Expert Perspectives on Student Errors in SQL

DAPHNE MIEDEMA and GEORGE FLETCHER, Eindhoven University of Technology
EFTHIMIA AIVALOGLOU, Leiden Institute of Advanced Computer Science, The Netherlands
and Open Universiteit

Prior studies in the Computer Science education literature have illustrated that novices make many mistakes in composing SQL queries. Query formulation proves to be difficult for students. Only recently, some headway was made towards understanding why SQL leads to so many mistakes, by uncovering student misconceptions. In this article, we shed new light on SQL misconceptions by analyzing the hypotheses of SQL experts on the causes of student errors. By examining the experts’ perceptions, we draw on their understanding of students’ misconceptions and on their experiences with studying and teaching SQL. For our analysis, we chose the Policy Delphi, a questionnaire instrument specifically designed for gathering opinions and evidence. Through a two-round process, our nineteen participants proposed and voted on underlying causes for SQL errors which resulted in a set of hypotheses per error. Our main contribution to this article is this new set of possible misconceptions. With them, we can design more complete educational approaches to address misconceptions underlying SQL errors made by students, leading to more effective SQL education.

CCS Concepts: • Information systems → Structured Query Language; • General and reference → Empirical studies; • Social and professional topics → Computing education; Information systems education;

Additional Key Words and Phrases: Errors, misconceptions, delphi study


1 INTRODUCTION

The Structured Query Language (SQL) is ubiquitous, both in practice and in education. It is an established language, introduced in 1970 for querying relational databases [12] and has been an ISO standard since 1987 [1]. Research in Database Education has shown that SQL is difficult for learners, as indicated by the high prevalence of errors in SQL queries. Various researchers have studied these errors; one of the first such studies was done by Reisner [36], while more recent work has provided a more extensive understanding of errors [2, 4, 29, 46].

Authors’ addresses: D. Miedema and G. Fletcher, Eindhoven University of Technology, Groene Loper 5, 5612 AE, Eindhoven, the Netherlands; emails: {d.e.miedema, g.h.l.fletcher}@tue.nl; E. Aivaloglou, Leiden Institute of Advanced Computer Science, Niels Bohrweg 1, 2333 CA, Leiden, The Netherlands and Open Universiteit, Valkenburgerweg 177, 6419 AT, Heerlen, The Netherlands; email: e.aivaloglou@liacs.leidenuniv.nl.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2022 Copyright held by the owner/author(s).
1946-6226/2022/12-ART11
https://doi.org/10.1145/3551392

Only recently have researchers started to explore the underlying reasons for these SQL errors: misconceptions. In total, there are four types of conceptions, and all can be applied in instructional approaches to help students learn: correct, incomplete, incorrect, and misconceptions [25]. Margulieux et al. propose the multiple conceptions theory, which suggests that “learners develop better conceptual knowledge when they are guided to compare multiple conceptions of a concept during instruction” [25, page 194]. For this reason, it is important to explore all existing conceptions of SQL. Since all types of conceptions for SQL are underexposed, much work needs to still be done.

A recent study that employed a think-aloud protocol to identify student misconceptions on SQL resulted in fourteen misconceptions in four categories: misconceptions based on previous course knowledge, generalization-based misconceptions, misconceptions based on language, and misconceptions due to an incomplete or incorrect mental model [27]. Building on these findings, we aim to explore the misconceptions of novices from a different viewpoint: that of experts. SQL experts can be educators, researchers, or practitioners. Each of these groups has a distinct view on SQL errors and their underlying causes, which strengthens the content validity of the study. Furthermore, as experts have worked with SQL significantly longer than novices, they should have correct and highly detailed mental models. This, in combination with their experience in teaching SQL, can give us unique insights into underlying causes of errors.

Research goals. Our aim is to explore the underlying causes of student errors from the perspective of SQL experts. The method through which we do this is a derivative of the Delphi technique [11]. We selected eleven errors of different categories identified by Miedema et al. [27] and formed a panel of nineteen SQL experts from diverse backgrounds. We then presented these errors to our panel and asked them to provide their hypotheses as to why students might make these errors. We aggregated their answers to find all distinct hypotheses and set up a second questionnaire in which we asked all experts to vote on the likelihood of each hypothesis posed by them and their colleagues. In particular, we aim to answer the following two research questions.

Q1. How do experts interpret the underlying causes of student errors in SQL?
Q2. Which are the differences in the perspectives of students and experts on errors in SQL?

Answers to these questions provide insights complementary to that of the student perspective, towards the design of effective educational approaches addressing and correcting the misconceptions underlying the SQL errors made by students.

The remainder of the article is structured as follows: we first introduce related work to our study in Section 2. Then, we introduce the errors that this article is analyzing, by putting them in the context of existing literature in Section 3. We present our research method in Section 4 and our results in Section 5. Finally, we discuss our findings, their validity, limitations and future work in Section 6.

2 RELATED WORK

2.1 Student Errors and Misconceptions

The errors novices make when writing SQL queries have been researched through several quantitative studies and have been organized in various categories [4, 7, 32, 44–46]. Commonly reported categories are syntax errors [2, 3, 32] and semantic errors [3, 4, 7, 32]. In their discussion of semantic errors, Brass and Goldberg distinguish between two types of answers [7]: those queries for which we need to know the question to see whether they are correct, and those where the query is incorrect regardless of the question. Taipalus and Siponen also use these two classes: they call the former category logical errors, and the latter semantic errors [46]. In addition, they add
complications to their categorization. Complications are those unnecessary additions that muddle understanding of a query but do produce the correct result table. Examining which errors are more persistent in student work, Taipalus and Perälä found that logical errors and complications had the highest incidence, and that syntax and semantic errors occurred least in the final attempts of students [45].

The underlying causes of SQL errors have not been as thoroughly examined. The first attempt was by Smelcer, who developed a model to explain basic errors such as omissions and misspellings [38], listing five underlying causes: working memory overload, absence of retrieval cues, procedural fixedness, incorrect procedural knowledge, and misperception. Building on this work, Taipalus in [44] discussed causes that could be behind the persistent errors explored in [45]. They map the errors to four of the cognitive explanations introduced by Smelcer [38]. Unfortunately, this mapping is speculative [44], as they only use the plain text query as submitted, without qualitative discussion with students. Qualitative insight into the causes of SQL errors was only provided recently, through a think-aloud study with SQL novices [27]. Misconceptions were found to stem from prior course knowledge, along with generalization-based misconceptions, language-based misconceptions, and misconceptions due to an incomplete or incorrect mental model.

Contrary to SQL misconceptions, the underlying causes of errors in introductory programming languages have been widely studied. Although SQL is a query language, not a programming language, parallels can be drawn. Research on programming misconceptions has focused on the understanding of concepts in imperative and object-oriented languages [24, 40]—an extensive list of misconceptions is presented by Sorva [41]. Apart from languages commonly taught in CS1, recent work on misconceptions has focused on visual programming languages used by younger learners [43], functional programming languages [14], as well as difficulties with the understanding of data structures [52]. Prior work also found that students’ prior knowledge in areas such as mathematics and natural language was transferred to programming, which caused misconceptions [10, 34]. Transfer of prior knowledge has been observed, for example, to affect the understanding of execution sequences [40]. Prior exposure to different programming languages has also been found to cause misconceptions [37], for example, due to the differences in the notations and their use for the definitions of variables and arithmetic operators.

