

Generating SQL Queries from Visual Specifications

George Obaido¹, Abejide Ade-Ibijola², and Hima Vadapalli¹

¹ School of Computer Science and Applied Mathematics
University of the Witwatersrand,
Johannesburg, South Africa

rabeshi.george@gmail.com, hima.vadapalli@wits.ac.za

² Formal Structures, Algorithms, and Industrial Applications Research Cluster
School of Consumer Intelligence and Information Systems
University of Johannesburg, Bunting Road Campus
Johannesburg, South Africa
abejideai@uj.ac.za, www.abejide.org

Abstract. Structured Query Language (SQL) is the most widely used declarative language for accessing relational databases, and an essential topic in introductory database courses in higher learning institutions. Despite the intuitiveness of SQL, formulating and comprehending written queries can be confusing, especially for undergraduate students. One major reason for this is that the simple syntax of SQL is often misleading and hard to comprehend. A number of tools have been developed to aid the comprehension of queries and improve the mental models of students concerning the underlying logic of SQL. Some of these tools employed visualisation and animation in their approach to aid the comprehension of SQL. This paper presents an interactive comprehension aid based on visualisation, specifically designed to support the SQL `SELECT` statement, an area identified in the literature as problematic for students. The visualisation tool uses visual specifications depicting SQL operations to build queries. This is expected to reduce the cognitive load of a student who is learning SQL. We have shown with an online survey that adopting visual specifications in teaching systems assist students in attaining a richer learning experience in introductory database courses.

Keywords: SQL comprehension, visual specification, learning via visualisation

1 Introduction

Structured Query Language (SQL) remains an integral part of the introductory relational database course curriculum in higher institutions of learning, and considered the most widely used *declarative* and *non-procedural* language for querying relational databases [35]. As a declarative language, many of its statements read a bit like English and are usually described as “English-like”. Secondly, SQL is expressed as a non-procedural language because data operations are specified by their intended result rather than the end result. Since it was adopted by the ANSI³ and ISO⁴ in the late

³ American National Standards Institute

⁴ International Organization for Standardization

70's, SQL has evolved as a standardised language for most relational database management systems or RDBMSs [24, 28]. SQL uses commands to define schema objects also known as DDL⁵ and manipulate data (or DML⁶) in a RDBMS [15].

With the vast application of SQL, formulating queries is an important skill that employers require for graduates in the computer sciences and related disciplines. This skill relates to knowledge of relational databases that enables the writing of useful and correct queries in SQL [25]. If students' knowledge of the SQL concept is relatively poor, their performance in business sectors would be questionable [5, 14]. Despite the trivial task of manipulating and retrieving information from relational databases, the task of writing correct queries still remains a well known problem for students [15, 20, 41]. Numerous studies have been conducted with regard to the challenges that students face while learning SQL queries [4, 20, 25]. Some of the challenges include the burden of having to memorise database schemas; the misleading declarative nature of SQL (especially, when learning it alongside a Procedural or Object-Oriented Programming language) and the incorrect perception that queries are easy to grasp [10, 20]. Another misconception that students find particularly difficult is using the SQL join and grouping functions [24]. Also, writing queries in DML expressions has shown to be difficult for undergraduate students [15]. In order to provide adequate support to address these problems, there is a need to build interactive platforms, either incorporated with *animation* or *visualisation* aids to support the understanding of SQL. In the past decades, a number of tools have been developed to provide support in learning SQL. Some of the existing tools employed interactive visualisations to aid SQL understanding.

In this paper, we have proposed the use of an interactive visualisation technique to aid the understanding of SQL. The visualisation technique ensures the interaction between visual specifications to build queries and will eliminate the need to memorise database schemas, which is a major problem faced by students learning SQL. In this work, we have developed an interactive visualisation aid which uses visual specifications by the 'drag and drop' interactions for generating SQL queries. Although, this approach have found applications in the programming languages paradigm such as Alice [13], Scratch [39] and StarLogo TNG [43], with little or none (to the best of our knowledge) applied to SQL queries.

Figure 1 shows the SQL query generation process. To use the SQL visualiser, images are represented as visual specifications to depict the database model. These images can be moved into the query box. When the moved images correspond to a standard SQL SELECT statement, a query is generated and displayed to the user.

In this research, we make the following contributions:

1. used predefined images that represent SQL commands to present a user with a drag and drop visualiser.

