Building Blocks Towards More Effective SQL Error Messages

Toni Taipalus
toni.taipalus@tuni.fi
Tampere University
Tampere, Finland
University of Jyväskylä
Jyväskylä, Finland

Hilkka Grahn
hilkka.grahn@jyu.fi
University of Jyväskylä
Jyväskylä, Finland

ABSTRACT

Reading and interpreting error messages are significant aspects of a software developer's work. Despite the importance and prevalence of error messages, especially for novices, SQL compiler error messages from various relational database management systems have seen limited development since their inception. This lack of progress may stem from the fact that it is not well-understood what constitutes an effective error message. With data from 568 participants across three student cohorts, we investigate whether novel SQL error message design guidelines can explain success in fixing SQL syntax errors. The results indicate that some of the guidelines indeed serve as building blocks toward more effective SQL error messages for novices. However, error messages that adhered to certain guidelines showed inconclusive or negative results. These findings can be applied to iterate on SQL error messages in SQL learning environments or SQL compilers.

CCS CONCEPTS

• **Applied computing** → **Education**; • **Social and professional topics** → **Computing education**; • **Information systems** → **Query languages**; • **Human-centered computing** → *Empirical studies in HCI*.

KEYWORDS

SQL, error, error message, compiler, database, database management system, relational database, human-computer interaction, computing education, novice, error message design

ACM Reference Format:
Toni Taipalus and Hilkka Grahn. 2024. Building Blocks Towards More Effective SQL Error Messages. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), July 8–10, 2024, Milan, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649217.3653552

1 INTRODUCTION

A significant aspect of software development involves reading and interpreting compiler error messages [2]. However, it is widely acknowledged that error messages are often unhelpful in resolving errors [5]. Several studies have proposed guidelines for designing general system messages [14], particularly for programming language error messages [4, 23], especially within the realm of computing education. Nevertheless, what contributes to more effective error messages remains largely unclear. Besides programming language compilers, other computer languages, such as query and markup language compilers and interpreters, produce error messages that are similarly unhelpful. SQL, in particular, falls into this category [15]. SQL is a challenging language to learn even without confusing error messages [16].

The goal of this study is to investigate whether novel SQL error message design guidelines [17] related to the clarity and comprehensibility of SQL error messages can explain the success or failure of participants in fixing SQL syntax errors. The resulting multilevel binary logistic regression model indicates that five of the nine design guidelines have a statistically significant association with the success or failure of error fixing. For instance, the model reveals that simply formatting error messages to place the most important information first increases the odds of successfully fixing a syntax error by a factor of 3.8. Additionally, the model suggests that as much as 15% of success in fixing syntax errors is attributed to individual differences. These findings can help us understand which aspects of SQL error messages assist or hinder learners in query formulation and how error messages should be crafted to cater to the needs of novice users in error recovery.

The rest of this study is structured as follows. In the next section, we discuss error messages and the novel SQL error messages design guidelines. In Section 3 we detail our data collection and participants, and in Section 4 present our resulting model. In Section 5 we discuss threats to validity, speculate the reasons behind the model, and provide practical implications of the results. Section 6 concludes the study.

2 BACKGROUND

2.1 System messages

It is rather widely accepted that system messages, i.e., natural language messages output by various programs, are problematic in many different ways [8, 12, 23]. In 1982, Shneiderman [14] introduced five guidelines for designing general system messages: *be brief*, *be positive*, *be constructive*, *be specific*, and *be comprehensible*. It seems justified to argue that the introduction of such simple guidelines propounds the view that system messages over 40 years ago were *not* brief, positive, constructive, specific, or comprehensible. Unfortunately, it seems that even these simple guidelines have not been widely adopted today [17].

