International Journal of Human–Computer Interaction

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/hihc20

NewSQL Database Management System Compiler Errors: Effectiveness and Usefulness

Toni Taipalus & Hilkka Grahn

To cite this article: Toni Taipalus & Hilkka Grahn (2022): NewSQL Database Management System Compiler Errors: Effectiveness and Usefulness, International Journal of Human–Computer Interaction, DOI: 10.1080/10447318.2022.2108648

To link to this article: https://doi.org/10.1080/10447318.2022.2108648

INTERNATIONAL JOURNAL OF HUMAN-COMPUTER INTERACTION
https://doi.org/10.1080/10447318.2022.2108648

NewSQL Database Management System Compiler Errors: Effectiveness and Usefulness

Toni Taipalus and Hilkka Grahn

University of Jyväskylä, Jyväskylä, Finland

ABSTRACT

Modern database management is often faced with a high number of concurrent end-users, and the need for database distribution to ensure fault tolerance and high throughput. To flexibly address these challenges, many modern database management systems (DBMS) provide highly automated and effortless, i.e., highly usable database distribution, deployment, and maintenance. However, the usability considerations are yet to extend from the aforementioned DBMS features to query language compilers. In this study, based on participant answers ($N = 157$), we compare the error message qualities of four modern DBMSs (CockroachDB, SingleStore, NuoDB, and VoltDB) using one objective and three subjective metrics. Our results show that some of the DBMSs provide the users with more useful error messages, even though many of these error messages violate even the most basic usability guidelines. These results (i) are applicable in further developing the usability aspects of query language compilers, (ii) provide a timely effort of bridging the gap between human-computer interaction and query language compilers, and (iii) offer suggestions on teaching novices, who require emphasized support in query formulation.

1. Introduction

Error messages are crucial for fixing errors in queries, yet errors are difficult to fix because of error messages' poor usability (Traver, 2010). Several decades ago, scholars have pointed out that especially novice users feel "*confused, dismayed, and discouraged from continuing*" when encountering confusing or even aggressive system messages (Shneiderman, 1982). The usability aspects of compilers and error messages have received ample scientific attention (Becker et al., 2019), but this attention has not been extended from programming languages to query languages. As the query language is an integral part of the process of retrieving data from a database, it is crucial that the query is written without errors. Furthermore, information retrieval from databases is an important topic in human-computer interaction (HCI) research. The increasingly emphasized role of data in information systems has led to the emergence of nascent subfields, such as human–data interaction (Victorelli et al., 2020).

At the same time, the importance of data is increasingly highlighted in rapidly growing fields, such as data mining and machine learning. Additionally, the rise of the highly competitive market of web and mobile applications has pressured technical data management solutions to meet demands for the high number of concurrent users, high volume and velocity of data, as well as high reliability (Ramakrishnan, 2012). Consequently, a large portion of data management has moved to cloud environments, which enable rapid prototyping, cost–efficiency, and automated resource allocation on demand (Buyya et al., 2019). Furthermore, the information technology field and related skills are becoming more and more common, and basic software development is introduced earlier and earlier as well as more and more broadly into various curricula (Lédeczi et al., 2021; Szabo et al., 2019). As the ubiquitousness of the information technology field is increasing, expert systems, such as DBMSs need to be accessible for novices as well as experts (Nicolaos & Katerina, 2015; Sobiesiak et al., 2002). As such, many vendors behind modern, distributed database management systems (DBMS) have made DBMS deployment and database distribution flexible, automated, and effortless for software developers (Hacigumus et al., 2002).

Given these considerations, it remains unclear whether usability extends from features, such as automated and flexible database distribution to other aspects of DBMSs. To this end, we set out to compare NewSQL database management systems from a scientifically neglected point of view of query language compiler usability. Specifically, we compare 16 retrieval query syntax error messages of CockroachDB, SingleStore, NuoDB, and VoltDB using error fixing success rate, error recovery confidence, and perceived usefulness of the error message for finding and fixing the error as performance metrics. Our results reveal modern DBMSs with many compiler errors designed against 40 year old HCI best practices, as well as statistically significant differences in

CONTACT Toni Taipalus toni.taipalus@jyu.fi University of Jyväskylä, Jyväskylä, Finland

error message usefulness between these four modern DBMS compiler errors.

The rest of this study is structured as follows. In the next section, we discuss the theoretical background and prior studies on data management, usability, and query languages. In Section 3 we describe our research setting and data collection, and state our hypotheses. In Section 4 we present our results from statistical analyses and in Section 5 the implications of our research, and some recommendations for the DBMS industry. Section 6 concludes our study.

2. Theoretical background

2.1. Data management in the cloud

Cloud computing is a growing technology model built around providing a high-level abstraction of distributed computing, usually offered as a subscription-based service to the end-user (Abbasi et al., 2019; Buyya et al., 2019). Effectively, the end-user pays for resources they utilize (e.g., storage space, computation, and network bandwidth), rather than investing in hardware, software and infrastructure outright. Consequently, some of the reasons for the popularity of the cloud computing model are the speed of deployment, scalability of computing resources, and cost-efficiency (Buyya et al., 2019). Depending on the cloud service provider, different service models are offered. These service models typically dictate which parts of the system are provided and maintained by the service provider, and which parts by the end-user. For example, the service provider may merely provide the infrastructure and (often virtual) hardware, the aforementioned complemented by an operating system, or all the aforementioned complemented by a database management system (Somu et al., 2017). Depending on their requirements, the end-user may choose a high level of abstraction while discarding control over low-level configurations. In contrast, by choosing a low level of abstraction, the end-user retains control—and responsibility—of low-level tasks, such as maintaining the operating system.

The role of the relational data model, SQL, and traditional relational DBMSs (RDBMS), such as Oracle Database, IBM DB/2, and Microsoft SQL Server has been challenged in the 2000s by new data models and query languages of numerous NoSQL data stores (Grolinger et al., 2013). While RDBMSs have favored data consistency at the cost of availability and transaction performance (Chaudhry & Yousaf, 2020; Pavlo & Aslett, 2016), many NoSQL data stores have been designed the other way around to serve, e.g., web applications with requirements for low response time and a high number of concurrent end-users (Ramakrishnan, 2012). In the 2010s, however, the industry leaders, such as Google deemed transaction support, data consistency, and the SQL language important enough to design a new DBMS to incorporate features from both traditional RDBMSs and NoSQL data stores (Corbett et al., 2013). In general, modern (i.e., in this case initially released after 2010) online transaction processing DBMSs that use the relational model, SQL, and distributed architecture are called NewSQL DBMSs. A recent study (Pavlo & Aslett, 2016) further defines NewSQL DBMSs as systems built from the ground up, rather than extensions or modifications of existing systems. The study concludes that while NewSQL systems do not offer new features or innovations per se, they skillfully integrate tested techniques into single systems. That is, *"NewSQL database systems are not a radical departure from existing system architectures but rather represent the next chapter in the continuous development of database technologies"* (Pavlo & Aslett, 2016, p. 53). When choosing a set of NewSQL DBMSs for this study, we adopt the definition of NewSQL systems provided above (Pavlo & Aslett, 2016).

The distributed architectures of NoSQL and NewSQL systems are a natural fit for cloud environments (Grolinger et al., 2013). While traditional RDBMSs also support database distribution and are offered by cloud service providers, some studies consider traditional RDBMS distribution difficult for various reasons (Pavlo & Aslett, 2016; Stonebraker, 2010). In practice, the new distribution implementations provide automatic distribution of data, automated data balancing between the distributed nodes, and with heterogeneous distribution models, automated primary/secondary elections during faults or other topology modifications.

2.2. Query language usability

According to the ergonomics of human-system interaction standard, "*Usability is relevant to regular ongoing use, to enable users to achieve their goals effectively, efficiently and with satisfaction; learning, to enable new users to be become effective, efficient and satisfied when starting to use a system, product or service*" (ISO, 2018).

