The Effects of Database Complexity on SQL Query Formulation

Toni Taipalus

PII: S0164-1212(20)30057-1
DOI: https://doi.org/10.1016/j.jss.2020.110576
Reference: JSS 110576

To appear in: *The Journal of Systems & Software*

Received date: 6 June 2019
Revised date: 3 January 2020
Accepted date: 16 March 2020

Please cite this article as: Toni Taipalus, The Effects of Database Complexity on SQL Query Formulation, *The Journal of Systems & Software* (2020), doi: https://doi.org/10.1016/j.jss.2020.110576

Highlights

- Increase in database complexity results in lower success rates in query formulation
- Increase in database complexity results in more unnecessary complications
- Teachers should not strive for more complex databases without consideration
- More complex exercise databases may emphasize differences in student skills

*Title page with author details

The Effects of Database Complexity on SQL Query Formulation

Toni Taipalus
University of Jyvaskyla

-

Corresponding author:

Toni Taipalus
toni.taipalus@jyu.fi
University of Jyvaskyla
Faculty of Information Technology
P.O. Box 35, FI-40014 Finland

-

Declarations of interest: none

The Effects of Database Complexity on SQL Query Formulation

Abstract

In Structured Query Language (SQL) education, students often execute queries against a simple exercise database. Recently, databases that are more realistic have been utilized to the effect that students find exercises more interesting and useful, as these databases more accurately mimic databases students are likely to encounter in their future work environments. However, using even the most engaging database can be counterproductive to learning, if a student is not able to formulate correct queries due to the complexity of the database schema. Scientific evidence on the effects of database complexity on student's query formulation is limited, and with queries from 744 students against three databases of varying logical complexity, we set out to study how database complexity affects the success rates in query formulation. The success rates against a simple database were significantly higher than against a semi-complex and a complex database, which indicates that it is easier for students to write SQL queries against simpler databases. This suggests, at least in the scale of our exercise databases, that educators should also consider the negative effects of more realistic databases, even though they have been shown to increase student engagement.

Keywords: Structured Query Language (SQL), database, database complexity, education, student learning

1. Introduction

Computer languages have been a major topic in ICT education curricula for decades. Even though most of these languages change over time, Structured Query Language (SQL) has proved especially resilient. Given the importance, long life, and pervasive nature of databases and query languages in the field of information technology, it is rather surprising that educational research on the topic is relatively scarce when compared to, e.g., programming languages. Furthermore, studies related to skills of professionals working with databases have pointed out the difficulties arising from the differences of database management system implementations of the SQL standard (McMinn et al., 2019), faults in database schema integrity constraint definition and enforcement (McMinn et al., 2015), and ill-designed database transactions (Warszawski and Bailis, 2017), all of which further emphasize the importance of effective SQL education.

In addition to teaching theoretical foundations, many university level database courses facilitate

SQL learning by providing the students an environment in which they can execute SQL queries against an exercise database (e.g., Mitrovic, 1998). Similarly to programming education, teaching SQL in practice is justified, as many students are expected to perform similar tasks in their future work environments. The used exercise databases are usually constructed by the teacher, or provided by a third party. One of such third party database is the Sakila[1] database of MySQL database management system (Sakila, 2019), which contains both structure and data for a movie rental business domain. Traditionally, these exercise databases have been relatively simple, possibly to shift the focus of the learning process from the structure of the database to the logic and semantics of SQL (Wagner et al., 2003). Recently, though, more realistic databases such as Sakila, and some of the databases of Teradata University Network (Jukic and Gray, 2008a; Watson and Hoffer, 2003) have been utilized, and research shows that students find more realistic databases more interesting and useful (Yue, 2013). In effect, educational research has provided support for the assertion that more complex databases have positive effects on database education. However, little research touches the potential negative effects of the structural complexity of a database on SQL learning, e.g., a student's failure to formulate SQL queries. This inability is a likely indication that a student has not acquired the necessary practical knowledge to write valid SQL, which is arguably one of the goals of SQL education. This study provides the field with a perspective on the potential negative side effects (i.e., lower success rates in query formulation) of increasing exercise database complexity. Given the consideration that, however interesting a more realistic exercise database might seem to a student, utilizing such a database may be counterproductive to learning, if the student cannot formulate correct SQL queries due to the structural complexity. This problem of database complexity potentially manifests in either as a failure to start the query formulation process due to perceived overwhelming complexity, or as a failure to successfully formulate the query despite one or several attempts. Although writing erroneous queries is part of any student's learning process, a student is able to correct some errors, but not necessarily all. An error left uncorrected is a common indication of some problem in knowledge, skill, or learning. In the vein of Taipalus and Perälä (2019), we call errors which are never corrected *persistent*. In this study, we set out to analyze differences in query writing performance (i.e., success rates) of three student cohorts with a total of 744 students. One cohort wrote SQL queries against a simple, one against a semi-complex, and one against a complex database. While the complexity of a database schema is both subjective and relative, we measured the complexity of a database schema according to previously established metrics (Calero et al., 2001), which effectively measure complexity by both the number of certain database objects, and the number of potential predictable joins (cf. Section 2.2 for a more detailed

[1] https://dev.mysql.com/doc/sakila/en/

description, and the Appendices for the database schemas).

Our results show statistically significant differences in success rates between the student cohorts. Based on our results, we recommend that researchers and educators also consider the negative implications of more complex exercise databases, rather than using the more complex databases available.

The rest of the study is structured as follows. In Section 2 we discuss prior SQL education research, database complexity metrics, and the frameworks used in this study. In Section 3, we present our hypotheses. In Section 4, we describe the course and exercises from which the data were collected, the exercise databases, and our research method. In Section 5 we present our results, and in Section 6 discuss practical implications and limitations of our study, and future research avenues. Finally, in Section 7 we present conclusions.

2. Theoretical background

2.1. Database complexity in education

A number of studies discuss database complexity and SQL learning (Jukic and Gray, 2008b; Wagner et al., 2003; Yue, 2013), but it is worth noting that none of these studies explore the effects of database complexity on query writing performance, but on student interest (Yue, 2013), or how to better prepare students for their future work (Jukic and Gray, 2008b; Wagner et al., 2003). Additionally, the effects of task complexity (Topi et al., 2005) and data model representation (Chan et al., 1997, 2005; Rho and March, 1997) on query formulation have been studied. However, given that there exists no scientific evidence regarding the effects of logical complexity of a relational database on SQL query learning, we address here studies that consider database complexity in education in general.

Intuitively, it may seem obvious that it is easier to write SQL queries against a simple rather than a complex database. We traced the argument for more complex exercise databases to 2003, when Wagner et al. (2003) concluded that “[...] using large scientific datasets in a database systems course has a number of benefits for students, and no discernible losses.” The authors claim that increased complexity better prepares students for their future employment, students learn that real-world data have problems, and students learn interdisciplinary work and communications skills. Even though the authors present little numerical evidence to support their argument, we can certainly agree with the part concerning the benefits for students. Similar argument for the benefits for students has also been presented later by Jukic and Gray (2008b). However, the latter part of the quotation concerning no discernible losses seems somewhat contradictory, as the same article reports students perceiving increased complexity more difficult. It is worth noting that Wagner et al. (2003) focus their discussion on the complexity of data (i.e., extension), not the complexity of the database structure (i.e., intension), and in this study, by complexity of a *database* we refer to the complexity of the logical structure, rather than complexity of data.

A more recent study by Yue (2013) argued for more complex exercise databases from the point of view of student interest. The study measured the perceived interestingness and usefulness when Sakila-based assignments were gradually integrated into a database course. The students perceived Sakila more interesting and useful when compared to instructor and textbook assignments. The study also commended Sakila for having the right balance of complexity, meaning that the database is structurally complex, but does not contain unnecessary domain intricacies. This is a noteworthy observation, and in line with Wagner et al. (2003).

For studying how database complexity influences query writing performance, we identified three crucial aspects. First, a set of metrics is needed to measure database complexity, second, a unified set of SQL exercises for the cohorts despite the fact that the three databases are different, and third, a framework to measure whether or not a student's query is correct or incorrect. Next, we discuss these three aspects in prior studies, and argue for the choices we made concerning this study.

2.2. Database complexity metrics

Although normal forms can be considered a method of determining the complexity of a relational database, a higher normal form does not implicitly result in a simpler or more complex database structure. Even though a higher normal form implies more tables, and thus a more complex database, a lower normal form presents different complexities for the query writer. Regarding relational database structure complexity metrics, we found two scientific proposals which complement normalization. First, a four part metric was proposed by Calero et al. (2001) which consists of the number of attributes in the schema (NA), depth referential tree (DRT), number of foreign keys in the schema (NFK), and cohesion of the schema (COS). When a database is presented as a graph G of tables (nodes) and foreign keys (directed edges), DRT is the number of edges on the longest path (not counting loops), and COS is the sum of the square of the number of nodes in each component of G. Second, Pavlic et al. (2008) proposed a database complexity measuring method which decrees that the complexity of a database is the sum of the number of all attributes, keys (i.e., primary and secondary keys), indices, and foreign keys in the database. We wanted to limit this study to the logical complexity of a database, and chose to use the former metrics (Calero et al., 2001), as indices are not a part of the relational model but a part of physical database design. We would like to add that while any of the metrics proposed by Calero et al. (2001) is insufficient to measure complexity by itself, together they consolidate into an adequate, high-level presentation of the logical complexity of a relational database. Two issues with database complexity metrics, Calero et al. (2001) included, is that there is no objective way to measure whether one database is more complex than the other, if one of the attributes (e.g., DRT) is higher, but another (e.g., COS) is lower. In contrast, database complexities measured by the metrics proposed by Pavlic et al. (2008) can be objectively compared by numbers, but these numbers do not represent objective complexity, as these numbers may be the same for different databases (compare a database with 9 attributes, 3 primary keys, and 5 foreign keys to a database of 12 attributes, 3 primary keys, and 2 foreign keys). The second issue is that these database complexity metrics measure only quantitative aspects of databases, but not the complexity of the business domain. Arguably, a genome database may be more complex for a layperson than a movie rental database, even if these two databases are of equal complexity by both of the discussed metrics.