⁵ Data Definition Language

⁶ Data Manipulation Language

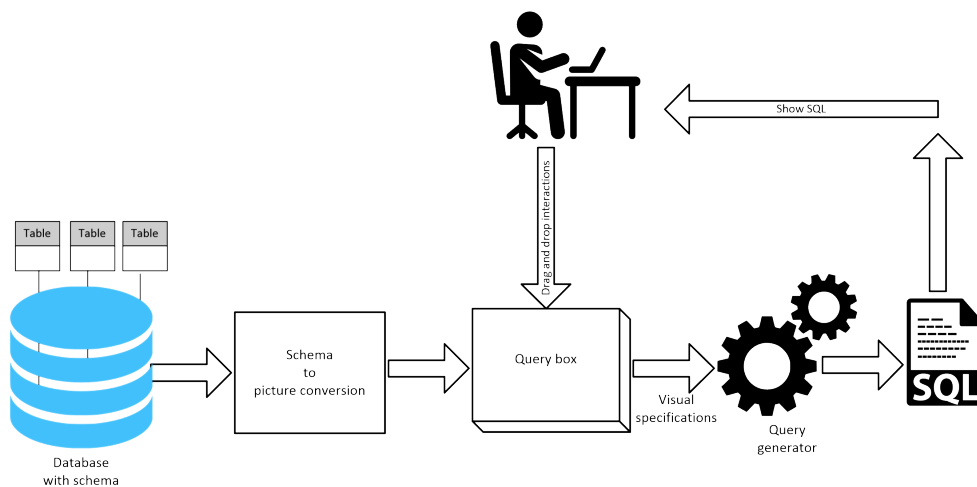


Fig. 1. The framework of the SQL query generation

2. evaluated the approach using human subjects to get feedback on their perceptions of the visualisation.

This paper is organised as follows. In Section 2, the background related to this work is discussed. Section 3 presents the methodology used in this work. Section 4 provides the results. Section 5 presents empirical evaluation of the SQL visualiser on a group of 121 participants. Section 6 concludes this work and presents future directions.

2 Related Work

In this section, we present the previous work related to this study. First, the pedagogy models of teaching SQL were identified. Also, the process of learning through visualisation was presented. Lastly, we discuss some of the tools used to aid the understanding of SQL.

2.1 Teaching Patterns for SQL

Different SQL pedagogy models have been proposed over the years. These teaching patterns are either traditional face-to-face instructor-led teaching or teaching through electronic aids. We discuss some of the pedagogy models for teaching SQL. Prior and Lister [37] proposed three-fold objectives for teaching SQL. The first objective aimed to ensure that students developed their query formulation skills, while the second objective described an approach to develop real-world systems using SQL. The third approach encouraged students to practice and develop their SQL skills using an online method. Renaud and van Biljon proposed another approach [38] which highlighted two pedagogical approaches for teaching SQL. The first method ensured that students used tools for learning SQL, while the second method suggested the use of an instruction-led method

of teaching SQL in class before exposing students to tools. Caldeira [9] approach was similar, as it aimed to ensure that students first understood SQL thoroughly by reading and understanding how to write SQL queries before they were exposed to electronic aids. A recent study by Ahadi *et al.* [4] presented the common semantics that instructors need to consider when teaching students how to write SQL queries. Their study emphasised the need for a deeper understanding of the common semantics to improve learning outcomes to assist students in the proper writing of SQL queries.

2.2 Learning through Visualisation

The idea of learning through visualisation is not a new concept. This technique has been extensively used in different application domains to present ideas [17, 26]. Ellis and Alan [16] defined visualisation as a systematic way of representing an abstract idea in a way that facilitates human understanding. This abstract idea is usually designed in a way that is sometimes ‘playful and aesthetically pleasing’, so that users can explore how to solve tasks. In terms of learning, visualisation can encourage active participation and lead students’ critical thought processes [6]. A study conducted by Kellems *et al.* [27] showed that visualisation can even help students with learning disabilities to grasp information more easily. Their study showed that visual aids can also better meet the academic demands of students with Autism Spectrum Disorder (ASD)⁷, since they do not require intensive training.

In the area of program comprehension, visualisation has been explored to aid the understanding of programming. Lee *et al.* [30] proposed the use of the drag and drop refactoring visualisation to assist programmers comprehend programs written in Java. Empirical evidence presented in their work showed that the approach was more efficient and less error-prone, and that it could help programmers comprehend programs easily. Studies conducted in block programming (a technique which represents programs as blocks and uses the drag and drop technique to generate a program) indicated that this type of visualisation increased student engagement and that it was indeed effective in knowledge transfer [33, 40]. A recent study conducted on serious games on the C programming language, also indicated that this type of learning visualisation applies the drag and drop approach and pave way for undergraduate students to learn programming [44]. The benefit offered by this visual aid is that it enhances self-pacing, while its entertaining component attracts the attention of students and engages them in the learning process. It is worth noting that most SQL comprehension tools employ visualisation to automatically comprehend SQL queries and database schemas through either textual or graphical representations [10, 19, 41].

2.3 SQL Learning Tools

Over the past decades, a number of tools have been developed to aid the understanding of SQL [10, 25, 34, 41]. The majority of these tools are either web-based, desktop or gaming applications, which use visualisation to represent SQL queries. We present some of the tools that have been used to aid the comprehension of SQL.

⁷ A neurological and developmental disorder, which result in communication and interaction difficulties.

