah16b] Ahadi, A. et al.: Students’ Syntactic Mistakes in Writing Seven Different Types of SQL Queries and its Application to Predicting Students’ Success. SIGCSE 2016, S. 401–406, 2016.

[BG05] Brass, S.; Goldberg, C.: Semantic errors in SQL queries: A quite complete list. J. Syst. Softw. 79, 5(2005), S. 630–644, 2005.

[Br10] Brusilovsky, P. et al.: Learning SQL Programming with Interactive Tools. ACM Transactions on Computing Education, 9(4), S. 1–15, 2010.

[CDU11] Cembalo, M.; De Santis, A.; Umberto, F. P.: SAVI: A new system for advanced SQL visualization. SIGITE’11, S. 165–170, 2011.

[Fa13] Faeskorn-Woyke, H., Bertelsmeier, B., Gawenda, D. Kasper, A., 2013. Erfahrungen mit dem kooperativen E-Learning-Datenbankportal edb. In: Breiter, A. Rensing, C. (Hrsg.), DeLFI 2013, (S. 285-288).

[HKM19] Hallweger, M.; Kaczmarczyk, D.; Märtens, F.: Überarbeitung der Fehlerausgabe des SQL Trainers, internal student project, 2019.

[KTS19] Kessler, J.; Tschuggnall, M.; Specht, G.: RelaX: A webbased execution and learning tool for relational algebra, BTW-GI 2019, S. 503–506, 2019.

[KS18] Kleerekoper, A.; Schofield, A.: SQL tester: An online SQL assessment tool and its impact. ITiCSE, S. 87–92, 2018.

[MO16] Mitrovic, A.; Ohlsson, S.: Implementing CBM: SQL-Tutor after Fifteen Years. International Journal of Artificial Intelligence in Education, 26(1), S. 150–159. 2018.

[Pr03] Prior, J. C.: Online Assesm. of SQL Query Form. Skills. Ace ’03, 20, S. 247–256, 2003.

[Ra09] Rakow, T.C. et al.: Tools für die Lehre im Fach Datenbanken. Datenbank-Spektrum 9. Jg., Nr. 29, S. 5-13, 2009.

[RF19] Rakow, T. C.; Faeskorn-Woyke, H.: Digitale Lehre im Fach Datenb. In: Meyer, H., Ritter, N., Thor, A., Nicklas, D., Heuer, A. Klettke, BTW 2019 –GI, S. 97-98, 2019.

[Sa17] Saatz, I. M.: Wo steckt nur der Fehler in der SQL-Anfrage? Semantische Prüfung von Lösungen. CEUR Workshop Proceedings, 2017(Abp), S. 1–9, 2017.

[Sa04] Sadiq, S. et al.: SQLator-An online SQL learning workbench. SIGCSE Bulletin, 36(3), S. 223–227. 2004.

[TSV18] Taipalus, T.; Siponen, M.; Vartiainen, T.: Errors and complications in SQL query formulation. ACM Transactions on Computing Education, 18(3), 2018.