An Experimental Investigation of Complexity in Database Query Formulation Tasks

Gretchen Irwin Casterella
Leo Vijayasarathy
Computer Information Systems Department
College of Business
Colorado State University
Fort Collins, CO 80525-1277
Gretchen.Irwin@business.colostate.edu, Leo.Vijayasarathy@business.colostate.edu

ABSTRACT

Information Technology professionals and other knowledge workers rely on their ability to extract data from organizational databases to respond to business questions and support decision making. Structured query language (SQL) is the standard programming language for querying data in relational databases, and SQL skills are in high demand and are taught in most introductory database courses. We examined students' performance on query formulation tasks, in an experimental setting which varied the complexity of the query and the ambiguity of the information request. Our results confirm the main effects of query complexity and request ambiguity found in prior studies (Borthick et al. 2001). In addition, we found an interaction effect between complexity and ambiguity, namely that low ambiguity is more important as tasks increase in complexity. We also found that students' confidence with entity-relationship diagrams corresponds to reduced time spent on query formulation, and their ability to evaluate the accuracy of their queries reduces as query complexity increases. We discuss the implications of these findings with some suggestions for future research.

Keywords: Query language, Database management systems (DBMS), Data modeling

1. INTRODUCTION

Information Technology professionals and other knowledge workers rely on their ability to extract data from organizational databases to respond to business questions and support decision making. While there are many graphical user interface tools that allow end-users to summarize and view organizational data, structured query language (SQL) is still the standard programming language for formulating ad hoc queries against relational databases (Allen & March, 2006). Query formulation with SQL is a skill that is in high demand and is taught in most introductory database courses. Query formulation can be a complex task because it often includes a high degree of requirements uncertainty (e.g., ambiguity in the request for information), multiple solution paths that produce the correct result, and a high degree of information overload when working with large data models (Bowen et al., 2009; Ashkanasy et al., 2007; Borthick et al., 2001; Campbell 1988).

In this study, we investigate two factors that impact query writing performance—the ambiguity in the information request and the complexity of the target solution. We examine performance in terms of the accuracy of the query solution, the time taken to produce the solution, and the writer's confidence in the quality of his solution. The purpose of the study is to confirm the main effects of ambiguity and solution complexity on performance (as in Borthick et al., 2001) and to evaluate the interaction effects of ambiguity and complexity on performance. Our goal is to use these findings to better understand why some queries are more difficult to formulate than others, and to identify potential teaching strategies and techniques to facilitate students' acquisition of SQL skills.

2. PRIOR RESEARCH ON QUERY FORMULATION

Reisner's (1981) classic model of the query formulation process is shown in Figure 1. According to this model, the query writer is given an information request (e.g., "Find the salary of Smith's manager") and generates a mental "query template" of an SQL SELECT statement. The template specifies the structural foundation for the query. The query writer then maps elements from the information request into SQL components that can be inserted into the appropriate "slots" of the template. The mapping involves three transformational activities: (1) replacing words from the information request with elements from the data model (e.g., replacing the "salary" with the column SAL), (2) adding elements to the SELECT statement beyond what is in the

information request (e.g., "Smith's manager" → NAME = (SELECT MGR WHERE NAME = 'Smith'), and (3) ignoring terms from the information request that are not needed in the SELECT statement.

This model of template-generation-plus-mapping provides a reasonable starting point for understanding the process of query formulation and two sources of complexity in query formulation tasks—structural complexity and (lexical) transformational complexity (Reisner, 1977). Structural complexity addresses questions about the query template, such as whether the FROM clause specifies an inner or outer join, or whether a WHERE, GROUP BY, or HAVING clause is needed.