Usability is a recurring theme in the evolution of cloud data management, and one of the main reasons behind new database distribution implementations was rooted in usability considerations (Shi et al., 2010; Stonebraker, 2010). First, arguably, in addition to performance and flexible scalability, the need for dynamic database schemas is one of the defining characteristics of many NoSQL data models. Dynamic schemas absolve the software developer from defining a strict database structure. Second, the need to return to strong transactional capabilities with NewSQL systems can be seen as a need to abstract the implementation of database transactions from the software developer to the DBMS. Finally, the abstraction of computer infrastructure, hardware, and partial software through cloud services all serve the demand for usability through cost-efficiency, flexibility, and rapid prototyping. On logical grounds, it seems interesting whether the demands for usability are also considered in other aspects of cloud database management systems, e.g., in compiler error messages, as usability, in general, has been argued to facilitate cost-efficiency through, e.g., improved productivity, reduced training, and documentation costs, lower support costs, and competitive edge (Donahue, 2001).

Data management solutions in cloud environments utilize several query languages and data models. Traditional RDBMSs and NewSQL systems utilize an implementation of SQL, while

NoSQL systems each usually have a distinct query language, e.g., Neo4j's Cypher (Francis et al., 2018), or Cassandra's SQL-based CQL (Wang & Tang, 2012). These proprietary query languages are sometimes complemented by SQL. As these NoSQL languages are designed for different data models and have different levels of expressiveness, usability comparison of different query languages is arguably problematic. Further, as some traditional RDBMSs offer implementations dating across four or five decades, we deemed it more interesting to focus on the usability of systems developed from the ground up in the last decade. As a contrasting example, Oracle Database 8i documentation from 1998 listed the same SQL error messages as Oracle Database 21c from 2021 (Oracle Corporation, 2021).

SQL is a language initially designed for data retrieval. However, in the decades following the initial release of the SQL standard, the language has evolved to encompass data manipulation, database structure definition, access control, and transaction management (Chamberlin, 2012). Data retrieval remains the most studied aspect of SQL (Taipalus & Seppänen, 2020), and because of this more established research background, this study focuses solely on data retrieval.

Possibly due to the increasingly ubiquitous nature of data, and the rising popularity of data analytics and data science, query languages, SQL in particular, have received increasing scholarly attention (Taipalus & Seppänen, 2020). Current educational research seems rather unanimous with the view that learning SQL is difficult (Miedema et al., 2021; Shin, 2020; Taipalus & Perälä, 2019). Usability concerns in query formulation have been explained by human factors, such as cognitive load (Shin, 2020; Smelcer, 1995), data model and real-world mismatch (Borthick et al., 2001; Sutcliffe et al., 2000), and different user characteristics (Ashkanasy et al., 2007; Bak & Meyer, 2011). Additionally, it has been shown that different environmental aspects, such as database complexity (Taipalus, 2020a) and database representation (Shin, 2020; Siau et al., 2004) have an effect on query writing. Finally, different measures for engaging and helping the end-user have been proposed in scientific literature, e.g., query visualization and previews (Taipalus, 2019; Tanin et al., 2000), cosmetic alterations (Dong & Khandwala, 2019), different natural language interfaces (Ribeiro & Moreira, 2003), and the facilitation of query reuse (Allen & Parsons, 2010; Toorn et al., 2022). However, error message research has not extended from programming languages to query languages, and the latest studies on the effects of SQL compiler error messages on query formulation seem to be published in the 1980s (Reisner, 1981; Welty & Stemple, 1981), until a recent comparison of SQL compilers of traditional RDBMS in 2021 (Taipalus et al., 2021). The differences in the SQL standard (ISO/IEC, 2016a, 2016b) between the 1980s and 2020s, as well as differences between SQL and imperative programming languages, and the potential threats to the generalizability of scientific results induced thereof have been highlighted in a previous study (Taipalus & Seppänen, 2020). Regarding usability, due to its declarative nature, SQL is arguably a "blacker box" to a software developer than an imperative programming language.

2.3. Error messages and error recovery

A large number of studies have shown the importance of compiler error messages for learning, and for software development in general in the context of programming languages (Becker et al., 2016, 2019; Wrenn & Krishnamurthi, 2017). The same studies have also argued that current compiler error messages are ineffective due to several reasons. From the perspective of error messages, DBMSs have a query parser that checks the syntax of the query and outputs an error message if necessary (Hellerstein et al., 2007). In the scope of this study, the query parser is the component that separates the usability aspects of different DBMSs from each other. Additionally, some DBMSs, such as MySQL allows pluggable storage engines that can be switched with relative ease. The storage engine typically contains the query parser, and thus the storage engine is often responsible for generating the SQL error messages. It is worth noting that while SingleStore is a NewSQL system, it utilizes the InnoDB storage engine also utilized by MySQL.

When an end-user, e.g., a software developer, writes an erroneous query and submits it to a DBMS, the DBMS outputs an error message. This is often referred to as *error detection* (van der Schaaf, 1995; Zapf & Reason, 1994). Next, the end–user tries to interpret the error message and find the erroneous part of the query. This phase is called *explaining*. Finally, the end-user attempts to *fix* the error, typically based on the feedback provided by the error message. This process of three phases is called *error recovery* (van der Schaaf, 1995; Zapf & Reason, 1994), and serves as a theoretical foundation for our chosen subjective metrics, i.e., *error recovery confidence*, and *error message usefulness for finding* and *fixing the error*.

A seminal study published in 1982 suggests that computer error messages should be "*brief, positive, constructive, specific, comprehensible*" (Shneiderman, 1982, p. 611), *positive* referring to refraining from using words, such as "*illegal, invalid, error*" in the error message, and *constructive* referring to hints or suggestions on the causes of the error and how to fix it. Considering that the `WHERE` clause is one of the most common SQL clauses, Figure 1 shows an SQL query with a simple typographical error in the keyword `WHERE`, and seven corresponding error messages from traditional RDBMSs and NewSQL systems. As software developers, especially novices, often consider the compiler the first authority in determining the quality of written software, the error messages in Figure 1 succeed in neither communicating why the query is erroneous nor adhering to all the suggestions presented 40 years ago.

3. Research setting

3.1. Study scope

In the previous section, we discussed the importance of effective error messages in the context of programming languages, and that prior works have attempted to explain and enhance said error messages to facilitate more effective software development. We also argued for the usability concern, which seems to be one of the driving factors behind the popularity of cloud environments and more effortless database distribution. However, in light of previous scientific

```
SELECT   name, price_usd, brand, model
FROM     product
WHRE     (brand LIKE 'S%' OR brand LIKE
    'C%')
AND      picture IS NULL
ORDER BY name DESC;
```

(a) Query with a typographical error (WHRE instead of WHERE)

```
Msg 321, Level 15, State 1, Server
    q7410, Line 4
"brand" is not a recognized table hints
     option.
```

(b) SQL Server error message

```
ERROR : syntax error at or near "LIKE"
LINE 3: WHRE (brand LIKE 'S%' OR brand
    LIKE 'C%')
                  ^
```

(c) PostgreSQL error message

```
ORA-00933: SQL command not properly
    ended
```

(d) Oracle Database error message

```
invalid syntax: statement ignored: at
    or near
"like": syntax error
DETAIL: source SQL:
SELECT   name, price_usd, brand, model
FROM     product
WHRE     (brand LIKE 'S%' OR brand LIKE
    'C%')
               ^
HINT: try \h <SOURCE>
```

(e) CockroachDB error message

```
SQL error while compiling query: SQL
    Syntax error in
"SELECT   name, price_usd, brand, model
FROM     product
WHRE     (brand LIKE 'S%' OR brand LIKE
    'C%')
AND      picture IS NULL
ORDER BY name DESC;"
unexpected token: LIKE required: )
```

(f) VoltDB error message

```
ERROR 1064 ER_PARSE_ERROR: You have an
    error in your SQL syntax; check the
    manual that corresponds to your
    MySQL server version for the right
    syntax to use near
'(brand LIKE 'S%' OR brand LIKE 'C%')
AND      picture IS NULL
ORDER BY name DE' at line 3
```

(g) SingleStore (with InnoDB storage engine) error message

```
Error 42000: syntax error on line 3
WHRE (brand LIKE 'S%' OR brand LIKE 'C
    %')
     ^ expected end of statement got
    parenthesis
```

(h) NuoDB error message

Figure 1. Erroneous query with a simple typographical error (a), and seven corresponding error messages generated by seven different DBMSs; three traditional RDBMSs (b–d), and four NewSQL DBMSs (e–h).

literature, and a preliminary inspection of DBMS error messages (Figure 1), it seems that even SQL compilers of modern DBMSs do not necessarily account for usability concerns or error message design guidelines and that the topic has not received much scientific attention in recent decades.