2.3. Exercises and query evaluation

A number of studies exploring SQL exercises have been published (Ahadi et al., 2016a,b; Prior and Lister, 2004; Smelcer, 1995; Taipalus et al., 2018), and many of these studies utilized exercises designed for each particular study. The query concepts in the SQL exercises reported in these five studies show similarities, e.g., exercises testing joins, expressions, ordering, and grouping with their respective clauses and predicates. In addition to reporting query concepts by name, Taipalus et al. (2018) provide example SQL queries of the exercises, and the number of tables needed for the formulation of each query. For its relative specificity, we designed our exercises for each database using the query concept framework presented by Taipalus et al. (2018). Although a query can be interpreted as any SQL statement, the scope of our chosen framework limits our study solely on data retrieval. This limitation would have also been the case, had we based our study on any other of the aforementioned studies' query concepts.

In order to measure success rates in student queries, we needed a framework to determine whether or not a student's query was correct or incorrect. Some studies discuss SQL error categorizations (Ahadi et al., 2016b,a), which are, however, not the results of the studies, but rather a vehicle for answering their respective research questions. Other studies, however, present SQL error categorizations as results of their respective studies. Brass and Goldberg (2006) present an extensive list of semantic errors and complications based on their teaching experience, and Taipalus et al. (2018) complement Brass and Goldberg's listing with syntax and logical errors, which are rooted in the SQL standard (ISO/IEC, 2016) rather than a single database management system's implementation. Taipalus et al. (2018) categorize 105 different errors into four error classes: 1) complications, which do not affect the result table, but hinder queries with readability or performance issues, 2) logical errors, which affect the result table, and make the query appear as if it was written to answer a different but valid data demand (i.e., natural language representation of the task), 3) semantic errors, which affect the result table, and make the query unsuitable for any valid data demand, and 4) syntax errors, which result in an error message instead of a result table. For this study, we chose this error categorization framework for its relative extensiveness, and for database management system independence. This error categorization framework (Taipalus et al., 2018) also fits our chosen database complexity metrics (Calero et al., 2001), as both of them disregard physical structure in its entirety.

3. Hypotheses

The discussion in the previous section has both highlighted the positive outcomes for using more complex exercise databases in teaching SQL, as well as the undercurrent of the possible negative effect of difficulty. We propose that as database complexity increases, so does the difficulty of successfully writing SQL queries that satisfy given data demands. The basic proposition to be tested is

H_1: The success rates for formulating correct SQL queries decrease as logical complexity of the database increases.

The error categorization framework (Taipalus et al., 2018) divides errors into four classes (syntax errors, semantic errors, logical errors, and complications), and an incorrect query may, in theory, exhibit as many as 105 different errors. A recent study (Taipalus and Perälä, 2019), in turn, indicated that logical errors and complications are more likely to persist than syntax errors and semantic errors, meaning that although syntax and semantic errors are committed, they are more likely corrected by students. Given the framework to measure these four error classes, and rather than only studying whether there exists an effect between success rates and database complexity, we wanted to explore if different database complexities invite different kinds of errors. Therefore, as auxiliary hypotheses, we propose that the number of persistent errors committed for each of these four error classes increase as database complexity increases.

H_2: The number of syntax errors committed in incorrect final SQL queries increase as logical complexity of the database increases.

H_3: The number of semantic errors committed in incorrect final SQL queries increase as logical complexity of the database increases.

H_4: The number of logical errors committed in incorrect final SQL queries increase as logical complexity of the database increases.

H_5: The number of complications committed in final SQL queries increase as logical complexity of the database increases.

Concerning the auxiliary hypotheses, it is worth noting that lower success rates do not necessarily imply higher numbers of persistent errors. Even if success rates increase, it is possible that students who are unable to write correct queries commit more persistent errors.

4. Research setting

4.1. Course and data collection

We collected the queries from an introductory database course targeted for second year students majoring in computer science or information systems, with no prior knowledge on SQL. The course was mandatory, but completing the exercises was not, and by completing the exercises the students could earn points toward a better grade. We collected the queries from three student cohorts (237, 280, and 227 students), and each cohort completed SQL exercises against a simple, semi-complex, or complex database (cf. Appendices), respectively. This study took place over a period of three years, and the first author taught the course for each student cohort. The course was given in Finnish.

We constructed exercises for each of the cohorts using the query concept framework presented by Taipalus et al. (2018). The framework contains 15 exercises, all of which test a student's skill in several query concepts. These query concepts per exercise are presented in Table 1. This framework allowed us to construct similar exercises for each cohort in terms of query concepts tested, and the number of tables required to formulate the correct query. According to the framework, a source table is a table which is used to project or calculate values into the result table, and a subject table is a table which is used to restrict the values that are accepted into the result table. For the complex database, we utilized the database structure and exercises presented by Taipalus et al. (2018), and, from there constructed the simple and semi-complex databases with respective business domains and exercises. Refer to Taipalus et al. (2018) and Appendix D for detailed descriptions and examples of the query concepts. It is worth noting that the data demands and database schemas presented in the Appendices are translations from Finnish to English, and the translations introduce some natural language considerations about the similarity of the data demands between the student cohorts.

Students completed the exercises using an interactive database management system (SQLite) prompt embedded on a web page, which, depending on the query submitted by the student, output either a result table or an error message from SQLite. The correct result table was visible during the whole query writing process, and the students could compare their result tables with the correct one. The exercises were completed over three weeks during the course, in three sets (cf. A, B, and C in Table 1), each with their weekly deadlines. The exercises in a set could be completed in any order.

Table 1: Query concepts for each exercise, the numbers of source tables and subject tables, and the total number of tables needed in the formulation of a correct query, based on Taipalus et al. (2018)

Exercise	Concepts	Source	Subject	Total
A1	single-table; expressions	1	1	1
A2	single-table; expressions; ordering	1	1	1
A3	single-table; wildcard; expressions with nesting	1	1	1
B4	multi-table; expressions; facing foreign keys	1	1	2
B5	multi-table; expressions; ordering	1	3	3
B6	multi-table; expressions with nesting; ordering	1	2	3
B7	multi-table; expressions; does not exist	1	2	2
B8	multi-table; does not exist; equal subqueries	1	2	3
B9	single-table; expressions; aggregate functions	1	1	1
B10	multi-table; expressions; multiple source tables	2	3	4
B11	multi-table; expressions; self-join; aggregate function evaluated against a column value; correlated subquery	1	2	2
B12	multi-table; expressions; aggregate function evaluated against a constant; uncorrelated subquery; parameter distinct	1	1	2
B13	multi-table; expressions; self-join	1	5	5
C14	multi-table; multiple source tables; aggregate functions; grouping	2	1	2
C15	multi-table; multiple source tables; aggregate functions; grouping; grouping restrictions; ordering	2	1	2

The students were given unlimited tries within the weekly deadlines, and were allowed to use whatever materials or ways of communication. The database schema as well as a short description of the business domain was also visible during the whole process, and students could obtain more information on the database objects using built-in SQLite commands. After a deadline had passed, the students were given example answers for the respective set of exercises. Course contents prior to and during data collection are presented in Table 2, and the structure is common for a database course (Topi et al., 2010), containing enhanced/extended entity-relationship model (EER), transformation from EER to relational schema, relational calculus, and SQL sublanguages data manipulation language (DML), data definition language (DDL), data control language (DCL), and transaction control language (TxCL). The course continues with database normalization, data warehousing, database distribution, and NoSQL. The course structure was the same for each of the three cohorts.

Table 2: Course activities prior to and during data collection

Week	Course activity (chronologically ordered for each week)
n	Lectures: general concepts in database systems, conceptual modeling with EER
n+1	Lectures: relational model, transformation from EER to relational schema
n+2	Lectures: relational calculus, DML Exercises #1 presented: conceptual modeling with EER
n+3	Lectures: DML, DDL Answers for exercises #1 presented Exercises #2 presented: transformation from EER to relational schema, SQL exercise set A
n+4	Lectures: DCL, TxCL Answers for exercises #2 presented Exercises #3 presented: SQL exercise set B
n+5	Lectures: database normalization Answers for exercises #3 presented Exercises #4 presented: SQL exercise set C, additional SQL exercises (DML, DDL, DCL)
n+6	Lectures: data warehousing Answers for exercises #4 presented Exercises #5 presented: database normalization
Course continues	

4.2. Databases

We implemented the exercise databases with hand-crafted data. For clearer distinguishability, we call these databases "simple", "semi-complex" and "complex", although, compared to real life databases, they are all relatively simple. The logical complexities of the databases, some summarizing information, and, for comparison, additional databases from literature are presented in Table 3. Our three databases were normalized to Boyce/Codd normal form. In terms of data, the tables in the simple database contained 17–73 rows, with the average of approximately 46 rows, the tables in the semi-complex database 5–105 rows, with the average of approximately 50 rows, and the tables in the complex database 5–125 rows, with the average of approximately 61 rows. We designed the data to contain no anomalies or errors, and null values were only present in obvious columns, e.g., in a customer's email or an actor's date of death, as opposed to null values in foreign key columns. For these reasons, it was more feasible to hand-craft the data, rather than using automatic data generation tools such as DBMonster[2] or Mockaroo[3].

Table 3: Database business domains and complexities (NT = number of tables, NA = number of attributes, NFK = number of foreign keys, DRT = depth referential tree, COS = cohesion of the schema) - databases marked with an asterisk can also be found in the Teradata University Network

	Business domain	NT	NA	NFK	DRT	COS
Simple	social media	5	22	8	3	25
Semi-complex	rally timing	7	32	8	3	49
Complex	movie rental	11	54	12	4	121
Hoffer et al. (2014)	order catalog	4	17	3	2	16
Kroenke and Auer (2016)	order catalog	5	27	2	1	11
Elmasri and Navathe (2016)*	company employees	6	28	6	3	36
Connolly and Begg (2015)	property rental	6	39	6	3	36
Hoffer et al. (2011)*	product lines	15	59	13	2	153
Sakila (2019)	movie rental	16	88	23	7	256

The textbook databases described in Table 3 are presented in their respective sections concerning SQL. These textbooks also present other databases in, e.g., sections addressing conceptual modeling or data warehousing. Note that in Elmasri and Navathe (2016), NFK, and hence DRT, are counted using the table creation statements (p. 211) presented in the textbook. If the schema complexity is

[2] http://dbmonster.sourceforge.net/

[3] https://mockaroo.com/

evaluated based on the database schema (p. 194), NFK = 8 and DRT = 5.

4.3. Protocol and method

After the last deadline for the last student cohort had passed, we collected all the submitted queries for the 15 exercises from our learning environment, a total of over 123,000 SQL queries. Some students had attempted to complete the course and exercises in a previous year or years. To achieve independence of observations, we removed all but first attempts from the data, i.e., if a student tried to complete the exercises in year *n*, *n+1* and *n+2*, we omitted their answers from years other than *n*. In order to study success rates and the numbers of persistent errors, we were only interested in the final queries from each student for each exercise. After omitting all non-final queries (i.e., queries submitted chronologically before the last query), we were left with 8,771 queries. Next, using the error categorization framework (Taipalus et al., 2018), we coded each final query with errors it exhibited, if any. We considered a query incorrect if it contained at least one syntax, semantic, or logical error. A query which contained only a complication or complications was considered correct.