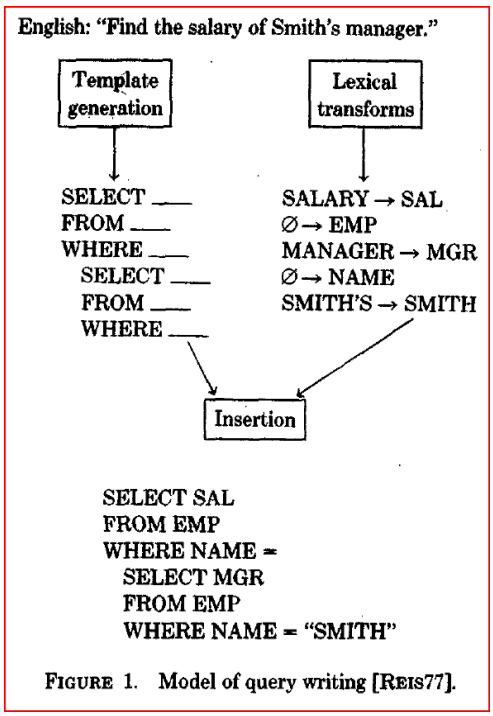


Figure 1. Reisner's (1981) Model of the Query Formulation Process

Transformational complexity stems from the complexity in the lexical mappings shown in Figure 1 – the replacing, adding, and omitting of lexical elements from the natural language request to fill in the "slots" in the query template. Transformational complexity increases as the "gap" between terms in the information request and elements in the data model increases, and is influenced by the degree of ambiguity in the information request (Borthick et al., 2001). For example, a request such as, "Which customers placed online orders over $3,000 last July?" could also be worded more precisely as, "List the customer's name and account number, if the customer placed an order between July 1, 2012 and July 30, 2012 with an order total greater than 3000 and an online order flag equal to 1." We would expect the former task wording to create more transformational complexity because the query writer has to know, for example, that "online orders" translates into "OnlineOrderFlag = 1" and that "placed last July" translates into "OrderDate BETWEEN '7/1/2012' AND '7/30/2012'. Thus, lexical or transformational complexity is related to the query writer's knowledge of the user's domain and of the data model (Allen & Parsons, 2010).

Borthick et al. (2001) provide an alternative model of the query formulation process, shown in Figure 2. According to this model, query formulation begins with an analysis of the information request, followed by an evaluation of the data representation, and these two sources are used to create a mental model of how the data will be manipulated to fulfil the information request (e.g., "tables x and y need to be joined on column z, and columns a and b need to be returned"). Presumably, this mental model may be consistent with Reisner's (1981) model of a query template with "slots" for the lexical data model elements. This mental model is then translated into specific query language syntax.

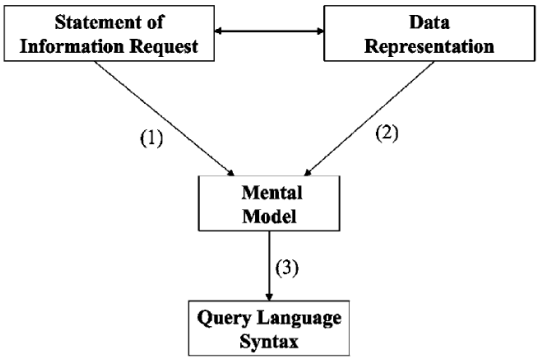


Figure 2. Borthick et al.'s (2001) Model of the Query Formulation Process

Borthick et al.'s model emphasizes two external sources of information—the information request and the data representation—that provide input to generating the correct mental model (Borthick et al., 2001). Characteristics of the information request and the data representation can hinder or facilitate the formation of an appropriate mental model of the query, and subsequently hinder or facilitate the formation of a correct solution. One source of query formulation complexity stems from the *information requirement distance*, which is the gap between the words in the information request and the operations and operators in the query language, shown by paths (1) and (3) in Figure 2. This is similar to Reisner's transformational complexity, which is higher when information requests have higher levels of ambiguity. Borthick et al. (2001) investigated the impact of ambiguity on query quality and found that participants performed better with pseudo-SQL (low ambiguity) requests than with manager-English (high ambiguity) requests. While their study supported the main effect between information request ambiguity and query performance, they did not study the interaction between ambiguity and query complexity.

3. RESEARCH MODEL & HYPOTHESES

Figure 3 shows our research model of the query formulation process, which extends Borthick et al.'s (2001) model. In our model, the query writer generates a mental model of the

SELECT statement based on the information request, an external representation of the database (e.g., an entity-relationship diagram or relational schema), and three *internal* sources of knowledge: (1) domain knowledge; (2) data model knowledge; and (3) query language knowledge.

Domain knowledge and data model knowledge are used to map elements in the information request into the appropriate tables and columns in the database (i.e., lexical mapping). Query language knowledge is needed to generate the correct SELECT statement template (i.e., structural mapping). In addition, the combination of these knowledge sources is needed if data needs to be transformed in the query (e.g., applying a YEAR function to a date column, or an AVERAGE function to a set of column values).The query writer formulates an SQL statement in a particular software tool based on his mental model, executes the query, and receives feedback in the form of error messages or a result set. The query writer may use this feedback to modify his mental model, which may signal the need for further examination of one or more knowledge sources.