Research on the programming errors of novices has also examined their cognitive causes. Limitations in the attention and memory resources for cognitive processing have been identified early as possible causes of programming errors [30]. Exploring the cognitive processing differences between expert programmers and novices, it was found that experts make more use of semantic memory and high-level plan knowledge to direct their programming activities [6] while, more recently, experts and novices were found to have similar thinking process in syntax debugging, but differences were identified in semantic debugging and logical structuring [50]. Examining the causes of different types of errors, Ko and Myers developed a model of programming errors in event-based systems that ties specific errors to their cognitive causes, which can be knowledge, strategic, or attentional, and can relate to specification, implementation, and debugging activities [22].

2.2 Teachers’ Perceptions on Student Errors

Related work in Computer Science education has examined the programming errors students make from the point of view of educators. Hristova et al. surveyed computer science professors about commonly made Java programming errors and identified the ones that are perceived as the most frequent and harder to find and fix for the students [19]. In a later study, Brown and Altadmri examined whether educators can make an accurate estimate and form a consensus on the frequency and time to fix 18 Java errors [8]. Comparing survey results with a dataset of compilation events, they found that educators’ estimates did not agree with one another or with the actual student
data and suggested that a different level of discourse may be required, as “educators may still be accurate about the cause of mistakes and a student’s conceptions of the mistakes, but just not about the frequency and time-to-fix of such mistakes.” [8, page 19].

Examining teachers’ understanding of the causes of student errors or their misconceptions is particularly important because it is a major factor in teachers’ pedagogical content knowledge, affecting planning, conducting instruction, and assessment [31]. Teachers should be able to identify student misconceptions and have an understanding of the context in which they are created [18]. One of the studies that have focused on programming misconceptions was made by Qian et al., who surveyed teachers about 37 misconceptions of high school students on Python [33]. The study examined their perceived frequency, importance in learning, and teachers’ confidence in addressing them, and found that misconceptions may be difficult for teachers to detect if they are latent, i.e., not immediately leading to error messages.

2.3 The Delphi Protocol
The Delphi protocol is also referred to in the literature as the Delphi technique, the Delphi method, or just the Delphi. It was introduced to optimize the methods of interaction with experts [11]. When compared to discussing with several experts back to back, or through round-table discussions, the Delphi protocol has the benefit of experts evaluating each other’s viewpoints, as well as the advantage that no loud voice can dominate the discussion [11]. Herman adds: “[E]xperts remain anonymous during the [Delphi], so that they are influenced by the logic of the arguments rather than by the reputations of other experts.” [17, page 50].

In the context of educational research, the Delphi protocol has been used as a starting point for the development of Concept Inventories (CIs) [16, 17, 42]. For CIs, typically researchers start by asking what concepts should be taught in a certain course. Then, for each concept that occurs in the list more than once, they ask their experts to rate them based on importance and student understanding. Specifically in the Computer Science education domain, Delphi has been applied in the works by Goldman et al. [15] and Kaczmarczyk et al. [20] on memory models and data assignment, the work by Herman on logic [17], the work by Wittie et al. on CS2 [49], and the work by Farghally et al. on algorithm analysis [13].

Instead of focusing on concepts, as the studies working towards a CI do, in our Delphi setup we aim to gather hypotheses. Experts vote on each other’s hypotheses to gain more insight into what the more likely hypotheses are. This method is a variation of the Policy Delphi, which, in contrast to conventional and real-time Delphi, asks experts for presenting all options and evidence for considerations [11]. This type of Delphi method is not necessarily about making decisions, but is more about being informed. The idea behind Policy Delphi is to identify differing opinions [51].

3 ERROR CONTEXT
In our study on misconceptions, we present our participants with 11 errors for their consideration. All of these errors have been identified and described in related literature. The errors are of varying severity and specificity. For example, some are syntax errors and some are complications, i.e., errors that would still lead to a correct results table [46]. Some errors are specific and some are broader and represent general issues, such as concerns about the understanding of GROUP BY, JOIN, or subqueries. Some of these may not have been covered literally but instead can be considered part of a general error category.

In our deliberation below, we do not discuss the work by Miedema et al. [27], as that is the source of these errors. The literature context for each error is summarized in Table 3.
3.1 Using ≠ Instead of != or <>

This error regards a non-standard operator used in SQL, one that is not recognized as correct. It is a syntax error, as parsing ≠ will lead to an error. This error is mentioned by Presler–Marshall et al. as a broken operator [32], and by Taipalus et al. as SYN-6 Common syntax error: nonstandard operators (36) [46]. In later work, Taipalus et al. highlight that errors of type SYN-6 are among the most persistent errors made by students [45]. Ahadi et al. specifically focus on SQL syntax errors in [2]. In their discussion of PostgreSQL error codes, one code that occurs often is that for operator does not exist. Although Ahadi et al. do not explicitly mention the error we describe here, it does fit the aforementioned operator category.

3.2 AVG in WHERE Instead of HAVING

The foundation for this type of error was laid by Ahadi et al. who identified student difficulties in all queries involving HAVING [3]. In a later paper, they write that “we noticed that the condition that is supposed to appear in the HAVING clause was often mistakenly written in the WHERE clause” [4, page 275], which is the generalized variant of our error. Taipalus et al. identify two categories in which aggregation is misplaced: SYN-4 Illegal aggregate function placement (14) and LOG-4 Expression error (69) [46]. AVG in WHERE is a syntax error, making SYN-4 our best fit.

3.3 JOIN in All Tables that One Needs the Primary Keys of

This mistake can be categorized as a complication: it does not lead to an incorrect result table but does increase the computational complexity. Brass et al. introduced this concept of complication, and more specifically introduce the mistake of unnecessary join [7]. In the work of Taipalus et al., this mistake is most closely related to complication 84—unnecessary join [46]. Another related error that they identify is the LOG-2 JOIN error (59), as their example shows the inclusion of an extra table. In later work, Taipalus et al. conclude that both complications and LOG errors are common [45]. The persistence of LOG-2 errors is unknown, but complications are typically corrected, according to them.

3.4 Missing Second Table for Self-join

Again, Ahadi et al. laid the groundwork to show that students struggle with self-joins: they found that 76% of students made (unidentified) mistakes with the self-join [3]. Although the missing second table for self-join was not mentioned specifically, Ahadi et al. do indicate that students did not understand that they needed a self-join [4]. Presler–Marshall et al. capture this error with their categories of incomplete query and table reference error [32]. In the categorization of Taipalus et al., this error is covered in SEM-3 Missing JOIN (48) and LOG-2 JOIN error (62). SEM-3 is among the most frequent and persistent errors, LOG-2 is frequent but with unknown persistence [45]. Taipalus also mapped errors to causes. For these two errors, the causes are the absence of cues and the lack of knowledge of the students [44].

3.5 Lack of Understanding of when to Apply GROUP BY

GROUP BY is another concept that students struggle with: these struggles are mentioned in almost all SQL error-related articles. Already in 1977, Reisner found many problems with the “major error” class in GROUP BY questions [35]. This high error rate was also found by Ahadi et al. [2, 3]. Furthermore, Ahadi et al. identified the errors irrelevantly used GROUP BY clause and including extra columns in GROUP BY clause [4].

Brass and Goldberg identified five types of errors with GROUP BY: unnecessary GROUP BY in EXISTS subquery, GROUP BY with singleton groups, GROUP BY with only a single group,
Unnecessary GROUP BY attribute, and GROUP BY can be replaced with DISTINCT [7]. All five of these reflect a lack of understanding of how GROUP BY works, and when to apply it properly.