We first conducted a chi-square test of homogeneity using count data with weighted cases to examine the relation between database complexity (simple, semi-complex, complex) and success rate in respective final queries. Not all students attempted all exercises, and we considered non-attempts as failures. We argue for our decision with a minimal counterexample of two students cohorts, ten students each: in cohort A, only one student tries to complete an exercise and succeeds (success rate = 100%), and in cohort B, all ten students try to complete an exercise but only three succeed (success rate = 33%). We consider our protocol to better reflect the equivalence of the research setting between the cohorts, as opposed to ignoring non-attempts. The chi-square test of homogeneity fit our data and research design, as we had a sufficiently large sample size, and, by design, independence of observations.

To test the auxiliary hypotheses, we compared the numbers of errors committed for each error class against the databases of different complexity. The data were not normally distributed between groups (simple, semi-complex, and complex), but the distributions of the number of errors committed were similar for all groups. Additionally, group sizes were not equal (745, 1,116, and 791 incorrect final queries). For these reasons, we ran a Kruskal-Wallis H test (one for syntax errors, one for semantic errors, and one for logical errors) to determine if there were differences in the number of errors committed between the database groups of different complexity. Finally, we ran a Kruskal-Wallis H test to determine if there were differences in the number of complications committed between the database groups. As complications by themselves do not constitute in making a query incorrect, we ran the test on final queries regardless of their correctness (2,870, 3,425, and 2,476 final queries).

5. Results

The null hypothesis for a chi-square test of homogeneity is that in all groups of the independent variable, the proportions are equal in the population, while the alternative hypothesis is that not all group population proportions are equal. There was a statistically significant difference between the three independent binomial proportions ($p < .001$). Therefore, we can reject the null hypothesis and accept the alternative hypothesis.

We analyzed 11,160 cases (8,771 queries and 2,389 cases of non-attempts) from a total of 744 students, each assigned to a group writing queries against either a simple, semi-complex or complex exercise database. The group of 227 students writing queries against the simple database had a higher success rate (62.4%) compared to the group of 280 students with the semi-complex database (55.0%), and to the group of 237 students with the complex database (47.4%). Post hoc analysis involved pairwise comparisons using the z-test of two proportions with a Bonferroni correction. All pairwise comparisons were statistically significant. The success rates for each database are visualized in Fig. 1, and the success rates for each exercise for each database in Fig. 2.

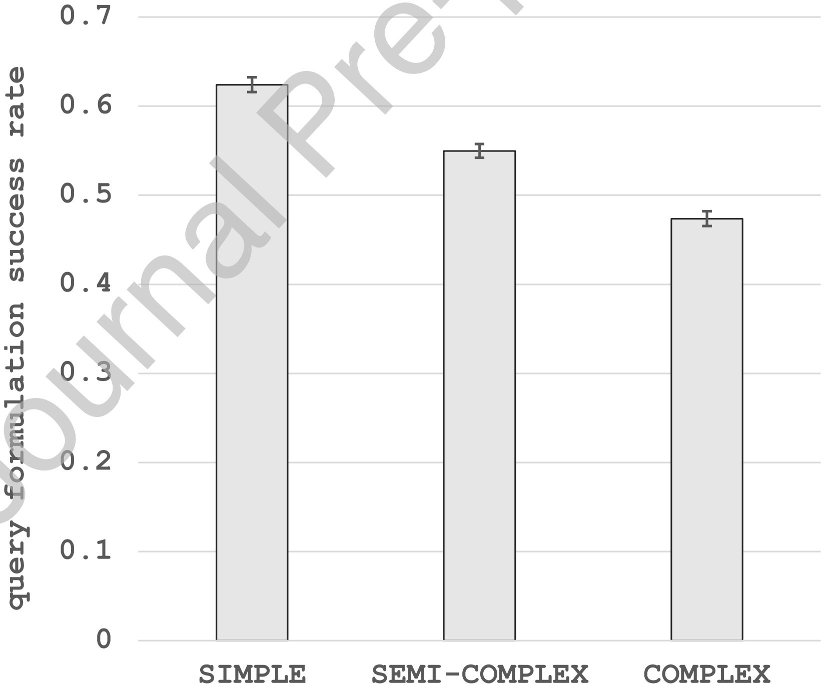


Figure 1: Success rates for each database

The null hypothesis for a Kruskal-Wallis H test is that the distribution of the number of errors (syntax, semantic, or logical) for the groups are equal, while the alternative hypothesis is that the

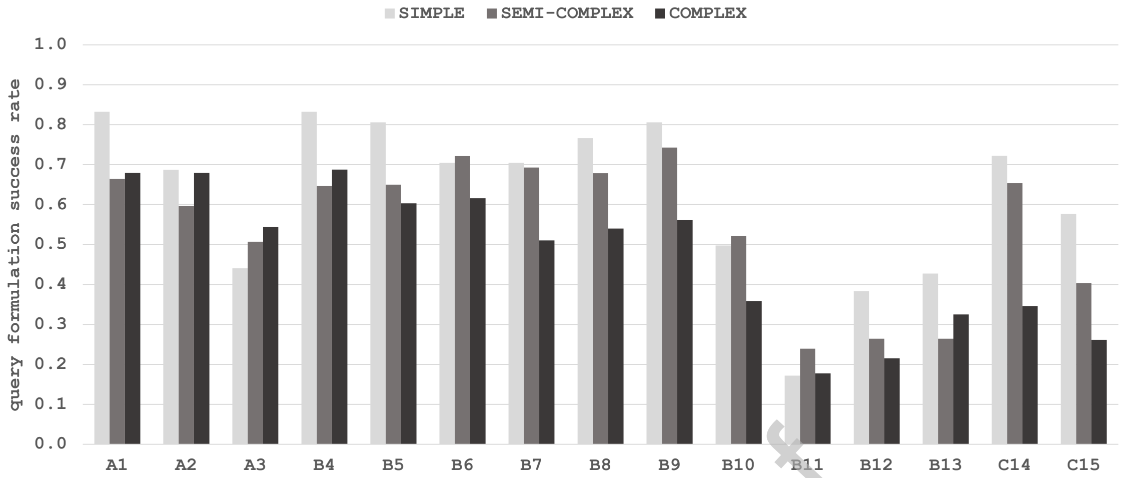


Figure 2: Success rates for each exercise for each database

distribution of the number of errors (syntax, semantic, or logical) are not equal. The Kruskal-Wallis H test is a common nonparametric alternative to one-way ANOVA (Ruxton and Beauchamp, 2008).

A Kruskal-Wallis H test was run to determine if there were differences in the number of syntax errors committed between three groups of students writing queries against four databases of different complexity: "simple" (N = 745), "semi-complex" (N = 1,116) and "complex" (N = 791), where N represents the number of incorrect final queries submitted by each group in total. Distributions of the number of syntax errors committed were similar for all groups, as assessed by visual inspection of a boxplot. Median number of syntax errors committed were statistically significantly different between groups, $H(2) = 23.481$, $p <$ .001. Subsequently, pairwise comparisons were performed using Dunn's (1964) procedure with a Bonferroni correction for multiple comparisons. Adjusted p-values are presented. This post hoc analysis revealed statistically significant differences in the number of syntax errors committed between the simple (mean rank = 1,279.10) and complex (mean rank = 1,420.85)($p <$ .001), and semi-complex (mean rank = 1,291.26) and complex ($p <$ .001) database complexity groups, but not between the simple and semi-complex database complexity group.

A Kruskal-Wallis H test was run to determine if there were differences in the number of semantic errors committed between four groups of students writing queries against four databases of different complexity: "simple" (N = 745), "semi-complex" (N = 1,116) and "complex" (N = 791). Distributions of the number of semantic errors committed were similar for all groups, as assessed by visual inspection of a boxplot. Median numbers of semantic errors committed were not statistically significantly different between groups, $H(2) = 5.314$, p = .070.

A Kruskal-Wallis H test was run to determine if there were differences in the number of logical errors committed between three groups of students writing queries against three databases of different complexity: "simple" (N = 745), "semi-complex" (N = 1,116) and "complex" (N = 791). Distributions of the number of semantic errors committed were similar for all groups, as assessed by visual inspection of a boxplot. The numbers of logical errors committed were statistically significantly different between groups, $H(2) = 14.280$, $p = .001$. Subsequently, pairwise comparisons were performed performed using Dunn's (1964) procedure with a Bonferroni correction for multiple comparisons. Adjusted p-values are presented. This post hoc analysis revealed statistically significant differences in the number of logical errors committed between the complex (mean rank = 1,250.20) and simple (mean rank = 1,341.41)($p = .031$), and complex and semi-complex (mean rank = 1,370.62)($p <$ .001) database complexity groups, but not between simple and semi-complex.