One of the earliest SQL learning aids was the **eSQL** system developed by Kearns *et al.* [25] to visualise the SQL **SELECT** statement (an area students usually find confusing). The goal of **eSQL** is to visualise the set-by-step method, where the query selects the output data and behaviour of query operators. The **eSQL** system displays an output data from an SQL query, hence, extensive knowledge of SQL is required to use the system. **SQLify** was developed by Dekeyser *et al.* [15] as an online interactive, visualisation and assessment tool to aid the SQL comprehension. This tool provides comprehensive feedback to students in an automated and interactive fashion. To use the system, a user is required to possess knowledge of SQL in order to test and visualise a database schema. **QueryViz**, a graph visualisation tool presented by Danaparamita and Gatterbauer [12], allows users to understand the logic behind the SQL syntax. The advantage posed by **QueryViz** is that it allows students to read and understand queries as fast as possible. Also, the tool enables query-reuse and provides a means of visualising schema objects.

SAVI, an online tool developed by Cembalo [10], uses visualisation to teach the semantics of SQL. Hence, SQL knowledge is required to interact with a target database. The goal of **SAVI** is to overcome difficulties via the way an SQL query interacts with the database. **eledSQL** was presented by Grillenberger and Brinda [22] to overcome the challenges of teaching SQL education in German secondary schools. This tool ensures that students are able to interact with databases without prior knowledge of SQL. **sAccess** is an interactive web-based learning tool developed by Nagataki *et al.* [36], which offers a simple, customised interface for learning database manipulation in relational databases. The tool provides a simple interface and eliminates the mastering of SQL syntaxes. As such, it is suitable for introductory database education.

While all these tools include visualisation to aid SQL comprehension, not one considers our approach of using images to generate an SQL query. Also, most of the tools discussed, require knowledge of SQL to present the visualisation. Table 1 reviews these tools and the approach undertaken in this research.

Table 1. Comparison of other existing tools and our approach

Features	eSQL	SAVI	SQLify	sAccess	eledSQL	QueryViz	Our Approach
Visualisation of database schema	✗	✓	✓	✗	✗	✓	✓
Visualisation of output data	✓	✓	✓	✓	✓	✗	✗
SQL query generation	✗	✗	✗	✗	✗	✗	✓
Feedback on query semantics	✗	✗	✓	✗	✗	✓	✓
Visual object representation	✗	✗	✗	✗	✗	✗	✓
Ideal for less knowledgeable users (undergraduate students)	✗	✗	✗	✓	✓	✗	✓

In the domain of program comprehension, visualisation tools have been developed to employ the *drag and drop* approach to aid the understanding of programming. **Scratch** [39], a tool developed to use blocks in order to form a program, minimises

the occurrences of syntax errors. Another tool that aids program comprehension is **Alice 2** [11], a 3D animation and interactive system, that uses blocks to build virtual worlds, which are primarily designed for novices to learn programming. The intention behind **DropLet**, designed by Bau [7], is to facilitate the learning of programs written in JavaScript, and to close the learning gap between using blocks and text to learn programming. **BlockC**, a tool developed by Kyfonidis *et al.* [29] in 2017, empowers students' learning of the C programming language. The **BlockC** tool guides them to focus on the programming logic before they learn the syntax of the language. Together, all these tools focus on using blocks to learn programming without first considering the syntax of the language. This syntax-free style of teaching was described by Fincher [18] and is generally regarded as the syntax-free approach (SFA) for program comprehension. This syntax-free approach has been applied in novice program comprehension [1–3]. We have extended this approach in generating queries using visual specifications without first considering the SQL syntax. The next section presents the methodology of the SQL visualiser.

3 Methodology

One aspect of queries which poses difficulties for students is the SQL **SELECT** construct. This type of query is used to extract data from a relational database [25]. Hence, our main goal is focused on using visual aids to easily generate queries by means of the SQL **SELECT** constructs. This idea can be extended to the system of queries.

It is a common perception that students are better in *recognising* visual constructs rather than in *writing* code for an application. Thus, this work is motivated by an intention to use visual specifications in order to generate SQL queries. Also, another motive of developing this SQL visualiser is to use it as a teaching and learning aid. We have found that database schemas pose difficulties for students [15], hence, our intention is to simplify the process of understanding database schemas. We identify three main points to distinguish our visualisation from other approaches.

Intuitive Our visualisation is intuitive to students who are learning SQL queries for the first time. The visualisation uses images to depict each query statement. It helps students better understand SQL queries since it helps them get a glimpse of the behaviour of the image when each image is selected. Hence, students do not require extensive training to understand how to use the visual aid.

Interactive The visualisation tool is interactive, which means that students are not required to write any query statement in the application. They can simply click and drag the images across panels. Query statements are generated at the same time.

Helpful A help facility is provided before using the visualiser. A user is provided with an instruction of the underlying database schema before using the application. Also, *hints* are provided to the user and are specified using colours (green or red). These colours show whether a query is wrong or correct. In addition, a textual suggestion is offered to the user to ensure that the correct object is selected.