3.6 Missing Parts of Keywords
Various examples of this error are available in the literature. Ahadi et al. found that sometimes students would forget the ON in JOIN ON [4]. Presler–Marshall et al. group this error under miscellaneous errors, with the example of a student writing BY instead of GROUP BY [32]. Taipalus classifies this error as SYN-6 Common syntax error (36) [46]. Their later work confirms SYN-6 errors to be among the most persistent [45].

3.7 Extra Condition when Not Required
This mistake can also be classified as a complication: the extra condition does not change the result table, but does make the query longer. Ahadi et al. include this mistake under the category irrelevant condition in the WHERE clause [4]. Brass and Goldberg group this under implied, tautological, or inconsistent sub-condition [7]. In the work of Taipalus et al., this mistake is related to both LOG-4 Expression error (68) and complication 82 [46]. As mentioned before, LOG errors have been shown to be common, and this error again is a persistent one. Taipalus maps error 68 to ignorance of the students [44].

3.8 Ambiguous Column Name Because of Missing Alias
This error is mentioned explicitly in various other works, either as an ambiguous column [2], column reference error [32], or SYN-1 Ambiguous Database Object (2) [46].

3.9 Using Comma Instead of AND in the WHERE Clause
Although this error is not mentioned in literature about the WHERE clause, several works do refer to similar errors. Reisner identifies a category of minor errors, which includes errors in spelling and punctuation [35]. Presler–Marshall et al. also identify punctuation errors, specifically for commas: they coin the errors extra commas and missing commas [32].

3.10 Thinking DISTINCT will Take the First Item of a List
This error is perhaps described in a non-optimal way. When searching through the literature, we found this error described in a more explanatory manner: subquery term that might return more than one tuple, by Brass and Goldberg [7]. The error of the student is that there was a subquery that may return more than one result, and this needs to be fixed in some way. The student then suggested DISTINCT, which does not work. It is noteworthy that we could only find one article that mentions this error, as this seems to be an error that may occur often for novices.

3.11 Lack of Understanding of Subqueries
Subqueries are another element of SQL that students seem to struggle with. Ahadi et al. find that “Students were not able to identify the skill required to answer [questions requiring a simple subquery], or they did not have the skill required to construct such a query.” [4, page 275] and that “the identification by students of the need for a correlated subquery was the most common problem.” [4, page 275].

From the perspective of subqueries as complications, Brass and Goldberg define the error IN/EXISTS condition can be replaced by comparison [7], and Taipalus et al. repeat this error under complication 90 [46].
4 METHOD

4.1 Research Design Overview

Our goal for this study was to capture expert hypotheses on SQL errors made by novices. To this end, we presented novice errors to our expert participants and obtained their perspectives on them.

The study consisted of two rounds of online questionnaires, implemented through Google Forms for convenient distribution. The first questionnaire contained eleven SQL errors examined in the study by Miedema et al. [27], introduced in Section 3. For each of these 11 problems, we asked the participants to formulate one or more hypotheses as to what the underlying causes could be for this problem.

For the second round, we collected all answers for each of the questions in Round 1 and aggregated them to produce lists of hypotheses. In the resulting questionnaire, we asked the participants to vote on the likelihood of each of these hypotheses. Furthermore, participants were able to comment on the questions and hypotheses of others. From the resulting qualitative input, we can draw conclusions from expert views on underlying causes for SQL errors made by novices, answering Q1. For deeper insights, we then mapped each hypothesis to causes introduced in the literature [27, 38, 44]. The mapping was done in collaboration by two of the authors, where any cases of disagreement were discussed. This led to the confirmation of existing mappings and the development of some new categories:

- Previous course knowledge [27]
- Generalizations [27]
- Language [27]
- Mental model [27]
- Lack of practice
- Sloppiness
- Not an error
- Working memory overload [38, 44]
- Absence of retrieval cue [38, 44]
- Procedural fixedness [38, 44]
- Ignorance [38, 44]
- Shown examples were too limited
- Avoidance (database anxiety)

One additional category by Smelcer is that of Misperceptions. However, we exclude that here as the point was to find misconceptions (misperceptions), and as such all hypotheses could potentially be mapped to this category.

Towards answering Q2, we extracted the misconceptions for each error from Miedema et al. [27] and compared them with the list of hypotheses generated from the above-mentioned process. We examined whether the two contained similar concepts or reasoning, to draw conclusions from the expert versus student perspective.

4.2 Participants

4.2.1 Recruitment Process. All participants were invited through an email, detailing the current state of research and the study. It included details on the type and content of the questionnaires, as well as an estimate for the amount of time each questionnaire would take.

The invitees then let us know via return email whether they would like to participate and were sent a link to the questionnaire. Both questionnaires were open for two weeks, with an invitation sent at the start of the first week, and a reminder halfway through the second week.

We aimed for 15–30 participants in this study, as we believed that this number would be appropriate to saturate the different types of hypotheses our participants would propose. The participants’ differing backgrounds with regard to working experience, culture, and interests made saturation more likely.

The participants were not financially compensated for participation in the study. The study design and data collection were approved by the Ethical Review Board of the first author’s institution.
Table 1. Participant Profiles

<table>
<thead>
<tr>
<th>Role</th>
<th>Round 1</th>
<th>Round 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Educator</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>Researcher</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Practitioner</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>User</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Developer/Architect</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Consultant</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Each participant belongs to one or more categories.

4.2.2 Participant Characteristics. Our participants were experts in SQL education (either in tertiary education or mentoring and training in industry), use, and research. The experts were recruited from the authors’ networks of researchers and educators and were invited personally with an introductory email. Those who elected to participate received the links to the questionnaires by email.

In total, we invited 27 people. Of the 27, 17 people work for (15 different) universities, 1 at a teaching college, and 9 in industry. They work in nine different countries (in alphabetical order): Belgium, Chile, France, Germany, Netherlands, Poland, Singapore, United Kingdom, and the United States. This geographic, cultural, and institutional spread increases the content validity and generalizability of the study. From this set of participants, we received 18 submissions to our first questionnaire, and 19 submissions to the follow-up.

To obtain deeper insights into population characteristics, we asked participants to identify their relationship with SQL. They could choose one or more relations out of the following five: educator, researcher, user, developer/architect, and consultant. Most of our participants classified themselves as educator, researcher, or both. An overview of participant characteristics is presented in Table 1.

4.3 Data Collection

In the first round, we presented the participants with the eleven errors (see the first column of Table 3) along with contextual information on them, such as the query formulation question and an example SQL query demonstrating the error. A text field was provided in which the participants could provide their hypotheses on the cause of the error. The questionnaire was kicked off with an example, such that the participants had some ideas about our expectations. The questionnaire also had a field at the end for any further notes that the participant might like to make.

The next phase included the aggregation of hypotheses from the first round. For this process, the first author selected all groups of 18 hypotheses (one hypothesis per participant, one group per error). They then went through each group from the first submission to the last and summarized the core of each hypothesis if it had not been introduced before. Some participants wrote down their ideas in one-sentence hypotheses, whereas others wrote down elaborate reasons. After the first author had made a pass, the second and third author compared the full list of answers to the aggregated list and checked whether all participants’ ideas were included. Each error was left with four to nine hypotheses.