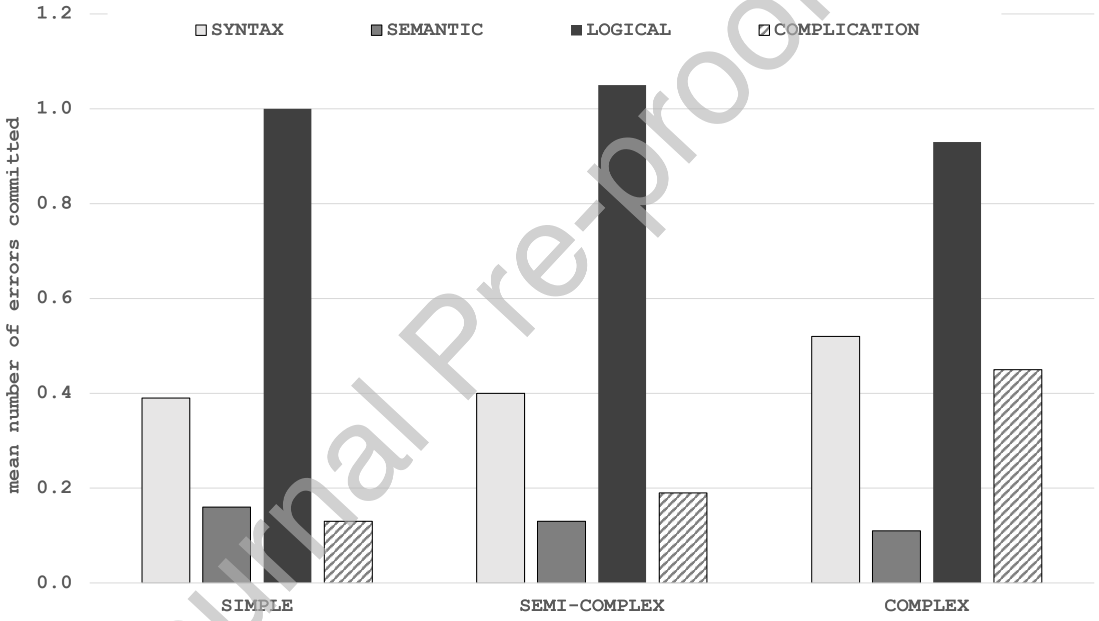


Figure 3: Means for each error class for each database

A Kruskal-Wallis H test was run to determine if there were differences in the number of complications committed between three groups of students writing queries against three databases of different complexity: "simple" (N = 2,870), "semi-complex" (N = 3,425) and "complex" (N = 2,476). Distributions of the number of complications committed were similar for all groups, as assessed by visual inspection of a boxplot. The numbers of complications committed were statistically significantly different between groups, $H(2) = 717.363$, $p <$ .001. Subsequently, pairwise comparisons were performed performed using Dunn's (1964) procedure with a Bonferroni correction for multiple comparisons. Adjusted p-values are presented. This post hoc analysis revealed statistically significant differences in the number of complications committed between the simple (mean rank = 3,932.88) and semi-complex (mean rank = 4,173.22)($p <$ .001), and simple and complex (mean rank = 5,205.56)($p <$ .001), and semi-complex and complex ($p <$ .001) database complexity groups. These numbers for four error classes for each of the three databases are visualized in Fig. 3 as means (rather than mean ranks) for readability.

Based on the aforementioned results, we can conclude that the basic proposition H_1, and auxiliary hypothesis H_5 were supported. Auxiliary hypothesis H_2 was supported, but the increase from simple database to semi-complex was not statistically significant. Auxiliary hypothesis H_3 was not supported, and had a negative, but statistically non-significant effect. Auxiliary hypothesis H_4 was not supported.

6. Discussion

6.1. Why the success rates differ

The chi-square test of homogeneity indicates that there is an association between database complexity and success rate, and on the basis of the evidence currently available, it seems fair to suggest that more often than not, a logically more complex relational database yields lower success rates than a simpler relational database when students try to write correct SQL queries. Even though this relationship does not appear uniform among all the exercises (Fig. 2), the results overall (Fig. 1) support the position that a more complex database results in lower success rates, and the question under scrutiny is not *if* but rather *why*.

Based on a set of studies by Reisner (1977, 1981, 1988), a seminal study on student errors in SQL query writing by Smelcer (1995) provided the field with six (later abstracted to four in the same study) cognitive explanations on why errors occur. First two, *absence of retrieval cue* and *imprecise retrieval cue* are closely related to the data demand. For example, the data demand "list the names of customers who have rented the same movie as John Doe has rented" lacks the cue to leave John Doe out of the results. However, in addition to the query concept framework, we designed the exercises for all cohorts to follow similar natural language expressions. Next three explanations, *misperception*, *procedural fixedness*, and *inaccurate procedural knowledge*, are closely related to human error, and lack of knowledge concerning the relational model, the business domain, or SQL. We believe that although these three explanations matter in the comparison of success rates between the cohorts, the differences of their effects between the cohorts are minor due to our research design, as explicated in Section 4.1. Finally, and in our opinion, most importantly, Smelcer (1995) explains SQL errors with *exceeding working memory's capacity* (Miller, 1956): when the number of query concepts, expressions, or database objects in a task increases, a student's working memory capacity exceeds, and errors (omission errors in particular) occur. Our results seem to support the observations presented by Smelcer (1995), although the connection is not straightforward. What is worth noting is that between our three cohorts, the query concepts, number of required tables, and database objects in a task are the same by design (cf. Table 1), and it is the complexity of the database which increases. The view that more database objects (were they merely present in the database, or also part of a query being written) cause more strain on working memory is in line with common sense. Based on the results by Smelcer (1995) and our research, we suggest that it is not only the complexity of the task that affects the success rate, but also the logical complexity of the exercise database. For future research, mapping errors to their cognitive explanations via e.g., student interviews would be a valuable addition to understanding *why* errors occur with databases of different complexities.

In this study, we did not consider student engagement, but intuitively, more interesting exercises should result in both more students trying to complete the voluntary exercises, and students engaging more in the exercises, e.g., a less interested student attempting 5 times, and a more interested student attempting 10 times to solve an exercise before giving up. In the analyses, we considered that a student had *attempted* to solve an exercise if they had written at least one SQL query. A post hoc inspection of attempt rates revealed that the cohort with the simple database had the highest attempt rates for 11 of the 15 exercises, while the cohort with the complex database had the lowest attempt rates for all exercises. This might suggest that the students in the cohort with the simple database (social media) were more interested in completing the exercises than the students in the cohort with semi-complex (rally timing) and the complex database (movie rental). This might be due to database complexity, but also due to the database business domain.

This leads to another point we feel compelled to make. Making mistakes is part of any learning process, and it is rare that a student is able to write the correct query on the first attempt. Moreover, even if a student is not able to formulate the correct query at all, the errors committed during the writing process constitute to learning, but non-attempts do not. This propounds the view that measuring success rates while ignoring non-attempts leaves out the factor of how many of the students in a cohort even attempted, thus possibly biasing the results towards higher success rates. In contrast, measuring the perceived interest and usefulness of the exercises, as studied by Yue (2013), leaves out the factor of success rates, as successfully formulating a query implies that a student has achieved the required level of knowledge in SQL, whereas failure to do so implies the opposite.

6.2. Considerations on lower success rates

Our results show statistically significant differences in success rates between the databases of different complexities. However, we do not wish to infer that a high success rate in query writing is a metric that educators should necessarily strive for, or that a high success rate conflates with learning. Arguably, the more a student commits errors, the more misconceptions are uncovered and uncertainties remedied. That being said, we did not consider the number of errors a student committed, only the number of persistent errors. As stated earlier, a persistent error arguably represents a misconception or uncertainty that is not remedied, at least not during the query writing process.

Prior studies have provided evidence on the positive effects of more complex databases, and while our results are not in conflict, they shed light on the possible negative effects. The data yielded by this study provides considerations for future research, as many questions regarding the matter of exercise database complexity remain open. If students learn SQL using more complex exercise databases, does that imply that the students are more familiar with complex databases, but do not have the skills to formulate correct SQL queries? In contrast, if students learn SQL using simpler exercises databases, does that imply that the students have the skills to formulate correct SQL queries, but not in complex database environments? If a more realistic database is demonstrated to cause positive feelings (i.e., it is interesting and useful, Yue, 2013) in students, does a low success rate in query formulation cause negative feelings in students towards SQL, query languages, or databases in general? These considerations also propound the future research question of how simple exercise database is too simple, and how complex is too complex.

Finally, as discussed by, e.g., Denny et al. (2012) in the context of programming languages, students have different levels of capability, and by making the task more difficult, performance decreases (Topi et al., 2005). With these considerations in mind, it is intuitive that students with high capability have a tendency to perform better than students with low capability, regardless of the task complexity. As our results have provided evidence that an increase in database complexity (as opposed to task complexity) also results in decrease in performance, it seems justified to foster debate whether more complex databases emphasize the capability differences between students.

6.3. Implications for research

Only one of the auxiliary hypotheses, H_5, was supported with a statistically significant effect. Consequently, while we cannot infer from our results that database complexity affects the number of syntax, semantic, or logical errors, complications seem to increase with a statistically significant effect as the the complexity of the database increases. According to the error categorization (Taipalus et al., 2018), complications can be, e.g., unnecessary joins, ordering in a subquery, or unused correlations names (i.e., aliases). As complications do not affect the result table, but query readability or computational performance, their severity is below that of other errors. Furthermore, it is theoretically possible to reliably identify complications in queries with computerized automation (Brass and Goldberg, 2006), as opposed to, e.g., identifying logical errors. Persistent and non-persistent SQL errors have been identified earlier (Taipalus and Perälä, 2019), but based on the evidence currently available, it seems reasonable to suggest that error persistence in regards to error class is not affected by database complexity.

An interesting set of studies by Bowen et al. (2004, 2009) investigated whether *ontological clarity* affects query writing performance. The authors effectively designed two relational databases with the same business domain. One database was designed following widely accepted design guidelines at the cost of ontological clarity, resulting in a simpler database structure. The other database was designed with the prioritization of ontological clarity, resulting in a more complex database structure. Their results indicated that the participants writing queries against the ontologically clearer database committed more semantic errors, took longer to write their queries, and were less confident in the accuracy of their queries than the participants writing queries against the ontologically less clear database. With the omittance of the factor of ontological clarity, our results provide an indication that increased structural complexity negatively affects query formulation performance.

6.4. Implications for teaching

Intuitively, there were three possible outcomes of this study; a more complex database either causes a decrease or an increase in success rates, or the success rates remain the same despite the change in database complexity. Depending on the results, and with Yue's study (2013) in mind, we encourage teachers to utilize simpler exercise databases now that the results suggest a decrease in success rates. We would like to point out that the two other possible outcomes would have been equally interesting, and in those cases we would have argued for the use of more complex databases. However, as discussed earlier, our results leave room for interpretation, and, given that a teacher has time, more than one exercise database can be utilized.

Although it is not apparent in the study by Yue (2013) whether the students found a more complex database more interesting and useful due to complexity or something else, for the sake of discussion, we would like to argue that structural complexity increases perceived usefulness and student interest. Furthermore, if an increase in structural complexity indeed implies decrease in success rates, we as researchers and teachers should either 1) consider other ways besides increasing structural complexity to convey interesting and useful exercise databases to students, or 2) support learning SQL in complex databases with a different or an auxiliary method. That said, if an interesting and useful database is inevitably also complex, we suggest utilizing both of the above.

Finally, if the differences in success rates between simple and complex databases are indeed caused by increased load on working memory, we propose that the earlier, rather ambiguously phrased *auxiliary method* could be considered a way to simplify the SQL syntax, semantics, and the database structure into a form that puts less strain on a student's working memory. As a possible solution, we are currently investigating how a notation for planning more complex SQL queries (Taipalus, 2019) affects SQL query formulation in more complex exercise databases. In addition to the environment, concerns about the relationship between language syntax and cognitive load have been raised in the context of programming languages (Kelleher and Pausch, 2005; Lister, 2011a,b). Ahadi et al. (2016a) conclude their study on SQL syntax errors by noting that while semantic errors require more creative problem solving, solving them is not feasible until possible syntax errors are fixed. With this in mind, the relative difference in the means of syntax and logical errors (Fig. 3) should not be considered an indicator that syntax errors are somehow less important.