3.1 Design of the SQL Visualiser

The SQL visualiser was implemented as a Windows Form Application and was included as part of the .NET framework for the purpose of creating rich client applications [31]. The visualisation tool consists of some components used for the generation of a query. These components include schema, query box and query generator.

3.1.1 Schema The schema shows the logical organisation of the data. In the visualiser, the schema consists of tables and associated fields. The SQL query is based on the underlying schema. In addition, if a schema is correct, the generated SQL query is correct, and vice-versa. Figure 2 depicts the schema for this model.

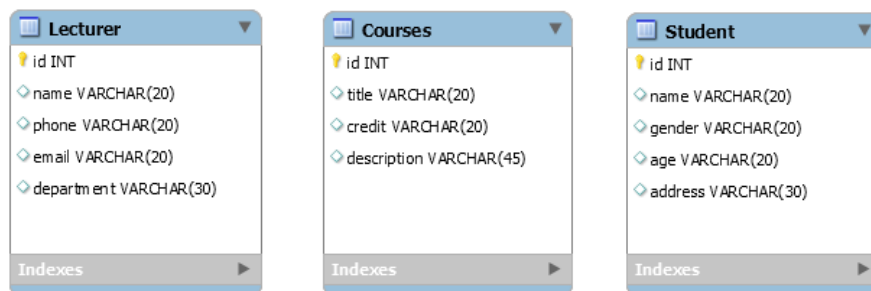


Fig. 2. Logical organisation of the data

In the schema, each table is shown by its name displayed at the top and its corresponding attributes shown at the bottom. For example:

Lecturer (id, name, phone, email, department)

Courses (id, title, credits, description)

Student (id, name, gender, age, address)

Each table is linked by a primary key. The unique identifier for the tables is the entity *id*. The visualisation tool relies on the schema to generate the SQL query.

3.1.2 Query box The query box is used to specify subsets of the schema that the user is interested in. The query box is also denoted as the building block for the query generator. In the query box, the schema (represented by the images) are extracted into a form used by the query generator to generate the SQL query. Each image is included with a caption for easy identification. Table 2, Table 3, Table 4 and Table 5 represent the pictures and descriptions used to represent the schema.

3.1.3 Query generator The query generator transforms the images in the query box and presents a query to the user. As more images are added, the query generator also adds the attribute to the query. The generator phase is very straightforward since the scope of this work is limited to a single-relation. The **SELECT** portion of the query

Table 2. Operation: Symbols and description


Symbol	SQL Block	Description
	SELECT	This icon represents the SELECT statement

Table 3. Entity and Attributes: Symbols and description of the Lecturer table













Symbol	SQL Block	Description
	*	This denotes “all” in rows
	id	This icon represents the primary key field
	name	This icon denotes the name field
	phone	A representation for a phone field
	email	This symbol denotes an email field
	Lecturer	A representation for a lecturer field

Table 4. Entity and Attributes: Symbols and description of the Courses table








Symbol	SQL Block	Description
	*	This denotes “all” in row
	id	This icon represents a primary key field
	title	This icon denotes a title field
	credit	A representation for a credit field
	description	This symbol represents a description field
	Courses	A representation for a course entity

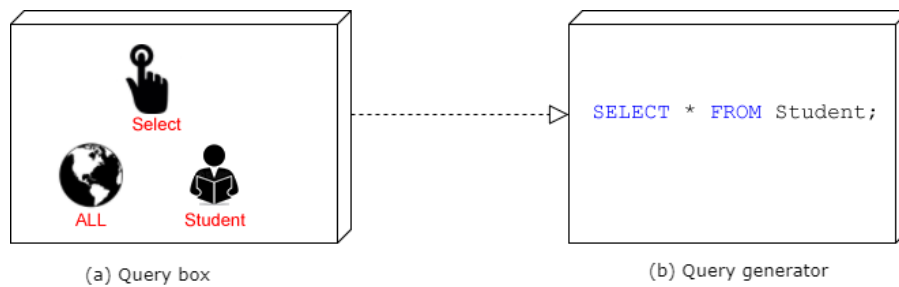
consists of tables and attributes; where the **FROM** clause defines a table and the **WHERE** clause is defined by a field attribute and its value. We illustrate this with an example.

Example 1. Consider a simple database table with the schema: *Student (id, name, age)*. Now, write a simple SQL query to display all information from the student table.

Figure 3 presents the process that the query generator uses to present queries. When a user adds the images into the query box (a), the query generator displays the

Table 5. Entity and Attributes: Symbols and description of the Student table

Symbol	SQL Block	Description
	*	This denotes “all” in rows
	id	This icon represents an identity field
	name	An illustration for a name field
	gender	A symbol for a gender field
	age	A representation for age field
	address	An icon for an address field
	Student	This symbol represents a student entity

**Fig. 3.** The process of generating an SQL query

visualisation from the images that were selected into the query box (b). More options used in this paper are presented in the next section.

