Aalto University

OPEN ACCESS

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Akhuseyinoglu, Kamil; Hardt, Ryan; Barria-Pineda, Jordan; Brusilovsky, Peter; Pollari-Malmi, Kerttu; Sirkiä, Teemu; Malmi, Lauri

A Study of Worked Examples for SQL Programming

Published in:
ITiCSE 2022 - Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education

DOI:
10.1145/3502718.3524813

Published: 07/07/2022

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Akhuseyinoglu, K., Hardt, R., Barria-Pineda, J., Brusilovsky, P., Pollari-Malmi, K., Sirkiä, T., & Malmi, L. (2022). A Study of Worked Examples for SQL Programming. In *ITiCSE 2022 - Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education* (pp. 82-88). ACM. https://doi.org/10.1145/3502718.3524813

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

A Study of Worked Examples for SQL Programming

Kamil Akhuseyinoglu
University of Pittsburgh
Pittsburgh, PA, USA
kaa108@pitt.edu

Ryan Hardt
University of St. Thomas
St. Paul, MN, USA
ryan.hardt@stthomas.edu

Jordan Barria-Pineda
University of Pittsburgh
Pittsburgh, PA, USA
jab464@pitt.edu

Peter Brusilovsky
University of Pittsburgh
Pittsburgh, USA
peterb@pitt.edu

Kerttu Pollari-Malmi
Aalto University
Espoo, Finland
kerttu.pollari-malmi@aalto.fi

Teemu Sirkiä
Aalto University
Espoo, Finland
teemu.sirkia@aalto.fi

Lauri Malmi
Aalto University
Espoo, Finland
lauri.malmi@aalto.fi

ABSTRACT

The paper focuses on a new type of interactive learning content for SQL programming - worked examples of SQL code. While worked examples are popular in learning programming, their application for learning SQL is limited. Using a novel tool for presenting interactive worked examples, Database Query Analyzer (DBQA), we performed a large-scale randomized controlled study assessing the value of worked examples as a new type of practice content in a database course. We report the results of the classroom study examining the usage and the impact of DBQA. Among other aspects, we explored the effect of textual step explanations provided by DBQA.

CCS CONCEPTS

• **Applied computing → Interactive learning environments**; • **Information systems → Structured Query Language**.

KEYWORDS

worked examples, textual explanations, computer science education, OLM, classroom study, SQL

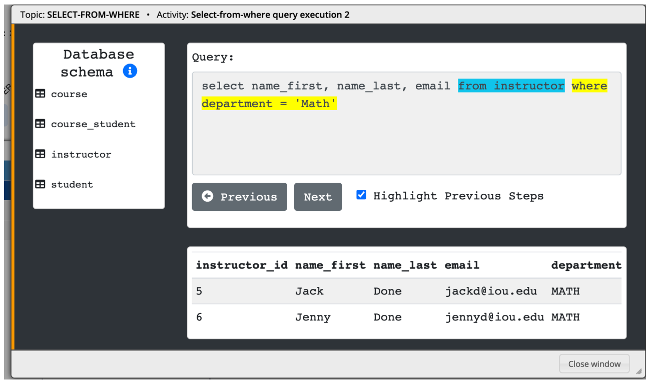
1 INTRODUCTION

Worked examples and problems are two key types of smart learning content for studying programming languages [5]. For learning programming languages like Java or Python, both types of learning content are well represented. However, for learning Structured Query Language (SQL), current work predominantly focuses on SQL problems with automatic assessment [6, 18, 22]. This paper attempts to bridge this gap by introducing and evaluating Database Query Analyzer (DBQA) [13], an online learning tool for SQL worked examples that interactively illustrates and explains the step-by-step execution of SQL *SELECT* statements. We report the results of a large-scale randomized control study where DBQA examples were used by learners among other types of interactive learning content through an online SQL Practice System. In our analysis, we focus on the usage and the impact of DBQA, as well as the effect of textual step explanations provided by DBQA.

2 RELATED WORK

Over the last 30 years, worked examples, also referred to as worked-out examples [2], have gradually emerged as an important instructional approach supported by learning technology. Worked examples are comprised of the presentation of a problem, the solution steps, and the final solution. Students use these examples as models of how to solve certain types of problems. A sizable body of research on the use of worked examples for acquiring cognitive skills has been accumulated in various domains, such as mathematics and physics [2, 27]. This research has consistently shown that in the early stages of skill acquisition when students typically have little or no domain knowledge, instruction that relies more heavily on studying worked examples is more effective for learning than the traditional approach of being focused on only problem-solving. Previous work has shown that early example-based instruction leads to better learning outcomes, which are reached in less time and with less effort [16, 28].