In the second round, the participants could then vote on the aggregated hypotheses on a five-point Likert scale, as can be seen in Figure 1. Each set of hypotheses also had space where our participants could elaborate on their answers, if they wished to do so.
Table 2. The Database Schema Used for the Questions in Section 5

<table>
<thead>
<tr>
<th>Table name</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>customer</td>
<td>cID, cName, street, city</td>
</tr>
<tr>
<td>store</td>
<td>sID, sName, street, city</td>
</tr>
<tr>
<td>product</td>
<td>pID, pName, suffix</td>
</tr>
<tr>
<td>shoppinglist</td>
<td>cID, pID, quantity, date</td>
</tr>
<tr>
<td>purchase</td>
<td>tID, cID, sID, pID, date, quantity, price</td>
</tr>
<tr>
<td>inventory</td>
<td>sID, pID, date, quantity, unit-price</td>
</tr>
</tbody>
</table>

Underlined attributes together form the primary key for each table.

5 RESULTS

In this section we aim at answering the research questions raised in Section 1:

Q1. How do experts interpret the underlying causes of student errors in SQL?

Q2. Which are the differences in the perspectives of students and experts on errors in SQL?

Before we dive into the participants’ hypotheses per error, we first introduce our results with a basic overview. In Table 3, we present each error, its reference in existing literature, the number of proposed hypotheses by our participants, the most popular expert hypothesis, and the proposed student hypothesis from Miedema et al. [27].

In the following subsections, we cover each of these errors and hypotheses in turn to answer Research Questions 1 and 2. To give the reader the appropriate context, we also present the elements our participants received, to support the understanding of the error: (1) the question the students tried to answer, (2) a (partial) query containing the error, and (3) context on the error. The database schema that the questions concern can be found in Table 2. The material presented in each section is exactly the same as the presentation to our participants.

The participants voted on each of the aggregated hypotheses in part two through a five-point Likert scale. We visualize this by means of stacked bar charts with a diverging color scale, where the width of each segment corresponds to the proportion of answers. Each category of votes is shown in a different color (likely hypotheses in blue, unlikely hypotheses in orange), with the neutral answers centered on the midline of the plot. Below each hypothesis, a grey badge indicates the corresponding category as introduced in Section 4.1.

The participants have been assigned random numbers and are referred to as Participant 1 to Participant 19.

5.1 Using ≠ Instead of != or <>

Findings for this error are displayed in Figure 2. Overall, the trend in these expert hypotheses is that ≠ is a common and intuitive symbol. Students may have used it for a number of reasons, reflected in the hypotheses presented above. Our Delphi participants are sympathetic:

“[..] in general, I sympathize with this one, ≠ is just the way it is written in all normal math on paper (and is also the way it should be in SQL if it wasn’t for the missing ≠-key on the keyboard...(and the lack of support for ≠ in ASCII)”—Participant 1

“It is a common symbol that will have been used in other courses, and the meaning is actually correct. Plus the <> and != in SQL are not that intuitive in their form in connection with their meaning.”—Participant 2
Table 3. This Table Presents an Overview of the Results, Containing All 11 Errors Explored in This Article, the Number of Hypotheses that Came Out of Delphi Round One, and the Most Popular Hypotheses That Came Out of Delphi Round Two

<table>
<thead>
<tr>
<th>Error</th>
<th>Similar error mentioned in:</th>
<th>Num of hypotheses</th>
<th>Most popular expert hypothesis</th>
<th>Student misconception according to Miedema et al. [27]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Using ≠ instead of != or &lt;&gt;</td>
<td>[2, 32, 46]</td>
<td>6</td>
<td>Students are familiar with ≠ and find it more intuitive than &lt;&gt; and !=</td>
<td>Interference of previous course knowledge from either Mathematics or Relational Algebra</td>
</tr>
<tr>
<td>2 AVG in WHERE instead of HAVING</td>
<td>[3, 46]</td>
<td>7</td>
<td>Students associate the application of a filter with WHERE</td>
<td>Misunderstanding SQL syntax and its internal (in)consistency</td>
</tr>
<tr>
<td>3 JOIN in all tables that you need the primary keys of</td>
<td>[7, 46]</td>
<td>7</td>
<td>Students have created a template query/pattern and apply it without understanding</td>
<td>Applying an incorrect query template</td>
</tr>
<tr>
<td>4 Missing second table for self-join</td>
<td>[3, 4, 32, 44, 46]</td>
<td>8</td>
<td>Students do not think of a second access as a separate instance of a table</td>
<td>Misunderstanding the scope of elements in a query, or thinking of the DBMS as a black box.</td>
</tr>
<tr>
<td>5 Lack of understanding of when to apply GROUP BY</td>
<td>[2–4, 7, 35]</td>
<td>7</td>
<td>Students do not understand the implication of grouping: there are no rows anymore and thus one cannot project on them</td>
<td>Misunderstanding SQL syntax and its internal (in)consistency</td>
</tr>
<tr>
<td>6 Missing parts of keywords</td>
<td>[4, 32, 46]</td>
<td>5</td>
<td>Students forget the correct syntax because most other SQL keywords are single-valued</td>
<td>Remembering only the core part of a keyword (as this captures the ‘main’ meaning)</td>
</tr>
<tr>
<td>7 Extra condition when not required</td>
<td>[4, 7, 44, 46]</td>
<td>5</td>
<td>Students do not realize that being a different customer follows from the different cities (as cID is the primary key and thus unique)</td>
<td>Lacking knowledge on primary keys</td>
</tr>
<tr>
<td>8 Ambiguous column name because of missing alias</td>
<td>[2, 32, 46]</td>
<td>4</td>
<td>Student knew which pID they mean and feel it is self-explanatory</td>
<td>No specific misconception for this error is provided. Instead, they attribute this to a lack of knowledge or experience.</td>
</tr>
<tr>
<td>9 Using comma instead of AND in WHERE clause</td>
<td>[32, 35]</td>
<td>6</td>
<td>Student was sloppy in writing the query</td>
<td>Misunderstanding SQL syntax and its internal (in)consistency</td>
</tr>
<tr>
<td>10 Thinking DISTINCT will take the first item of a list</td>
<td>[7]</td>
<td>4</td>
<td>Students mix the semantics of ‘a single’ and ‘the first’</td>
<td>Believing DISTINCT takes the first item of a list (because of being presented with very selective examples of DISTINCT queries)</td>
</tr>
<tr>
<td>11 Lack of understanding of subqueries</td>
<td>[4, 7, 46]</td>
<td>7</td>
<td>Students prefer a loop-style query, as you would do in programming</td>
<td>No specific misconception for this error is provided. Instead, they attribute this to a lack of knowledge or experience</td>
</tr>
</tbody>
</table>

For each error, we also compare to the misconception underlying this error from the student perspective, taken from [28].
Fig. 2. Expert hypotheses for using ≠ instead of != or <>.

The fact that ≠ is common in other areas of study is also reflected in the mapping of hypotheses: two of them can be classified as previous course knowledge and one under generalizations. Furthermore, one hypothesis can be mapped to lack of practice, and two to mental model.

Regarding Q2, the students and experts have similar perspectives. The fact that students have learned of the ≠ symbol months to years before learning SQL, means it is more accessible in memory than more newly introduced symbols such as <> or !=.

5.2 AVG in WHERE Instead of HAVING

Provided problem statement

**Question:** Find the names of store chains that on average sell products in quantities of more than 4.

**Error context:** This placement of AVG does not lead to a correct result as the average should be taken over the grouped attribute.