As shown in Table 2.2, our databases are somewhat similar in complexity to those presented in learning environments and textbooks. When a teachers chooses an exercise database for a course, the appropriate structural complexity depends on the difficulty of the planned exercises, student backgrounds (e.g., majoring in business analytics versus software engineering), as well as teacher skill and experience. Furthermore, a single database course is not necessarily limited to a single exercise database. A teacher may utilize a simple database to teach query concepts in theory and through examples, yet utilize a complex database against which the students can practice query formulation. Finally, although in the vein of Yue (2013), we have effectively treated a more realistic database as a synonym for a more complex database, this connection does not necessarily hold true. The growing trend of, e.g., microservice architectures (Alshuqayran et al., 2016) and mobile applications are often concerned with subsets of business domains, and do not necessarily address structurally complex databases. This puts forward the topical view that more realistic databases are not necessarily more complex, and educators should consider using databases which are both realistic (and thus engaging), and relatively simple (and thus query formulation is likely more successful). That said, what is an engaging business domain among students remains an open question. While the answer is changing and subjective, student engagement to different database domains is an interesting future research topic. In conclusion, the database feature of being more or less realistic is simply a student's *perception* of realistic. If educators can demonstrate that a simple exercise database indeed reflects the structure of a realistic database, that might positively affect student engagement without negatively affecting query formulation.

6.5. Limitations

There are two main limitations that affect the generalizability of the results of this study. First, the data were collected from one university, and a single course which took place three times over three years. This presents the question whether similar results could be obtained from students taking other database courses in other universities or under other teachers. As this study was to our knowledge the first to explore the effects of database complexity on query writing performance, it is not possible to compare the our results to other studies. Second, only a subset of SQL concepts, even in the scope of data retrieval, were studied. Then again, a narrower study scope does not necessarily imply weaker research, as argued for by Siponen and Klaavuniemi (2019).

6.6. Threats to validity

We wanted to study the effects of database complexity on SQL query formulation. Prior to the study, we identified seven control variables that could affect our results (cf. Fig. 4), and designed our research setting to mitigate the effects of these variables. Next, we discuss how these variables might have affected the results of this study, describe the measures (labels *a-g* in Fig. 4) we took to mitigate these effects, and argue for the choices we made concerning the research setting.

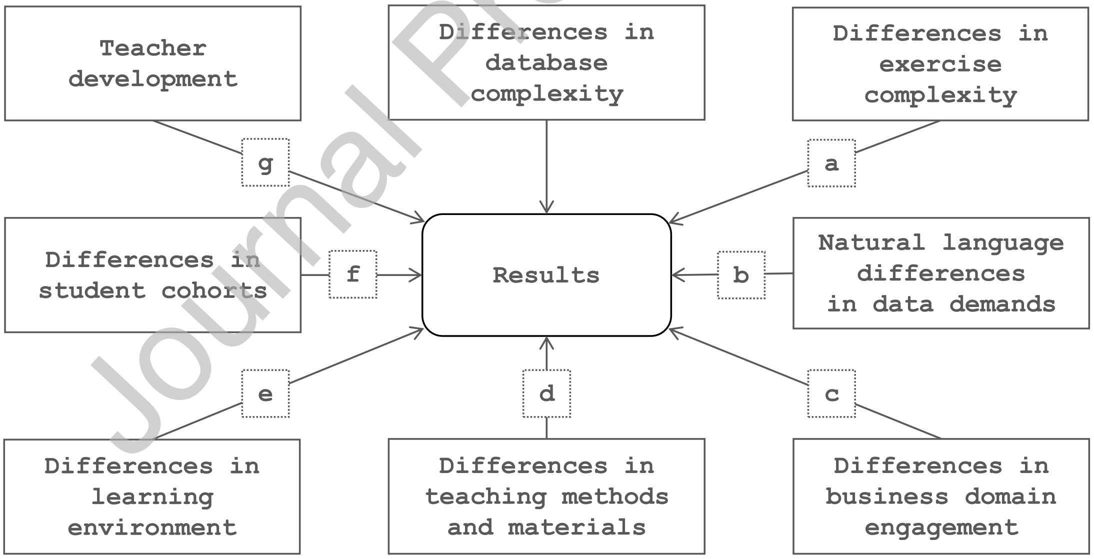

Teacher development
Differences in database complexity
Differences in exercise complexity
g
a
Differences in student cohorts
f
Results
b
Natural language differences in data demands
e
d
c
Differences in learning environment
Differences in teaching methods and materials
Differences in business domain engagement

Figure 4: Variables potentially affecting the results

As we studied three students cohorts, each with their respective database and exercises, the effects of differences in exercise complexity (Fig. 4a) needed to be mitigated. We designed the exercises for each cohort according to the query concept framework, which lists query concepts and the number of source and subject tables needed for each of the 15 exercises. Designing the exercises with this framework allowed us to control the complexity of the exercises, thus mitigating the risk that one cohort had easier or more difficult exercises than another.

Each of the three cohorts wrote queries against a database with a different business domain. Consequently, the data demands for each cohort were different from each other (Fig. 4b), e.g., one cohort had to list the names of social media users, one of rally drivers, and one of customers. Although these natural language considerations are relatively minor due to the fact that the exercises were designed using the same framework, natural language entails ambiguity (Borthick et al., 2001; Casterella and Vijayasarathy, 2013; Reisner, 1981). We tried to minimize the effects of natural language on query writing by providing the students with the correct result table, and we did not consider the number of tries a student needed to formulate the correct query. We hoped that if students saw that their result table differed from the correct result table, it would effectively steer students toward the correct interpretation of the data demand. Similarly, the effects of different database business domains (Fig. 4c) may have affected the number of students who decided to attempt the exercises. It has also been shown that understanding the business domain affects query writing performance (Siau et al., 2004). We considered using a single business domain and a single database, and modularly adding (or subtracting) tables and attributes for each cohort. However, we found it increasingly difficult to utilize the query concept framework and come up with at least somewhat realistic data demands. We also considered using the modular approach with same data demands for each cohort. This was not deemed feasible for two reasons. First, it would have meant that that some of the tables would not have been utilized in any query, for any cohort. In our study, the simple database contained five tables and the complex database eleven, and with same data demands, the remaining six tables of the complex database would not have been used in any of the queries. Second, based on our previous teaching experiences, some students have shared the example answers from previous years in different forums. Even the most diligent student may be tempted to look up an example answer to an exercise they could not solve, thus achieving more course points. For these reasons, we designed new exercises and databases for each student cohort, and strived to utilize business domains that are at least some way familiar to students.

We mitigated the effects of differences in teaching methods and materials (Fig. 4d), and in the learning environment (Fig. 4e) by using the same teaching materials (slides, handouts), not making adjustments to the teaching methods, and retaining the course outline (cf. Fig. 2) for all cohorts. All cohorts used the same e-learning environment and database management system (SQLite) even though the pedagogical shortcomings of SQLite became increasingly apparent during the study. The first author taught the course for each cohort, and also coded the queries according to the error categorization framework. It is possible that there were misinterpretations of the framework, but possible misinterpretations were at least consistent between the cohorts.

As we elaborated in Section 4.1, the students formulated the queries in a minimally controlled environment, and there is a possibility that the student cohorts studied were, in some unforeseeable way, different from each other (Fig. 4f). Perhaps there was a growing trend that students communicate with each other more and more, perhaps students are more and more skilled in utilizing internet search engines, or perhaps students have more and more certain personal characteristics - a factor which has been studied to affect query writing performance (Ashkanasy et al., 2007). Additionally, and although we used the same teaching materials for each cohort, it is possible, even likely, that the teacher's skills develop over time (Fig. 4g), thus possibly positively affecting the development of success rates over time. If such trends or development exist, we tried to mitigate the effects by following a schedule. Instead of chronologically gradually increasing or decreasing the database complexity for the cohorts, we utilized the complex database for the first cohort, the simple for the second, and the semi-complex for the third. Furthermore, the teacher had taught the same course for years before the first cohort subject of this study, so major developments in teaching skills were not likely.

In summary, we made deliberate choices to favor a more natural environment for the students to write their queries. Although, as opposed to a more controlled environment, this presented several threats to validity, but in turn allowed us to study query writing that more accurately reflects the students' future work environments. Additionally, our data collection method allowed for a relatively large number of students and queries to be studied, whereas a more controlled experiment, at least in our experience, would possibly have yielded significantly less participants. We believe that the relatively large sample sizes compensate for the margin of error presented possibly by the less controlled environment.

7. Conclusion

In this study, we set out to investigate whether the logical structural complexity of a relational database affects the success rates of students writing SQL queries against three databases of varying complexity. Overall, the results show statistically significant differences between the different databases, which indicates that students are less likely to formulate correct SQL queries if the exercise database is complex. Rather than suggesting the usage of simpler databases when teaching SQL, we encourage educators to consider the potential negative effects of more complex databases on SQL learning, as it has been demonstrated that more complex databases also bring beneficial effects to teaching.

Acknowledgements

The authors would like to thank Hilkka Grahn for her invaluable advice regarding the data analysis and grammar, and the associate editor and anonymous reviewers for their helpful insights and comments on how to improve the paper.

Declarations of interest

None.

Funding

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

References

References

Ahadi, A., Behbood, V., Vihavainen, A., Prior, J., Lister, R., 2016a. Students' Syntactic Mistakes in Writing Seven Different Types of SQL Queries and its Application to Predicting Students' Success, in: Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16), ACM Press, New York, New York, USA. pp. 401–406. doi:10.1145/2839509.2844640.

Ahadi, A., Prior, J., Behbood, V., Lister, R., 2016b. Students' Semantic Mistakes in Writing Seven Different Types of SQL Queries, in: Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '16), ACM Press, New York, New York, USA. pp. 272–277. doi:10.1145/2899415.2899464.

Alshuqayran, N., Ali, N., Evans, R., 2016. A systematic mapping study in microservice architecture, in: 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), IEEE. pp. 44–51. doi:10.1109/SOCA.2016.15.

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, 53–82. doi:10.2308/jis.2007.21.1.53.

Borthick, A., 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, 3 – 25. doi:10.1016/S0167-9236(01)00097-5.