In the domain of learning programming, the majority of research on worked examples has focused on *animated code examples*. Animated examples do not limit the example presentation to just showing students the code of example programs, but allow the



```
select name_first, name_last, email from instructor where
department = 'Math'
```

instructor_id	name_first	name_last	email	department
5	Jack	Done	jackd@iou.edu	MATH
6	Jenny	Done	jennyd@iou.edu	MATH

Figure 1: An SQL *SELECT* example presented in DBQA *without* the step explanation support. The yellow color highlights the currently executed step in the query text while the blue-color highlights previously executed steps if any.

students to see the code executed step-by-step using *program visualization* [26]. The use of step-wise program visualization enables students to see the internal state and intermediate results or program execution, which are usually hidden [11, 19, 25], and thus better understand the behavior of program constructs [29]. More recently, there have been several attempts to develop other types of worked examples that focus on code exploration rather than code animation [8, 31]. However, direct comparison of other types of worked examples with animated examples [15] has indicated higher effectiveness of animated examples.

The use of worked examples in the domain of SQL programming is much less explored. While static code examples were found effective in combination with problem-focused SQL-Tutor [23] and interactive annotated examples were a popular component of Database Exploratorium [6], we were not able to find any studies of SQL code examples that visualize step-by-step execution of SQL code.

3 PRESENTING SQL EXAMPLES WITH DBQA

In this study, SQL code examples were illustrated using *Database Query Analyzer* (DBQA) [13], a learning tool that uses data-oriented visualizations to illustrate the effects that clauses and conditions have on SQL *SELECT* statements. DBQA illustrates intermediate data sets similar to those maintained by the database management system during the processing of the query (Figure 1). Upon query submission, DBQA processes clauses in an SQL *SELECT* statement in the following order (if present): `FROM`, `WHERE`, `SELECT`, `GROUP BY`, `HAVING`, and `ORDER BY`. For each clause and condition, DBQA highlights the query component being processed and updates the intermediate data set displayed to the student accordingly. “Next” and “Previous” buttons allow the student to step forward and backward in the query execution, allowing for procedural-like query execution and illustration. Students can highlight already executed clauses by checking the “highlight previous steps" checkbox. Additional features in DBQA allow the learner to view all tables, columns, and keys in a database, as well as view all data in those tables.

DBQA can be used as a learning tool as a stand-alone system allowing students to explore custom queries that they want to execute on a database, or as an interoperable service [12] that can be integrated with other systems as a provider of reusable smart learning content through the LTI protocol, which was the case for this study. DBQA was integrated into the SQL Practice System (presented in Section 4.1) as a tool to deliver interactive worked examples for SQL *SELECT* statements that cover multiple SQL topics such as subqueries and aggregate operators. For example, given the following query (Figure 1):

```
SELECT name_first, name_last, email
FROM instructor
WHERE department = 'Math'
```

DBQA would present the following steps to the student: (1) highlight the `FROM` clause and display all of the data in the instructor table, (2) highlight the condition in the `WHERE` clause and remove all rows in the intermediate data set that do not match the specified condition, and (3) highlight the `SELECT` clause and remove all columns in the intermediate data set that are not in the select list. Students should explore the three query step executions presented above to fully explore this particular SQL example.

To better convey the modifications on the intermediate data set by each query clause and condition, text-based query step explanations presented in Table 1 were added to DBQA for the purpose of this study. Figure 2D shows an instance where the textual explanation “Retrieve all rows from: course” was shown as a yellow-marked text for the current execution step `FROM course`.

Clause	Explanation(s)
`SELECT`	“Remove all columns other than...” “Retain all columns from...”
`FROM`	“Retrieve all rows from...” “Retrieve all rows from the Cartesian product of...”
`JOIN`	“Add columns from *table* that satisfy the following condition...”
`WHERE` `HAVING`	“Remove all rows other than those that satisfy the following column condition...”
`GROUP BY`	“Combine rows (according to the grouping function) with matching values in the following column(s)...”
`ORDER BY`	“Sort all rows according to values in the following column(s)...”

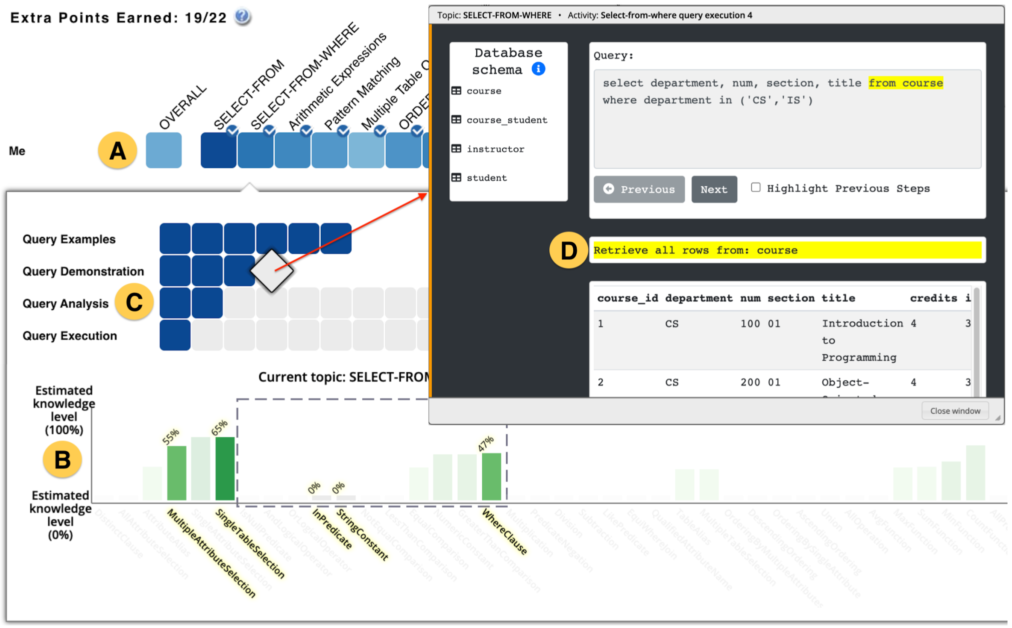
Table 1: DBQA query step explanations.