As explained in Section 2, we deemed comparing DBMSs utilizing SQL with DBMSs utilizing some other query language difficult for internal validity. On the other hand, a recent study (Taipalus et al., 2021) compared SQL compiler usability of traditional RDBMS. For these reasons, in this study, we chose to focus on NewSQL systems using the SQL compiler usability framework reported in a previous study (Taipalus et al., 2021). We deemed it more interesting to focus on popular NewSQL systems, even though measuring *popularity* is rather difficult. Based on three NewSQL studies (Kaur & Sachdeva, 2017; Pavlo & Aslett, 2016; Schreiner et al., 2019), we identified four popular NewSQL database management systems for this study: CockroachDB (v19.2.2), SingleStore (7.0.10, previously known as MemSQL), NuoDB (build 4.0.4-2), and VoltDB (Community 9.2.2). All these systems implement relational or semi-relational data models, use SQL as their query language, and are built from the ground up in the 2010s (Grolinger et al., 2013). Additionally, DB-Engines[1] ranks these four DBMSs high in popularity among NewSQL systems, when NewSQL systems are defined as in Section 2.1. In regard to different types of errors, we focus on syntax errors, and based on a previously reported framework (Taipalus et al., 2018), we focus on the 16 most common syntax errors in SQL queries. These previously reported syntax errors and our corresponding tests are reported in Table 1. These tests and queries within are in turn based on those reported in a previous study (Taipalus et al., 2021), but adjusted to account for the chosen four NewSQL systems. In the next subsections, we detail the data collection, hypotheses, and analyses, which are summarized in Figure 2.

3.2. Data collection

To focus on the differences in the selected DBMS usability in fixing erroneous queries, we chose not to use database experts as participants. We speculated that experts might

have former experience on one or several of the DBMSs studied and that their expertise would result in successful error fixing regardless of the error message, thus skewing the results toward a ceiling effect (i.e., results are not statistically significantly different because tests were too easy for selected participants). Furthermore, experts are arguably less dependent on the error messages, and more able to fix erroneous queries regardless of the error message. Because we wanted to specifically study the effects of different compiler error messages, we recruited our study participants from a database management course given at the authors' university. The participants majored in software engineering or information systems science and had acquired basic SQL knowledge from the course. The students were promised extra course points for taking the survey. Taking the survey was not mandatory, and if a student also chose to do so, their answers were anonymized and used in this study. Participating in the study was not required for extra course points, and the students were shown a full privacy statement before answering. Out of the 188 students who answered the survey, 157 (84%) chose to participate in the study.

Next, a participant was randomly assigned to one of the four database management system groups—i.e., CockroachDB ($n = 32$), NuoDB ($n = 44$), SingleStore ($n = 39$), and VoltDB ($n = 44$)—and shown a set of 20 tests, one test at the time. The first four tests were control questions measuring participant skill in error fixing, and these four tests were the same for all participants, regardless of the group the participant was assigned to. Next, the test suite of 16 tests (cf. tests T01–T16 in Table 1) was shown, test by test, and in a randomized order for each participant. Each of the 16 tests consisted of a database structure diagram, a data demand, an erroneous SQL query, an error message generated by the DBMS, a free text input box in which the participant was instructed to write the fixed query, and a set of five-point Likert scale (1 = strongly disagree, 5 = strongly agree) questions pertaining to subjective indicators of the usability qualities of the error message (cf. hypotheses H_2, H_3 and H_4 in the next section). Depending on the group the participant was assigned to, they were shown corresponding error messages, e.g., for participants assigned to VoltDB group, VoltDB generated error messages were shown. Answering could be paused or stopped altogether, yet none of the participants chose to do so. The participants could use any materials for support during the tests. For more details on the tests, error messages, database structure, and questions, please refer to the supplementary Appendices. After all the participants had answered the tests, the first author coded the queries submitted by the participants as correct or incorrect. A query that contained at least one syntax error was considered incorrect.

Table 1. Test suite consists of 16 most common syntax errors (Taipalus et al., 2018).

Test	Syntax error name
T01	Ambiguous column
T02	Omitting quotes around character data
T03	IS where not applicable
T04	Confusing the syntax of keywords
T05	Confusing the logic of keywords
T06	Too many columns in subquery
T07	Undefined column
T08	Misspellings
T09	Failure to specify column name twice
T10	Using an aggregate function outside `SELECT` or `HAVING`
T11	Grouping error: extraneous grouping column
T12	Non-standard operators
T13	Using `WHERE` twice
T14	Non-standard keywords or standard keywords in wrong context
T15	Synonyms
T16	Curly, square, or unmatched brackets

3.3. Hypotheses

To study usability considerations of four NewSQL database management systems, we formulated two sets of hypotheses. Hypotheses H_1–H_4 compare objective query fixing success rates, as well as subjective error recovery confidence, usefulness for error finding, and usefulness for error fixing with a between-subjects study design. Hypotheses H_5–H_7 test correlation of error message qualities regardless of the database management system group. We chose to test the particular correlations between the objectively measured variable (i.e., success rate) and the subjectively measured variables (i.e., perceived usefulness for finding and fixing the error, and error recovery confidence) because of the nature of how these variables were measured. In other words, we did not test correlations between subjectively measured variables.

H_1: the medians of query formulation success rates are different for the database management system groups.

H_2: the medians of error recovery confidence are different for the database management system groups.

H_3: the medians of perceived usefulness for error finding are different for the database management system groups.

H_4: the medians of perceived usefulness for error fixing are different for the database management system groups.

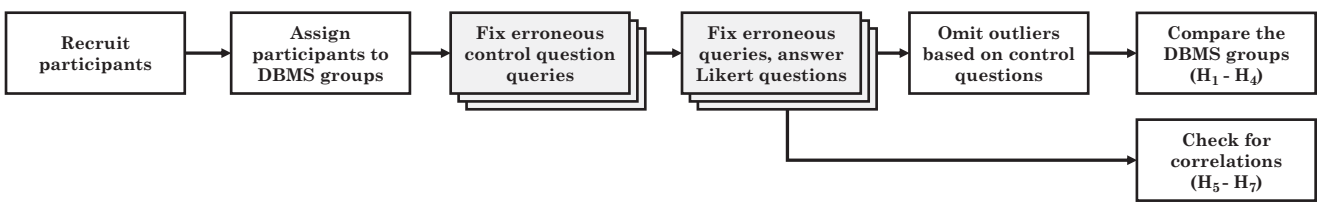


Figure 2. Overview of the data collection and analysis process; white rectangles refer to actions performed by us and grey rectangles to actions performed by the study participants.

Table 2. Summary of results.