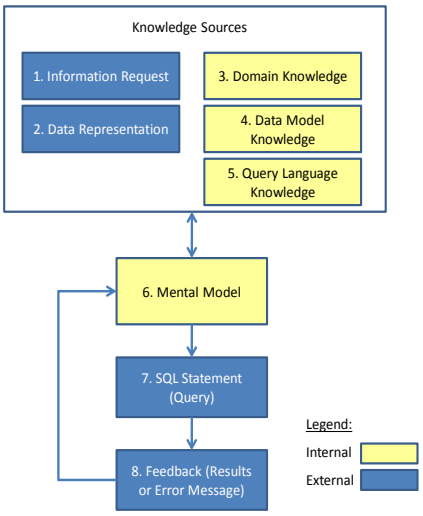


Figure 3. Research Model of the Query Formulation Process

To illustrate the process, consider the example from a sales order database shown in Figure 4. In order to formulate the correct query, the writer has to use domain and data model knowledge to determine, for example, that "order numbers" corresponds to the SalesOrderPK column in the SalesOrderHeader table and that "online orders" will be those where the OnlineOrderFlag column has a value of 1. These are examples of lexical mappings.

The writer also needs query language knowledge to recognize that WHERE and GROUP BY clauses are needed but a HAVING clause is not. These are examples of generating the appropriate structural components of the query. In addition, the writer has to recognize that the "total quantity" of an order is not stored in the database, but can be derived by summing the OrderQty values after grouping by SalesOrderPK, which involves both structural (the SUM function, GROUP BY clause) and lexical (OrderQty) elements.

The feedback loop (see Figure 3) is important because users often accept or revise their initial queries (and, implicitly, their mental models of the query) based on the results or error messages they receive. If the mental model needs changing, the user may revisit the information request or the data model (external information sources), and/or revise his/her understanding of the problem domain, the data model, or the query language (internal knowledge sources).

With reference to our research model, this study varies one characteristic of the information request, namely, the level of ambiguity, across several query formulation tasks of increasing complexity (in terms of the complexity of the query solution). We hold constant the data model representation, and we control for differences in domain, data model, and SQL knowledge. We also control for other individual characteristics. Our hypotheses examine the impact of our independent variables on query *outcomes*:

H1: (Main Effect): Query formulation performance is inversely related to the difficulty of the SELECT statement solution.

H2: (Main Effect): Query formulation performance is higher with a low-ambiguity request than with a high-ambiguity request.

H3: (Interaction Effect) Request ambiguity has a more pronounced effect on query formulation performance as the queries increase in complexity.

4. RESEARCH METHOD

This section discusses the experimental design and the measures, participants, tools, and procedure we used in this study.

4.1 Subjects

The participants were thirty-three undergraduate juniors and seniors enrolled in a database management course in the Computer Information Systems department of a large US public university. Similar to prior studies, we controlled for some individual differences (e.g., age, educational background) by choosing subjects from a fairly homogenous pool of students and randomly assigning them to experimental conditions (Bowen et al., 2009). We also used self-reported measures of GPA, comfort reading ER diagrams, and comfort with SQL as covariates, to control for other individual differences (Bowen et al., 2004; Allen & March, 2006).

4.2 Data Collection Tool – CeeKwel

We used Microsoft development technologies to build a software tool, called CeeKwel, with a tabbed-interface with a query editor for writing and executing SELECT statements, and a feedback area for displaying error messages or query results. CeeKwel created a participant-specific log of every query that was executed, along with a timestamp and the results of the execution (i.e., result set or error message). This tool is similar to that used in other studies of query formulation (e.g., Allen & Parsons, 2010; Bowen et al., 2009; Allen & March, 2006; Bowen et al., 2006) in that it is an online tool and participants are allowed to revise their queries as often as they like, based on the query results or error messages they receive.

External Knowledge Source	Example
Information Request	"List the order numbers, total quantity, and total dollars for all online orders over $5,000 that were placed in 2012."
Data Representation	Entity-relationship diagram and data dictionary (see Figure 5).
(Correct) Query Statement*	SELECT *SalesOrderPK*, SUM(*OrderQty*), *TotalDue* FROM *SalesOrderHeader* JOIN *SalesOrderDetail* ON *SalesOrderPK*= *SalesOrder* FK WHERE *OnlineOrderFlag*= *1* AND *TotalDue*>*5000* AND YEAR (*OrderDate*) = *2012* GROUP BY *SalesOrderPK, TotalDue*