4 CLASSROOM STUDY

We evaluated the impact of the SQL example tool in a large-scale empirical study where the tool was made available as a component of an SQL Practice System. In this section, we describe the course context, study design, and relevant parts of the practice system.

4.1 SQL Practice System

4.1.1 Open Learner Model Interface. The goal of the SQL Practice System is to provide access to four types of interactive learning content that can help students to practice SQL topics. Since the system was designed to be used in a free mode, i.e., students can choose when and how to practice, the interface of the practice system was designed to help students with monitoring their progress and choosing the most appropriate content to practice. The access to all four types of interactive learning content is organized by topics and augmented by an open learner model (OLM) [4, 9].



```
select department, num, section, title from course
where department in ('CS','IS')
```

Retrieve all rows from: course

course_id	department	num	section	title	credits	i
1	CS	100	01	Introduction to Programming	4	3
2	CS	200	01	Object-	4	3

Figure 2: The interface of the SQL Practice System. The topic-based (A) and concept-based OLM (B). The list of available learning contents for the topic *SELECT-FROM-WHERE* is shown and grouped under four content types (C). A DBQA activity accessed by a student in *textual-DBQA* condition and a query execution step is shown with textual explanation for *FROM* clause(D).

The goal of OLM is to increase students' awareness of their learning progress. As seen in Figure 2, there is a topic-based (A) and concept-based (B) OLM. The grid (Fig. 2A me row) represents the average progress of a learner for each topic, which increases by completing activities within a topic (Fig. 2C). The darker color of the cell indicates a higher level of completion in a topic or an activity. In parallel, as the student works with the problem-solving activities (i.e., Query Analysis and Query Execution), estimations of their knowledge level on the different SQL concepts are calculated based on their correct/incorrect attempts [30]. These estimations are shown through the length and the color of the concept bar chart shown in Fig. 2B, where each bar represents an SQL concept covered in the course. The longer and the greener the bar, the higher the estimated knowledge of this SQL concept. Students can explore the OLM by (1) mousing over the topics to get an overview of the concepts that are introduced in each of these topics, or also by (2) mousing over the learning activities to know which concepts can be practiced through each activity.

4.1.2 Practice Content. Figure 2 shows four types of interactive content available in the SQL Practice System. *Query Examples* are *annotated* SQL examples served through the WebEx tool [8], which allow students to interactively examine textual annotations for each SQL clause and condition, line by line. From the *Query Demonstration* row, students can access worked-out examples presented through DBQA. *Query Analysis* row provides access to SQL problems served through SQL-Tutor [22], a constraint-based tutor for teaching SQL, where students have a structured SQL select query template to answer a given SQL problem. SQL-Tutor provides multiple support and feedback options such that students can see the errors associated with their queries based on well-defined constraints, and even ask for a partial solution. Finally, *Query Execution* activities are problems that assess learners' SQL knowledge and are solved by writing free-form SQL select statements for a given problem text. The submitted query is evaluated against a model solution using a sample database, and immediate correct/incorrect feedback is provided. These problems are served through *SQL Assessment Tool (SAT)* [6] and generated through pre-defined *templates.* Every time a student accesses a problem, the problem text is randomly selected from a predefined problem set. SAT has another unique feature, *query build mode* that allows learners to run their SQL queries multiple times and lets them see the actual query result before submitting it as an answer for assessment. Attempts to both SQL-Tutor and SAT are used by the student modeling component to update concept-level knowledge estimates. In this study, students had access to 63 annotated examples, 42 DBQA examples, 52 SQL-Tutor problems, and 55 SAT problems through the practice system.

4.2 Study Context

The study was conducted from February to May 2021 on an undergraduate database management course at a major research university in Finland. The course is compulsory in two majors: Computer science, and Industrial engineering and management. It is also very popular among students from other bachelor's and master's programs, with 587 students starting the course in the spring of 2021. The course content covers relational modeling, relational algebra, UML modeling, SQL, and transaction management. The course material and related exercises were presented using the local course management system (CMS)[17]. To pass the course, students have to take a compulsory exam and implement a team project in which a small database is designed and implemented using SQLlite.