Hypothesis	Short description	Supported	Test statistic	Effect size
H_1	Different effectiveness	No	$H(3) = 5.254$, $p = .154$	
H_2	Different recovery confidence	No	$H(3) = 0.157$, $p = .984$	
H_3	Different usefulness for error finding	Yes	$H(3) = 24.396$, $p < .001$	$\eta^2 = .174$
H_4	Different usefulness for error fixing	Yes	$H(3) = 9.870$, $p = .020$	$\eta^2 = .071$
H_5	Effectiveness $\iff$ recovery confidence	Yes	$r_{rb}(2486) = .283$, $p < .001$	
H_6	Effectiveness $\iff$ error finding	Yes	$r_{rb}(2486) = .238$, $p < .001$	
H_7	Effectiveness $\iff$ error fixing	Yes	$r_{rb}(2486) = .215$, $p < .001$	

H_5: $\rho \neq 0$; the correlation coefficient between query formulation success rate and error recovery confidence is not equal to zero.

H_6: $\rho \neq 0$; the correlation coefficient between query formulation success rate and perceived usefulness for finding the error is not equal to zero.

H_7: $\rho \neq 0$; the correlation coefficient between query formulation success rate and perceived usefulness for fixing the error is not equal to zero.

3.4. Data preparation and mitigation of control variables

Due to random participant assignment, it is possible that participants with higher (or lower) querying skills were assigned to the same group. This assignment presents a threat to internal validity, potentially skewing the results regardless of the qualities of the dependent variable (i.e., the error messages). To mitigate the effect of imbalance in participant assignment, we included four control questions in the survey. Based on the control questions, we omitted outliers from further between-subjects analyses based on query fixing success rate in the control questions. After the outliers were removed, and because the data were not normally distributed, we ran a Kruskal-Wallis H test to determine if there were differences in control question scores between the four groups of participants using different database management systems: CockroadDB ($n = 25$), NuoDB ($n = 44$), SingleStore ($n = 28$), and VoltDB ($n = 44$). Distributions of control question scores were similar for all groups, as assessed by visual inspection of a boxplot. There were no significant differences in the medians of control question scores between groups, $H(3) = 4.987$, $p = .173$. Hence, we considered the groups equal in terms of query fixing skills. For hypotheses H_5–H_7, which were not concerned with between–subjects comparison, we analyzed all data ($N = 157$).

SingleStore SQL compiler tolerated syntax errors in tests T09 and T11. The lack of an error message in these two tests was compensated in the questionnaires by made up error messages. The test results of tests T09 and T11 for SingleStore were omitted from the statistical analyses.

4. Results

4.1. A summary of results

In the following sections, we present the analyses in more detail, i.e., system per system, and describe the chosen statistical tests. A significance level of $\alpha = .05$ was chosen for all the statistical tests. A summary of results presented in Table 2 shows that hypotheses H_3–H_7 were supported, and hypotheses H_1 and H_2 were not supported. Please refer to Figure 3 for an overlook of the DBMS comparison.

4.2. Database management system group differences

For each of the hypotheses H_1, H_2, H_3, and H_4, we ran a Kruskal-Wallis H test to determine if there were differences in error message effectiveness (measured in error fixing success rates, H_1), error recovery confidence (H_2), and perceived usefulness of the error message in terms of finding (H_3) and fixing (H_4) the error between four groups of participants with different database management systems: CockroachDB ($n = 25$), SingleStore ($n = 28$), NuoDB ($n = 44$), and VoltDB ($n = 44$). Distributions of the answers for all hypotheses were similar for all groups, as assessed by visual inspection of a boxplot. Subsequently, pairwise comparisons were performed using Dunn's (1964) procedure with a Bonferroni correction for multiple comparisons. Adjusted p-values are presented in Table 3, and the results are visualized in Figure 3.

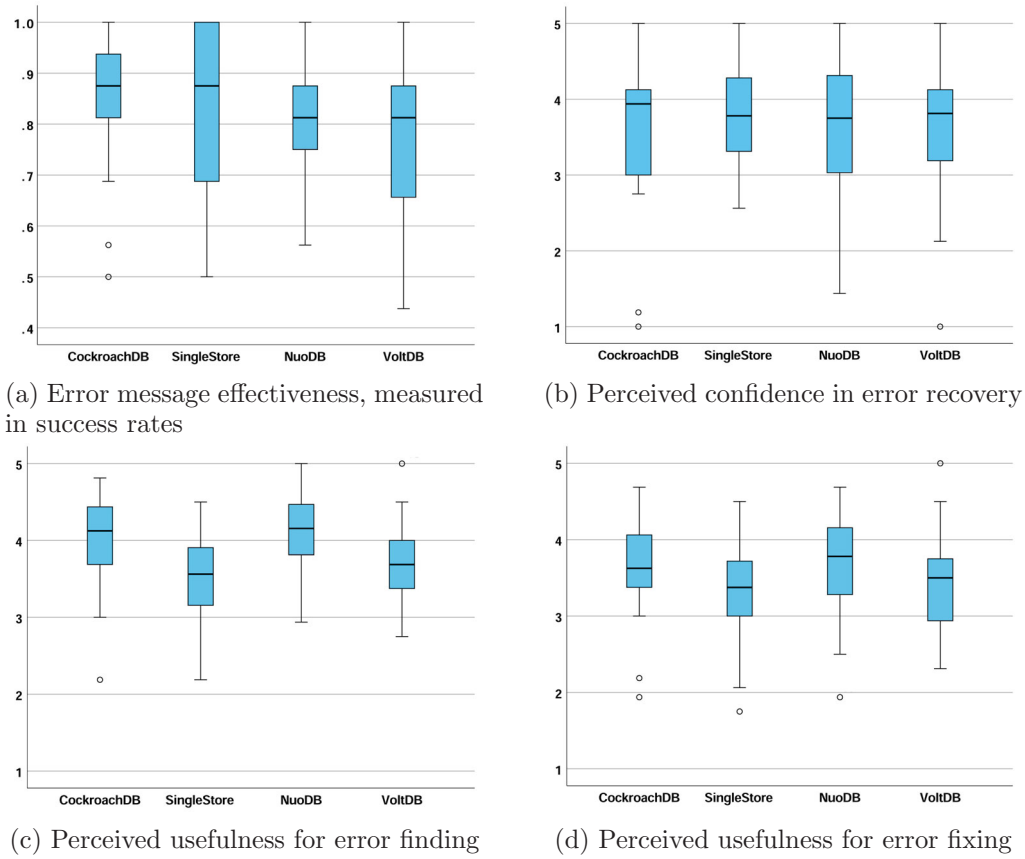
4.3. Correlations

For each of the hypotheses H_5, H_6 and H_7, we ran a rank biserial correlation to assess the relationship between error message effectiveness (measured in success rate, H_5) and error recovery confidence; between query formulation success rate and perceived usefulness for finding the error (H_6); and between query formulation success rate and perceived usefulness for fixing the error (H_7) ($N = 157$). For all three hypotheses, and for individual database management systems, the results were all statistically significant with a weak positive correlation. The test statistics are presented in Table 4.

5. Discussion

5.1. Implications for research

The results show no statistically significant differences in error message effectiveness between the DBMSs (hypothesis H_1). Although this observation implies that none of the DBMSs studied has more effective error messages than another, it is worth noting that success rate may be considered as one metric for effectiveness, rather than the sole metric. For example, in the context of programming



(a) Error message effectiveness, measured in success rates

(b) Perceived confidence in error recovery

(c) Perceived usefulness for error finding

(d) Perceived usefulness for error fixing

Figure 3. Between–subjects comparison of 16 tests regarding error message effectiveness, error recovery confidence, and error message usefulness for error finding and fixing—the boxplots represent interquartile range, and whiskers minimum and maximum values, excluding outliers. (a) Error message effectiveness, measured in success rates, (b) perceived confidence in error recovery, (c) perceived usefulness for error finding, and (d) perceived usefulness for error fixing.

Table 3. Test statistics for hypotheses H_1–H_4; *post-hoc* analyses were performed only if the Kruskal Wallis *H* test was statistically significant; DBMS names have been abbreviated as (Si)ngleStore, (Co)ckroachDB, (Nu)oDB, and (Vo)ltDB.

