ScaffoldSQL: Using Parson's Problems to Support Database Pedagogy

Otto Borchert
Missouri Southern State University
borchert-o@mssu.edu

Gursimran Singh Walia
Georgia Southern University
gwalia@georgiasouthern.edu

Abstract

This paper examines ScaffoldSQL, an interactive tool for helping students learn SQL through a system of interactive scaffolded exercises using Parson's problems. In the system, students are posed with a problem to solve using SQL. They start by attempting to answer the question using free-form text. If they get the problem wrong, they can use a Parson's problem interface to simplify the problem. After completing the problem, students are given one of two "secret words," which allows instructors to track student progress without the need to install anything beyond their typical LMS. The system is designed to help instructors of flipped classrooms identify students who are struggling early, while simultaneously providing immediate feedback for students as they are learning. The system also provides tools for content creation and data gathering for research and development purposes.

1. Introduction

SQL (Structured Query Language) is an ANSI-specified declarative programming language used to build, modify, and retrieve information from structured databases [1]. SQL is currently one of the most popular programming languages in the world [2], yet research shows that students struggle learning SQL concepts [3, 4, 5, 6, 7, 8]. While teaching paradigms have been investigated for object-oriented programming languages (e.g., Java, Python), there has not been as much work done on teaching students how to understand and write SQL, a declarative language.

Based on published reports from interviews of hiring managers and software developers, "most fresh grads will have none to very basic SQL skills" [9]. Research reports also indicate that recent CS/SE graduates have difficulty interacting with databases and tying in the code meant to interface with the database, difficulties even creating and designing databases [10, 11]. This implies that current teaching methods, tools, and pedagogies are not addressing database concepts in an acceptable fashion. To address this, we propose a novel pedagogical tool called ScaffoldSQL, which combines flipped classroom instruction with scaffolding, Parson's problems, and automated test cases in a single-tier database application.

One recent teaching paradigm that has gained popularity is the flipped classroom [12]. In this approach, students watch lectures outside of class and do more complex homework in class. When working outside of class in the flipped approach, students do not have immediate access to the instructor. However, research shows that scaffolded instruction and immediate feedback works best for learning.

Scaffolded instruction is "the systematic sequencing of prompted content, materials, tasks, and teacher and peer support to optimize learning" [13]. Immediate feedback provides students with answers to their questions as they are asking them. This technique has been shown to provide learning benefits in artificial language learning environments [14], among others.

Parson's problems [15] provides a unique method for immediate feedback and scaffolding. In a Parson's problem, students are asked to solve a programming problem. Rather than typing out a full computer program - potentially taxing a novice student's cognitive load - students click and drag a series of lines of code, creating the code line-by-line. Distractors can be added to increase the difficulty level. Most importantly, the problems can be auto-graded, so that students get immediate feedback about what they did right and wrong.

Parson's problems are typically compared to full code writing exercises, where a student is posed with a problem and must write the code free-form and from scratch. Parson's problems take less time to complete and are shown to provide the same learning performance and student retention levels as a code writing exercise in Python [16].

Another option for providing immediate feedback for flipped classroom situations is auto-graded free-form assignments using test cases. In this method, instructors begin by creating a problem they would like students to solve, then creating a series of test cases to determine if the student successfully solved the problem or not. Auto-grading tools like *Gradescope* and *Web-CAT* utilize this method and have shown to support student learning [17, 18].

While flipped classrooms have been deployed in SQL classrooms [19, 20], tools that require less instructor overhead are needed for widespread adoption. One way to reduce instructor overhead is by focusing on tool architecture. SQL is typically deployed using one of three tiers of architecture [21]. In a three-tier system, a client connects to a middleware layer that connects to a server. This provides excellent security for the database but requires the most effort to implement. In a two-tier architecture, a client connects directly to a server database. Again, not impossible, but has some setup costs. A one-tier system simply has the user connect directly to the database on their own personal computer. This approach can support students with no or poor Internet connections. This is important for rural students especially - only 70% of households near our campus have broadband on average [22, 23]. A no-Internet option would be useful for other learners as well.

ScaffoldSQL is a learning tool for SQL instruction that integrates into flipped classrooms, provides scaffolded instruction, and uses a one-tier system for no or poor Internet connection support. It runs using an SQLite database through a web browser. A prototype of this system, first described in [24], was developed by the first author and used for the 2019-2020 school year at Missouri Southern State University, a regional teaching university in southwest Missouri in the United States. This prototype consisted of an SQL simulator built into an interactive textbook. Students were able to type in free-form SQL queries and enter their response into an essay-style question in the online textbook.

Despite students anecdotally enjoying the first prototype, when these students came to class, they were unprepared for the more complex material being presented in class. They were unsure of the ordering and execution of clauses, did not understand concepts like aggregation, logical operators, and join operations. These issues led to the conclusion that they required further scaffolded instruction. Further improvements described in this paper include the Parson's problem interface, a content creation tool, research tool, and UI improvements.

In this paper, we will begin by identifying relevant literature in SQL teaching and learning and Parson's problems in Section 2. Section 3 will look at ScaffoldSQL's novel approach to combining techniques from computer science education research used in other contexts but applied to database pedagogy. Section 4 will examine a previous version of ScaffoldSQL used in a flipped classroom setting. Section 4 will also describe the most recent iteration of ScaffoldSQL and its specific implementation of interventions in depth. Section 5 will identify opportunities for future work, while Section 6 will offer some concluding remarks.