Two groups of learning and assessment tools were provided for students to gain and practice their database knowledge: CMS exercises and SQL Practice System. Neither of the tools was mandatory for passing the class, but their use was incentivized through the mechanism of *exercise points* that play an important role in course grading. By collecting exercise points, students can improve their (passing) exam grade by up to 1.5 grades on a scale of 0-5, where 1 denotes a pass and 5 is the best grade. The opportunities to earn exercise points differed considerably between CMS exercises and the SQL Practice System, reflecting their role in the course. CMS exercises on various course topics were designed more as an assessment tool. Each solved exercise earned exercise points and was the means by which students could earn most of their points. In contrast, SQL Practice System was offered for additional practice for students who needed more support to gain SQL knowledge. It was accessible through a link from the CMS, and the incentives to use it were much weaker to avoid "practicing for points" rather than knowledge. While the practice system offered a large number of problems and worked examples for 11 SQL topics, only up to 22 exercise points (about 8.4% of the maximum available exercise points) could be earned through this system. More precisely, students could earn up to 2 points per topic by solving two SQL problems (1 for each type), and no points could be earned by working with examples. The status of the earned points was visible to students as shown in the top-left corner of Figure 2 (e.g., *Extra Points Earned: 5/22*).

4.3 Study Design and Conditions

While the main goal of the study was to assess the DBQA tool in a realistic context, we also used it as an opportunity to find some optimal settings for the tool, namely, the presence of a concept-based open learner model (OLM) and the presence of textual explanations in the step-wise query execution. In our past work with other programming languages, we found that both concept-based OLM [10] and explanatory visualization [20] could make example exploration more valuable. We hoped that they could also be helpful in the context of learning SQL. To explore these two aspects, the study was designed as a two-by-two randomized control experiment with the following conditions:

OLM Conditions:

(1) Topic-based OLM only (*topic-OLM*)
(2) Topic-based OLM and concept-based OLM (*concept-OLM*)

DBQA Conditions:

(1) Visual query execution steps (*visual-DBQA*)
(2) Visual query execution steps and textual query step explanations (*textual-DBQA*)

Students who were assigned to the *concept-OLM* condition used the SQL Practice System version shown in Figure 2A, and those who were assigned to *the topic-OLM* condition could only access the topic-based OLM without concept-level knowledge estimations. Similarly, students in the *textual-DBQA* condition explored worked examples in DBQA with textual support whereas the *visual-DBQA* condition had access to other visual demonstrations without the textual support. Students were assigned randomly to each of the four study conditions. Prior to the experiment, we hypothesized that augmenting query execution illustrations with textual explanations might improve the learners' understanding of visualization and raise learners' eagerness to explore more execution steps [20, 25]. Similarly, we expected that the concept-based OLM would provide additional navigational support and could potentially increase student engagement with the learning tools [7, 9].

4.4 Dataset and Overall System Usage

The dataset includes the interaction logs collected through the SQL Practice System. To assess the effectiveness of the practice system and its components, we also administered pre- and post-tests. The pre-test was administered before students started working with the SQL exercises. The post-test was administered at the end of the semester. Both tests had 10 problems, including 5 multiple-choice and 5 SQL *fill-in-the-blank* problems, which concentrated on SQL *SELECT* statements. Pre and post-test problems were isomorphic [1].

At the beginning of the course, we informed students about this research and asked them to give their consent for participation. In this paper, we considered only data from students who gave their consent. Table 2 presents the summary statistics about the practice system usage for students who attempted at least one problem and viewed at least one example (N=131). Specifically, students solved on average 28 unique problems in total considering both SQL-Tutor and SAT, which is beyond the limit for earning the extra exercise points in full (22 problems), and almost half of the students exceeded this limit. Beyond problems, students were also engaged with both types of worked-out examples and viewed on average 21 unique examples in total. It is evident that the students practiced with the SQL Practice System substantially and allowed us to investigate further both DBQA usage and effects of study conditions.

Table 2: Summary statistics for the practice system usage of the learning activities by students who attempted at least one activity (N=131)

Content	Usage Metric	Mean	SD
Annotated Examples	Distinct accesses	14.2	13.3
	Annotated line views	38.9	56.0
DBQA	Distinct accesses	7.0	10.3
	Execution step views	25.3	45.6
SQL-Tutor	Problem solving attempts	46.1	34.9
	Successful attempts	13.7	10.8
	Distinct problems solved	12.9	10.1
SQL Assessment Tool	Problem solving attempts	31.0	21.2
	Successful attempts	16.1	13.9
	Distinct problems solved	15.1	10.5
	Attempts in *query build mode*	17.5	24.1

5 RESULTS

In this section, we present our results starting from a broader perspective and later share our findings from deeper analyses. First, we explore if practicing with the SQL Practice System is associated with student learning (Section 5.1). Then, we investigate the possible impact of practicing with worked examples on problem-solving activities within the practice system (Section 5.2). Next, we

[1]There were no significant differences in pre/post-test scores between study conditions.

inspect the effects of randomized experiment conditions (i.e., OLM-condition and DBQA-condition) on the usage of DBQA (Section 5.3). Finally, we analyze the effect of textual step explanations on DBQA usage over the course period (Section 5.4).