3.1.4 More Options

Operators and Values In the SQL visualiser, we have explored some comparison operators, such as the (= or *equal to*, < or *less than* and > or *greater than*). The comparison operators are used within the generated query with values between 0–100 to show the relationship. For each operator, the user can interactively determine which option to select. More-so, a user can select the preferred choice of value to use within the query by using the scroll option provided. Once the scroll option is selected, it changes the value within the generated query.

Colours In the Human-Computer Interaction (HCI) interface design specifications, colours have been described to convey information [8]. Within this specification, the association of colours may be used for many purposes if this is implemented

conservatively. Colours have salient features, which are useful in human perception [23]. For example, the colour *red* strongly indicates an error, while *green* indicates a normal or acceptable condition. These colours were explored within the visualiser to indicate either an acceptable condition or to respond critically to a user’s error as presented in Figure 4.

Red (indicator of an error)	Green (acceptable condition)
Incorrect query, drag the required field	Fantastic! Your query is correct

Fig. 4. Colours used to show annotation

4 Results

We present the result from using the visualisation tool. Figure 5 shows the feedback received from adding only the `SELECT` operation into the query box at runtime. The feedback received will assist the user to specify the required visuals before the query can be generated. This example shows that, if the user inserts the correct table and its attributes, the query will be successfully generated. The field “ID” was chosen as the primary key for each of the table, and the value, “50” was selected. See Figure 6. The help facility showing the instruction on how to use the SQL visualiser is presented in Figure 7.

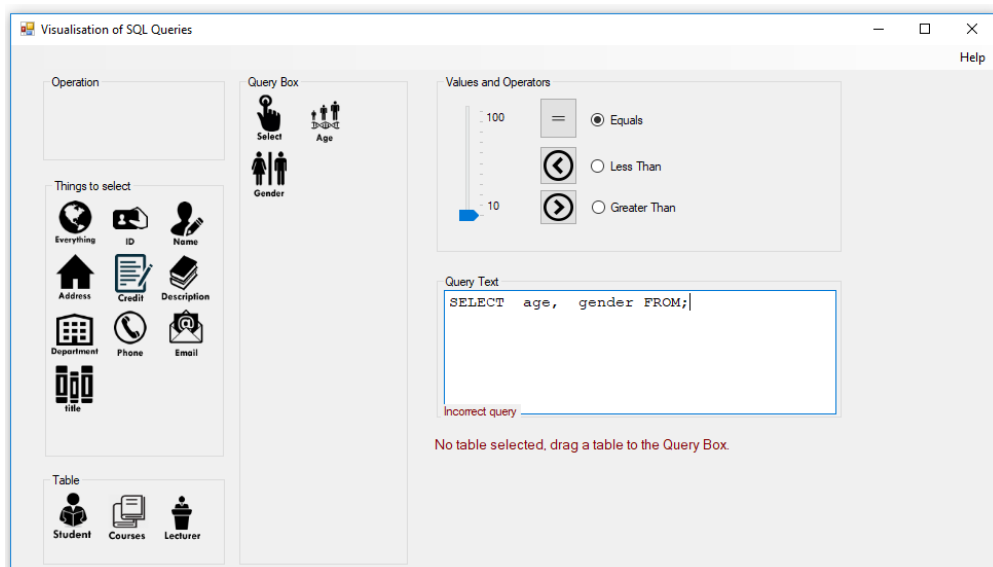


Fig. 5. SQL Visualiser: Hints provided to the user



Fig. 6. SQL Visualiser: A successfully generated query

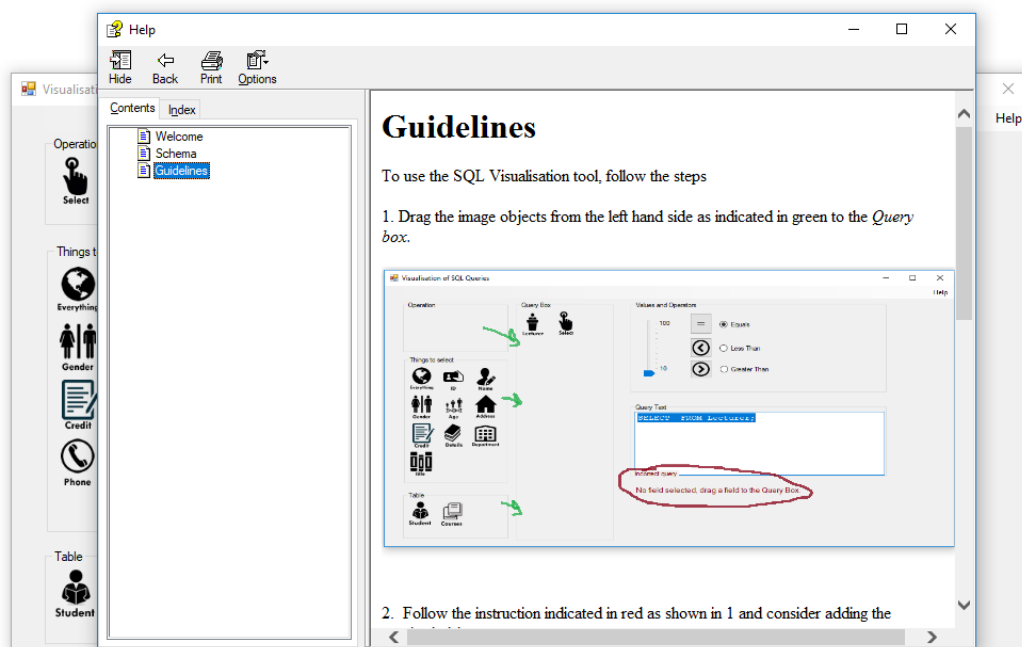


Fig. 7. SQL Visualiser: The help facility option presents the instruction to the user