** Italicized terms are lexical elements; other elements are structural.*

Figure 4. Example Query Formulation Task

4.2 Experimental Tasks and Independent Variable Measures

The two independent variables of interest in this study were query difficulty and information request ambiguity. Query difficulty was measured by calculating the Halstead (1977) difficulty measure of the query solution for each task (see Borthick et al., 2001). For information request ambiguity, we used a dichotomous measure: pseudo-SQL (low ambiguity) or manager-English (high ambiguity) wording (Borthick et al., 2001). Each task was written with both wordings. The managerial version was a natural language request, such as, "How many products do we manufacture in-house?" The pseudo-SQL version was written to facilitate the mapping between user-requested information and specific table names, column names, and data values in the query solution. For example, a pseudo-SQL version of the previous request would be, "Show the count of products that have a value of 1 for the MakeFlag column."

Figure 5 shows the design of the database used in our study, which was a modified subset of Microsoft's SQL Server AdventureWorks database. In addition to the Entity-Relationship Diagram shown in Figure 5, we provided a data dictionary with attribute definitions and data types. Figure 6 lists some of the experimental query tasks for this database, with the pseudo-SQL and manager-English versions, the corresponding query solution, and the Halstead difficulty measure for the solution.

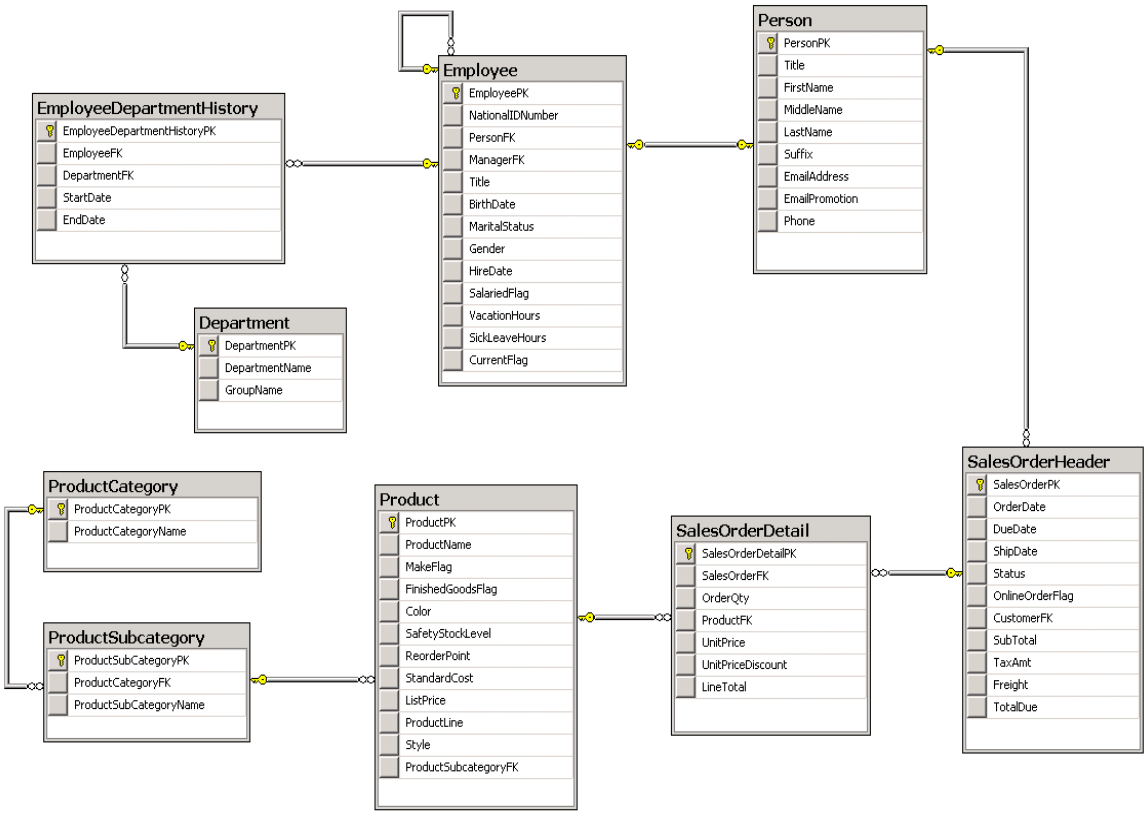


Figure 5. Entity-Relationship Diagram for Experimental Tasks.