Bowen, P., O'Farrell, R., Rohde, F., 2004. How does your model grow? An empirical investigation of the effects of ontological clarity and application domain size on query performance, in: Proceedings of the International Conference on Information Systems (ICIS), p. 7. URL: https://aisel.aisnet.org/icis2004/7.

Bowen, P.L., O'Farrell, R.A., Rohde, F.H., 2009. An empirical investigation of end-user query development: The effects of improved model expressiveness vs. complexity. Information Systems Research 20, 565–584. doi:10.1287/isre.1080.0181.

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

Calero, C., Piattini, M., Genero, M., 2001. Metrics for controlling database complexity, in: Developing Quality Complex Database Systems: Practices, Techniques and Technologies. IGI Global, pp. 48–68. doi:10.4018/9781878289889.ch003.

Casterella, G.I., Vijayasarathy, L., 2013. An Experimental Investigation of Complexity in Database Query Formulation Tasks. Journal of Information Systems Education 24, 211–221. URL: http://jise.org/Volume24/24-3/pdf/Vol24-3pg211.pdf.

Chan, H., Siau, K., Wei, K.K., 1997. The effect of data model, system and task characteristics on user query performance: an empirical study. SIGMIS Database 29, 31–49. doi:10.1145/506812.506820.

Chan, H.C., Teo, H.H., Zeng, X., 2005. An evaluation of novice end-user computing performance: Data modeling, query writing, and comprehension. Journal of the American Society for Information Science and Technology 56, 843–853. doi:10.1002/asi.20178.

Connolly, T., Begg, C., 2015. Database Systems (6th. ed.). Pearson.

Denny, P., Luxton-Reilly, A., Tempero, E., 2012. All syntax errors are not equal, in: Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE), ACM, New York, NY, USA. pp. 75–80. doi:10.1145/2325296.2325318.

Dunn, O.J., 1964. Multiple comparisons using rank sums. Technometrics 6, 241–252. doi:10.2307/1266041.

Elmasri, R., Navathe, S.B., 2016. Fundamentals of Database Systems (7th. ed.). Pearson.

Hoffer, J.A., Ramesh, V., Topi, H., 2011. Modern database management. Upper Saddle River, NJ: Prentice Hall.

Hoffer, J.A., Topi, H., Ramesh, V., 2014. Essentials of Database Management. Pearson Education.

ISO/IEC, 2016. ISO/IEC 9075-2:2016, "SQL - Part 2: Foundation". URL: https://www.iso.org/standard/63556.html.

Jukic, N., Gray, P., 2008a. Teradata university network: A no cost web-portal for teaching database, data warehousing, and data-related subjects. Journal of Information Systems Education 19, 395–402. URL: http://jise.org/Volume19/n4/JISEv19n4p395.html.

Jukic, N., Gray, P., 2008b. Using real data to invigorate student learning. SIGCSE Bulletin 40, 6–10. doi:10.1145/1383602.1383604.

Kelleher, C., Pausch, R., 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys 37, 83–137. doi:10.1145/1089733.1089734.

Kroenke, D., Auer, D.J., 2016. Database Processing: Fundamentals, Design, and Implementation (14th. ed.). Pearson Education.

Lister, R., 2011a. Programming, syntax and cognitive load (part 1). ACM Inroads 2, 21–22. doi:10.1145/1963533.1963539.

Lister, R., 2011b. Programming, syntax and cognitive load (part 2). ACM Inroads 2, 16–17. doi:10.1145/2003616.2003622.

McMinn, P., Wright, C.J., Kapfhammer, G.M., 2015. The effectiveness of test coverage criteria for relational database schema integrity constraints. ACM Transactions on Software Engineering and Methodology 25, 8:1–8:49. doi:10.1145/2818639.

McMinn, P., Wright, C.J., McCurdy, C.J., Kapfhammer, G.M., 2019. Automatic detection and removal of ineffective mutants for the mutation analysis of relational database schemas. IEEE Transactions on Software Engineering 45, 427–463. doi:10.1109/TSE.2017.2786286.

Miller, G.A., 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review 63, 81. doi:10.1037/0033-295x.101.2.343.

Mitrovic, A., 1998. Learning SQL with a computerized tutor, in: Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Science Education, ACM, New York, NY, USA. pp. 307–311. doi:10.1145/273133.274318.

Pavlic, M., Kaluza, M., Vrcek, N., 2008. Database complexity measuring method, in: Central European Conference on Information and Intelligent Systems, Faculty of Organization and Informatics Varazdin. URL: http://archive.ceciis.foi.hr/app/index.php/ceciis/2008/paper/view/84/84.

Prior, J.C., Lister, R., 2004. The backwash effect on SQL skills grading. SIGCSE Bulletin 36, 32–36. doi:10.1145/1026487.1008008.

Reisner, P., 1977. Use of psychological experimentation as an aid to development of a query language. IEEE Transactions on Software Engineering SE-3, 218–229. doi:10.1109/tse.1977.231131.

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

Reisner, P., 1988. Query languages, in: Helander, M. (Ed.), Handbook of Human-Computer Interaction. Elsevier, New York, pp. 257–280.

Rho, S., March, S.T., 1997. An analysis of semantic overload in database access systems using multi-table query formulation. Journal of Database Management 8, 3–15. URL: https://www.igi-global.com/gateway/article/51176.

Ruxton, G.D., Beauchamp, G., 2008. Time for some a priori thinking about post hoc testing. Behavioral Ecology 19, 690–693. doi:10.1093/beheco/arn020.

Sakila, 2019. Sakila sample database (accessed February 2019). URL: https://dev.mysql.com/doc/sakila/en/sakila-structure.html.

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, 276–281. doi:10.1109/TSMCA.2003.820581.

Siponen, M., Klaavuniemi, T., 2019. Narrowing the theory's or study's scope may increase practical relevance, in: Proceedings of the Annual Hawaii International Conference on System Sciences, University of Hawai'i at Manoa. pp. 6260–6269. URL: https://scholarspace.manoa.hawaii.edu/handle/10125/60060.

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

Taipalus, T., 2019. Teaching Tip: A Notation for Planning SQL Queries. Journal of Information Systems Education 30, 160–166. URL: http://jise.org/Volume30/n3/JISEv30n3p160.pdf.

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, ACM, New York, NY, USA. pp. 198–203. doi:10.1145/3287324.3287359.

Taipalus, T., Siponen, M., Vartiainen, T., 2018. Errors and complications in SQL query formulation. ACM Transactions on Computing Education 18, 15:1–15:29. doi:10.1145/3231712.

Topi, H., Kaiser, K.M., Sipior, J.C., Valacich, J.S., Nunamaker, Jr., J.F., de Vreede, G.J., Wright, R., 2010. Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. Technical Report. ACM/AIS. New York, NY, USA. URL: https://dl.acm.org/citation.cfm?id=2593310.

Topi, H., Valacich, J.S., Hoffer, J.A., 2005. The effects of task complexity and time availability limitations on human performance in database query tasks. International Journal of Human-Computer Studies 62, 349–379. doi:10.1016/j.ijhcs.2004.10.003.

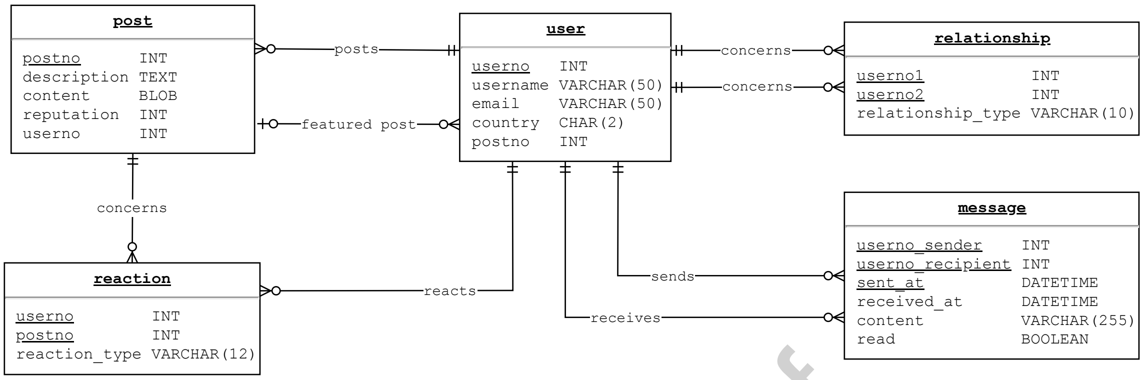
Wagner, P.J., Shoop, E., Carlis, J.V., 2003. Using scientific data to teach a database systems course, in: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, ACM, New York, NY, USA. pp. 224–228. doi:10.1145/611892.611975.

Warszawski, T., Bailis, P., 2017. ACIDRain: Concurrency-related attacks on database-backed web applications, in: Proceedings of the 2017 ACM International Conference on Management of Data, ACM, New York, NY, USA. pp. 5–20. doi:10.1145/3035918.3064037.

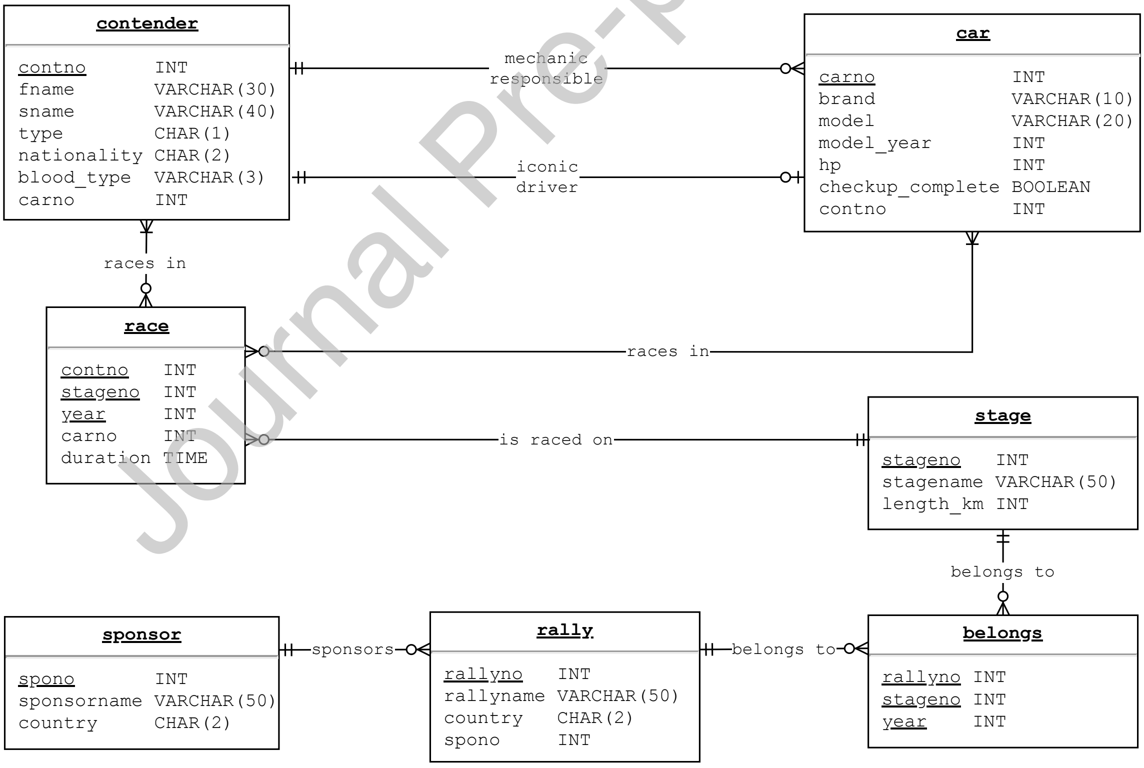
Watson, H.J., Hoffer, J.A., 2003. Teradata university network: A new resource for teaching large data bases and their applications. Communications of the Association for Information Systems 12, 9. URL: https://aisel.aisnet.org/cais/vol12/iss1/9/.