2. Literature Review

There have been many efforts to assess student understanding of SQL. Most of these efforts can be divided into four main categories: SQL converted to natural languages, auto-grading of SQL queries, SQL visualizations, and feature-rich interactive tutoring systems.

In [25], the authors describe a system to help students learn SQL by converting queries into natural language descriptions using regular expressions. SQL involving subqueries are not captured by this system, but it is still able to translate 96% of test queries into an English representation. ScaffoldSQL does not support the conversion of SQL statements into their English equivalents.

Several systems have been developed to assist instructors in grading SQL queries. In [26], the authors describe XDaTa, a system that compares the results of a student's query to the results of the solution query. The system can also generate test set data for SQL queries. It does not currently support SQLite. ScaffoldSQL specifically does not include a solution query to avoid cheating that can occur in a one-tier architecture.

Visualization is another avenue of teaching SQL. Early visualization efforts include eSQL [8], a query visualization tool for Oracle databases [27], and the Animated Database Courseware (ADbC) system [28]. SAVI (System for Advanced SQL Visualization) [3] is a tool for visualizing basic selection with the WHERE operator, joins, grouping, and aggregates. It also runs on any HTML5 compliant browser, but still requires an extensive back-end server setup. SQL in Steps (SiS) [29] uses a visualization strategy to help students build the individual clauses of an SQL query. Mastery Grids [30] is another, more recent method of visualization using smart content for teaching Java, SQL, and Python. ScaffoldSQL does not provide visualization support, but this would be an interesting avenue for future research.

CS/SE educational research has a long history of interactive tutoring combined with automated assessment as an effective means to support pedagogy in classrooms. In terms of grading and tutoring for SQL queries, some of the tools include SQL-Tutor [7], AssessSQL [6], aSQLg [31], LEARN-SQL [32], SQLator [4], and unnamed systems like [5]. ScaffoldSQL's free-form automated test case suite looks most like these tools, with the addition of the Parson's problem interface – making it unique.

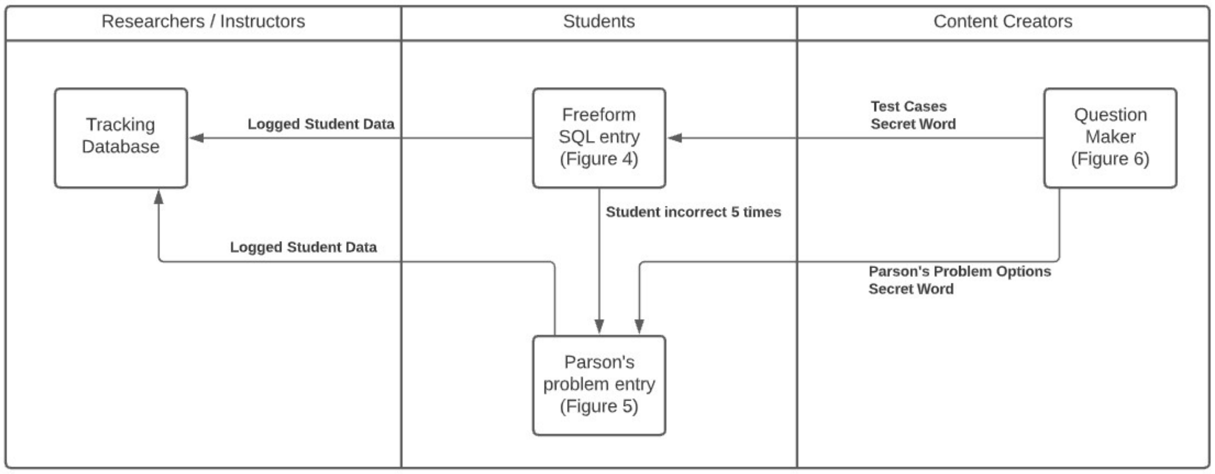

Researchers / Instructors
Students
Content Creators
Tracking Database
Logged Student Data
Freeform SQL entry (Figure 4)
Test Cases
Secret Word
Question Maker (Figure 6)
Student incorrect 5 times
Logged Student Data
Parson's Problem Options
Secret Word
Parson's problem entry (Figure 5)

Figure 1. Flow diagram for ScaffoldSQL. Students start in the Freeform SQL entry interface.

3. Proposed Approach: Supporting Database Pedagogy using ScaffoldSQL

ScaffoldSQL (Figure 1) uniquely combines several pedagogical approaches from non-database domains and applies them to student learning. ScaffoldSQL combines (1) automated test cases, as in [4, 5, 6, 7, 31, 32], (2) Parson's problems [15, 16], (3) support for low or offline Internet connectivity using "secret words", (4) a series of hooks for tracking student progress for researchers, (see [33, 34]); and (5) a tool for developing question, test case, and Parson's problem content, similar to [35].

Automated test cases have been used to help teach students database concepts in many contexts [4, 5, 6, 7, 31, 32]. We see this as a baseline functionality for any new learning tool designed to teach students about SQL syntax and semantics.

Parson's problems have been well established as tools for helping students learn imperative programming languages [15]. Our system seeks to determine the effectiveness of applying Parson's problems to a new domain - learning SQL.

```
SELECT Items.itemName, COUNT(Items.itemName)
FROM Guilds JOIN GuildTreasury
    ON Guilds.guildID = GuildTreasury.guildID
JOIN Item
   ON Items.itemID = GuildTreasury.itemID
WHERE Guilds.guildName = 'Shocking Power'
GROUP BY Guilds.guildName, Items.itemName
ORDER BY COUNT(Items.itemName) DESC
```

Figure 2. A SQL query that has been indented and placed on separate lines.