We have presented a visualisation tool that applies visual specifications to generate queries. The technique presented in this paper will find applications in teaching and learning systems. The benefits offered by the visualiser will facilitate human comprehension of SQL queries. This work will particularly aid undergraduate students who are learning SQL queries for the first time.

Another application area of this work can be extended to commercial business systems, where the visualiser may be used to assist non-professional users comprehend SQL queries. While such users may be aware of databases, their knowledge of SQL queries may be limited. We believe that our technique's clear communication and visualisation-focus will help users to easily understand SQL queries.

5 Evaluation

The evaluation of the SQL visualiser was carried out using an online survey from 121 students from the University of the Witwatersrand. The respondents were mostly undergraduate Computer Science students, and majority of them had knowledge of SQL. The questionnaire is split into two parts: the first required the students to answer general questions about their knowledge of visualisers, while the second focused on the perception of the SQL visualiser we have designed in this research. In addition, the students were asked the following open ended question: *Please provide feedback to improve the SQL visualiser*, and constructive feedbacks were received. The results of the evaluation are presented in Section 5.1.

5.1 Result of the Survey

Out of the 121 responses, 89.3% admitted to have knowledge of SQL, 7.4% affirmed no knowledge of SQL and 3.3% were unsure about their responses – this is presented in Figure 8(a). 94.2% agreed that the SQL visualiser was user-friendly, 4.1% admitted that the visualiser was not user-friendly, and 1.7% were unsure about their responses (see Figure 8(b)). Furthermore, the students were asked if they were able to synthesise basic SQL queries using the visualiser (in Figure 8(c)). 95% agreed that the visualiser helped them comprehend SQL queries, 4% admitted that they find the visualiser difficult to use and 1% stayed indifferent. In addition, 92.6% admitted that visual specifications helped them understand the syntax of the SQL queries, 5% did not agree and 2.4% stayed indifferent (see Figure 8(d)). The feedback the respondents made were helpful, and some respondents highlighted some limitations of the SQL visualiser. Examples of the positive comments made were:

1. “The visualizer was extremely helpful. I liked the way the pictures assisted in displaying the query. As a learner, I think it will help other students understand SQL queries better and improve our knowledge of the SQL concept”,
2. “Icons were a good idea and helpful. It is not boring”,
3. “Far much better visualiser I have used so far”,
4. “It seems simple and straight-forward, the user does not have to try hard to understand its functionality and operation”.

Some of the limitations mentioned by the respondents include:

1. “There should be explanations, the use of comments in the query text box would be extremely useful. I have used SQL before, therefore this is mostly targeted at novice users. Other methods should be integrated to cater for expert users”,
2. “The icons colour choice is boring”,
3. “Possibly increase text sizing for the visually impaired user”,
4. “This tool should allow users to choose their own icons (e.g. students could be a different icon)”.

The majority of the students commended the use of visual specifications to aid their comprehension. These results are consistent with the evaluation carried out on a visualiser [42] for program comprehension, where users perceptions supported the usefulness and importance of visual specifications. We believe that adopting this tool in higher institutions of learning will improve student’s comprehension of SQL.