Based on professional opinions and empirical research, the general system message guidelines have been further particularized

ITiCSE 2024, July 8–10, 2024, Milan, Italy

ACM ISBN 979-8-4007-0600-4/24/07.
https://doi.org/10.1145/3649217.3653552

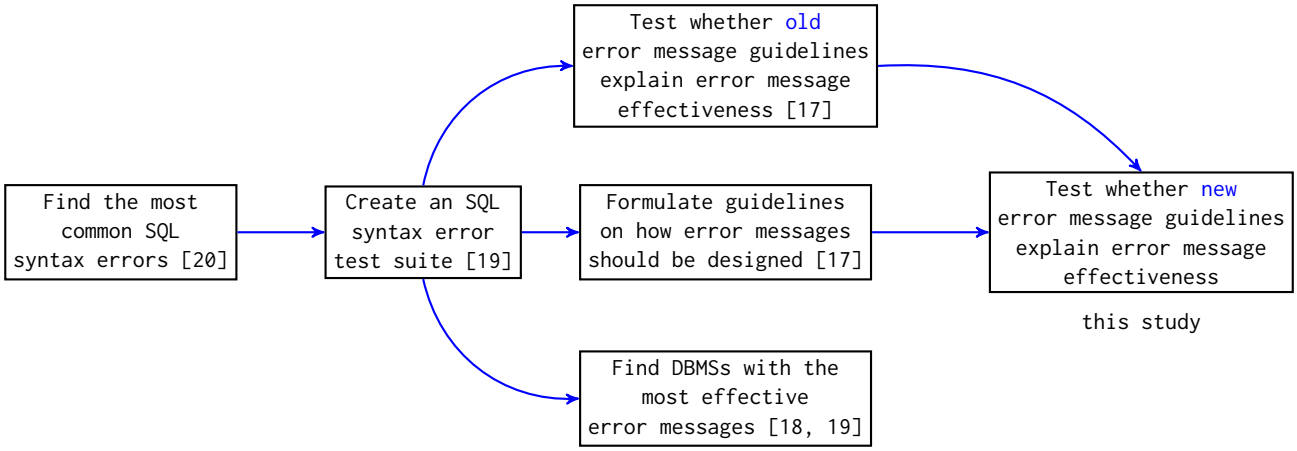

Figure 1: Research goals and questions of the most relevant prior studies leading up to this study, which is depicted in the rightmost rectangle

[23] and composed [4] especially for programming language compiler error messages. These guidelines suggest, e.g., *clarity and brevity*, using *programmer language* and *locality* of error messages [23], as well as *showing examples* and *allowing dynamic interaction* in error recovery [4]. Compared to the number of suggested improvements for programming language error messages, the number of studies providing quantitative evidence on the effects of enhanced error messages is on par with the amount of anecdotal evidence presented [4]. Empirical studies have shown that enhanced programming language error messages may reduce the number of committed errors [e.g., 3], and that the end-users may prefer enhanced error messages [e.g., 1, 21]. Many of these guidelines have similarities, but also contradict each other. However, several studies have reported inconclusive or negative results [e.g., 9, 10, 13]. Recent works have also investigated factors such as focusing on error message readability [12] and the use of large language models in enhancing programming language error messages [7].

2.2 SQL error messages

Compared to programming language error messages, error messages produced by the SQL compilers of various database management systems (DBMS) have received little scientific attention. For example, a comprehensive literature review from 2019 lists over 300 scientific studies on programming language error messages [4], while studies on SQL (or query languages in general) error messages number under a dozen.

Fig. 1 outlines particularly influential prior studies that have contributed to our study. We will discuss these studies in a chronological order, highlighting their contributions to this work. In 2018, a large-scale empirical study analysed errors that novices committed in SQL query formulation [20]. The analysis uncovered, among other things, the most frequent syntax errors, as well as syntax errors that were the most difficult to fix (i.e., the most *persistent* syntax errors). Based on these findings, a subsequent study created an SQL syntax error test suite consisting of sixteen erroneous SQL queries [19]. Each query contained one of the sixteen most persistent syntax errors. The purpose of the test suite was to enable testing of how well study participants can fix syntax errors given a corresponding error message, and how the participants would rate the error messages according to several subjective metrics.