As a declarative language, SQL does not typically use control flow, however, SQL can be formatted into "clauses" on separate lines, forcing students to think about the "steps" of the problem (Figure 2). This format is not required and can be replaced with a single less-readable line (Figure 3). However, industry-standard practice focuses on using clause-based formatting. For this reason, Parson's problems were chosen in this approach as a tool to help students scaffold their understanding of SQL.

```
SELECT Items.itemName, COUNT(Items.itemName)
FROM Guilds JOIN GuildTreasury ON
Guilds.guildID = GuildTreasury.guildID JOIN
Item ON Items.itemID = GuildTreasury.itemID
WHERE Guilds.guildName = 'Shocking Power'
GROUP BY Guilds.guildName, Items.itemName
ORDER BY COUNT(Items.itemName) DESC
```

Figure 3. The same query as in Figure 2 without indentation or line breaks.

The COVID-19 pandemic has illuminated issues with Internet equity, especially in remote and rural areas [36]. SaffoldSQL aims to reduce this discrepancy with respect to SQL pedagogy by including a series of "secret words". When creating a question, the instructor selects two words – one for the Parson's problem interface and one for the free-form interface. These words are encrypted and hidden from the student. When the student answers a question correctly in an interface, that secret word is encrypted and shown to the student. These secret words can be sent to instructors via non-Internet means - whether through SMS messages, paper, or verbally. This allows the instructor to see which students needed to use the Parson's problem interface and which students were able to answer the question using the free-form interface – allowing for more appropriately scaffolded instruction in low or no Internet scenarios.

Recent efforts in computer science education have focused on developing common frameworks for sharing research data. For example, PEML (Programming Exercise Markup Language) [37] for sharing programming exercises, and ProgSnap2 (Programming Snapshots) [38] for sharing student process data captured during student interactions with computer-based programming learning tools. ScaffoldSQL seeks to support the broader computer science education research field by using these existing frameworks, rather than inventing a new one.

One of the major disadvantages of platforms like ScaffoldSQL is the difficulty in developing content that is bug-free and syntactically valid [35]. Instructors do not necessarily have the time, expertise, or patience to learn the specific test case and Parson's problem format. Thus, ScaffoldSQL provides a content creation tool, allowing an instructor or other content creator to specify what test cases, Parson's problem data, and secret words should be included for their problems, without knowing the specifics of the file format.

4. ScaffoldSQL: Implementation and Field Study

ScaffoldSQL has been prototyped, evaluated, and has undergone significant iterative changes. This section provides details regarding the two versions in the following subsections.

4.1. ScaffoldSQL v1

At the university using the initial prototype, SQL is considered a third programming course after two semesters focused on C#. Approximately half of the course is involved in teaching SQL, while the other half focuses on writing GUI applications in C# that use databases.

To reduce the effort required to maintain a server architecture and with a long-term plan of supporting offline deployment, SQLite was chosen to implement ScaffoldSQL v1. SQLite is "a small, fast, self-contained, high-reliability, full-featured, SQL database engine" [39]. ScaffoldSQL uses a Javascript version of SQLite to provide the students with an interface to edit SQL [40].

In the summer of 2019, the author developed an interactive website and textbook in Top Hat. Top Hat is a web-based platform for course management. It includes features like attendance tracking, lecture recording, polls and quizzes, assignments, and textbooks [41]. The textbook allowed the instructor to "flip" the database course. Students completed a series of readings and SQL queries on a simple database, then came to class to work on more difficult SQL and GUI problems in a pair programming environment. The textbook includes an iframe ability, displaying external web site content in the textbook. This "external website" was ScaffoldSQL v1.

The website was built in GitHub Pages using sql.js [40] and used in the 2019-2020 school year. Students were able to query the database and see the results of their query. They would then enter their responses into a Top Hat question which had to be graded manually.

While this was somewhat effective, students would often have the wrong answer and come to class inadequately prepared to answer more complicated SQL questions. Analysis of students' responses when using ScaffoldSQL v1 and their content understanding led to the development of the new system (v1.2).

4.2. ScaffoldSQL v1.2

During Fall 2021, progress was made developing ScaffoldSQL to include more features. The new version combines the following:

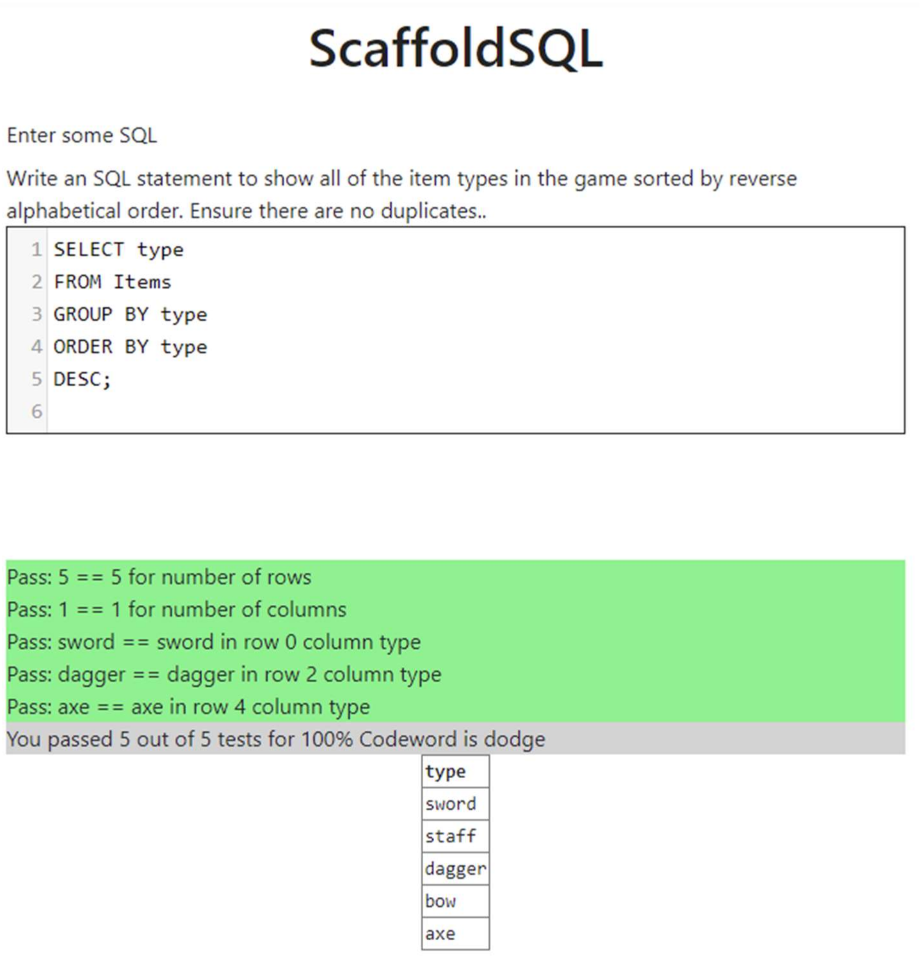
- a "secret word" system for improving the experience for students with poor or no Internet connectivity;
- automatic grading using test cases;
- a Parson's problem hint interface;
- a "Question Maker" so instructors or content creators can build questions, Parson's problems, and test cases; and
- research infrastructure for gathering data.

This new version is a standalone system that does not require Top Hat or a connected Internet server.

4.2.1. Offline Support - "Secret Words". One of the design goals of ScaffoldSQL is to create a product that worked in locations with poor or no Internet connectivity but could also scale - adding more features when the Internet is available. The secret word system is designed to assist students in this situation.

In a normal client-server platform, an Internet-based server would store whether they got a particular query right or wrong. In ScaffoldSQL, upon completing a question, the program provides the student with a secret word that they can give to their instructor over a non-Internet communication channel.

There are two sets of secret words. One set corresponds to the student correctly answering an SQL query using the free-form interface, and another that corresponds to the student successfully answering a


ScaffoldSQL

Enter some SQL

Write an SQL statement to show all of the item types in the game sorted by reverse alphabetical order. Ensure there are no duplicates..

```
SELECT type
FROM Items
GROUP BY type
ORDER BY type
DESC;
```

```
Pass: 5 == 5 for number of rows
Pass: 1 == 1 for number of columns
Pass: sword == sword in row 0 column type
Pass: dagger == dagger in row 2 column type
Pass: axe == axe in row 4 column type
You passed 5 out of 5 tests for 100% Codeword is dodge
```

type
sword
staff
dagger
bow
axe

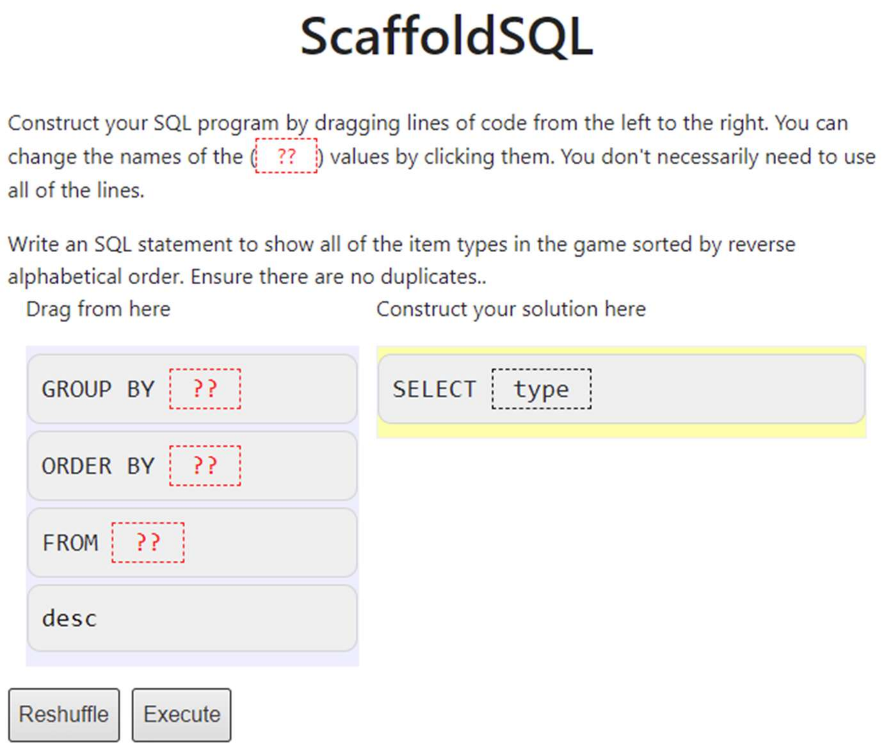
Figure 4. The free-form interface of ScaffoldSQL filled in, after a student has successfully answered the question. The codeword can be entered into an LMS or Top Hat textbook response to help instructors identify who needed a hint.

question after using the Parson's problem interface. This way, an instructor can keep track of student progress, even without the direct support of an Internet connection. These secret words could also eventually be used as part of a gamification system (see Future Work).