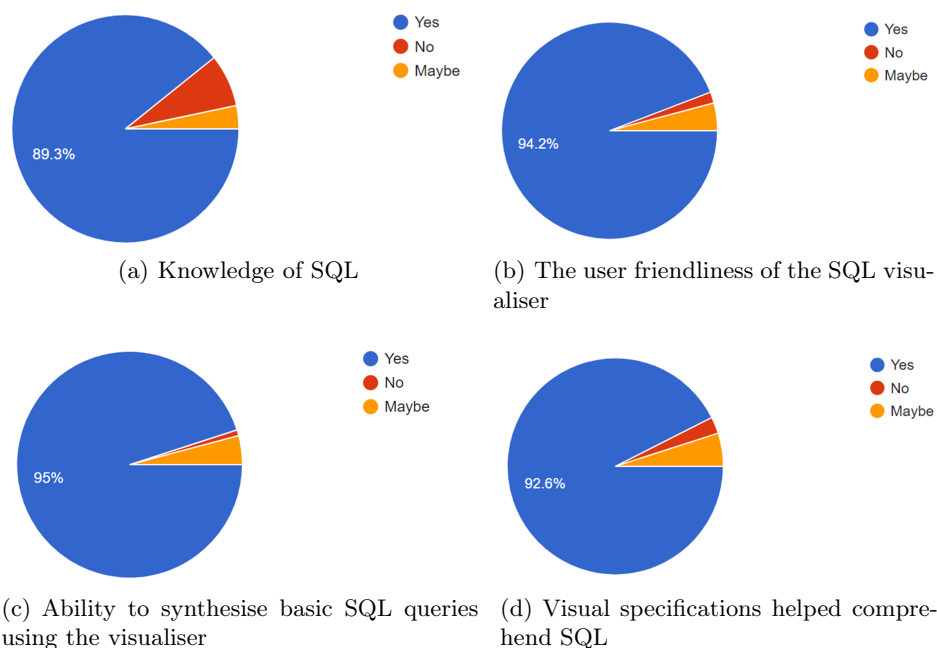


Fig. 8. The result of the evaluation

6 Conclusion and Future Directions

In this paper, we presented an interactive visualisation tool that uses visual specifications to build SQL queries. The visualisation tool considered the SQL `SELECT` constructs

in a bid to improve the comprehension process. It is generally agreed that visualisation can encourage active participation and also lead to critical thought processes in students [21, 32].

Currently, we have explored just a few SQL `SELECT` constructs to aid SQL comprehension. In future, we will create an extended visualiser in an attempt to support other SQL `SELECT` statements such as `JOIN`, `ORDER BY`, `GROUP BY` and more aggregate functions. Another aspect to explore would be to generate nested queries. Thus far, we have only explored simple SQL query generation. Other areas of exploration would be to allow students define their own schemas and then use the information to generate SQL queries. This will provide a richer learning experience to the learner. We also anticipate an extended GUI that attempts to use generated queries to manipulate data in a database and then produce a result. This will promote a more extensive use of the tool.

7 Acknowledgements

The authors would like to appreciate the Department of Science and Technology (DST) and the Council for Scientific and Industrial Research (CSIR) Inter-bursary support programme for funding this research.

References

1. Ade-Ibijola, A.: New finite automata applications in novice program comprehension. LAP Lambert Academic Publishing (2017)
2. Ade-Ibijola, A., Ewert, S., Sanders, I.: Abstracting and narrating novice programs using regular expressions. In: Proceedings of the Annual Conference of the South African Institute for Computer Scientists and Information Technologists. pp. 19–28. ACM (2014)
3. Ade-Ibijola, A.O.: Automatic novice program comprehension for semantic bug detection. Ph.D. thesis (2016)
4. Ahadi, A., Behbood, V., Vihavainen, A., Prior, J., Lister, R.: Students’ syntactic mistakes in writing seven different types of SQL queries and its application to predicting students’ success. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education. pp. 401–406. ACM (2016)
5. Aken, A., Michalisin, M.D.: The impact of the skills gap on the recruitment of MIS graduates. In: Proceedings of the 2007 ACM SIGMIS conference on computer personnel research: The global information technology workforce. pp. 105–111. ACM (2007)
6. Allenstein, B., Yost, A., Wagner, P., Morrison, J.: A query simulation system to illustrate database query execution. ACM SIGCSE Bulletin 40(1), 493–497 (2008)
7. Bau, D.: Droplet, a blocks-based editor for text code. Journal of Computing Sciences in Colleges 30(6), 138–144 (2015)
8. Brown, C.M.: Human-Computer Interface design guidelines. Intellect Books (1998)
9. Caldeira, C.P.: Teaching SQL: a case study. In: ACM SIGCSE Bulletin. vol. 40, pp. 340–340. ACM (2008)
10. Cembalo, M., De Santis, A., Ferraro Petrillo, U.: SAVI: a new system for advanced SQL visualization. In: Proceedings of the 2011 conference on Information technology education. pp. 165–170. ACM (2011)