The test suite has been used in several studies on SQL error messages. One study used the test suite to compare the error messages of four traditional relational DBMSs using several metrics such as error fixing success rate, as well as metrics pertaining to participants' subjective experiences such as error recovery confidence and error message usefulness [19]. Another study used the test suite to compare error messages of four NewSQL systems [18]. NewSQL systems are relational DBMSs that are built from the ground-up using some best practices introduced by NoSQL systems. Finally, a recent, mixed-methods study [17] used the test suite in coding the respective error messages of eight DBMSs according to Shneiderman's [14] general system message guidelines. The results indicated that the general system message guidelines poorly explain SQL error message effectiveness on syntax error fixing. The same study collected qualitative data on what an SQL error message should contain, and formulated a framework of nine guidelines for SQL error message design.

2.3 SQL error message guidelines

The framework for SQL error message design [17] consists of nine guidelines divided into four groups. The *where* group instructs the error message to *provide line number* (guideline #1) on which the error occurs. Due to the nature of SQL, line breaks are not syntactically necessary, and the guidelines also instruct to *specify error position* (#2). This is done by a cadet symbol in some DBMSs (cf. Fig. 2c, second line).

The *what* group advises that the error message should both *explain what causes the error* (#3) and *explain why the error occurs* (#4). For example, an error message should say that an erroneous SQL statement contained more than one `WHERE` clause (i.e., this caused the error), and that placing more than one `WHERE` in the same statement is not necessary, as multiple expressions should be separated with `AND` and `OR` operators (i.e., why this is an error).

```
SELECT id, name, status
FROM   project
WHERE  name LIKE ('H%', 'J%', 'K%')
AND    manager_id IN
  (SELECT id
  FROM    employee
  WHERE   sname = 'Smith');
```

(a) An erroneous SQL query (test suite test T04) [19]

```
ORA-00907: missing right parenthesis
```

(b) Oracle Database error message

```
Line 3: name LIKE ('H%', 'J%', 'K%')
                         ^
LIKE cannot be used with a list of values.

HINT: If you are using wildcards, consider using
    logical operators AND or OR between
    expressions. Alternatively, if you are not
    using wildcards, try replacing LIKE with IN.

EXAMPLES:
SELECT *
FROM product
WHERE name LIKE 'A%'
OR name LIKE 'B%';

SELECT *
FROM product
WHERE name IN ('Alpha', 'Beta');
```

(c) Iterated error message [17]

Figure 2: An erroneous SQL query and corresponding error message from Oracle Database, and a corresponding error message iterated according to the SQL error message design guidelines [17]

Additionally, *most important information should be placed first* in the error message (#5).

The *how* group guides the error message towards constructive suggestions. The error message should *provide suggestions on how to fix the error* (#6), and *provide working examples of similar query concepts* (#7). Arguably, these guidelines contradict Shneiderman's guideline of *being brief*, yet the framework justifies these based on empirical evidence.

Finally, two guidelines cover the nature of the error message as a whole. These guidelines advise to *remove unnecessary elements* (#8) such as error codes and host names, or move them to the end of the message, as per guideline #5. Finally, the guidelines instruct to *use plain English* (#9). Fig. 2 shows an example of the application of the guidelines on an SQL error message.

3 RESEARCH SETTING

3.1 Data collection instrument

The test suite [19], i.e., our data collection instrument in this study consists of sixteen tests. One test consists of five elements: *(i)* a database schema diagram representing the underlying database, *(ii)* a task in natural language, e.g., *find the IDs, names and status of projects which start with an H, J, or K, and have a manager whose surname is Smith*, *(iii)* an SQL query that corresponds to the task, but contains a syntax error, *(iv)* a respective error message yielded by a DBMS, and *(v)* a free-form text field in which the participant is asked to fix the syntax error.

The test suite can be customized in various ways. One way to customize the test suite is to incorporate error messages from different DBMSs in order to, e.g., compare the effectiveness of error messages of different DBMSs with each other, as described in Section 2.2. We chose to utilize the modified test suites reported in previous studies [18, 19], which incorporate error messages from a total of eight different relational DBMSs: MySQL, PostgreSQL, Oracle Database, SQL Server, NuoDB, CockroachDB, SingleStore, and VoltDB. Effectively, this means that we utilize eight different test suites, which differ from each other only by the error messages they contain.