Since the secret words are stored locally, one worry is that students could simply look up the secret words and enter those into the offline communication channel. A dedicated student could open the files associated with the ScaffoldSQL program and use the secret words directly, rather than completing the exercises. To alleviate this risk, secret words are encrypted using AES. This is not a perfect solution but should keep the casual student from cheating during the exercises.

4.2.2. Automatic Grading via Test Cases. In the initial ScaffoldSQL prototype, students needed to either write a precisely matching SQL query or have the instructor manually grade their SQL query to get full credit for a solution. This led to a second design goal of ScaffoldSQL: a product where students did not have to exactly match the provided solution, giving more flexibility to student creativity and reducing instructor grading burdens. To satisfy this requirement, a system of test cases was developed that would be executed on the results of a student's SQL query.

A typical use of the ScaffoldSQL system would begin with a student being posed with an SQL-based problem or question in a free-form interface (Figure 4). The simulator loads a specific test case file using a GET request to a locally running web server. The student begins the problem by attempting to answer the SQL query by typing out their response. When they are satisfied with their answer, they click the Execute button, which executes the query on the database. Rather than matching the student's submission against a provided solution via a simple string comparison, the results returned from the database are compared to these test cases to determine if their output matches. If the student passes all the test cases, their answer is deemed



ScaffoldSQL

Construct your SQL program by dragging lines of code from the left to the right. You can change the names of the (??) values by clicking them. You don't necessarily need to use all of the lines.

Write an SQL statement to show all of the item types in the game sorted by reverse alphabetical order. Ensure there are no duplicates..

Figure 5. The Parson's Problem interface in ScaffoldSQL. Clicking and dragging lines from left to the right solves the problem. Clicking the ?? buttons switches between options. Clicking Reshuffle sets the problem back to its original state, but re-ordered. Clicking the Execute button runs the SQL query on the test cases with the database results appearing as output.

correct, the secret word is decrypted and provided to the student.

If the student gets the free-form question incorrect once, they are allowed to enter the Parson's problem interface. If the student is incorrect five times, they are automatically put into the Parson's problem interface. Future versions will allow the instructor to tailor how many free-form attempts should be allowed before the hint interface is exposed or when students are automatically switched to the interface.

The test case format of ScaffoldSQL is a text file divided into three main sections - introduction, automated test case information, and Parson's problem information.

The first section consists of five lines.

1. The text of the question.
2. A true/false value indicating if this question includes a Parson's problem.
3. A true/false value indicating if the question includes automated test cases.
4. The encrypted secret word for completing the Parson's problem.
5. The encrypted secret word for completing the automated test cases (without the Parson's problem hint).

The second section contains a list of automated tests. The test cases can check if the results contain a given number of rows and columns, if there is a specific value in a specific cell in the results, and if the name of a particular column matches a particular value.

There is a single line with the constant "Parsons" to separate the automated test cases from the Parson's problem data. The final section describes the information to be displayed to the user in the Parson's problem interface, using the js-parsons format, described in the next section.

4.2.3. Parson's Problem Hint Interface. Upon clicking the Hint button or being automatically directed to the Parson's problem interface, the student sees a list of SQL query clauses with toggles on the left-hand side. The student must click and drag the clauses from the left to the right, place the clauses in the correct order, and select the appropriate toggle to answer the question (Figure 5).

After completing the Parson's problem, the student clicks the Execute button, which causes the SQL query to be executed and tested against the same series of test cases as the free-form interface. If all the test cases pass, the student is given the secret word specific to the

Parson’s problem. This way, an instructor can identify students that answered the free-form question correctly from those who needed the hint interface to answer the question, regardless of Internet availability.

To load a Parson's problem, ScaffoldSQL reads the test case file from the hard drive or available Internet connection. The js-parson’s format consists of a series of lines indicating which lines of code are included in the problem. In ScaffoldSQL’s case, this corresponds to the series of SQL clauses the student might use in their solution.

In addition to these clauses, toggles need to be specified (See the ?? buttons in Figure 5). These toggles allow the student to select between different field names, table names, expression symbols (ex: <, <=, >, >=, =) and literal values within the SQL query. These toggles start with the token $$toggle, and end with the token $$. Each toggle choice is separated by two colons. For example, a toggle that includes all of the expression symbols shown above would be represented via $$toggle<::<=::>::>=::=$$.

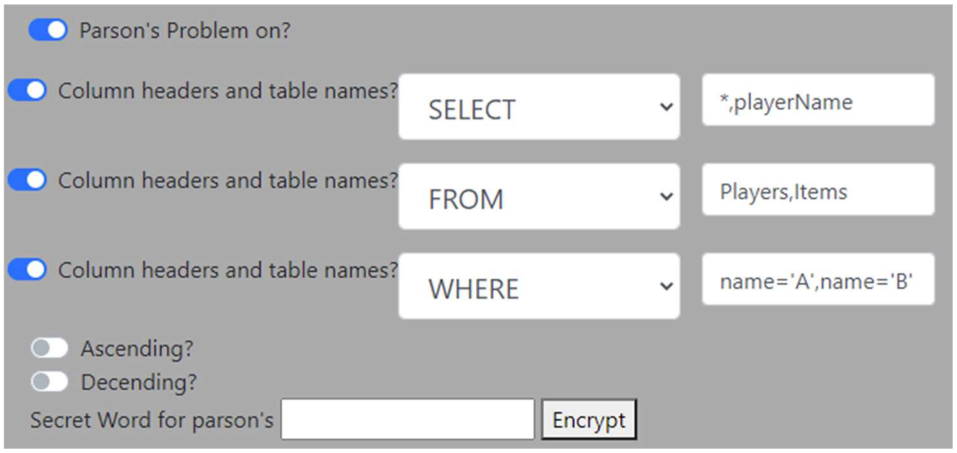
4.2.4. Instructor Tools. Building the test cases and Parson’s problems by hand is tedious. The Question Maker interface (Figure 6) allows instructors or other content creators the ability to create these test case files without needing to know the specific format requirements. The creator can enter the question number, question name, test case specifics, Parson’s problems specifics, and secret words for the problem. The test case format is then written to the hard drive and can be distributed with ScaffoldSQL installations as needed.