Task #	Task Wording (Manager-English (M) or Pseudo-SQL (P))	Correct SQL Statement	Halstead Difficulty
1	(M) How many products are manufactured in-house? (P) Show the number of products that are manufactured in-house. (Hint: Products manufactured in house have a value of 1 for the MakeFlag column.)	SELECT COUNT(*) FROM Product WHERE MakeFlag = 1	4.8
3	(M) List the name, job title, and total available vacation and sick leave hours for the employee(s) with no manager. (P) List the first name, last name, job title, and total available hours for the employee(s) with no manager. Total available hours is calculated as the sum of an employee's vacation hours and sick leave hours. (Hint: Employees with no manager will have a NULL value for the ManagerFK column.)	SELECT FirstName, LastName, Employee.Title,(VacationHours + SickLeaveHours)AS UnpaidHours FROM Person JOIN Employee ON PersonPK = PersonFK WHERE ManagerFK ISNULL	16.7
4	(M) Which sales orders over $3,500 placed in 2001 were either placed online or placed by customer #17584? For each of these orders, list the sales order number, order subtotal, and whether it was an online order. Sort the results so that the highest subtotal amount is first and the lowest subtotal amount is last. (P) Create a report with columns for the sales order primary key, the order subtotal and the online order flag. List only those sales orders that have an order date in the year 2001, and have a subtotal greater than 3500, and were either online orders or placed by customer number 17584. Sort the results by subtotal in descending order.	SELECT SalesOrderPK, SubTotal, OnlineOrderFlag FROM SalesOrderHeader WHEREYEAR(OrderDate)= 2001 AND SubTotal > 3500 AND (OnlineOrderFlag = 1 OR CustomerFK = 17584) ORDER BY SubTotal DESC	16.7
5	(M) Which sales orders were placed in July of 2003 and contained more than 3 line items? For each of these orders, list the sales order number, the order subtotal, and the number of line items on the order. (P) List the sales order primary key, the order subtotal, and the number of line items for those sales orders with an order date between July 1, 2003 and July 31, 2003. Include only those sales orders that had more than 3 line items. (Hint: The line items for a sales order are stored in the Sales Order Detail table.)	SELECT SalesOrderPK, SubTotal,COUNT(*)AS NumberOfItems FROM SalesOrderHeader JOIN SalesOrderDetail ON SalesOrderPK = SalesOrderFK WHERE OrderDate BETWEEN '7/01/2003' AND '7/31/2003' GROUPBY SalesOrderPK, SubTotal HAVING COUNT(*)> 3	20

Figure 6. Sample Query Tasks, Solutions, and Halstead Complexity Score

4.4 Procedure

We collected data over two seventy-five minute class periods. During the first class period, students were given an overview of the study, a demonstration of CeeKwel, and a training exercise. The experimental session was conducted two days later during the next class period. Participants were given the ERD and a Data Dictionary excerpt for a sales database (Figure 5). They studied the database design and completed a short, data model comprehension quiz in CeeKwel. Then they were given a series of six query formulation tasks.

For each task, CeeKwel displayed an information request, and the participant wrote a SELECT statement in the editor, executed it, received feedback, and then either revised the SELECT statement, or requested to move on to the next task. Before the next task was displayed, the participant had to rate his/her confidence in the accuracy of the completed task, on a scale of 1 (not at all confident) to 5 (very confident). At the conclusion of the session, each participant completed a background survey (e.g., age, GPA, comfort with ERDs, comfort with SQL).

Each participant received the same six tasks in the same order, with easier tasks first. However, each participant saw only one version of each task, either the pseudo-SQL or the managerial wording. Thus, our experiment was a combination of between-subject (for wording) and within-subject (for task difficulty) designs, which is similar to the design of previous query formulation studies (Allen & March, 2006; Chan et al., 2005; Borthick et al., 2001; Rho & March, 1997; Chan, 1999).

4.5 Dependent Variable and Covariate Measures

To examine query performance, we were primarily interested in query quality or accuracy. We graded each participant's final query attempt for each task, using a grading scheme based on the percentage of correct elements in the participant's query (Bowen et al., 2009; Allen & March, 2006; Borthick et al., 2001). For example, the following figure shows the accuracy coding for one participant's solution to the fourth query task. A trained research assistant performed the query assessment.