11. Cooper, S., Dann, W., Pausch, R.: Teaching objects-first in introductory computer science. In: ACM SIGCSE Bulletin. vol. 35, pp. 191–195. ACM (2003)
12. Danaparamita, J., Gatterbauer, W.: QueryViz: helping users understand SQL queries and their patterns. In: Proceedings of the 14th International Conference on Extending Database Technology. pp. 558–561. ACM (2011)
13. Dann, W.P., Cooper, S., Pausch, R.: Learning to program with Alice 2. Prentice Hall Press (2008)
14. Davis, P.: What computer skills do employees expect from recent college graduates? The Journal of Technological Horizons in Education) 25(2), 74 (1997)
15. Dekeyser, S., de Raadt, M., Lee, T.Y.: Computer assisted assessment of SQL query skills. In: Proceedings of the eighteenth conference on Australasian database-Volume 63. pp. 53–62. Australian Computer Society, Inc. (2007)
16. Ellis, G., Dix, A.: A taxonomy of clutter reduction for information visualisation. IEEE transactions on visualization and computer graphics 13(6), 1216–1223 (2007)
17. Ellis, G., Mansmann, F.: Mastering the information age solving problems with visual analytics. In: Eurographics. vol. 2, p. 5 (2010)
18. Fincher, S.: What are we doing when we teach programming? In: Frontiers in Education Conference. vol. 1, pp. 12A4–1. IEEE (1999)
19. Folland, K.A.T.: viSQLizer: An interactive visualizer for learning SQL. Master’s thesis (2016)
20. Garner, P., Mariani, J.A.: Learning SQL in steps. Journal on Systemics, Cybernetics and Informatics 13(4), 19–24 (2015)
21. Gray, C., Malins, J.: Visualizing research: A guide to the research process in art and design. Routledge (2016)
22. Grillenberger, A., Brinda, T.: eledSQL: a new web-based learning environment for teaching databases and SQL at secondary school level. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education. pp. 101–104. ACM (2012)
23. Jost, T., Ouerhani, N., Von Wartburg, R., Müri, R., Hügli, H.: Assessing the contribution of color in visual attention. Computer Vision and Image Understanding 100(1-2), 107–123 (2005)
24. Kawash, J.: Formulating second-order logic conditions in SQL. In: Proceedings of the 15th Annual Conference on Information technology education. pp. 115–120. ACM (2014)
25. Kearns, R., Shead, S., Fekete, A.: A teaching system for SQL. In: Proceedings of the 2nd Australasian conference on Computer science education. pp. 224–231. ACM (1997)
26. Keim, D.A., Mansmann, F., Schneidewind, J., Thomas, J., Ziegler, H.: Visual analytics: Scope and challenges. In: Visual data mining, pp. 76–90. Springer (2008)
27. Kellems, R.O., Gabrielsen, T.P., Williams, C.: Using visual organizers and technology: Supporting executive function, abstract language comprehension, and social learning. In: Technology and the Treatment of Children with Autism Spectrum Disorder, pp. 75–86. Springer (2016)
28. Kriegel, A.: Discovering SQL: a hands-on guide for beginners. John Wiley & Sons (2011)
29. Kyfonidis, C., Moumoutzis, N., Christodoulakis, S.: Block-C: A block-based programming teaching tool to facilitate introductory C programming courses. In: IEEE Global Engineering Education Conference. pp. 570–579. IEEE (2017)
30. Lee, Y.Y., Chen, N., Johnson, R.E.: Drag-and-drop refactoring: intuitive and efficient program transformation. In: Proceedings of the 2013 International Conference on Software Engineering. pp. 23–32. IEEE Press (2013)
31. Liberty, J.: Programming C#: Building .NET Applications with C. O’Reilly Media, Inc. (2005)
32. Lye, S.Y., Koh, J.H.L.: Review on teaching and learning of computational thinking through programming: What is next for k-12? Computers in Human Behavior 41, 51–61 (2014)

33. Malan, D.J., Leitner, H.H.: Scratch for budding computer scientists. In: ACM SIGCSE Bulletin. vol. 39, pp. 223–227. ACM (2007)
34. Mitrovic, A.: An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education* 13(2-4), 173–197 (2003)
35. Myalapalli, V.K., Shiva, M.B.: An appraisal to optimize SQL queries. In: *International Conference on Pervasive Computing*. pp. 1–6. IEEE (2015)
36. Nagataki, H., Nakano, Y., Nobe, M., Tohyama, T., Kanemune, S.: A visual learning tool for database operation. In: *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*. pp. 39–40. ACM (2013)
37. Prior, J.C., Lister, R.: The backwash effect on SQL skills grading. *ACM SIGCSE Bulletin* 36(3), 32–36 (2004)
38. Renaud, K., Van Biljon, J.: Teaching SQL: Which pedagogical horse for this course? In: *British National Conference on Databases*. pp. 244–256. Springer (2004)
39. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al.: Scratch: programming for all. *Communications of the ACM* 52(11), 60–67 (2009)
40. Rizvi, M., Humphries, T., Major, D., Jones, M., Lauzun, H.: A cs0 course using Scratch. *Journal of Computing Sciences in Colleges* 26(3), 19–27 (2011)
41. Sadiq, S., Orlowska, M., Sadiq, W., Lin, J.: SQLator: an online SQL learning workbench. In: *ACM SIGCSE Bulletin*. vol. 36, pp. 223–227. ACM (2004)
42. Satyanarayan, A., Heer, J.: Lyra: An interactive visualization design environment. In: *Computer Graphics Forum*. vol. 33, pp. 351–360. Wiley Online Library (2014)
43. Wang, K., McCaffrey, C., Wendel, D., Klopfer, E.: 3D game design with programming blocks in StarLogo TNG. In: *Proceedings of the 7th international conference on Learning sciences*. pp. 1008–1009. International Society of the Learning Sciences (2006)
44. Yassine, A., Chenouni, D., Berrada, M., Tahiri, A.: A serious game for learning C programming language concepts using solo taxonomy. *International Journal of Emerging Technologies in Learning* 12(03), 110–127 (2017)