The Question Maker interface contains three distinct sections available for question creation, (1) The question text posed for a student attempt and its corresponding number, (2) The test cases used to evaluate the correctness of an attempted query, and (3) The Parson’s problems commands and values, including the potential to add distractors.

Upon entering the application, a creator is first asked to set the question number and question text. The question number is used as the name of the file that will be loaded by the student interface. The question text is the scenario posed to the student.

The second section is focused on test case development. The creator can select whether test cases will be used for this problem. If test case evaluation is enabled, the instructor enters test case data, including the number of columns and rows that should result from a valid SQL query, the column headers that should appear in the result, and specific evaluation criteria for row/column cell values - for example, cell 0, 2 should contain the value 20. Finally, the creator enters a secret word for the test case data and is prompted to encrypt the secret word.

The third section of Question Maker focuses on Parson’s problem development. As with the test case evaluation, creators can enable or disable the Parson’s problem interface for students. A series of combo boxes indicates which clauses to include in the Parson's problem interface. Clauses available are SELECT, FROM, INSERT INTO, VALUES, DELETE FROM, COUNT, IN, NOT, LIKE, CROSS JOIN, HAVING, ORDER BY, GROUP BY, OFFSET, LIMIT, RIGHT JOIN, UNION, FULL OUTER JOIN, ON, LEFT JOIN, NATURAL JOIN, JOIN, BETWEEN, AND, UPDATE, SET, and WHERE. There is also support for sorting data in ascending or descending order. The creator also sets the possible table, column, and literal values for each clause. Finally, the creator can select a secret word for the Parson’s problem interface, encrypt the word, and submit the question. Submitting saves the question file to the local hard drive.


Parson's Problem on? [✓ on]

Column headers and table names? [✓ on]	SELECT	*,playerName
Column headers and table names? [✓ on]	FROM	Players,Items
Column headers and table names? [✓ on]	WHERE	name='A',name='B'

Ascending? [off]
Decending? [off]
Secret Word for parson's

Figure 6. A screenshot of ScaffoldSQL’s Question Maker interface. This shows a portion of the Parson’s problem generation tool.

4.2.5. Researcher Tools. ScaffoldSQL includes tools to help with computer science education research through a series of logging hooks. When a student clicks the hint or execute buttons, an entry is stored in a research-only external database with the question number, question text, if the student was using the Parson's interface, the date and time they clicked the Hint button (if applicable), the date and time they clicked the Execute button, and the number of attempts for that student. No student demographics or tracking information is included. This information is only sent to the database when the student is connected to the Internet.

5. Contribution and Future Work

While ScaffoldSQL has not yet been comprehensively validated, students at Missouri Southern State University used ScaffoldSQL v1 and were pleased with the interactive features (e.g., having a built-in simulator for practice) but would have liked to get more immediate feedback during their readings which motivated the addition of new features in v1.2.

Since the design of ScaffoldSQL is grounded in pedagogical literature, we believe that this tool will provide scaffolds for student learning (that can allow instructors to assess and intervene in a timely manner), reducing students' cognitive load towards solving SQL queries and reducing instructor overhead. We anticipate that students will demonstrate higher learning gains, especially on more complex queries like joins and subqueries. Based on the student feedback on ScaffoldSQL v1, students appreciate the extra support during out-of-class sessions in a flipped classroom modality where they can practice and self-assess their understanding of content covered during the classroom.

As we plan to conduct large scale studies using ScaffoldSQL v1.2 and make subsequent improvements, we will seek opportunities for ways to enable widespread adoption and evaluation. Some of these include integration into Learning Management Systems using LTI, creation of a shared database of questions and test cases, improvement of research tool interoperability, addition of gamification elements, creation of a tool to help instructors of large classes manage secret word communication, development of self-paced competency-level mastery modules, improvement in Parson's problem scaffolding, and integration of ScaffoldSQL with other data pedagogical tools.

One request of many instructors when dealing with learning tools is integration with university learning management systems (Blackboard, Canvas, Moodle, etc). This is typically done via an LTI (Learning Tools Interoperability) interface. ScaffoldSQL currently does not support LTI, but it will need to be included to support broader adoption of the technology.

Instructor overhead is still high using the current version of ScaffoldSQL, especially with respect to content creation. A shared repository of Parson's problems, test cases, and questions will alleviate the burden of instructors needing to develop this content on their own. This database could also be used to identify difficulty level using student answers to questions in the ScaffoldSQL interface.

While a rudimentary system for saving research data exists, it will be important for ScaffoldSQL to become part of the broader research landscape. For this to occur, research data gathered from ScaffoldSQL needs to be read and shared in some common data format. Options include PEML [37] and ProgSnap2 [38], the former for sharing the actual questions embedded in ScaffoldSQL, while the latter for sharing student progress while they use ScaffoldSQL.