Clause	Elements	Required Elements for This Query	Max. Count	Actual Count
SELECT	Clause	Select	1	1
	Attributes	SalesOrderPK, SubTotal, OnlineOrderFlag	3	3
	Keywords			
	Arithmetic Operators			
	Scalar Functions			
	Aggregate Functions			
FROM	Clause	From	1	1
	Tables	SalesOrderHeader	1	1
	Join Conditions			
WHERE	Clause	Where	1	1
	Join Conditions			
	Attributes	OrderDate, Subtotal, OnlineOrderFlag, CustomerFK	4	4
	Logical Operators	And, And, ( ), Or	4	1
	Comparison Operators	=, >, =, =	4	4
	Arithmetic Operators			
	Scalar Functions	Year	1	0
	Values	2001, 3500, 1, 17584	4	4
GROUP BY	Clause			
	Attributes			
HAVING	Clause			
	Attributes			
	Keywords			
	Logical Operators			
	Comparison Operators			
	Arithmetic Operators			
	Scalar Functions			
	Aggregate Functions			
	Values			
ORDER BY	Clause	Order By	1	1
	Attributes	SubTotal	1	1
	Keywords	Desc	1	1
		Total = 23/27 = 85.2%	27	23

Figure 7. Accuracy Coding for Participant #8020's solution to query task #4

In addition, the authors randomly selected and assessed about one-third of the queries each. The Pearson *r* between the assistant's scores and the authors' was 0.99, confirming the consistency of the scoring process. In addition to query quality, we also examined performance in terms of **query formulation time** (the difference, in seconds, between the final query's submission and the task's opening), **the number of query attempts** (a count of query tries that were executed), and the participant of t's **confidence level** (on a scale of 1, for "not at all confident," to 5, for "very confident"). Figure 8 shows summary statistics for each dependent variable by task difficulty and wording.

Two other performance-related measures we used were the mean probability score and the judgment bias score (Allen & Parsons, 2010). **Mean probability scores** reflect the relationship between the confidence expressed by subjects in their queries and their actual correctness. We followed Allen & Parsons' (2010) procedure for computing the probability score (Yates, 1990), which ranged from 0 to 1. A score of 0 indicates perfect prediction, i.e., high confidence and a correct query, whereas a score of 1 indicates poor prediction, i.e., high confidence and an incorrect query. Mean probability score for a query task is the average of all subjects' probability scores for that task.

The **judgment bias score** is the raw difference between confidence and correctness, and provides an assessment of a subjects' under- or over-confidence in their query (Allen and Parsons, 2010; Yates, 1990). Again, we followed Allen and Parsons' procedure for calculating this score, which ranged from -1 to 1. Negative scores indicate under-confidence and positive scores reflect over-confidence.

Dependent Variable	Statistic	Query Task (Halstead Difficulty Score)												Total M n=92	Total P n=94	Total n=186
		Task 1 (4.17)			Task 2 (11.11)			Tasks 3 & 4 (16.67)			Tasks 5 & 6 (20.00)					
		M[1] n=16	P[2] n=16	Total n=32	M n=16	P n=16	Total n=32	M n=31	P n=32	Total n=63	M n=29	P n=30	Total n=59			
Quality	Mean	0.97	0.97	0.97	0.90	0.99	0.95	0.84	0.89	0.86	0.61	0.71	0.66	0.80	0.86	0.83
	SD	0.14	0.11	0.12	0.22	0.03	0.16	0.14	0.21	0.18	0.26	0.23	0.25	0.24	0.21	0.23
Time[3]	Mean	138	195	166	591	472	531	456	484	470	414	516	466	411	443	427
	SD	63	165	126	473	254	378	236	257	246	281	306	296	316	281	298
# of Tries	Mean	2.75	4.25	3.50	10.94	10.06	10.50	8.29	10.09	9.21	7.38	9.60	8.51	7.50	8.94	8.23
	SD	1.98	3.84	3.10	6.63	7.20	6.82	6.40	7.50	6.99	7.07	7.55	7.34	6.56	7.20	6.91
Confidence[4]	Mean	4.50	4.44	4.47	3.81	3.75	3.78	3.61	3.47	3.54	2.72	2.53	2.63	3.52	3.38	3.45
	SD	0.52	0.73	0.62	1.47	1.61	1.15	1.48	1.52	1.49	1.65	1.46	1.54	1.53	1.55	1.54

[1]: Manager-English Wording (High Ambiguity); [2]: Pseudo-SQL Wording (Low Ambiguity); [3]: In Seconds; [4]: Scale of 1 (Not Very Confident) to 5 (Very Confident).