5.1 External Value of Practice System Usage

We started our analysis by checking the relationship between the practice system usage and student learning. Following our prior research findings [3], we concentrated on success rate related features. We considered using the overall success rate and first-attempt success rate in both the SQL assessment tool (SAT) and SQL-Tutor as features. To test such connections, we can only consider students who had both pre/post-test scores and used the SAT and SQL-Tutor at least once (N=110). We performed a backward step-wise feature selection process to pick the important usage metrics. This process picked the overall success rate in SAT and pre-test scores as features. Then, we fitted a linear regression model to predict post-test scores using these features. We found a significant model ($F(2, 107) = 14.8,\ adj.R^2 = .20,\ p < .001$) and the results indicated that the overall success rate was a statistically significant predictor ($B = 7.50, t = 3.433, p < .001, \eta_p^2 = .10$) of post-test scores after controlling for pre-test scores ($B = 9.15, t = 4.189, p < .001, \eta_p^2 = .14$), such that a 10% increase in success rate was associated with a 4.8% increase in post-test scores. We believe that given the intelligent-tutoring-system nature of the SQL-Tutor and multiple feedback/hint options (e.g., partial solution request), the success rate in SQL-Tutor problems was not a significant factor in this analysis. Following these findings, we concentrated more on success rate related metrics in SAT in the following analyses to assess the value of the worked-out example tools.

5.2 Internal Value of Worked Examples

Following the prior work on worked examples [14, 15], we extended our analysis to see if practice with worked examples was associated with performance in problem-solving activities, specifically the success rate in SAT (see the previous section). As we discussed in Section 4.4, students used the practice system substantially. However, some students decided not to use some of the available learning tools. For this reason, we only considered students who practiced with all types of learning content (N=91) to assess the value of worked examples.

We have extracted multiple usage metrics for each example tool to summarize learners' engagement levels. But, since these metrics for each example tool were highly correlated (i.e., Pearson's r > .8), we can only use one for each example tool in our regression models. Thus, we selected the number of execution step views for *DBQA* and the number of annotated line views for *Annotated Examples* to summarize the learners' engagement with the example tools.

We first fitted a linear regression model to predict the overall success rate in SAT using the features mentioned earlier. The regression model was statistically significant ($F(2, 88 = 3.65)$, $adj.R^2 = .06,\ p = .030$)[2]: viewing more execution steps in DBQA was significantly and positively associated with higher success rate in SAT problems ($B = .07, t = 2.212, p = .029, \eta_p^2 = .07$). Viewing more annotated lines was not significant ($B = -.03,\ t = -.978,\ p = .331$).

[2]The pre-test scores did not improve the model fit and were removed.

In our prior work [1], we found out that learners exhibited two divergent practice behaviors in a similar SQL Practice System. Some students tended to learn by exploring worked-out examples and then solving SQL problems. Others preferred to solve problems through multiple attempts and by executing SQL queries in the *execution mode* for exploration and debugging purposes. Hence, we further hypothesized that exploring worked examples might affect students' problem-solving behavior in this study as well. To test our hypothesis, we fitted negative binomial generalized linear models (due to over-dispersion) to predict the number of attempts using the *query build mode* in SAT. We controlled for prior knowledge and the number of distinct SQL problems attempted (i.e., students who practiced with more problems might need to use query build mode more). The model that used the number of execution steps viewed in DBQA as a feature improved the overall fit when compared to a model that did not use this feature ($\chi^2(1) = 4.870, p = .027$). However, the model that used the number of line views in Annotated Examples (AE) did not improve the overall fit ($\chi^2(1) = .099, p = .753$). To summarize, students who explored more execution steps in DBQA used the query build mode in SAT significantly fewer times, suggesting that they used DBQA to understand how SQL queries work and needed fewer query building attempts for debugging and exploring their mistakes.

In the learning process, the persistence in problem-solving activities (i.e., not giving up after failing to solve a problem but keep trying to solve it) is constantly associated with better learning outcomes [21, 24]. In light of these research findings, we explored if work with examples affected student persistence in solving SQL problems in SAT. We calculated the ratio of problems that students abandoned (*abandon-ratio*) after their first failed attempt. Then, we fitted a linear regression model to predict *abandon-ratio* and added the number of execution step views in DBQA and line views in AE as features. The model was statistically significant ($F(2, 88) = 5.427, p = .006$). Both features related to example tools were significant predictors: viewing more execution steps was associated with lower *abandon-ratio* ($B = -.05, t = -3.133, p = .002, \eta_p^2 = .07$) while viewing more annotated lines was associated with higher *abandon-ratio* ($B = .03, t = 2.042, p = .044, \eta_p^2 = .05$). These findings suggest that DBQA might help students to be more persistent during problem-solving whereas AE might not be that beneficial.