	Mdn				Pairwise comparison (*p*-value)					
	Co	Si	Nu	Vo	Co-Si	Co-Nu	Co-Vo	Si-Nu	Si-Vo	Nu-Vo
Effectiveness	.875	.875	.813	.813						
Recovery confidence	3.94	3.78	3.75	3.81						
Error finding	4.13	3.56	4.16	3.68	.004	1	.023	<.001	1	.003
Error fixing	3.63	3.36	3.78	3.50	.256	1	.665	.048	1	.140

Table 4. Test statistics for hypotheses H_5–H_7; correlations between error message effectiveness and error recovery confidence (r.c.), and perceived error message usefulness for finding and fixing the error.

	Effectiveness ⟺ r.c.	Effectiveness ⟺ finding	Effectiveness ⟺ fixing
CockroachDB	$r_{rb}(510) = .315, p < .001$	$r_{rb}(510) = .306, p < .001$	$r_{rb}(510) = .316, p < .001$
SingleStore	$r_{rb}(566) = .336, p < .001$	$r_{rb}(566) = .240, p < .001$	$r_{rb}(566) = .258, p < .001$
NuoDB	$r_{rb}(702) = .260, p < .001$	$r_{rb}(702) = .174, p < .001$	$r_{rb}(702) = .198, p < .001$
VoltDB	$r_{rb}(702) = .239, p < .001$	$r_{rb}(702) = .178, p < .001$	$r_{rb}(702) = .208, p < .001$

language compiler error messages, it has been suggested that the messages affect error recovery time rather than success (Ahmed et al., 2019).

There were no statistically significant differences in error recovery confidence between the DBMSs (hypothesis H_2). This suggests that error messages, although different, do not necessarily affect novice confidence in error recovery. Arguably, some error messages highlight the erroneous part of the query, yet fail to identify why the query contains an error, or may even provide false information on why the query is erroneous (Figure 1). Based on the results, it remains unclear why there were no significant differences in error recovery confidence. Similarly, the results yielded by this study support the notion that SQL error recovery confidence may have a similar relationship with error message effectiveness (hypothesis H_5) as confidence more generally has with success (Fleming et al., 2010; Martino et al., 2013). A rather underwhelming result of a *weak* positive correlation between success rate and error recovery confidence may indicate that it might be unexpectedly common that either a participant was confident in their fixed query, yet the query was incorrect, or that a participant was unsure

whether they had correctly fixed the query, yet the query was correct. At least in the scope of this study, this relationship suggests that the theories of neuroscience are particularize in relatively specific domains, such as SQL learning and error message research.

Regarding error message usefulness for error finding (hypothesis H_3), the results show relatively useful messages (i.e., those of CockroachDB and NuoDB), and relatively unuseful messages (i.e., those of SingleStore and VoltDB). Both CockroachDB and NuoDB use a free-standing circumflex (Figures 1(e,h)) to denote the assumed position of the error, while SingleStore and VoltDB (Figures 1(f,g)) replicate the erroneous query in its entirety or partially. This observation propounds the view that in text-only query formulation, such a simple approach makes finding errors easier. Rather similarly, a prior study has argued for the importance of positioning the error message close to the erroneous part (Hundhausen et al., 2017). It is worth noting that in this example, CockroachDB and NuoDB place the circumflex in a different position. Although there were no statistically significant differences in error message effectiveness, the results show a weak positive correlation between error message effectiveness and error message usefulness for error finding (hypothesis H_6).

NuoDB error messages were perceived as more useful than SingleStore error messages in regards to fixing erroneous queries, while other differences between the DBMSs were not statistically significant (hypothesis H_4). Again, looking at the corresponding error messages demonstrated in Figures 1(g,h), SingleStore does not provide a reason for the error message, while NuoDB suggests to the user what is erroneous. What is worth noting, in this case, is that the error messages in Figure 1 are merely a single example and that the NuoDB suggestion is rather confusing, and does not point to the typographical error, but to a non-erroneous part after the error. SingleStore, on the other hand, highlights a part of the query which does not contain the error—a principle argued against in a previous study (Wrenn & Krishnamurthi, 2017). As with usefulness for error finding, error message effectiveness was shown to have a weak positive correlation with usefulness in error fixing (hypothesis H_7). On these grounds, it seems fair to suggest that the issue of error message effectiveness might not be as clear-cut as the rejected hypothesis H_1 implies. As suggested in a previous study (Prather et al., 2017), an error message might be well designed, but the query writer does not understand it in context. The results yielded by this study (Hypotheses H_1–H_4) are in line with a previous study (Taipalus et al., 2021) that reported SQL compiler usability comparison in the context of traditional RDBSs using the same four metrics (effectiveness, recovery confidence, and perceived usefulness for error finding and fixing). The similarity of the results of these studies implies that whether the SQL compiler is more or less modern seems to play no part regarding the observation that some error messages are perceived as more useful than others, but this difference is not necessarily reflected in error message effectiveness in terms of error fixing success, or on error recovery confidence.

Finally, NewSQL systems often—and understandably—claim standard-conforming SQL implementations. To test whether the SQL test suite of 16 tests and four control questions were executable in the first place, we ran the queries without the syntax errors in all four DBMSs subject to this study, and all the queries passed the syntax checks. Although the implementation of the SQL standard is not unequivocal (Taipalus et al., 2018), and there is no SQL conformance testing anymore (Randolph, 2003), this testing provides confirmatory evidence that these four NewSQL DBMSs execute basic SQL queries without problems. This observation may be considered a by-product of this study, rather than an attempt of conformance testing, yet we are not aware of up-to-date scientific reports of SQL conformance of different DBMSs, or up-to-date test suites.

5.2. Recommendations for industry

In summary, we draw four recommendations for industry from our results, and suggest that DBMS vendors (i) utilize human-computer interaction literature in error message design, (ii) address usability concerns ubiquitously, i.e., usability should extend from aspects, such as installation and database distribution to querying, (iii) put forward the consideration that first impressions matter, especially early in the professional career, and that (iv) DBMS vendors should consider pinpointing the factors that hold down error message iteration, and consider whether these factors can be overcome by feature prioritization. We discuss these four recommendations in the following subsections.

5.2.1. Utilize human-computer interaction literature

Human-computer interaction is a field specifically addressing concerns, such as how a machine should interact with a human, and vice versa. Because of the current state of even NewSQL compiler error messages, we deemed enough to point to old guidelines which are still recognized, rather than discussing more recent error message design trends, such as those presented by Barik (2018) or Traver (2010). At the very least, we recommend that DBMS vendors consider the five guidelines introduced by Shneiderman (1982) 40 years ago when designing compiler error messages. First, error messages should be *brief*. In the context of programming languages, longer error messages have been shown to lengthen end-user response time (Nienaltowski et al., 2008), even though longer messages arguably provide more information to the end-user. Second, the error messages should be formulated in a *positive* tone, avoiding strict or negative words, such as *error* or *invalid*. Third, the error message should be *constructive*. If possible, the error message should provide a hint on how to fix the error. If it is not possible to reliably provide an accurate hint, the error message should not provide one at all. Fourth, the error message should be *specific* regarding both the position of the erroneous part, as well as the suggestion of what is erroneous and how to fix it. A general suggestion of referring to the DBMS manual is not helpful. Fifth, the use of *comprehensible*

language is recommended. For example, we cannot argue for "*Msg 321, Level 15, State 1, Server q7410*" in Figure 1(b) being comprehensible or helpful for a novice. Finally, Shneiderman (1982) suggests positioning the error code (if any) to the end of the error message. This serves as a reference point to the manual, or as a starting point in search engine utilization, yet does not confuse a novice user at the beginning of the error message.

5.2.2. Address usability concerns ubiquitously

As argued in Section 2.2, usability—as in usability in software development—has been one of the selling points of NewSQL systems, and usability should not be limited to flexible installation and maintenance in the cloud or relatively effortless database distribution. Again, if an end-user deems query formulation in an otherwise usable DBMS difficult, this might negatively affect the user experience. If the DBMS runs in a cloud environment, many cloud service providers provide the end-user with many DBMS options that are relatively effortless to deploy. In this regard, the use of a cloud service might be a positive factor for the end-user, but negative for the vendor of the DBMS the end–user deemed too difficult to use. The usability aspects of cloud computing, in general, have been necessitated previously (Buyya et al., 2019).