Figure 8. Descriptive Statistics for Dependent Variables by Query Difficulty and Task Wording

5. RESULTS

We tested the overall effects of task difficulty and task wording using multivariate analysis of covariance (MANCOVA), as shown in Figure 9. The model included the four indicators of query performance – query quality, total time spent on the task, number of query attempts, and confidence in the query quality. We controlled for differences among our study participants by introducing their data model comprehension scores, and comfort levels with SQL and ERDs as covariates in the model. The results indicate that both query difficulty (F: 9.99; $p < 0.000$) and request ambiguity (F: 2.39; $p < 0.053$) had significant effects on query performance, which is consistent with prior research and supports our first two hypotheses.

The univariate tests and the post-hoc pair wise comparisons are summarized in 10. The results reveal that query difficulty had a significant effect on all four performance indicators, while ambiguity had a significant effect on query quality alone (F: 5.22; $p < 0.024$).

Effect	Value	F	Sig.
Task Difficulty	0.56	9.99	0.000
Task Ambiguity	0.05	2.39	0.053
Difficulty * Ambiguity	0.04	0.65	0.803
Data Model Comp.[1]	0.01	0.46	0.763
ERD Comfort[1]	0.07	3.04	0.019
SQL Comfort[1]	0.26	14.75	0.000

[1] Covariates

Figure 9. MANCOVA Multivariate Test Results.[1]

An examination of the covariates shows that SQL comfort level had a significant effect on the four facets of query performance, ERD comfort level was related solely to query formulation time, and data model comprehension did

not have any relevant impact. Participants who were more comfortable with SQL wrote better queries, in less time, with fewer attempts, and were more confident in their queries. The negative relationship between ERD comfort level and query time can be expected since participants had to comprehend the logical structure of the database through its ER representation. Although lower ERD proficiency may have increased the time to complete the query tasks, it did not have a negative impact on other performance indicators. A possible explanation for the lack of significant differences by data model comprehension could be the lack of variance in this measure. The median score on the four questions used to assess this control variable was 3 (out of 4), indicating that most participants had a reasonably good grasp of the data model.

Although our data analysis supported the hypothesized main effects, it did not provide evidence for the expected interaction-effects between task complexity and task wording. Allen and Parsons (2010) argue that rather than using a query quality score which is a relative indicator of performance, it may be more appropriate to assess query performance in absolute terms or as a dichotomy. Following their suggestion, we coded query performance as a binary outcome – a final query was either *correct* (it produced the correct results) or *incorrect* (the query either didn't execute or generated incorrect results). We then conducted an analysis of covariance (ANCOVA) with this alternate query performance measure as the dependent variable. The ANCOVA results, shown in Figure 11, not only corroborate the MANCOVA findings but also expose the interaction effect between the two independent variables (F: 3.65; P < 0.014).

Source	Dependent Variable	Mean Square	F	Sig.
Corrected Model	Quality	0.39	11.84	0.000
	Time	404109.46	5.68	0.000
	# Of Tries	134.70	3.15	0.001
	Confidence	15.42	9.57	0.000
Task Difficulty	Quality	0.96	29.53	0.000
	Time	917436.08	12.90	0.000
	# Of Tries	316.95	7.41	0.000
	Confidence	27.65	17.17	0.000
Task Ambiguity	Quality	0.17	5.22	0.024
	Time	12104.30	0.17	0.680
	# Of Tries	55.23	1.29	0.258
	Confidence	0.46	0.29	0.593
Difficulty * Ambiguity	Quality	0.01	0.31	0.822
	Time	64939.83	0.91	0.436
	# Of Tries	11.19	0.26	0.853
	Confidence	0.78	0.48	0.695
Data Model Comprehension[1]	Quality	0.01	0.18	0.672
	Time	710.04	0.01	0.921
	# Of Tries	13.15	0.31	0.580
	Confidence	0.50	0.31	0.580
ERD Comfort[1]	Quality	0.00	0.13	0.909
	Time	592570.71	8.33	0.004
	# Of Tries	59.55	1.39	0.240
	Confidence	2.52	1.57	0.213
SQL Comfort[1]	Quality	0.82	25.30	0.000
	Time	410774.88	5.78	0.017
	# Of Tries	191.70	4.48	0.036
	Confidence	67.84	42.12	0.000

[1] : Covariates

Figure 10. MANCOVA Univariate Test Results

Source	Mean Square	F	Sig.
Task Difficulty	6.03	45.56	0.000
Task Ambiguity	1.05	7.96	0.005
Difficulty * Ambiguity	0.48	3.65	0.014
Data Model Comp.[1]	0.03	0.23	0.635
ERD Comfort[1]	0.00	0.03	0.864
SQL Comfort[1]	2.05	15.51	0.000

[1]: Covariates

Figure 11. ANCOVA Results for Query Correctness (measured as a dichotomous variable)

Figure 12 shows the difference in quality for manager- versus pseudo-SQL wording as tasks increase in difficulty. Specifically, the effect of task wording on query correctness was contingent on the difficulty of the task. For the easier tasks, wording did not matter. However, as tasks became more difficult, the pseudo-SQL wording was helpful in formulating correct queries. This finding lends support for our third hypothesis (H3).