3.2 Participants

We recruited study participants from a database and data management course given at the authors' university. The course followed AIS/ACM curriculum guidelines for information systems [22] course on databases, and consisted of lectures and practical exercises on topics such as Entity-Relationship modeling, database normalization, and SQL. We recruited the participants mid-course, after all SQL topics had been covered. A total of 568 participants were recruited from a total of three cohorts, n = 302, n = 184 and n = 82, respectively, an overall response rate of 78%.

3.3 Data collection

Potential participants who showed willingness to participate in the study were shown a full privacy and data collection statement prior to participation. Participation yielded no advantages or disadvantages, and was voluntary. After a participant chose to participate, they were randomly assigned to one of the eight test suites, i.e., one participant was assigned to the test suite containing error messages output by MySQL, another to a test suite containing error messages output by PostgreSQL, etc. Next, a participant was shown one test in the test suite of sixteen tests. After the participant had completed the test, the next test was shown until all sixteen tests in the assigned test suite were completed. All tests were shown in random order for each participant.

3.4 Data preparation

After data collection, we coded all 128 error messages (16 tests × 8 DBMSs) in the tests suites according to the nine error message guidelines. That is, we considered whether an error message adheres to each of the guidelines. With the exception of guideline #7 (*provide working examples of similar query concepts*), all error message guidelines were adhered to by at least one of the 128 error messages. Since guideline #7 was constant in all error messages, it was omitted from the subsequent analysis. Therefore, each error message received eight codes. The coding was binary. We also considered the correctness of each submitted, fixed SQL query by running the queries through their respective DBMSs, e.g., the queries submitted by participants in the MySQL test suite were run on MySQL to assess their correctness. If a query contained at least

Table 1: Multilevel binary logistic regression model predicting success in fixing the queries; the values in the expected odds column indicate how error messages adhering to a particular guideline explain error fixing success – values below one indicate that adhering to the guideline decreases the odds of successfully fixing errors, and values greater than one indicate the opposite; confidence intervals (CI) are for expected odds

Fixed effects	**Coefficient**	**Std. error**	***p***	**Exp. odds**	**95% CI (lower)**	**95% CI (upper)**
Intercept	1.541	.066	< .001	4.667	4.087	5.329
Remove unnecessary elements 1	-1.405	.273	< .001	.245	.144	.419
Remove unnecessary elements 0	0*					
Place the most important information first 1	1.345	.284	< .001	3.839	2.201	6.697
Place the most important information first 0	0*					
Explain why the error occurs 1	-.401	.067	< .001	.669	.587	.763
Explain why the error occurs 0	0*					
Provide suggestions on how to fix the error 1	.369	.092	< .001	1.447	1.209	1.732
Provide suggestions on how to fix the error 0	0*					
Explain what causes the error 1	.174	.065	.008	1.190	1.047	1.353
Explain what causes the error 0	0*					
Random effects	**σ^2**	**Std. error**	***p***			
Intercept (participant)	.585	.060	< .001			
Intraclass correlation (ICC)						
Participant	0.151					

*The factor above is compared to the factor that gets the value of zero (i.e., intercept)

one syntax error, the query was marked incorrect, i.e., the participant had failed to fix the query. If a query was deemed syntactically correct by the DBMS, we evaluated the query's logical correctness without any computational automation. If a query contained at least one logical or semantic error, the query was marked incorrect. If the query contained no errors, the query was marked correct. The error message coding served as independent variables (i.e., fixed factors or predictors) and query correctness as the dependent variable (i.e., predicted variable) in our subsequent analysis.

4 RESULTS

We developed a multilevel binary logistic regression model to examine the influence of the previously explained SQL error message design guidelines on the probability of query fixing accuracy. We conducted the analysis using IBM SPSS 28.0. Multilevel binary logistic regression is a statistical approach utilized when the outcome variable (i.e., success in query fixing) is binomial, with 0 indicating that the query was not fixed correctly, and 1 that the query was fixed correctly. An α-level of .05 was chosen.