5.2.3. First impressions matter

All software developers are novices at some point in their careers. Although DBMSs are expert systems requiring a considerable amount of knowledge to learn and effectively use, it is not in the best interest of novices that parts of the DBMS are—intentionally or not—obscured, as query formulation is difficult as is (Taipalus, 2020b). Arguably, creating database structures and querying them is one of the first stages when a DBMS provides *value* to the end-user. It seems fair to speculate that a negative user experience in DBMS installation, setup, or the first querying negatively affects how the DBMS is perceived. A growing trend in K12 education is the introduction of programming principles and languages in general education (Lédeczi et al., 2021; Szabo et al., 2019). For the sake of discussion, it seems a matter of time when database topics are introduced to complement programming, or as a separate topic. If an educator deems a DBMS too difficult for students to learn, they may choose another DBMS that more effectively satisfies usability concerns. In contrast, positive experiences with a DBMS may prompt novices toward using the specific DBMS later in their careers as, e.g., software developers, database administrators, consultants, or project managers.

5.2.4. Consider what holds down better error message design

All being said in this section, we recognize that enhancing compiler error messages is a difficult task, and depending on the DBMS internals, designing more effective error messages is a task of varying complexity. Previously, the reluctance of enhancing error messages in programming language compilers has been explained by at least three concerns. First, the compiler performance degrades with more accurate diagnostics. Second, the software developers developing the compilers and designing error messages are experts who find it difficult to speculate how a novice would interpret an error message. Finally, the development of better error messages is simply not a high priority (Alexandrescu, 1999). Given these three considerations, and the arguments given in the previous sections, we implore DBMS vendors to reconsider whether error message enhancement is feasible, whether the reasons behind not enhancing error messages are rooted in justifiable causes, and whether error messages could be provided on different levels of specificity and according to a user's personal needs, as proposed in earlier studies (McIver & Conway, 1996; Traver, 2010).

5.3. Threats to validity

We chose to use novices as participants in this study, and consequently, the test suite tests syntax error recovery with relatively simple SQL queries. Arguably, it is possible that these results do not generalize to expert users, or to more complicated SQL queries. However, Figure 3(a) shows only a nascent ceiling effect, indicating the complexity of the test suite queries was not too low or too high for novice participants. Furthermore, we chose not to utilize expert participants, as experts are likely able to fix syntax errors based more on their personal experience, and less on the error message displayed. Finally, experts are likely experts using one or several DBMSs. If some of the hypothetical expert participants were familiar with one of the four tested NewSQL DBMSs, this would have introduced another threat to validity. In contrast, in the course from which the novice participants were recruited for this study, the instructor familiarized the students with SQL with SQLite DBMS. This approach ensured to a degree that the participants were not more familiar with any of the DBMSs than the other. We believe that using students as participants instead of experts is appropriate considering the study design is concerned with novice perceptions. Utilizing appropriate participants have been argued in several prior studies (Falessi et al., 2018; Feldt et al., 2018; Tahaei & Vaniea, 2022).

As explained in Section 3.2, we randomized the order in which the sixteen tests were shown to participants. With this, we aimed to mitigate the effect of skill improvement during the study. Arguably, if we had shown the tests in the same order for each participant, it would have been possible that the success rates would have risen toward the end of the study. The participants were randomly assigned to one of the four DBMS groups. To mitigate the possibility of skill differences of participants between the groups, the sixteen tests were preceded by four control questions, and we compared participant skills in the four groups based on the scores from these control questions. Consequently, we adjusted the groups accordingly, as detailed in Section 3.4.

The participants were studied in an unnatural environment. In a natural query formulation setting, a DBMS end-user initiates a feedback loop with the DBMS—a query is written and executed, an error message is received, the query is fixed or rewritten, and executed until the query returns a set of data. In our study, the query formulation process was initiated in the middle of the loop: the participant was shown an erroneous query and a corresponding error message, and the participant fixed the query without receiving further feedback on the correctness of the fixed query. As this study was to our knowledge the first published scientific study on SQL compiler error messages in the last 30 years, we chose this as a starting point for timely research. Additionally, studying the feedback loop in its entirety would have required the participants to start query writing without a preliminary (and erroneous) query. Depending on the participant, this setting would arguably have resulted in different syntax errors—not necessarily all we wanted to study, and possibly others outside the scope of this study. Furthermore, this type of research setting would have been more complex to manage. Finally, and as a deliberate choice, the chosen test suite only tests retrieval queries containing a single syntax error. Furthermore, the test suite does not cover all syntax errors.

5.4. Future research

As discussed in Section 5.1, subsequent studies could take error recovery time into account to provide supplementary insights on error message effectiveness. To the best of our knowledge, at the time of writing, there are no such studies regarding query languages, and studies on more detailed looks at the features of SQL compiler error messages are warranted. In particular, it might be interesting to compare the results of novices with experts to see whether our results generalize to a wider base of end-users. However, as argued in Section 3.2, usability concerns among experts might not be as important as usability is for novices. On the other hand, an interesting topic for further research would be to compare the error messages of NewSQL systems with those of traditional RDBMSs. Yet another potential future research topic is usability in the context of the query formulation feedback loop. Further studies could investigate the feedback loop between the end-user and the DBMS by, e.g., studying how many attempts it requires to fix a query. Consequent studies could also extend the error studies to logical (Taipalus et al., 2018) and semantic errors (Brass & Goldberg, 2006), other types of syntax errors, or errors in other types of SQL statements, e.g., updates, deletes, or inserts—a research dearth indicated earlier (Taipalus & Seppänen, 2020). Finally, the DBMS features that are more and more emphasized in cloud environments or distributed DBMSs in the general warrant for usability studies regarding e.g., database distribution, MapReduce, and the usability aspects of different application programming interfaces.

6. Conclusion

In this study, we set out to study the differences in usability of query language compiler error messages of four popular, modern NewSQL DBMSs. The results showed statistically significant differences in perceived usefulness of the error messages for finding and fixing the error, and query fixing success rates were shown to have a weak positive correlation with error recovery confidence as well as perceived usefulness for error finding and error fixing. Additionally, the results showed no statistically significant differences in query formulation success rates between the DBMSs, indicating that although the error messages of said systems are different, they do not have an impact on whether an erroneous query is eventually fixed. Similarly, there were no differences in perceived confidence in error recovery between the four DBMSs, which indicates that despite the different error messages, the participants were confident (or lacked confidence) in their skill in fixing erroneous queries. This is a rather interesting observation, since clearly many error messages erroneously identified the position of the error, or provided the user with erroneous hints on the nature of the error. These results indicate that the usability of the error messages of some of the DBMSs studied is higher than that of others and that success rate alone may not be a sufficient metric for measuring compiler usability. In general, these results highlight the need for usability considerations in not just programming languages, but in DBMS query language compilers as well.

Note

1. https://db-engines.com/en/ranking.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

ORCID

Toni Taipalus http://orcid.org/0000-0003-4060-3431
Hilkka Grahn http://orcid.org/0000-0001-7567-7807

References

Abbasi, A. A., Abbasi, A., Shamshirband, S., Chronopoulos, A. T., Persico, V., & Pescape, A. (2019). Software-defined cloud computing: A systematic review on latest trends and developments. *IEEE Access 7*, 93294–93314. https://doi.org/10.1109/ACCESS.2019.2927822

Ahmed, U. Z., Sindhgatta, R., Srivastava, N., & Karkare, A. (2019). Targeted example generation for compilation errors. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. https://doi.org/10.1109/ASE.2019.00039

Alexandrescu, A. (1999). Better template error messages. *C/C++ Users Journal*, *17*, 37–47.