5.3 Effect of Textual Query Step Explanations

In the preceding sections, we reported the benefit of the SQL Practice System and the worked-out example learning content on learning and problem-solving performance. As summarized in Section 4.3, in this study we had two by two study designs, i.e., OLM (topic-OLM/concept-OLM) and DBQA (textual/visual) conditions. In this section, we present the effect of these randomized experiment conditions on the usage of DBQA. We used the same set of learners from Section 5.2 (N=91).

We checked the effect of both OLM and DBQA conditions by conducting two separate two-way ANOVAs on the ratio of fully explored DBQA examples (i.e., exploring all the execution steps in a DBQA example) to the number of examples that at least one execution step was viewed (*exploration-ratio*); and the average number of explored execution steps per unique DBQA example, which is calculated by dividing the total number of explored steps by the total number of accessed DBQA examples. We also added the interaction effect of study conditions. We did not add the pre-test scores as the covariate due to the missing linear relationship between pre-test scores and the dependent variables.

ANOVA on exploration ratio found a significant interaction effect between study conditions ($F(1, 87) = 5.172, p = .025$) with small-to-medium effect size ($\eta_p^2 = .06$), and marginal main effect of OLM type ($F(1, 87) = 3.252, p = .075$), but no significant main effect of step explanations ($F(1, 87) = 2.434, p = .122$). Similarly, ANOVA on the average execution steps revealed a significant interaction effect ($F(1, 87) = 7.357$, $p = .008$) with small-to-medium effect size ($\eta_p^2 = .08$), but no main effect of OLM type ($F(1, 87) = .197$, $p = .659$) and of step explanations ($F(1, 87) = .938, p = .336$). As shown in Figure 3, post-hoc analyses on average step executions revealed that students who received textual step explanations explored significantly more steps ($M = 3.35$) compared to students who did not have access to these explanations ($M = 2.32$), but only in concept-based OLM condition ($F(1, 87) = 6.6, p = .012$). A similar conclusion was reached from the post-hoc analysis of the exploration ratio. Thus, the effect of textual explanations on both *average step executions* and exploration ratio depends on the type of OLM visualization. To summarize, students who had access to textual step execution explanations tended to view more steps on average and fully explore a higher number of examples. However, such an effect only exists when students had access to fine-grained OLM design.

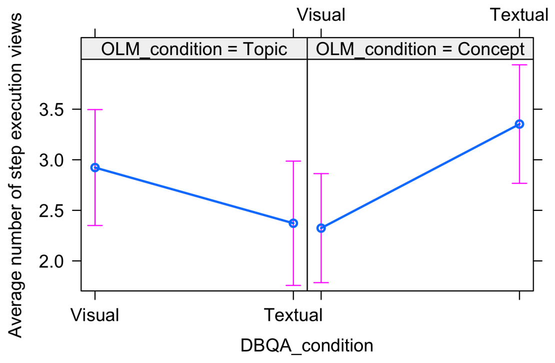


Figure 3: The average number of explored query execution steps by study conditions (marginal effects) in the ANOVA model. Purple bars denote 95% confidence interval.

5.4 Time-based Effect of Step Explanations

Following the interaction effect between study conditions on DBQA usage metrics, it is clear that students were engaged with DBQA at different levels of intensity, but we do not know if this was the case during the whole course period. Hence, in this section, we explore the effect of textual explanations throughout the course. We computed the average number of execution step views per week per student for the concept-based OLM condition. As shown in Figure 4, students in textual explanation condition worked with DBQA much more at the beginning of the semester (i.e., during the first two weeks). Then, this gap between conditions disappeared for the 3rd week of the semester. After the 3rd week, students engaged with DBQA examples consistently more when represented with textual explanations. To confirm our visual inspection, we further conducted an ANOVA on the average step count with week number and DBQA condition as factors. The ANOVA revealed a significant main effect of the DBQA condition ($F(1, 13) = 8.963, p = .010$) with a large effect size ($\eta_p^2 = .408$) but the week number was not significant ($F(1, 13) = .478, p = .501$).

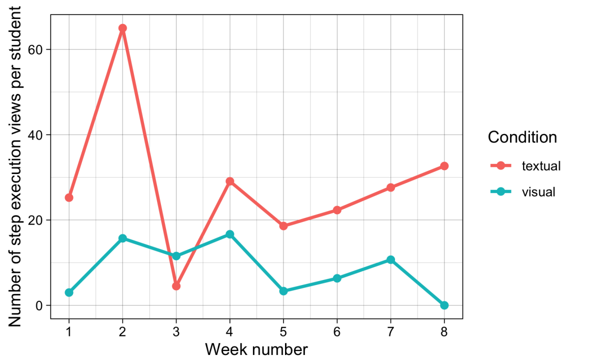


Figure 4: The number of DBQA step-execution views per student per week for textual and visual DBQA conditions.