The model comprises 9,088 attempts to fix queries contributed by 568 participants. In the intercept-only model, we found the intraclass correlation (ICC) to be 14.6%. This indicates the appropriateness of employing a multilevel model, as the ICC value significantly deviates from zero [cf. e.g., 11]. The ICC measures the proportion of the total variation in the outcome that can be attributed to between-group differences – in this case, differences between participants. According to the intercept-only model, the unconditional probability of a query being fixed successfully was 80.8%.

As fixed factors, we added the codings of eight error message guidelines that were previously discussed. Among these, three (namely, *providing line number*, *specifying the error position*, and *using plain English*) did not show statistical significance as predictors, and were consequently removed from the model. In the final model, the expected success rate for successful query fixing was 82.4%. The final model is presented in Table 1.

The intercept represents the baseline probability of an SQL query being fixed successfully. In this case, the odds of a successful fix are approximately 4.7 times higher than the odds of an unsuccessful fix. The strongest predictor for query success in the model was *remove unnecessary elements*. However, the negative regression slope suggests that when the error message contained no unnecessary elements, the odds of a query being fixed successfully decreased by approximately 75.5% (expected odds = .245). The second strongest predictor was *place the most important information first*, which indicates that queries with this factor have significantly higher odds (3.839) of being fixed successfully. The third strongest predictor was *explain why the error occurs*. However, the regression slope for this variable is negative, meaning that when this variable is present in the error message, it decreases the likelihood of successful fix. The next predictor, *provide suggestions on how to fix the error*, had a positive regression slope. This suggests that when suggestions in the error messages are present, the odds of successfully fixing the query are 1.447 times higher compared to when suggestions are not present. The final predictor in the model is *explain what causes the error*, which indicates that with an explanation in the error messages, the odds of fixing the query successfully are 1.190 time higher that without an explanation. The ICC in the final model was 0.151, indicating that approximately 15.1% of the total variation in query fixing success is due to differences between participants.

5 DISCUSSION

5.1 General discussion

Overall, the results show that SQL error messages that follow some guidelines facilitate successful error recovery, while following other guidelines decrease query fixing success. Arguably, some of these results appear counter-intuitive. Such observation may be explained by the ICC, which suggests that a significant amount of successful error recovery rests on individual differences, i.e., some error message design guidelines help some learners while hindering others.

Designing error message design guidelines is difficult due to complexity of the tasks and individual differences. Despite the fact that some design guidelines are formulated based on empirical data, the guidelines may be in conflict with other guidelines or even with themselves [6]. For example, Shneiderman's [14] guidelines for *constructiveness* and *brevity* may be seen to contradict each other. It has been suggested that different design guidelines are not to be followed strictly, but for limiting design space away from unusable systems [6, p. 259]. In other words, design guidelines provide perspective for the designer to consider important aspects of error messages.

5.2 Implications guideline-by-guideline

5.2.1 Provide line number. The lack of statistical significance suggests that line number alone may not explain error fixing success. It is possible that other factors play a more crucial role here, or that the relatively short and simple SQL statements in the test suite diminished the importance of line numbers in the error messages. In order to assess this factor further, the SQL statements in the test suite could be made more complex, while still keeping the syntax errors the same. This could be done by increasing the complexity of the database schema and the tasks.

5.2.2 Specify error position. Similar to the previous guideline, the lack of statistical significance here might indicate that participants rely on other cues to identify error location, or that the relative simplicity of the SQL statements in the test suite diminish the impact of this factor. It may be that some of the error messages implied the error position (e.g., *syntax error near ","*), and since the erroneous SQL statement contained commas only in the `SELECT` clause, finding the error position was relatively easy.

5.2.3 Explain what causes the error. This factor increasing the odds of success by 1.190 suggests that explaining the cause of the error helps participants understand the underlying issue better. This additional context likely aids in more accurate and efficient error resolution.