Allen, G. N., & Parsons, J. (2010). Is query reuse potentially harmful? Anchoring and adjustment in adapting existing database queries. *Information Systems Research*, *21*(1), 56–77. https://doi.org/10.1287/isre.1080.0189

Ashkanasy, N., Bowen, P. L., Rohde, F. H., & Wu, C. Y. A. (2007). The effects of user characteristics on query performance in the presence of information request ambiguity. *Journal of Information Systems*, *21*(1), 53–82. https://doi.org/10.2308/jis.2007.21.1.53

Bak, P., & Meyer, J. (2011). The effect of user characteristics on the efficiency of visual querying. *Behaviour & Information Technology*, *30*(6), 809–819. https://doi.org/10.1080/0144929X.2010.511264

Barik, T. (2018). *Error messages as rational reconstructions* [Doctoral dissertation]. North Carolina State University. Retrieved from https://repository.lib.ncsu.edu/handle/1840.20/35439

Becker, B. A., Denny, P., Pettit, R., Bouchard, D., Bouvier, D. J., & Harrington, B. (2019). Compiler error messages considered unhelpful. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*. ACM. https://doi.org/10.1145/3344429.3372508

Becker, B. A., Glanville, G., Iwashima, R., McDonnell, C., Goslin, K., & Mooney, C. (2016). Effective compiler error message enhancement for novice programming students. *Computer Science Education*, *26*(2–3), 148–175. https://doi.org/10.1080/08993408.2016.1225464

Borthick, A. F., Bowen, P. L., Jones, D. R., & Tse, M. H. K. (2001). The effects of information request ambiguity and construct incongruence on query development. *Decision Support Systems*, *32*(1), 3–25. https://doi.org/10.1016/S0167-9236(01)00097-5

Brass, S., & Goldberg, C. (2006). Semantic errors in SQL queries: A quite complete list. *Journal of Systems and Software*, *79*(5), 630–644. https://doi.org/10.1016/j.jss.2005.06.028

Buyya, R., Srirama, S. N., Casale, G., Calheiros, R., Simmhan, Y., Varghese, B., Gelenbe, E., Javadi, B., Vaquero, L. M., Netto, M. A. S., Toosi, A. N., Rodriguez, M. A., Llorente, I. M., Vimercati, S. D. C. D., Samarati, P., Milojicic, D., Varela, C., Bahsoon, R., Assuncao, M. D. D., … Shen, H. (2019). A manifesto for future generation cloud computing: Research directions for the next decade. *ACM Computing Surveys*, *51*(5), 1–38. https://doi.org/10.1145/3241737

Chamberlin, D. D. (2012). Early history of SQL. *IEEE Annals of the History of Computing*, *34*(4), 78–82. https://doi.org/10.1109/MAHC.2012.61

Chaudhry, N., & Yousaf, M. M. (2020). Architectural assessment of NoSQL and NewSQL systems. *Distributed and Parallel Databases*, *38*(4), 881–926. https://doi.org/10.1007/s10619-020-07310-1

Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P., Hsieh, W., Kanthak, S., Kogan, E., Li, H., Lloyd, A., Melnik, S., Mwaura, D., Nagle, D., Quinlan, S., … Woodford, D. (2013). Spanner. *ACM Transactions on Computer Systems*, *31*(3), 1–22. https://doi.org/10.1145/2491245

Donahue, G. (2001). Usability and the bottom line. *IEEE Software*, *18*(1), 31–37. https://doi.org/10.1109/52.903161

Dong, T., & Khandwala, K. (2019). The impact of "cosmetic" changes on the usability of error messages. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM. https://doi.org/10.1145/2F3290607.3312978

Dunn, O. J. (1964). Multiple comparisons using rank sums. *Technometrics*, *6*(3), 241–252. https://doi.org/10.1080/00401706.1964.10490181

Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A., & Oivo, M. (2018). Empirical software engineering experts on the use of students and professionals in experiments. *Empirical Software Engineering*, *23*(1), 452–489. https://doi.org/10.1007/s10664-017-9523-3

Feldt, R., Zimmermann, T., Bergersen, G. R., Falessi, D., Jedlitschka, A., Juristo, N., Münch, J., Oivo, M., Runeson, P., Shepperd, M., Sjøberg, D. I. K., & Turhan, B. (2018). Four commentaries on the use of students and professionals in empirical software engineering experiments. *Empirical Software Engineering*, *23*(6), 3801–3820. https://doi.org/10.1007/s10664-018-9655-0

Fleming, S. M., Weil, R. S., Nagy, Z., Dolan, R. J., & Rees, G. (2010). Relating introspective accuracy to individual differences in brain structure. *Science*, *329*(5998), 1541–1543. https://doi.org/10.1126/science.1191883

Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., & Marsault, V. (2018). Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data* (p. 1433–1445). Association for Computing Machinery. https://doi.org/10.1145/3183713.3190657

Grolinger, K., Higashino, W. A., Tiwari, A., & Capretz, M. A. (2013). Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Applications*, *2*(1), 22. https://doi.org/10.1186/2192-113X-2-22

Hacigumus, H., Iyer, B., & Mehrotra, S. (2002). Providing database as a service. In *Proceedings 18th International Conference on Data Engineering* (pp. 29–38).

Hellerstein, J. M., Stonebraker, M., & Hamilton, J. (2007). Architecture of a database system. *Foundations and Trends in Databases*, *1*(2), 141–259. https://doi.org/10.1561/1900000002

Hundhausen, C. D., Olivares, D. M., & Carter, A. S. (2017). IDE-based learning analytics for computing education. *ACM Transactions on Computing Education*, *17*(3), 1–26. https://doi.org/10.1145/3105759

ISO (2018). *ISO 9241-11:2018, "Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts"*. ISO. Retrieved from https://www.iso.org/standard/63500.html

ISO/IEC (2016a). *ISO/IEC 9075-1:2016, "SQL – Part 1: Framework"*. ISO/IEC. Retrieved from https://www.iso.org/standard/63555.html

ISO/IEC (2016b). *ISO/IEC 9075-2:2016, "SQL – Part 2: Foundation"*. ISO/IEC. Retrieved from https://www.iso.org/standard/63556.html

Kaur, K., & Sachdeva, M. (2017). Performance evaluation of NewSQL databases. In *2017 International Conference on Inventive Systems and Control (ICISC)*. IEEE. https://doi.org/10.1109/ICISC.2017.8068585

Lédeczi, A., Grover, S., Catete, V., & Broll, B. (2021). Beyond CS principles: Bringing the frontiers of computing to K12. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (p. 1379). Association for Computing Machinery. https://doi.org/10.1145/3408877.3439542

Martino, B. D., Fleming, S. M., Garrett, N., & Dolan, R. J. (2013). Confidence in value-based choice. *Nature Neuroscience*, *16*(1), 105–110. https://doi.org/10.1038/nn.3279

McIver, L., & Conway, D. (1996). Seven deadly sins of introductory programming language design. In *Proceedings 1996 International Conference Software Engineering: Education and Practice*. IEEE Computer Society Press. https://doi.org/10.1109/SEEP.1996.534015

Miedema, D., Aivaloglou, E., & Fletcher, G. (2021). Identifying SQL misconceptions of novices: Findings from a think-aloud study. In *Proceedings of the 17th ACM Conference on International Computing Education Research*. ACM. https://doi.org/10.1145/3446871.3469759

Nicolaos, P., & Katerina, T. (2015). Simple-talking database development: Let the end-user design a relational schema by using simple words. *Computers in Human Behavior*, *48*, 273–289. https://doi.org/10.1016/j.chb.2015.02.002

Nienaltowski, M.-H., Pedroni, M., & Meyer, B. (2008). Compiler error messages. *ACM SIGCSE Bulletin*, *40*(1), 168–172. https://doi.org/10.1145/1352322.1352192

Oracle Corporation (2021). *Oracle8i Release 8.1.6 Documentation*. Author. Retrieved from https://www.oracle.com/database/technologies/oracle8i.html