Yue, K.B., 2013. Using a semi-realistic database to support a database course. Journal of Information Systems Education 24, 327–336. URL: http://jise.org/Volume24/n4/JISEv24n4p327.html.

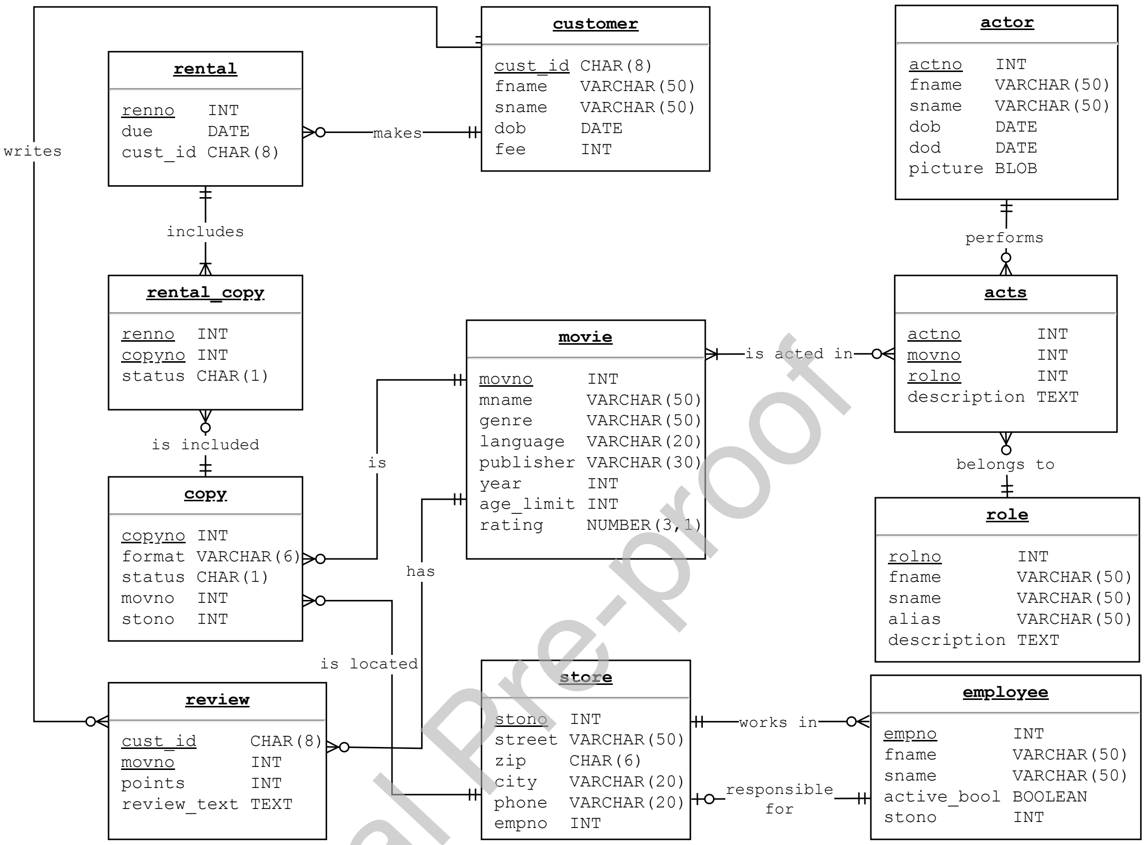
Appendix A. Simple database schema



Appendix B. Semi-complex database schema



Appendix C. Complex database schema (Taipalus et al., 2018)



Appendix D. Data demands and queries

	Simple	Semi-complex	Complex (Taipalus et al., 2018)

A1	List all information regarding users from Finland (FI) and Sweden (SE).
	SELECT *
	FROM user
	WHERE country IN ('FI', 'SE');
		List all information regarding cars from Opel and Toyota.
		SELECT *
		FROM car
		WHERE brand IN ('Opel', 'Toyota');
			List all information regarding stores in Helsinki and Tampere.
			SELECT *
			FROM store
			WHERE city IN ('Helsinki', 'Tampere');

A2	List the user numbers, user names, countries and emails of users who are Australian (AU) but have no featured post. Sort the results according to user name in ascending order.
	SELECT userno, username, country, email
	FROM user
	WHERE country = 'AU'
	AND postno IS NULL
	ORDER BY username ASC;
		List the names, nationalities and blood types of co-drivers who are not Finnish (FI). Sort the results according to surname in ascending order.
		SELECT fname, sname, nationality, bloodtype
		FROM contender
		WHERE type = 'c'
		AND nationality <> 'FI'
		ORDER BY surname ASC;
			List the names, age limits and years of movies that are in English but are not published by Goldeneye BC. Sort the results according to the name of the movie in ascending order.
			SELECT mname, age_limit, year
			FROM movie
			WHERE language = 'English'
			AND publisher <> 'Goldeneye BC'
			ORDER BY mname ASC;

A3	List the post numbers, descriptions and user numbers who made the post, of posts which description starts with an S or an R, and that have been posted by users whose user number is 1001 or 1003.
	SELECT postno, description, userno
	FROM post
	WHERE (description LIKE 'S%' OR description LIKE 'R%') AND (userno = 1001 OR userno = 1003);
		List the names and nationalities of contenders whose surname starts with an A or a C, and who are from the United Kingdom (UK) or the United States (US).
		SELECT fname, sname, nationality
		FROM contender
		WHERE (surname LIKE 'A%' OR surname LIKE 'C%') AND (nationality = 'UK' or nationality = 'US');
			List the names, dates of birth and death of actors whose surname starts with an F or an S, and whose date of birth is unknown, or who have a date of death.
			SELECT fname, sname, dob, dod
			FROM actor
			WHERE (sname LIKE 'F%' OR sname LIKE 'S%')
			AND (dob IS NULL OR dod IS NOT NULL);

B4	List user names and emails of users who have a featured post with no description.
	SELECT u.username, u.email
	FROM user u, post p
	WHERE u.postno = p.postno
	AND p.description IS NULL;
		List the car brand and model of the car of which Ari Vatanen is the iconic driver.
		SELECT c.brand, c.model
		FROM car c, contender d
		WHERE c.contno = d.contno
		AND d.fname = 'Ari' AND d.sname = 'Vatanen';
			List the city and phone number of the store in which Jaakko Mattila works.
			SELECT s.city, s.phone
			FROM store s, employee e
			WHERE s.stono = e.stono
			AND e.fname = 'Jaakko' AND e.sname = 'Mattila';

	Simple	Semi-complex	Complex (Taipalus et al., 2018)

B5	List the post numbers and descriptions of posts which at least one Finnish (FI) user has considered funny. Sort the results according to post number in descending order.
	SELECT p.postno, p.description
	FROM post p
	INNER JOIN reaction r ON (p.postno = r.postno)
	INNER JOIN user u ON (r.userno = u.userno)
	WHERE u.country = ’FI’
	AND r.reaction_type = ’funny’
	ORDER BY p.postno DESC;
		List the names and countries of rallies which included at least one stage of over 20 kilometers in 1998. Sort the results according to rally name in descending order.
		SELECT r.rallyname, r.country
		FROM rally r
		INNER JOIN belongs b ON (r.rallyno = b.rallyno)
		INNER JOIN stage s ON (b.stageno = s.stageno)
		WHERE b.year = 1998
		AND s.length_km > 20
		ORDER BY r.rallyname DESC;
			List the names of actors whose date of death is known and who have acted in at least one movie released after 2010. Sort the results according to surname in descending order.
			SELECT a.fname, a.sname
			FROM actor a
			INNER JOIN acts ac ON (a.actno = ac.actno)
			INNER JOIN movie m ON (ac.movno = m.movno)
			WHERE a.dod IS NOT NULL
			AND m.year > 2010
			ORDER BY a.sname DESC;

B6	List the contents and user numbers of the receivers of messages which were sent by an user with user number 1001 or 1003, and who has reacted to some post at least once. Sort the results according to the user number, and then according to content, both in ascending order.
	SELECT m.content, m.userno_receiver
	FROM message m
	WHERE EXISTS
	(SELECT *
	FROM user u
	WHERE m.userno_sender = u.userno
	AND (u.userno = 1001 OR u.userno = 1003)
	AND EXISTS
	(SELECT *
	FROM reaction r
	WHERE u.userno = r.userno)
	)
	ORDER BY m.userno_receiver ASC, m.content ASC;
		List the brands and models of cars which have been driven at least once on stage called Sweet Lamb 1 or Sweet Lamb 2. Sort the results according to brand, and then according to model, both in ascending order.
		SELECT c.brand, c.model
		FROM car c
		WHERE EXISTS
		(SELECT *
		FROM race r
		WHERE c.carno = r.carno
		AND EXISTS
		(SELECT *
		FROM stage s
		WHERE r.stageno = s.stageno
		AND (s.stagename = ’Sweet Lamb 1’
		OR s.stagename = ’Sweet Lamb 2’))
		)
		ORDER BY c.brand ASC, c.model ASC;
			List the names of actors who have acted a role as himself or herself. Sort the results according to surname, and then according to first name, both in ascending order.
			SELECT a.fname, a.sname
			FROM actor a
			WHERE EXISTS
			(SELECT *
			FROM acts ac
			WHERE a.actno = ac.actno
			AND EXISTS
			(SELECT *
			FROM role r
			WHERE ac.rolno = r.rolno
			AND (r.alias = ’Himself’
			OR r.alias = ’Herself’))
			)
			ORDER BY a.sname ASC, a.fname ASC;