5.2.4 Explain why the error occurs. The decrease in the odds of success by 1.331 (i.e., expected odds = .669) could imply that overly detailed explanations about why the error occurred might confuse participants or distract them from the task of fixing the error. It is possible that a balance between clarity and brevity is essential in such explanations. However, the logistic regression model built in a previous study [17] indicated that error message *brevity* (as per Shneiderman's definition [14]) fails to explain SQL error message fixing success in 15 out of 16 cases.

5.2.5 Place the most important information first. The significant increase in odds by 3.839 suggests that organizing error messages with the most critical information upfront greatly benefits novice participants. According to the qualitative analysis that formed the basis for the error message design guidelines, the most important pieces of information are the error position, presented as a line number or by replicating the line on which the error occurs, as well as the specific error position, as well as information on what causes the error [17].

5.2.6 Provide suggestions on how to fix the error. This factor increasing the odds of success by 1.447 aligns with the arguably intuitive idea that offering concrete solutions or hints for fixing the error can guide participants effectively. Such guidance likely reduces the trial-and-error aspect of error recovery.

5.2.7 Provide working examples of similar query concepts. Since none of the coded 128 error messages adhered to this guideline, the coding in the data was constant, and this factor was omitted from the model entirely. According to the practical examples of the applications of this guideline, the guideline does not instruct *dynamic* suggestions, i.e., the suggestions provided in the new error messages are concrete SQL examples, but not necessarily related to the current task but rather to the query concepts such as using `LIKE` or `GROUP BY`.

5.2.8 Remove unnecessary elements. The decrease in the odds of success by 1.755 (i.e., expected odds = .245) suggests that the removal of potentially helpful information may hinder participants. It is possible that some non-essential-deemed details can still aid in understanding and resolving the error. What separates many of the eight DBMSs studied here is that error messages of PostgreSQL, CockroachDB and VoltDB contain no error codes or environmental variables. The error messages of the rest of the DBMSs either always or sometimes contain error codes, and these error codes are always placed at the beginning of the error message. One reason why adherence to this guideline does not positively affect error fixing success might be that these unnecessary elements are merely small parts of the error message. The examples of iterated error messages [17] never contain error codes, possibly due to the fact that the iterated error messages are not produced by any particular DBMS, but rather general SQL error messages.

5.2.9 Use plain English. The lack of statistical significance here could indicate that, in this specific research context, the use of plain English in error messages does not have a measurable impact on participants' ability to fix syntax errors. It is possible that the novice participants in this study were accustomed enough to technical SQL terms and were able to work with the error messages regardless of whether technical or plain language was used.

5.3 Practical implications

To our understanding, this is the first time multilevel modeling has been applied in database education research, and we encourage computing education researchers to utilize the method in understanding the effects of individual student differences. In this study, the increase in ICC from 14.6% to 15.1% suggests that the explanatory variables included in the multilevel regression model explain

some of the variation in query fixing success, but there are still individual differences among participants that affect the outcome. The increase signifies that the model accounts for more of the variation, but some of the variation remains unexplained, which is typical in complex datasets. This simply means that some students are naturally better at fixing errors. Yet, considering the predictors and the increase in ICC, this means that while some students are naturally better at fixing errors, some students do better because they receive better error messages. However, there are still some differences we cannot explain. Perhaps the liveliness of the research field surrounding error messages in general explains how difficult it is to understand what is an effective error message, despite how easy it seems to point out what is problematic in error messages.

Despite the indications that subjective metrics of what is perceived helpful in an SQL error message seem to correlate with objective metrics such as error fixing success [18], the results of this study may indicate that the aspects novices seem to value in SQL error messages may not necessarily all be aspects that facilitate error fixing success. We speculated in Section 5.2 that some of the results may be explained by relatively simple SQL statements in the test suite. It has also been speculated that binary error fixing success may not be the most fitting metric to measure error message effectiveness [18, 19], and using e.g., time taken to fix errors may yield more detailed results.