Findings for this error are displayed in Figure 3. The hypotheses cover a very wide spectrum of underlying issues. Again, we see a reflection of the “this is more common” approach, but we also see considerations of the concept of groups, the concept of aggregates, and the concept of a filter. Furthermore, a translation could be an issue, as well as plain forgetfulness. Interestingly enough, there seems to be no correlation between the background of our participants, and the type of hypothesis they present for this error.

Selection at different stages was central to many participants’ answers, for example:

“WHERE and HAVING both do some form of selection, but just at a different stage. The type of statements (Boolean combinations) also look very similar. So it is not so strange to confuse them. [ . . . ]”—Participant 2

The difference between processing individual rows and groups is the main distinction between WHERE and HAVING. This processing difference—combined with having highly similar syntax and functionality—might be confusing, so Participant 18 suggests renaming HAVING to “WHERE GROUP HAS” in the SQL syntax.

Also related to this is the “flow” or the order of processing of a query, and how students may not be familiar with it. Not knowing the order of processing and the implicit restrictions that come with it, may lead to this error. Participant 8 illustrated this by writing:

Student (partial) answer:

```
GROUP BY t.sID
FROM transaction as t
WHERE AVG(t.quantity) > 4;
```
"[...] student did not understand that WHERE clause is evaluated before GROUP BY is applied"—Participant 8

As mentioned above, these hypotheses cover a wide spectrum of issues. As a result, most of them fit into different cause categories: only two were mapped to the same category of lack of practice. The other categories of causes for this error are language, procedural fixedness, ignorance, mental model, and generalizations.

The top hypothesis of the experts and the students reflect similar thoughts. The concept of filter seems to be mainly associated with WHERE by both student and expert users. If there is a condition, students will write this in the WHERE as that is consistent with what they know. The consistency of this misconception is also reflected in the hypotheses listed 5th and 7th in Figure 3.

5.3 JOIN in All Tables that you Need the Primary Keys of

Provided problem statement

**Question:** List all customer IDs, dates, and quantities of transactions containing products named Apples.

**Error context:** Here the participant joined the customer table for cID, although cID is already available in the transaction as a foreign key.

Findings for this error are displayed in Figure 4. For the first time, we see some hypotheses that skew to neutral or even disagreement.

Most of the expert answers in round one discussed ways in which students approach the problem. Participants 2 and 16 suggest an approach that students may take:

"The thinking about the result is probably in two tiers: (1) which entities do I need, and (2) which attributes of these entities do I need. So if the question asks for the IDs of customers in the result, which it more or less does, then the reflex is to ensure that the record for customer is in the result, and from that select that the ID."—Participant 2
“The question is for the "customer" IDs. Therefore, I think the student automatically decides to use the customer table (and the transaction table for the other part of the question), because that is the table that has information on customers. Not realizing that ID’s are often stored in other tables as well. Possibly this also points to a very rigorous application of a step-by-step solution plan, without thinking what you are doing.”—Participant 16

The first two hypotheses reflect a template-based way of working. The template (or step-by-step work plan) comes out of long-term memory when it seems applicable to the question, but the reuse of queries draws from similarity to a recently answered question, which is closer to short-term memory. Therefore, one of these hypotheses is classified under generalizations, whereas the other falls under procedural fixedness, a category that is closely related because it focuses on the student becoming accustomed to writing one type of query. Other hypotheses capture lack of practice (2), mental model, and ignorance. Finally, one hypothesis is classified as not an error.

The hypothesis of the students is similar to the two most agreed-upon hypotheses by experts. They both suppose there is some query template being applied here, either from memory or from previous questions.

### 5.4 Missing Second Table for Self-join

**Provided problem statement**

**Question:** List all pairs of customer IDs who live on a street with the same name but in a different city.

**Error context:** Here, the student should have applied a self-join, but only took the customer table once.

**Student answer:**

```sql
SELECT cName, cName AS name_1, name_2
FROM customer
WHERE (street_1 == street_2
AND city_1 != city_2 );
```

Findings for this error are displayed in Figure 5. For this error, we see a high level of agreement between experts. For the first three hypotheses, none of the experts disagreed. What makes it even more interesting is that the answers represent a wide variety of hypotheses on misconceptions. We see misconceptions on tables/relations, on restrictions in the FROM clause, on the WHERE clause, and on attributes. Participants also mention possible influences of programming.
"In the world of programming languages, such a construct would be implemented with two nested iterators defined over the same (shared) data structure. Students may apply this concept to tables as well."—Participant 18

Some of the hypotheses in Figure 5 explore a more black-box mental model of the DBMS, such as the top hypothesis. They reflect incomplete knowledge on the part of the student regarding how tables are accessed. Incomplete knowledge matches the student misconception by Miedema et al. [27]. Their other proposed misconception for this error is misunderstanding the scope of elements in a query, which is mostly agreed upon by the experts.

We categorize two of the expert hypotheses as *previous course knowledge*, two as *mental model* issues, two as *ignorance* and one as *shown examples were too limited*. Finally, as Taipalus maps this error to *absence of retrieval cue*, we apply that category for the hypothesis *Students do not realize they need a JOIN*.

In the extra comment space on this error, Participant 1 mentioned:

“I would also think that because SQL allows for “implicit” joins via the WHERE clause, the students do not clearly see whether something is a filter on a row, vs. a join condition. I’d blame this on SQL’s syntax, not on the students.”

Such an error is also partially due to the teaching style of the students’ lecturers and instructors. Some are strict on teaching explicit JOINs whereas others mostly teach implicit JOINs through Cartesian products.

### 5.5 Lack of Understanding of when to Apply GROUP BY

**Provided problem statement**

**Question:** Return the stores table ordered alphabetically on the city.

**Error context:** The student added GROUP BY when it was not required, leading to an incorrect answer.

**Student answer:**

```sql
SELECT city, street, sName, sID
FROM store
GROUP BY city
ORDER BY city ASC;
```
Expert Perspectives on Student Errors in SQL

Findings for this error are displayed in Figure 6. All hypotheses for this error skew towards neutrality, with many experts both agreeing and disagreeing with each hypothesis. Perhaps the error title was too general, and we should have focused on this specific error context. Nevertheless, we received some interesting insights in the first questionnaire:

“Student expecting the relationship between GROUP BY and ORDER BY, possibly caused by the misinterpretation of examples where both co-occur (I guess it’s not unlikely to have ORDER BY clauses over columns that have been grouped because groups are often the “primary” sort key—which of course doesn’t imply that GROUP BY is a pre-requisite for ORDER BY).”—Participant 9

It is very likely that a biased selection of examples sampled by the instructor (or read by the student) introduces misconceptions; as such examples have a significant impact on the students’ mental model of the concept.

“[...] the student could have seen something was wrong with their answer by seeing that the select statement asks values from non-grouped columns, leading to a malformed query. So maybe the biggest misconception is that you can group on column A and then still project values from a different column B (depending on the DBMS this is only true if B has the same values for every identical value of A).”—Participant 16

Participant 16 focuses on how the student could have evaluated their result, and therefore fixed their problem. As the student apparently did not do that, they must have a misconception of the order of operations during SQL execution. This leads us to the previously identified misconception of misunderstanding SQL syntax and its internal (in)consistency in [27]. The student may not have realized that after applying GROUP BY, not all actions are possible anymore, which is also reflected in the top hypothesis by the experts. Furthermore, the explanation of participant 16 shows that the student did not reflect on the result table. This may show the ignorance that Taipalus maps this error to [44]. Although different wording is used in all three cases, the students and experts most likely feel similarly about this error.