A potential future use of secret words is in the development of a gamified interface. Gamification has been used in a wide variety of disciplines and has been shown to improve student motivation and learning [42, 43, 44, 45]. ScaffoldSQL secret words could be used as part of a "madlib" type interface, where each secret word becomes part of a story that the student would be motivated to complete.

In addition to gamification of secret words, an interface could be built to collect and organize secret words from students. Presently, there is no way to manage what secret word came from what student. This would be especially important for instructors who are teaching large classes.

Rather than using the tool as a supplement to a flipped database course, it could also be self-contained and self-paced. Students would be able to complete a series of problems showing they have mastery over each of the individual learning outcomes within the tool.

More intentional and focused scaffolding between Parson's problems and the automated test suite is also possible. One such example would be a failed test case that induces a specific follow-up Parson problem. For example, if a test failed because of too many columns, the Parson's problem could focus on the SELECT clause. If a test failed because of too many rows, the associated Parson's problem could focus on the WHERE clause. Another example would be providing the student with specific study resources if they incorrectly answer a Parson's problem in common ways.

6. Conclusion

The ScaffoldSQL tool provides a system of scaffolding for students to learn SQL through a series of Parson's problems in a flipped classroom environment.

Specific student submissions allow instructors to identify students who require more attention during in-class activities.

Successful validation of ScaffoldSQL would include examining the order effects that might be present when using tool combinations. For example, is it better for students to learn using a visualization tool [3, 8, 27, 28, 29] first, then use Parson's problem interface? Is it better for students to write a Parson's problem, then have the SQL be translated into English as in [25]? These more complicated questions can only be answered with solid cross-researcher collaboration.

ScaffoldSQL is publicly available on GitHub at https://github.com/OttoBorchert/ScaffoldSQL.

7. Acknowledgements

Thank you to the Spring 2021 capstone group at Georgia Southern University that helped improve the ScaffoldSQL system, Dr. Andrew Allen for supervising the capstone students, and students at Missouri Southern State University for providing feedback on v1.

8. References

[1] "SQL Standard," [Online]. Available: https://blog.ansi.org/2018/10/sql-standard-iso-iec-9075-2016-ansi-x3-135/#gref. [Accessed 3 September 2021].

[2] "TIOBE Index," [Online]. Available: https://www.tiobe.com/tiobe-index/. [Accessed 3 September 2021].

[3] M. Cembalo, A. De Santis and U. Ferraro Petrillo, "SAVI: a new system for advanced SQL visualization," in *Proceedings of the 2011 conference on Information technology education*, West Point, 2011.

[4] S. Sadiq, M. Orlowska, W. Sadiq and J. Lin, "SQLator: an online SQL learning workbench," in *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, Leeds, 2004.

[5] G. Russell and A. Cumming, "Improving the Student Learning Experience for SQL Using Automatic Marking," in *Cognition and Exploratory Learning in Digital Age (CELDA)*, Lisbon, 2004.

[6] J. R. Prior, "Online assessment of SQL query formulation skills," in *Australasian Computing Education Conference*, 2003.

[7] A. Mitrovic, "A knowledge-based teaching system for SQL," in *World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA)*, Freiburg, 1998.

[8] R. Kearns, S. Shead and A. Fekete, "A teaching system for SQL," in *Proceedings of the 2nd Australasian conference on Computer science education*, 1997.

[9] D. Chopra, "A Recruiter's Guide to Screening and Hiring SQL Developers," 2020. [Online]. Available: https://www.adaface.com/blog/recruiter-guide-to-screening-sql-developers/. [Accessed 3 September 2021].

[10] A. Radermacher, G. Walia and D. Knudson, "Investigating the Skill Gap Between Graduating Students and Industry Expectations," in *Proceedings of the 36th ACM International Conference on Software Engineering*, Hyderabad, 2014.

[11] A. Radermacher, G. Walia and D. Knudson, "Missed Expectations: Where CS Students Fall Short in the Software Industry," *CrossTalk – The Journal of Defense Software Engineers,* vol. 28, no. 1, pp. 4-8, 2015.

[12] J. L. Bishop and M. A. Verleger, "The flipped classroom: A survey of the research," in *ASEE national conference proceedings*, Atlanta, 2013.

[13] S. V. Dickson, D. J. Chard and D. C. Simmons, "An integrated reading/writing curriculum: A focus on scaffolding," in *LD Forum*, 1993.

[14] B. Opitz, N. K. Ferdinand and A. Mecklinger, "Timing matters: the impact of immediate and delayed feedback on artificial language learning," *Frontiers in human neuroscience,* vol. 5, p. 8, 2011.

[15] P. Denny, A. Luxton-Reilly and B. Simon, "Evaluating a new exam question: Parsons problems," in *Proceedings of the fourth international workshop on computing education research*, 2008.

[16] B. J. Ericson, L. E. Margulieux and J. Rick, "Solving parsons problems versus fixing and writing code," in *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, 2017.

[17] A. Singh and e. al, "Gradescope: a fast, flexible, and fair system for scalable assessment of handwritten work," in *Proceedings of the fourth (2017) ACM conference on learning@ scale*, 2017.

[18] S. H. Edwards, "Improving student performance by evaluating how well students test their own programs," *Journal on Educational Resources in Computing (JERIC),* vol. 3, no. 3, pp. 1-es, 2003.

[19] S. Prabhu and S. Jaidka, "SQL and PL-SQL: Analysing teaching methods," in *CITRENZ Conference*, 2019.