5.4 Limitations and threats to validity

A threat to validity regarding the test suite is that it introduces an unnatural environment in which errors are fixed. Typically, a query writer engages in a feedback loop with the DBMS by writing a query, receiving an error message, and repeating the process until the query compiles and is syntactically and logically correct. With the test suite, participants do not write the initial queries themselves, but the queries are provided by the test suite. Additionally, participants do not receive feedback whether they fixed the query correctly. However, it is understandable that by letting participants formulate the initial queries, it would be more difficult to control that the participants indeed commit syntax errors. This would, in turn, introduce more confounding variables.

Although this study merely considers the error message qualities of eight relational DBMSs, these DBMSs arguably represent some of the most popular traditional relational and NewSQL systems. All the test suites used in this study are publicly available as supplementary files of previously published studies [18, 19].

6 CONCLUSION

Despite the age of several SQL compilers, it remains unclear what makes an SQL error message effective. In this study, we explored whether adhering to specific SQL error message design guidelines plays a role in the effectiveness of SQL error messages and whether this adherence can predict the success or failure of fixing SQL syntax errors. The results indicate that placing the most important information first in the error message, providing suggestions on how to fix the error, and explaining what causes the error contribute to error messages that enhance the success of fixing SQL syntax errors. The results suggest that when error messages adhere to these guidelines, they facilitate successful error fixing. However, the study results do not fully explain the role of some guidelines, raising the possibility that either some of the guidelines are ineffective, the test suite statements might be too simple, or that error fixing success may not be a suitable metric to measure the effectiveness of the error message design guidelines. The results yielded by this study may be utilized in formulating more effective SQL error messages in SQL compilers and learning environments.

ACKNOWLEDGMENTS

The authors thank all who participated in the study, and the participants of the *Human Factors Impact of Programming Error Messages (22052)* Dagstuhl Workshop for fruitful discussions.

REFERENCES

[1] Titus Barik, Denae Ford, Emerson Murphy-Hill, and Chris Parnin. 2018. How Should Compilers Explain Problems to Developers?. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Lake Buena Vista, FL, USA) *(ESEC/FSE 2018)*. Association for Computing Machinery, New York, NY, USA, 633–643. https://doi.org/10.1145/3236024.3236040

[2] Titus Barik, Justin Smith, Kevin Lubick, Elisabeth Holmes, Jing Feng, Emerson Murphy-Hill, and Chris Parnin. 2017. Do Developers Read Compiler Error Messages?. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE. https://doi.org/10.1109/icse.2017.59

[3] Brett A. Becker. 2016. An Effective Approach to Enhancing Compiler Error Messages. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 126–131. https://doi.org/10.1145/2839509.2844584

[4] Brett A. Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J. Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter-Michael Osera, Janice L. Pearce, and James Prather. 2019. Compiler Error Messages Considered Unhelpful: The Landscape of Text-Based Programming Error Message Research. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education* (Aberdeen, Scotland Uk) *(ITiCSE-WGR '19)*. Association for Computing Machinery, New York, NY, USA, 177–210. https://doi.org/10.1145/3344429.3372508

[5] Brett A. Becker, Cormac Murray, Tianyi Tao, Changheng Song, Robert McCartney, and Kate Sanders. 2018. Fix the First, Ignore the Rest: Dealing with Multiple Compiler Error Messages. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 634–639. https://doi.org/10.1145/3159450.3159453

[6] A. Dix, J. Finlay, G. D. Abowd, and R Beale. 2005. *Human-Computer Interaction*. Prentice-Hall.