6 CONCLUSION

We presented an online learning tool for SQL programming that illustrates the step-by-step execution of SQL SELECT statements. The tool can also provide textual explanations for the SQL clauses and conditions. In this study, we evaluated DBQA in a large-scale randomized control experiment as a part of an online non-mandatory SQL Practice System with multiple open learner model features.

The findings showed that when working with the SQL Practice System, the success rate in the SQL Assessment Tool (SAT) was associated with higher post-test scores even after controlling for the pre-test scores. After observing a positive connection between the system usage and learning outcomes, we evaluated the impact of worked examples on problem-solving activities. Exploring more query execution steps in DBQA was associated with a higher success rate, fewer query building attempts, and higher persistence during problem-solving in SAT. Finally, we explored the effect of textual explanations and open learner model features on the usage of DBQA and found out that students who accessed textual explanations viewed more steps on average in DBQA and fully explored a higher number of examples. However, this effect was only observed for students who used the concept-based OLM design.

While bringing some interesting and promising results, our study had limitations. The main limitation originates from our intention to assess a tool in a realistic context - a semester-long classroom study. Even if we checked the effect of prior knowledge on our analyses, we could not control for other factors that might influence learners' usage behavior such as learning through other resources. Finally, given that the majority of the practice system usage was non-mandatory, our results might be subject to self-selection bias.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1740775.

REFERENCES

[1] Kamil Akhuseyinoglu and Peter Brusilovsky. 2022. Exploring Behavioral Patterns for Data-Driven Modeling of Learners' Individual Differences. *Frontiers in Artificial Intelligence* 5 (2022). https://doi.org/10.3389/frai.2022.807320

[2] Robert K Atkinson, Sharon J Derry, Alexander Renkl, and Donald Wortham. 2000. Learning from examples: Instructional principles from the worked examples research. *Review of educational research* 70, 2 (2000), 181–214.

[3] Jordan Barria-Pineda, Kamil Akhuseyinoglu, Stefan Želem-Ćelap, Peter Brusilovsky, Aleksandra Klasnja Milicevic, and Mirjana Ivanovic. 2021. Explainable Recommendations in a Personalized Programming Practice System. In *Artificial Intelligence in Education*, Ido Roll, Danielle McNamara, Sergey Sosnovsky, Rose Luckin, and Vania Dimitrova (Eds.). Springer International Publishing, Cham, 64–76.

[4] Jordan Barria-Pineda, Julio Guerra-Hollstein, and Peter Brusilovsky. 2018. A Fine-Grained Open Learner Model for an Introductory Programming Course. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization* (Singapore, Singapore) *(UMAP '18)*. Association for Computing Machinery, New York, NY, USA, 53–61. https://doi.org/10.1145/3209219.3209242

[5] Peter Brusilovsky, Stephen Edwards, Amruth Kumar, Lauri Malmi, Luciana Benotti, Duane Buck, Petri Ihantola, Rikki Prince, Teemu Sirkiä, Sergey Sosnovsky, Jaime Urquiza, Arto Vihavainen, and Michael Wollowski. 2014. Increasing Adoption of Smart Learning Content for Computer Science Education. In *Proceedings of the Working Group Reports of the 2014 on Innovation and Technology in Computer Science Education Conference*. ACM, 31–57.

[6] Peter Brusilovsky, Sergey Sosnovsky, Danielle Lee, Michael Yudelson, Vladimir Zadorozhny, and Xin Zhou. 2010. Learning SQL programming with interactive tools: from integration to personalization. *ACM Transactions on Computing Education* 9, 4 (2010), Article No. 19, pp. 1–15. https://doi.org/10.1145.1656255.1656257

[7] Peter Brusilovsky, Sergey Sosnovsky, and Michael Yudelson. 2009. Addictive links: The motivational value of adaptive link annotation. *New Review of Hypermedia and Multimedia* 15, 1 (2009), 97–118. http://dx.doi.org/10.1080/13614560902803570

[8] Peter Brusilovsky and Michael Yudelson. 2008. From WebEx to NavEx: Interactive Access to Annotated Program Examples. *Proc. IEEE* 96, 6 (2008), 990–999.

[9] Susan Bull and Judy Kay. 2010. Open Learner Models. In *Advances in Intelligent Tutoring Systems*, Roger Nkambou, Jacqueline Bourdeau, and Riichiro Mizoguchi (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 301–322.

[10] Julio Guerra-Hollstein, Jordan Barria-Pineda, Christian Schunn, Susan Bull, and Peter Brusilovsky. 2017. Fine-Grained Open Learner Models: Complexity Versus Support. In *25th Conference on User Modeling, Adaptation and Personalization*. ACM, 41–49. https://doi.org/10.1145/3079628.3079682

[11] Philip Guo. 2013. Online python tutor: embeddable web-based program visualization for cs education. In *the 44th ACM technical symposium on Computer Science Education (SIGCSE 2016)*. ACM Press, 579–584. https://doi.org/10.1145/2839509.2844639

[12] Ryan Hardt and Kamil Akhuseyinoglu. 2020. Database Query Analyzer Integration. In *Proceedings of SPLICE 2020 Workshop co-located with 7th ACM Conference on Learning at Scale*.