Mapping of the hypotheses reveals that various themes were touched upon: we categorized two hypotheses as absence of retrieval cue, two hypotheses as shown examples were too limited, one as ignorance (as in Taipalus’ work mentioned above), one as language and one as mental model.
5.6 Missing Parts of Keywords

**Provided problem statement**

**Question:** Return the stores table ordered alphabetically on the city.

**Error context:** Missing parts of keywords (GROUP BY, ORDER BY, INSERT INTO). The student wrote ORDER instead of ORDER BY.

**Student answer:**

```
SELECT * 
FROM store AS s 
ORDER s.city ASC
```

Findings for this error are displayed in Figure 7. In round one, most of our participants mentioned students being careless, not knowing the syntax, or not remembering it correctly. For example, participant 3 writes:

“Student simply forgot the concrete syntax of SQL (where ORDER BY is a single token).” — Participant 3

The first four hypotheses in Figure 7 reflect this sentiment, although for different underlying reasons. Hypothesis 5 is the only one that falls outside of this category, but a large number of votes for (highly) unlikely shows that this is an unpopular opinion. Regardless of the hypotheses having similar sentiments, they do represent different types of underlying causes: we mapped them to **previous course knowledge, generalizations, language, lack of practice, and sloppiness**.

This is the first error that the students and experts interpret in different ways. For the experts, the focus is on commonalities (Hypothesis 1) and practice (Hypothesis 2). However, from the student’s perspective, this may have more to do with Natural Language effects. Although this hypothesis was posed by the experts, many of them disagreed with this statement.

5.7 Extra Condition when Not Required

**Provided problem statement**

**Question:** List all pairs of customer IDs who live on a street with the same name but in a different city.

**Error context:** In this case, the participant added cID <> cID even though that could not occur with our schema.

**Student answer:**

```
SELECT c1.cID, c2.cID 
FROM customer c1, customer c2 
WHERE c1.cID <> c2.cID 
AND c1.street = c2. street 
AND c1.city <> c2. city ;
```

Findings for this error are displayed in Figure 8. The most agreed-upon hypothesis reflects on either student’s conception of the schema in general or their conception of foreign keys. The other
Fig. 8. Expert hypotheses for Extra conditions when not required.

four hypotheses focus on the process of writing and editing. Participant 2 explicitly focuses on the writing process and working memory in their round one answer:

“Mental compartmentalization: the student wants to ensure that (1) the two customers are not the same customer and (2) have different city fields. And they do not realize that (1) follows from (2). What in my experience also plays a role is that in another question there was also asked for two <something> where it was important they were distinct, and where this did not follow from the additional selection criterium.”—Participant 2

The hypotheses of the experts again capture a broad range of categories: mental model, procedural fixedness, lack of practice, sloppiness, and not an error all occurred once.

An extra remark from Participant 7 leads us to believe that the hypothesis on laziness and remembering the schema could have had more experts agreeing:

“I actually kind of like parts of [...] “students forget the db schema”, but I cannot get behind the value judgment of “Laziness””—Participant 7

This is a valuable comment. In retrospect, we should have framed this hypothesis in a more discerning manner.

Regarding Q2, Miedema et al. [27] identified this as lacking knowledge on primary keys, a hypothesis that has not been explicitly mentioned by our experts. However, the primary keys are mentioned in the highest-ranking hypothesis. Therefore, although the angle is not the same, both experts and students attribute this error to primary keys.

5.8 Ambiguous Column Name Because of Missing Alias

Provided problem statement

Question: Find the names of all inventory items that have a higher unit price than Bananas.

Error context: It is not clear which pID is meant in pID=pID, because the participant did not add the alias.

Student (partial) answer:

```
SELECT unit_price
FROM inventory
INNER JOIN product
ON pID = pID
WHERE pName = "Bananas"
```

Findings for this error are displayed in Figure 9. In the first three hypotheses, the experts propose that students assume that the DBMS works similarly to a human brain. In that case, if something
is interpretable to the user, it should also be clear to the DBMS. This explains why the student missed that there is ambiguity in the query. As Participant 8 says it:

"Not double checking all of the details—assuming that the query has the same context as the query writer’s brain."—Participant 8

For this error we have only four hypotheses, over three categories: mental model (2), ignorance, and lack of practice. In the case of the hypotheses here, these three categories are closely related: a lack of practice may lead to an incorrect mental model, and ignorance may lead to similar mistakes.

The student may not have been able to evaluate the query from the correct perspective because of a lack of knowledge or experience. This is the explanation given by Miedema et al. [27] for this error. It is not a misconception in itself, but more of an explanation of why the misconception hypotheses as suggested by the experts occur.

5.9 Using Comma Instead of AND in WHERE Clause

Provided problem statement

Question: Find the names of all inventory items that have a higher unit price than Bananas.

Error context: Error occurs in the main query (comma after i.unit_price) but not in the sub-query.

Findings for this error are displayed in Figure 10. In Round 1, many of our participants mentioned that this mistake comes from sloppiness or confusion of what the student was writing where signaled by to lack of indentation. For example, participant 13 writes:

" Possibly a careless mistake. The student knows how to write a where clause but probably got confused due to the way he/she placed the subquery in the where clause before the join over pid. That is, the mistake possibly arises from poorly-structured where clause in the main query."—Participant 13

In Round 2, the experts benefited from the introduction of other experts’ opinions and found other hypotheses relatively likely too. These describe how existing knowledge interferes with learning and the writing process. We can categorize the hypotheses as follows: two hypotheses
Expert Perspectives on Student Errors in SQL

Fig. 10. Expert hypotheses for *Using a comma instead of AND in WHERE clause.*

Regarding Q2, here is another agreed-upon hypothesis that matches the findings of Miedema et al. [27]. They attribute this error to a misunderstanding of SQL syntax and its internal (in)consistency, where the inconsistency is between the SELECT, FROM, and WHERE clauses. This hypothesis is similar to the “list clause” hypothesis, which is broadly agreed upon by the panel of experts.

5.10 Thinking DISTINCT will Take the First Item of a List

**Provided problem statement**

**Question:** For this question: Find the names of all inventory items that have a higher unit price than Bananas. There was a follow-up question: What happens if there are two products named bananas?

**Student’s original query:**

```sql
SELECT B.pName
FROM inventory A, product B
WHERE A.pID = b.pID
AND B.unit_price > (SELECT C.unit_price
                    FROM inventory C, product D
                    WHERE C.pName = "Banana"
                    AND D.pID = C.pID)
```

On the question *What happens if there are two products named Bananas?* they then elaborated:

“[Suppose] There will be two prices [for two bananas]. If I would have used a DISTINCT before, on the unit_price, then you’d have taken the first unit_price and compared it with the unit_price of the primary query. [...] If I used DISTINCT over this C.unit_price, I’d have given only one unit_price and probably that would have been the first one, because that is what it will find first. And then it would just compare [the price] with that one.”

Findings for this error are displayed in Figure 11. In Round 1, most of the participants considered misunderstandings, carelessness, or low familiarity as reasons for this error. It is in such cases that the Delphi methodology is advantageous: there were some participants that went beyond conclusions of carelessness, and the inclusion of their hypotheses led our other participants to reflect on a deeper level.
Fig. 11. Expert hypotheses for Thinking DISTINCT will take the first item of a list.