[7] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A. Becker. 2023. Using Large Language Models to Enhance Programming Error Messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 563–569. https://doi.org/10.1145/3545945.3569770

[8] Davin McCall and Michael Kolling. 2014. Meaningful categorisation of novice programmer errors. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. IEEE. https://doi.org/10.1109/fie.2014.7044420

[9] Marie-Hélène Nienaltowski, Michela Pedroni, and Bertrand Meyer. 2008. Compiler Error Messages: What Can Help Novices? *SIGCSE Bull.* 40, 1 (2008), 168–172. https://doi.org/10.1145/1352322.1352192

[10] Raymond S. Pettit, John Homer, and Roger Gee. 2017. Do Enhanced Compiler Error Messages Help Students? Results Inconclusive. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (Seattle, Washington, USA) *(SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 465–470. https://doi.org/10.1145/3017680.3017768

[11] James L. Peugh. 2010. A practical guide to multilevel modeling. *Journal of School Psychology* 48, 1 (2010), 85–112. https://doi.org/10.1016/j.jsp.2009.09.002

[12] James Prather, Paul Denny, Brett A. Becker, Robert Nix, Brent N. Reeves, Arisoa S. Randrianasolo, and Garrett Powell. 2023. First Steps Towards Predicting the Readability of Programming Error Messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 549–555. https://doi.org/10.1145/3545945.3569791

[13] James Prather, Raymond Pettit, Kayla Holcomb McMurry, Alani Peters, John Homer, Nevan Simone, and Maxine Cohen. 2017. On Novices' Interaction with Compiler Error Messages: A Human Factors Approach. In *Proceedings of the*

2017 ACM Conference on International Computing Education Research (Tacoma, Washington, USA) *(ICER '17)*. Association for Computing Machinery, New York, NY, USA, 74–82. https://doi.org/10.1145/3105726.3106169

[14] Ben Shneiderman. 1982. Designing computer system messages. *Commun. ACM* 25, 9 (1982), 610–611. https://doi.org/10.1145/358628.358639

[15] Toni Taipalus. 2023. Query Execution Plans and Semantic Errors: Usability and Educational Opportunities. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI EA '23)*. Association for Computing Machinery, New York, NY, USA, Article 239, 6 pages. https://doi.org/10.1145/3544549.3585794

[16] Toni Taipalus. 2023. SQL: A Trojan Horse Hiding a Decathlon of Complexities. In *Proceedings of the 2nd International Workshop on Data Systems Education: Bridging Education Practice with Education Research* (Seattle, WA, USA) *(DataEd '23)*. Association for Computing Machinery, New York, NY, USA, 9–13. https://doi.org/10.1145/3596673.3603142

[17] Toni Taipalus and Hilkka Grahn. 2023. Framework for SQL Error Message Design: A Data-Driven Approach. *ACM Trans. Softw. Eng. Methodol.* (2023). https://doi.org/10.1145/3607180 Just Accepted.

[18] Toni Taipalus and Hilkka Grahn. 2023. NewSQL Database Management System Compiler Errors: Effectiveness and Usefulness. *International Journal of Human–Computer Interaction* (2023), 1–12. https://doi.org/10.1080/10447318.2022.2108648 arXiv:https://doi.org/10.1080/10447318.2022.2108648

[19] Toni Taipalus, Hilkka Grahn, and Hadi Ghanbari. 2021. Error messages in relational database management systems: A comparison of effectiveness, usefulness, and user confidence. *Journal of Systems and Software* 181 (2021), 111034. https://doi.org/10.1016/j.jss.2021.111034

[20] Toni Taipalus, Mikko Siponen, and Tero Vartiainen. 2018. Errors and Complications in SQL Query Formulation. *ACM Transactions on Computing Education* 18, 3, Article 15 (2018), 29 pages. https://doi.org/10.1145/3231712

[21] Emillie Thiselton and Christoph Treude. 2019. Enhancing Python Compiler Error Messages via Stack. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–12. https://doi.org/10.1109/ESEM.2019.8870155

[22] Heikki Topi, Kate M. Kaiser, Janice C. Sipior, Joseph S. Valacich, J. F. Nunamaker, Jr., G. J. de Vreede, and Ryan Wright. 2010. *Curriculum Guidelines for Undergraduate Degree Programs in Information Systems*. Technical Report. New York, NY, USA. https://dl.acm.org/citation.cfm?id=2593310

[23] V. Javier Traver. 2010. On Compiler Error Messages: What They Say and What They Mean. *Advances in Human-Computer Interaction* (2010), 1–26. https://doi.org/10.1155/2010/602570