[13] Ryan Hardt and Esther Gutzmer. 2017. Database Query Analyzer (DBQA): A Data-Oriented SQL Clause Visualization Tool. In *Proceedings of the 18th Annual Conference on Information Technology Education* (Rochester, New York, USA) *(SIGITE '17)*. Association for Computing Machinery, New York, NY, USA, 147–152. https://doi.org/10.1145/3125659.3125688

[14] Roya Hosseini, Kamil Akhuseyinoglu, Peter Brusilovsky, Lauri Malmi, Kerttu Pollari-Malmi, Christian Schunn, and Teemu Sirkiä. 2020. Improving Engagement in Program Construction Examples for Learning Python Programming. *International Journal of Artificial Intelligence in Education* (2020).

[15] Roya Hosseini, Teemu Sirkiä, Julio Guerra, Peter Brusilovsky, and Lauri Malmi. 2016. Animated Examples as a Practice Content in Java Programming Course. In *the 47th ACM technical symposium on Computer Science Education (SIGCSE 2016)*. ACM Press, 540–545. https://doi.org/10.1145/2839509.2844639

[16] Slava Kalyuga, Paul Chandler, Juhani Tuovinen, and John Sweller. 2001. When Problem Solving Is Superior to Studying Worked Examples. *Journal of Educational Psychology* 93, 3 (2001), 579–588.

[17] Ville Karavirta, Petri Ihantola, and Teemu Koskinen. 2013. Service-oriented approach to improve interoperability of e-learning systems. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*. IEEE, 341–345.

[18] Claire Kenny and Claus Pahl. 2005. Automated tutoring for a database skills training environments. In *36th SIGCSE Technical Symposium on Computer Science Education*. 58–62.

[19] Ronit Ben-Bassat Levy, Mordechai Ben-Ari, and Pekka A. Uronen. 2003. The jeliot 2000 Program Animation System. *Computers and Education* 40, 1 (2003), 1–15.

[20] Tomasz Loboda and Peter Brusilovsky. 2010. User-Adaptive Explanatory Program Visualization: Evaluation and Insights from Eye Movements. *User Modeling and User-Adapted Interaction* 20, 3 (2010), 191–226. https://doi.org/10.1007/s11257-010-9077-1

[21] Cristie McClendon, Robin Massey Neugebauer, and Amanda King. 2017. Grit, Growth Mindset, and Deliberate Practice in Online Learning. *Journal of Instructional Research* 8 (2017), 8–17.

[22] Antonija Mitrovic and Kurt Hausler. 2000. An Intelligent SQL Tutor on the Web. In *International Journal of Artificial Intelligence in Education*. 37–44.

[23] Amir Shareghi Najar, Antonija Mitrovic, and Bruce M. McLaren. 2016. Learning with intelligent tutors and worked examples: selecting learning activities adaptively leads to better learning outcomes than a fixed curriculum. *User Modeling and User-Adapted Interaction* 26, 5 (2016), 459–491. https://doi.org/10.1007/s11257-016-9181-y

[24] Juan Pinto, Yingbin Zhang, Luc Paquette, and Aysa Fan. 2021. Investigating Elements of Student Persistence in an Introductory Computer Science Course.. In *5th Educational Data Mining in Computer Science Education (CSEDM) Workshop at EDM2021*.

[25] Teemu Sirkiä. 2018. Jsvee & Kelmu: Creating and tailoring program animations for computing education. *Journal of Software: Evolution and Process* 30, 2 (2018). https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1924

[26] Juha Sorva, Ville Karavirta, and Lauri Malmi. 2013. A Review of Generic Program Visualization Systems for Introductory Programming Education. *ACM Transactions on Computing Education* 13, 4 (2013). https://doi.org/10.1145/2490822

[27] John Sweller and Graham A Cooper. 1985. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and instruction* 2, 1 (1985), 59–89.

[28] John Sweller, Jeroen JG Van Merrienboer, and Fred GWC Paas. 1998. Cognitive architecture and instructional design. *Educational psychology review* 10, 3 (1998), 251–296.

[29] Jaime Urquiza-Fuentes and J. Ángel Velázquez-Iturbide. 2009. A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems. *The ACM Transactions on Computing Education* 9, 2 (2009), Article No. 9.

[30] Michael Yudelson, Peter Brusilovsky, and Vladimir Zadorozhny. 2007. A User Modeling Server for Contemporary Adaptive Hypermedia: An Evaluation of the Push Approach to Evidence Propagation. In *Proceedings of the 11th International Conference on User Modeling (UM '07)*, Cristina Conati, Kathleen McCoy, and Georgios Paliouras (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 27–36.

[31] Rui Zhi, Thomas W. Price, Samiha Marwan, Alexandra Milliken, Tiffany Barnes, and Min Chi. 2019. Exploring the Impact of Worked Examples in a Novice Programming Environment. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. ACM, 98–104. http://doi.acm.org/10.1145/3287324.3287385