Pavlo, A., & Aslett, M. (2016). What's really new with NewSQL? *ACM SIGMOD Record*, *45*(2), 45–55. https://doi.org/10.1145/3003665.3003674

Prather, J., Pettit, R., McMurry, K. H., Peters, A., Homer, J., Simone, N., & Cohen, M. (2017). On novices' interaction with compiler error messages: A human factors approach. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 74–82). Association for Computing Machinery. https://doi.org/10.1145/3105726.3106169

Ramakrishnan, R. (2012). CAP and cloud data management. *Computer Magazine*, *45*(2), 43–49. https://doi.org/10.1109/MC.2011.388

Randolph, G. B. (2003). The forest and the trees: Using Oracle and SQL Server together to teach ANSI-standard SQL. In *Proceedings of the 4th ACM Conference on Information Technology Curriculum (CITC)* (pp. 234–236). ACM. https://doi.org/10.1145/947121.947174

Reisner, P. (1981). Human factors studies of database query languages: A survey and assessment. *ACM Computing Surveys*, *13*(1), 13–31. https://doi.org/10.1145/356835.356837

Ribeiro, R. A., & Moreira, A. M. (2003). Fuzzy query interface for a business database. *International Journal of Human-Computer Studies*, *58*(4), 363–391. https://doi.org/10.1016/S1071-5819(03)00010-7

Schreiner, G. A., Knob, R., Duarte, D., Vilain, P., & Santos Mello, R. d (2019). NewSQL through the looking glass. In *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications & Services*. ACM. https://doi.org/10.1145/2F3366030.3366080

Shi, Y., Meng, X., Zhao, J., Hu, X., Liu, B., & Wang, H. (2010). Benchmarking cloud-based data management systems. In *Proceedings of the Second International Workshop on Cloud Data Management* (pp. 47–54). Association for Computing Machinery. https://doi.org/10.1145/1871929.1871938

Shin, S.-S. (2020). Structured query language learning: Concept map-based instruction based on cognitive load theory. *IEEE Access*, *8*, 100095–100110. https://doi.org/10.1109/ACCESS.2020.2997934

Shneiderman, B. (1982). Designing computer system messages. *Communications of the ACM*, *25*(9), 610–611. https://doi.org/10.1145/358628.358639

Siau, K. L., Chan, H. C., & Wei, K. K. (2004). Effects of query complexity and learning on novice user query performance with conceptual and logical database interfaces. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *34*(2), 276–281. https://doi.org/10.1109/TSMCA.2003.820581

Smelcer, J. B. (1995). User errors in database query composition. *International Journal of Human-Computer Studies*, *42*(4), 353–381. https://doi.org/10.1006/ijhc.1995.1017

Sobiesiak, R., Jones, R. J., & Lewis, S. M. (2002). DB2 universal database: A case study of a successful user-centered design program. *International Journal of Human-Computer Interaction*, *14*(3), 279–306. https://doi.org/10.1207/S15327590IJHC143&4_02

Somu, N., Kirthivasan, K., & Sriram, V. S. (2017). A computational model for ranking cloud service providers using hypergraph based techniques. *Future Generation Computer Systems*, *68*, 14–30. https://doi.org/10.1016/j.future.2016.08.014

Stonebraker, M. (2010). SQL databases v. NoSQL databases. *Communications of the ACM*, *53*(4), 10–11. https://doi.org/10.1145/1721654.1721659

Sutcliffe, A., Ryan, M., Doubleday, A., & Springett, M. (2000). Model mismatch analysis: Towards a deeper explanation of users' usability problems. *Behaviour & Information Technology*, *19*(1), 43–55. https://doi.org/10.1080/014492900118786

Szabo, C., Sheard, J., Luxton-Reilly, A., Simon Becker, B. A., & Ott, L. (2019). Fifteen years of introductory programming in schools: A global overview of K-12 initiatives. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. Association for Computing Machinery. https://doi.org/10.1145/3364510.3364513

Tahaei, M., & Vaniea, K. (2022). Recruiting participants with programming skills: A comparison of four crowdsourcing platforms and a cs student mailing list. In *CHI Conference on Human Factors in Computing Systems (CHI '22)*. ACM. https://doi.org/10.1145/3491102.3501957

Taipalus, T. (2019). Teaching tip: A notation for planning SQL queries. *Journal of Information Systems Education*, *30*(3), 160–166. http://jise.org/Volume30/n3/JISEv30n3p160.pdf

Taipalus, T. (2020a). The effects of database complexity on SQL query formulation. *Journal of Systems and Software*, *165*, 110576. https://doi.org/10.1016/j.jss.2020.110576

Taipalus, T. (2020b). Explaining causes behind SQL query formulation errors. In *2020 IEEE Frontiers in Education Conference (FIE)* (pp. 1–9). https://doi.org/10.1109/FIE44824.2020.9274114

Taipalus, T., & Perälä, P. (2019). What to expect and what to focus on in SQL query teaching. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE)* (pp. 198–203). ACM. https://doi.org/10.1145/3287324.3287359

Taipalus, T., & Seppänen, V. (2020). SQL education: A systematic mapping study and future research agenda. *ACM Transactions on Computing Education*, *20*(3), 1–33. https://doi.org/10.1145/3398377

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

Taipalus, T., Siponen, M., & Vartiainen, T. (2018). Errors and complications in SQL query formulation. *ACM Transactions on Computing Education*, *18*(3), 1–29. https://doi.org/10.1145/3231712

Tanin, E., Lotem, A., Haddadin, I., Shneiderman, B., Plaisant, C., & Slaughter, L. (2000). Facilitating data exploration with query previews: A study of user performance and preference. *Behaviour & Information Technology*, *19*(6), 393–403. https://doi.org/10.1080/014492900750052651

Toorn, C. V., Kirshner, S. N., & Gabb, J. (2022). Gamification of query-driven knowledge sharing systems. *Behaviour & Information Technology*, *41*(5), 959. https://doi.org/10.1080/0144929X.2020.1849401

Traver, V. J. (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

van der Schaaf, T. (1995). Human recovery of errors in man-machine systems. *IFAC Proceedings Volumes*, *28*(15), 71–76. https://doi.org/10.1016/s1474-6670%2817%2945211-6

Victorelli, E. Z., Reis, J. C. D., Hornung, H., & Prado, A. B. (2020). Understanding human-data interaction: Literature review and recommendations for design. *International Journal of Human-Computer Studies*, *134*, 13–32. https://doi.org/10.1016/j.ijhcs.2019.09.004

Wang, G., & Tang, J. (2012). The NoSQL principles and basic application of Cassandra model. In *2012 International Conference on Computer Science and Service System* (pp. 1332–1335).

Welty, C., & Stemple, D. W. (1981). Human factors comparison of a procedural and a nonprocedural query language. *ACM Transactions on Database Systems*, *6*(4), 626–649. https://doi.org/10.1145/319628.319656

Wrenn, J., & Krishnamurthi, S. (2017). Error messages are classifiers: a process to design and evaluate error messages. In *Proceedings of the 2017 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. ACM. https://doi.org/10.1145/3133850.3133862

Zapf, D., & Reason, J. T. (1994). Introduction: Human errors and error handling. *Applied Psychology*, *43*(4), 427–432. https://doi.org/10.1111/j.1464-0597.1994.tb00838.x

About the Authors

Toni Taipalus is a researcher and teacher at the Faculty of Information Technology, University of Jyväskylä, Finland. His research has been published in journals such as *Journal of Systems and Software* and *ACM Transactions on Computing Education*. His current research interests include query languages, database management systems, and computing education.

Hilkka Grahn is a researcher and teacher at the Faculty of Information Technology, University of Jyväskylä, Finland. Her research has been published in journals such as *International Journal of Human–Computer Studies* and *Accident Analysis & Prevention*. Her current research interests include human–computer interaction, driver distraction, and human factors.