[20] S. M. Dol, "Use of Self-Created Videos for Teaching Structured Query Language (SQL) using Flipped Classroom Activity," *Journal of Engineering Education Transformations,* vol. 33, pp. 368-375, 2020.

[21] Javatpoint, "DBMS Architecture," [Online]. Available: https://www.javatpoint.com/dbms-architecture. [Accessed 3 September 2021].

[22] BroadbandNow. [Online]. Available: https://broadbandnow.com/Missouri. [Accessed 19 September 2021].

[23] BroadbandNow. [Online]. Available: https://broadbandnow.com/Kansas. [Accessed 19 September 2021].

[24] O. Borchert, "ScaffoldSQL: SQL Test Cases + Parson's Problems," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 2021.

[25] A. Ade-Ibijola and G. Obaido, "S-NAR: generating narrations of SQL queries using regular expressions," in *Proceedings of the South African Institute of Computer Scientists and Information Technologists*, 2017.

[26] A. Bhangdiya, B. Chandra, B. Kar, B. Radhakrishnan, K. V. Maheshwara Reddy, S. Shah and S. Sudarshan, "The XDa-TA system for automated grading of SQL query assignments," in *2015 IEEE 31st International Conference on Data Engineering*, 2015.

[27] B. Allenstein, A. Yost, P. Wagner and J. Morrison, "A query simulation system to illustrate database query execution," in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, 2008.

[28] M. Murray and M. Guimaraes, "Animated database courseware: using animations to extend conceptual understanding of database concepts," *Journal of Computing Sciences in Colleges,* vol. 24, no. 2, p. 144, 2008.

[29] P. Garner and J. Mariani, "Learning SQL in steps," *Journal of Systemics, Cybernetics and Informatics,* vol. 13, no. 4, pp. 19-24, 2015.

[30] P. Brusilovsky, S. Edwards, A. Kumar, L. Malmi, L. Benotti, D. Buck, P. Ihantola, R. Prince, T. Sirkiä, S. Sosnovsky, J. Urquiza, A. Vihavainen and M. Wollowski, "Increasing Adoption of Smart Learning Content for Computer Science Education," in *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*, Uppsala, 2014.

[31] C. Kleiner, T. Tebbe and F. Heine, "Automated grading and tutoring of SQL statements to improve student learning," in *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*, 2013.

[32] A. Abelló, M. E. Rodríguez, T. Urpí, X. Burgués, M. J. Casany, C. Martín and C. Quer, "LEARN-SQL: Automatic assessment of SQL based on IMS QTI specification," in *Eighth IEEE International Conference on Advanced Learning Technologies*, 2008.

[33] K. R. B. Sanders, J. E. Moström, V. Almstrum, S. Edwards, S. Fincher, ... and L. Thomas, "DCER: sharing empirical computer science education data," in *Proceedings of the Fourth international Workshop on Computing Education Research*, 2008.

[34] M. Yudelson, "Interoperable Data Collection," 6 June 2021. [Online]. Available: https://cssplice.github.io/interdc/. [Accessed 4 September 2021].

[35] G. Hokanson and B. M. Slator, "Development Tools for Content Creation in Virtual Environments," in *Proceedings of E-LEARN 2013 - World Conference on E-Learning*, Las Vegas, 2013.

[36] M. Lieberman, "Internet Access is a Civil Rights Issue," 23 September 2020. [Online]. Available: https://www.edweek.org/technology/internet-access-is-a-civil-rights-issue/2020/09. [Accessed 3 September 2021].

[37] "PEML," [Online]. Available: https://cssplice.github.io/peml/. [Accessed 3 September 2021].

[38] "ProgSnap2," [Online]. Available: https://cssplice.github.io/progsnap2/. [Accessed 3 September 2021].

[39] "SQLite," [Online]. Available: https://www.sqlite.org/index.html. [Accessed 3 September 2021].

[40] "SQL.js," [Online]. Available: https://github.com/sql-js/sql.js/. [Accessed 3 September 2021].

[41] "Top Hat," [Online]. Available: https://tophat.com/features/. [Accessed 3 September 2021].

[42] S. Deterding, D. Dixon, R. Khaled and L. Nacke, "From game design elements to gamefulness: defining gamification," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, 2011.

[43] J. Thom, D. Millen and J. DiMicco, "Removing gamification from an enterprise sns," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, 2012.

[44] Z. Fitz-Walter, D. Tjondronegoro and P. Wyeth, "Orientation passport: using gamification to engage university students," in *Proceedings of the 23rd Australian Computer-Human Interaction Conference*, 2011.

[45] M. R. Narasareddygari, G. S. Walia, D. M. Duke, V. Ramasamy, J. Kiper, D. L. Davis, ... and H. W. Alomari, "Evaluating the Impact of Combination of Engagement Strategies in SEP-CyLE on Improve Student Learning of Programming Concepts," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019.

[46] B. Shneiderman, "Improving the human factors aspect of database interactions," *ACM Transactions on Database Systems (TODS),* vol. 3, no. 4, pp. 417-439, 1978.

[47] K. Renaud and J. Van Biljon, "Teaching SQL—Which Pedagogical Horse for This Course?," in *British National Conference on Databases*, Edinburgh, 2004.

[48] V. Matos, R. Grasser and P. Jalics, "The case of the missing tuple: teaching the SQL outer-join operator to undergraduate information systems students," *Journal of Computing Sciences in Colleges,* vol. 22, no. 1, pp. 23-32, 2006.