For example, Participant 4 questioned the examples that were used to teach DISTINCT:

“Was the example with which they were taught distinct done using just one type of distinct value?”—Participant 4

This questioning led to the inclusion of Hypothesis 3, as did the following text from Participant 2:

“A DISTINCT does have the connotation of reducing a multitude to something smaller, so if I get many, and only want it to be one (so fewer), it sort of seems to come close to that. The student may have seen an example where a query returned a result with all duplicates, and the DISTINCT was used to reduce that to one record, which comes close to what they want here.”—Participant 2

Overall, these four hypotheses capture three categories of causes: **language** (2), **lack of practice** and **shown examples were too limited**.

We have seen this focus on examples already, suggested by our experts for the errors with GROUP BY. It was also suggested by Miedema et al. [27], who described the misconception as **Believing DISTINCT takes the first item of a list**. As the upper three hypotheses in the figure are ranked similarly, we can conclude that the students and experts agree on this hypothesis.

### 5.11 Lack of Understanding of Subqueries

**Provided problem statement**

**Question:** List all customer IDs, dates, and quantities of transactions containing products named Apples.

**Error context:** There is no need for the authors of these queries to write a subquery. It increases complexity and reduces readability.

<table>
<thead>
<tr>
<th>Student answers:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT t.cID, t.date, t.quantity</td>
</tr>
<tr>
<td>FROM transaction as t</td>
</tr>
<tr>
<td>WHERE (SELECT pName</td>
</tr>
<tr>
<td>FROM product</td>
</tr>
<tr>
<td>WHERE pID = t.pID</td>
</tr>
<tr>
<td>AND pName = &quot;Apples&quot;)</td>
</tr>
</tbody>
</table>

SELECT t.cID, t.date, t.quantity |
FROM transaction t |
WHERE t.pID in (SELECT p.pID |
FROM product p |
WHERE pName = "Apples") |

Findings for this error are displayed in Figure 12. For this section, our participants debated whether this is actually an error. Do we expect our students to consider the efficiency of their queries? Participant 18 did not think so:
"Not sure why this is considered an error. Semantically, the query is correct. Subqueries might execute slower in some DB engines, but I would not expect the students to know this."—Participant 18

Participant 6 was of the same opinion, arguing that efficiency depends on which DBMS is used:

"I do not agree with the premise of the question. The efficiency argument is not robust: this varies on different DBMSs. In fact, there are DBMSs where the subquery variant is actually more efficient, and so it might be that the student was exposed earlier to information from that context."—Participant 6

However, these participants only focus on the efficiency of this example, and did not touch upon the readability of subqueries. Two experts who did, are participants 1 and 16.

"Subqueries using identifiers from the upper-level query are hard and confusing. They are simply difficult to read."—Participant 1

"Honestly, I find (especially) the second query immensely readable. I prefer subqueries over joins mostly for readability. This is a common split among students as well. About a third typically like subqueries, while the majority prefer joins. If you consider the (previously introduced) idea that there can be multiple product with the same name, then I really like the second query."—Participant 16

These participants disagree on whether the queries presented here are readable or not, which might have to do with the query’s size.

Participants also mentioned translating of queries, either from Natural Language (Hypothesis 5) or First Order Logic (Hypothesis 7). The plot in Figure 12 does not indicate high agreement within the participant group, but it represents an interesting question nonetheless: how do students decompose a more complex problem, and where do they start the formulation of their query?

Overall, these seven hypotheses can be grouped into five categories: previous course knowledge (2), lack of practice (2), ignorance, mental model, and the new category of avoidance and database anxiety.

Regarding Q2, Miedema et al. [27] did not attribute this error to a specific misconception but instead explained this by a lack of knowledge or experience. Given the discussion by experts of
whether this should even count as an error, and whether that is DBMS-dependent or not, a lack of student insight could be a fitting explanation.

6 DISCUSSION AND CONCLUSION

In this article, we contribute by providing additions to the list of misconceptions held by students learning SQL. After validation, these misconceptions provide further insight on how we can design interventions to improve SQL education.

To recap, in this article we posed the following research questions:

Q1. How do experts interpret the underlying causes of student errors in SQL?
Q2. Which are the differences in the perspectives of students and experts on errors in SQL?

To answer these questions, we took a Policy Delphi approach, in which we asked a panel of SQL experts for their opinions on novice errors. The subsequent votes on the likelihood of each hypothesis gave us a list of possible misconceptions that we can use as a basis for the design of educational approaches to reduce the prevalence of these misconceptions, consequently improving the learning process of novices.

We answered Q1 by iterating over eleven errors made by novices, and listing the hypotheses as generated by our panel of experts. The different backgrounds of our participants, and their expertise in various aspects of SQL, led to the generation of a wide range of misconception hypotheses per question in round 1. Sometimes the participants were all focused on the same type of hypothesis but approaching it from different angles, such as in the case of Using \( \neq \) instead of \(!=\) or \(<=>\) or Using comma instead of AND in WHERE clause. Other times, the participants were divided in their perspective, such as for AVG in WHERE instead of HAVING. In that case, they suggested hypotheses related to the concepts of groups, filters, and aggregates, translation issues, or forgetfulness. Hypotheses related to forgetting, lack of experience, and lack of focus were common. However, the mistakes students make should not just be attributed to such factors.

When answering Q2, we found that in most cases, experts and students were in agreement on the underlying causes, although sometimes there was a difference in viewpoint. For eight out of eleven errors tested in this study, there was an agreement between the groups to some extent, although the student explanation may not have been the most agreed-upon expert hypothesis. For two errors, no student misconceptions were suggested and thus the question could not be answered. Finally, for error 6, Missing parts of keywords, the explanations were different. The experts focused on what is common in SQL, and on student practice, whereas the student perspective focused on Natural Language.

The method that we applied in this study resulted in capturing more input per error than presented by Miedema et al. [27]. Each error had between four and eight misconception hypotheses. In the mapping of these hypotheses to causes introduced by Miedema et al. and Taipalus [44], we found that they covered several different topics. Each error had hypotheses from three to six categories, scaling with the number of hypotheses. We found representations of all classes identified in previous work, except for working memory overload. This is likely due to the setup of the experiment, providing small snippets of information to our experts without a broader context. Beyond these classes from literature, there were representations of the classes lack of practice, sloppiness, and not an error.

Finally, in this work, we introduce two new causes of errors. The first is shown examples were too limited. This reflects a lack of depth in the material that the teachers use to introduce concepts to their students. Examples may be too simplistic, and as such do not showcase the full range of opportunities for the concept. The second is avoidance and database anxiety. This is the database
equivalent of *math anxiety*. Math anxiety has been shown to disrupt cognitive processes by monopolizing resources of working memory [5]. A similar process may occur in SQL querying when students need to apply more complex concepts such as JOINs, aggregation and subqueries. Future work should examine the extent to which database anxiety influences student performance.

### 6.1 Implications for Teaching

To improve upon our teaching of SQL, the first step is to identify where possible problems in knowledge transfer occur. This identification of faulty transfer, through the uncovering of misconceptions, is an indication of where course material and instruction need to be adapted to the students’ previous knowledge. Teachers who have an understanding of misconceptions can work towards knowledge refinement and reorganization, to build students’ knowledge of the taught concepts [39].

One central theme that came up regularly in our findings was both a lack of practice, and a lack of appropriate examples demonstrating SQL concepts. Such issues can be relieved by increasing the opportunities that students have to interact with SQL, as Participant 4 expressed:

“Students need more practical experience, just like any other programming language. We need to stop treating databases as an extension of math.”