	Simple	Semi-complex	Complex (Taipalus et al., 2018)

B7	List the post numbers, contents and reputations of posts which have a reputation greater than 0, but which no one has ever considered funny.
	SELECT p.postno, p.content, p.reputation
	FROM post p
	WHERE p.reputation > 0
	AND NOT EXISTS
	(SELECT *
	FROM reaction r
	WHERE p.postno = r.postno
	AND r.reaction_type = ’funny’);
		List the names and nationalities of co-drivers who have never raced in 1986-2010.
		SELECT c.fname, c.sname, c.nationality
		FROM contender c
		WHERE c.type = ’c’
		AND NOT EXISTS
		(SELECT *
		FROM race r
		WHERE c.contno = r.contno
		AND r.year BETWEEN 1986 AND 2010);
			List the movie numbers, names and years of movies that have been released in the first decade of the 2000s, but of which there exists no copy in BluRay format.
			SELECT m.movno, m.mname, m.year
			FROM movie m
			WHERE m.year BETWEEN 2000 AND 2009
			AND NOT EXISTS
			(SELECT *
			FROM copy c
			WHERE m.movno = c.movno
			AND c.format = ’BluRay’);

B8	List the user names, emails and countries of users who have never posted anything but who have at least once reacted to a post.
	SELECT u.username, u.email, u.country
	FROM user u
	WHERE NOT EXISTS
	(SELECT *
	FROM post p
	WHERE u.userno = p.userno)
	AND EXISTS
	(SELECT *
	FROM reaction r
	WHERE u.userno = r.userno);
		List the stage numbers, names and lengths of stages which have never been a part of any rally but on which someone has raced at least once.
		SELECT s.stageno, s.stagename, s.length_km
		FROM stage s
		WHERE NOT EXISTS
		(SELECT *
		FROM belongs b
		WHERE s.stageno = b.stageno)
		AND EXISTS
		(SELECT *
		FROM race r
		WHERE s.stageno = r.stageno);
			List the names and dates of birth of customers who have never rented a movie but who have given at least one review.
			SELECT c.fname, c.sname, c.dob
			FROM customer c
			WHERE NOT EXISTS
			(SELECT *
			FROM rental rt
			WHERE c.cust_id = rt.cust_id)
			AND EXISTS
			(SELECT *
			FROM review rv
			WHERE c.cust_id = rv.cust_id);

B9	List the average of post reputations with a reputation greater than 0. Rename the column in the result table descriptively.
	SELECT AVG(repuration) AS ”average positive reputation”
	FROM post
	WHERE reputation > 0;
		List the number of stages with the length between 4 and 10 kilometers. Rename the column in the result table descriptively.
		SELECT COUNT(*) AS ”number of 4-10 km stages”
		FROM stage
		WHERE length_km BETWEEN 4 AND 10;
			List the number of movies released between the years 1970-2000. Rename the column in the result table descriptively.
			SELECT COUNT(*) AS ”movies released in 1970-2000”
			FROM movie
			WHERE year BETWEEN 1970 AND 2000;

	Simple	Semi-complex	Complex (Taipalus et al., 2018)
B10	List the user names, emails, countries, and reaction types and post numbers which the reactions concern, but only from users who are married. Rename the columns in the result table descriptively. SELECT u.username AS ”user name”, u.email AS ”email”, u.country AS ”country”, r.reaction_type AS ”reaction”, p.postno AS ”post number” FROM user u, reaction r, post p, relationship s WHERE u.userno = r.userno AND r.postno = p.postno AND u.userno = s.userno1 AND s.relationship_type = ’marriage’;	List the names and types of contenders, and the car brands and models with which they have raced on a stage called Ouninpohja. Rename the columns in the result table descriptively. SELECT c.fname AS ”first name”, c.sname AS ”surname”, c.type AS ”type”, a.brand AS ”brand”, a.model AS ”model” FROM contender c, car a, race r, stage s WHERE c.contno = r.contno AND r.stageno = s.stageno AND r.carno = a.carno AND s.stagename = ’Ouninpohja’;	List the names of actors who have acted in the movie Physics 101 and list the names of the roles they have played in that movie. Rename the columns in the result table descriptively. SELECT a.fname AS ”actor’s first name”, a.sname AS ”actor’s surname”, r.fname AS ”character’s first name”, r.sname AS ”character’s surname”, r.alias AS ”character’s alias” FROM movie m, actor a, acts ac, role r WHERE m.movno = ac.movno AND ac.rolno = r.rolno AND a.actno = ac.actno AND m.mname = ’Physics 101’;

B11	List the contents, sender user number, and the time the message was sent of the oldest unread message.
	SELECT content, userno_sender, sent_at
	FROM message
	WHERE read = False
	AND sent_at =
	(SELECT MIN(sent_at)
	FROM message
	WHERE read = False);
		List the car number, model year and horse powers of the oldest Toyota.
		SELECT carno, model_year, hp
		FROM car
		WHERE brand = ’Toyota’
		AND model_year =
		(SELECT MIN(model_year)
		FROM car
		WHERE brand = ’Toyota’);
			List the name, year and genre of the oldest movie published by Goldeneye BC.
			SELECT mname, year, genre
			FROM movie
			WHERE publisher = ’Goldeneye BC’
			AND year =
			(SELECT MIN(year)
			FROM movie
			WHERE publisher = ’Goldeneye BC’);

B12	List the user names and emails of users who have sent messages to exactly six different users.
	SELECT u.username, u.email
	FROM user u
	WHERE 6 =
	(SELECT COUNT(DISTINCT m.userno_recipient)
	FROM message m
	WHERE u.userno = m.userno_sender);
		List the names of contenders who have raced with at least three different cars.
		SELECT c.fname, c.sname
		FROM contender c
		WHERE 2 <
		(SELECT COUNT(DISTINCT r.carno)
		FROM race r
		WHERE r.carno = c.carno);
			List the actor numbers and full names of actors who have acted in at least five different movies.
			SELECT a.actno, a.fname, a.sname
			FROM actor a
			WHERE 4 <
			(SELECT COUNT(DISTINCT ac.movno)
			FROM acts ac
			WHERE a.actno = ac.actno);

	Simple	Semi-complex	Complex (Taipalus et al., 2018)
B13	List the user names and countries of users who have posted a post which has received the at least one similar type of reaction as any post made by user 1004. SELECT u.userno, u.country FROM user u, post p1, post p2, reaction r1, reaction r2 WHERE u.userno = p1.userno AND p1.postno = r1.postno AND r1.reaction_type = r2.reaction_type AND r2.postno = p2.postno AND p2.userno = 1004 AND u.userno <> 1004;	List the numbers and names of rallies which have at least one stage which is of the same length as some stage that has been part of the Rally of Wales (rallyno = 201), whenever. SELECT r.rallyno, r.rallyname FROM rally r, belongs b1, belongs b2, stage s1, stage s2 WHERE r.rallyno = b1.rallyno AND b1.stageno = s1.stageno AND s1.length_km = s2.length_km AND s2.stageno = b2.stageno AND b2.rallyno = 201 AND r.rallyno <> 201;	List the names of customers who have rented exactly the same movie copy that Robert Butler (rbutler1) has rented, whenever. SELECT c.fname, c.sname FROM customer c, rental r1, rental_copy rc1, rental_copy rc2, rental r2 WHERE c.cust_id = r1.cust_id AND r1.renno = rc1.renno AND rc1.copyno = rc2.copyno AND rc2.renno = r2.renno AND r2.cust_id = 'rbutler1' AND c.cust_id <> 'rbutler1';

C14	List the numbers of users by country and relationship type. Sort the results by country in ascending order. Make sure that the structure of the result table is as below [example given].
	SELECT u.country, s.relationship_type, COUNT(u.userno) AS total
	FROM user u, relationship s
	WHERE u.userno = s.userno1
	GROUP BY u.country, s.relationship_type
	ORDER BY u.country ASC;
		List the numbers of stages by rally name and year. Sort the results by year in ascending order. Make sure that the structure of the result table is as below [example given].
		SELECT r.rallyname, b.year, COUNT(b.stageno) AS total
		FROM rally r, belongs b
		WHERE r.rallyno = b.rallyno
		GROUP BY r.rallyname, b.year
		ORDER BY b.year ASC;
			List the numbers of movie copies located in stores by city and status of the copy. Sort the results by city in ascending order. Make sure that the structure of the result table is as below [example given].
			SELECT s.city, c.status, COUNT(c.copyno) AS total
			FROM store s, copy c
			WHERE c.stono = s.stono
			GROUP BY s.city, c.status
			ORDER BY s.city ASC;

	Simple	Semi-complex	Complex (Taipalus et al., 2018)

C15	List the numbers of sent messages by the sender's country and message read status. Disregard senders with less than four sent messages, regardless of the message read status. Sort the results according to the number of messages sent in descending order.
	SELECT u.country, m.read, COUNT(m.userno_sender) AS total
	FROM user u, message m
	WHERE u.userno = m.userno_sender
	GROUP BY u.country, m.read
	HAVING COUNT(m.userno_sender) > 3
	ORDER BY total DESC;
		List the numbers of raced stages by contender number and nationality. Disregard contenders with less than seven raced stages. Sort the results according to the number of stages raced in descending order.
		SELECT c.contno, c.nationality, COUNT(r.stageno) AS total
		FROM contender c, race r
		WHERE c.contno = r.contno
		GROUP BY c.contno, c.country
		HAVING COUNT(r.stageno) > 6
		ORDER BY total DESC;
			List the numbers of movie copies by movie number and movie name. Disregard movies of which there are less than six copies, regardless of the status of the copy. Sort the results according to the number of the copies in descending order.
			SELECT m.movno, m.mname, COUNT(c.movno) AS total
			FROM movie m, copy c
			WHERE m.movno = c.movno
			GROUP BY m.movno, m.mname
			HAVING COUNT(c.movno) > 5
			ORDER BY total DESC;

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Author contributions

Toni Taipalus: sole author.

Toni Taipalus is a university teacher at the Faculty of Information Technology, University of Jyvaskyla. He teaches databases, data management, application programming, and system development. His research interests include pedagogical aspects of query languages, data models, and agile software development.