Typically, in Computer Science curricula, there are several courses centered around programming. Students are taught about programming languages, design patterns, algorithms and data structures, and they get the opportunity to practice their programming skills through several courses and projects. Why then do we teach only the basics with regards to Databases? With the increasing need for Data Science topics in Computer Science curricula, there should be more focus on teaching students about data storage and retrieval.

### 6.2 Comparing Prior Theories and Research Findings

Existing research is conflicted about whether teachers and other experts have enough insight to discuss their students’ misconceptions. Goldman et al. find that teachers have an incomplete understanding of student learning [15]. From a practical perspective, Qian et al. discuss the latent nature of misconceptions, and how this may make it difficult for teachers to detect them [33]. Does this then mean that educators have no insight into where students may misinterpret material? Brown et al. [8] instead hypothesize that although educators are not accurate in predicting the frequency of (programming) mistakes, they may still be accurate about the cause of mistakes and a student’s conception of the mistake [8]. To validate this, we recommend finding the prevalence of the misconceptions presented in this article in a future study.

### 6.3 Strengths and Limitations

Our main limitation is the number of errors we consider in this article. As participants in any study should not be overwhelmed, and our format required open-ended questions, we decided to limit ourselves to 11 errors. Although these errors have been shown in Section 3 to be common and persistent, there are many additional errors that we could have examined, generating more hypotheses for misconceptions. Future work should consider examining misconceptions for these errors. Additional hypotheses might have also come up if we had more participants in our study.

Although our study only included two rounds versus the typical three or more of traditional Delphi, we do not think adding extra rounds would have changed our findings. An extra round could show how our participant population thought of the various hypotheses, such that
participants could change their minds. But, as our focus is on qualitative over quantitative results, the scores themselves are not the most important thing, the hypotheses and quotes are.

Instead of the Policy Delphi approach we adopted, we could have also run a focus group. An advantage of Policy Delphi over a focus group is that it allowed asynchronous “discussion” between participants on various continents, with different time zones, who may not have been able to match up schedules otherwise. Furthermore, the anonymous nature of Delphi was an advantage to make sure that each voice counted equally.

The anonymous approach of our participants with regards to filling in the survey means that we did not have the exact same group of respondents for Rounds 1 and 2. We cannot tell to which extent our groups overlapped, although the different number of respondents indicates that the groups do not match. Although Delphi is about anonymity, the participants only need to be anonymous to the other respondents, not necessarily to the researchers. However, we do not think this is a threat to validity, as our main contribution is the list of hypotheses presented above. This is a result of Round 1 only, and thus the differing groups does not have a large impact.

Finally, we could have strengthened our results for Q1 by asking the participants to indicate which group they identified most heavily with (educator, researcher, and practitioner), instead of indicating all they identified with. As a result of this choice, we could not run statistical tests on our data to see if different groups had different opinions.

6.4 Validity

In this discussion, we follow the descriptions of validity for qualitative work by Joseph Maxwell [26]. He defines the nature of five dimensions of validity: descriptive validity, interpretive validity, theoretical validity, (internal and external) generalizability, and evaluative validity [26].

First of all, the setup of the study, with participants writing down their answers in a Google Form, means we record the participants’ opinions as quotes. As such, our results are an accurate representation of our participants’ opinions, meaning our study has high descriptive validity. Using quotes also means that we do not have to interpret what our participants mean with their actions, meaning the interpretative validity of the study is high too. One area where some interpretive validity may have been lost is the aggregation step between rounds one and two. An alternative setup to avoid this would have transferred all participants’ quotes directly from the round one form to round two. However, this would have led to many overlapping answers, which would distort the hypotheses voting results. As all three authors translated and aggregated the answers in an iterative process, and the participants voted on the newly generated hypotheses, we believe the interpretative validity of the study was upheld.

Given the direct reporting of participant perspectives in this article, presented as a stepping stone towards more research on SQL misconceptions, the authors believe that the concepts of theoretical and evaluative validity may not be applicable.

Finally, the generalizability of the study has been taken into account by inviting participants of varying backgrounds. The participants are of different geographical and cultural backgrounds, have different types of jobs with various levels of experience, work at different companies and institutions, and have different ages and gender identities. This provides us with confidence regarding the external generalizability (to different communities). The study was not designed to have internal generalizability (the hypotheses do not apply to different errors), future work should consider possible misconceptions for errors that we did not discuss here. One note on the external generalizability is that student misconceptions most likely depend on the teaching method (types of examples, database schema, teacher preference etcetera). Therefore, not all misconceptions identified in a study such as ours will be applicable to all students. However, the significant hindrance that misconceptions cause, makes it important to keep all possibilities in mind.
6.5 Future Work

We consider four avenues for future work. A first avenue is to follow up on this work by validating the proposed hypotheses. This can be done by generating multiple-choice questions testing for each of the misconceptions with distractor answers that indicate the misconception. For more information on this method, we refer to the work by Treagust [48].

Second, the SQL errors could be researched from a cognitive process point of view, following a model similar to the one developed for event-based systems [22]. The causes of SQL errors, as interpreted by experts in our work, could be similarly tied to knowledge, strategic, and attentional causes for activities that relate specifically to query formulation. This line of work could highlight how these problems in the various stages of query formulation affect the novices’ problem-solving processes.

Third, additional study is needed to gain deeper insights into which misconceptions are responsible for the biggest obstacles in learning SQL and their prevalence and interactions. Furthermore, the design of materials and interventions to correspondingly address these misconceptions should be investigated. We sketch two possible ways to go about designing new materials. The first is a discussion on knowledge refinement and reorganization, as argued by Smith et al. [39]. However, discussion material is relatively difficult to design, as discussion content, and context depends on the student, the material and the teacher. Other options are counter examples and refutation texts. Refutation texts consist of either two or three elements: a misconception, an explanation of the correct concept, and (optionally) a cue [47]. The cue helps the student understand that the misconception is incorrect. These refutation texts for common SQL misconceptions can then be distributed to SQL teachers and students, improving the accuracy of SQL education.

Finally, an intrinsic complication in teaching SQL is its declarative nature. The research on programming misconceptions in imperative languages has shown that decomposition of problem descriptions is difficult for students [34]. One way in which a student can decompose a problem is to trace the execution of that problem. For SQL, this is even more difficult as execution in declarative languages cannot be traced because query plan generation is non-deterministic. This characteristic of SQL may limit the extent to which a student can generate an appropriate mental model of query execution, which in turn might lead to a “black box” perception of the DBMS. One direction of research that can help students explore query execution is to develop query visualization tools for query explanation [9, 21], interpretation [23], and building [28]. On the other hand, it could be highly useful to research a notional machine for declarative languages, to gain more insight into the link between student misconceptions and DBMS processing.

ACKNOWLEDGMENTS

The authors are immensely grateful to all experts who participated in our study.

REFERENCES


Science Education. Association for Computing Machinery, New York, NY, 272–277. DOI: https://doi.org/10.1145/2899415.2899464


[42] Ruth A. Streveler, Barbara M. Olds, Ronald L. Miller, and Mary A. Nelson. 2003. Using a delphi study to identity the most difficult concepts for students to master in thermal and transport science. In Proceedings of the ASEE Annual Conference. 4447–4454. DOI: https://doi.org/10.18260/1-2--12592


Received 29 October 2021; revised 2 July 2022; accepted 14 July 2022