It is important to understand how well users are able to assess the correctness of their queries because it has a direct bearing on whether their reliance on query results for decision-making is justified (Allen and March 2006; Allen and Parsons 2010). We conducted two final ANCOVAs to examine the influence of task complexity and ambiguity on mean probability score and judgement bias score, respectively. Mean probability score reflects the accuracy of participants' confidence in their queries, while judgement bias score indicates the extent to which participants are under- or over-confident about their queries. Results from these ANCOVAs are shown in Figures 13 and 14.

The ANCOVA results show significant differences in mean probability score by task complexity (F: 5.73; $p < 0.001$), suggesting that participants' ability to assess the correctness of their queries diminished as the query tasks increased in complexity, which is consistent with prior research. The judgment bias score offers additional insight into participants' assessments of their query quality, by determining whether their assessments are under- or overconfident (Allen and Parsons, 2010). The results indicate that judgment bias score differed by both task complexity and task ambiguity. Specifically, our subjects tended to be over-confident in their assessment of their queries' correctness for more complex (F: 13.19; $p < 0.000$) and more ambiguous (F: 9.16; $p < 0.003$) tasks. In addition to the main effects, the interaction effect between the two independent factors was also significant (F: 3.60; $p < 0.015$), indicating that the difference in judgment bias scores between the managerial and pseudo-SQL group was contingent on task complexity, which lends support for our third hypothesis (H3).

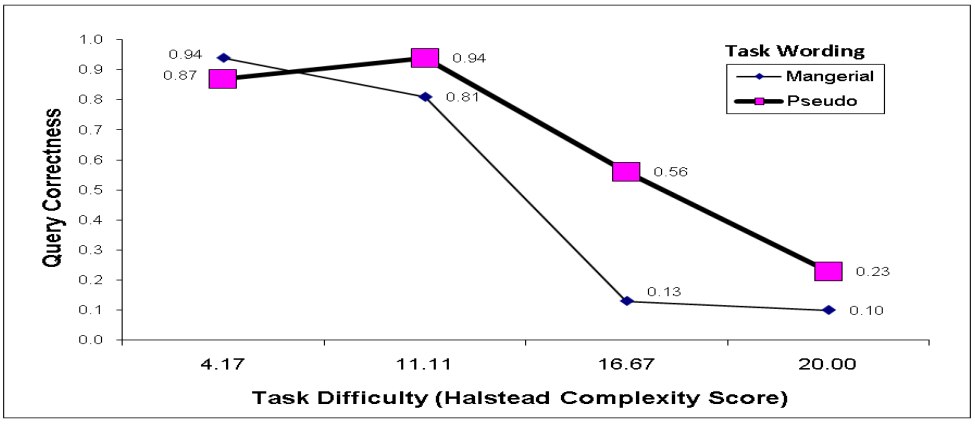


Figure 12. Interaction Effect between Task Difficulty and Task Wording

Source	Mean Square	F	Sig.
Task Difficulty	0.57	5.73	0.001
Task Ambiguity	0.32	3.22	0.074
Difficulty * Ambiguity	0.20	2.00	0.117
Data Model Comp.[1]	0.02	0.16	0.694
ERD Comfort[1]	0.05	0.50	0.481
SQL Comfort[1]	0.29	2.87	0.092

[1]: Covariates

Figure 13. ANCOVA Results for Mean Probability Score

Source	Mean Square	F	Sig.
Task Difficulty	2.06	13.19	0.000
Task Ambiguity	1.43	9.16	0.003
Difficulty * Ambiguity	0.56	3.60	0.015
Data Model Comp.[1]	0.00	0.00	0.994
ERD Comfort[1]	0.21	1.35	0.247
SQL Comfort[1]	0.39	2.51	0.115

[1]: Covariates

Figure 14. ANCOVA Results for Judgement Bias Score

6. DISCUSSION, LIMITATIONS, AND FUTURE RESEARCH

The main contribution of our study is the examination of the interaction between task complexity and information request ambiguity on query writing performance. For queries that involved simple SELECT-FROM-WHERE clauses, it did not matter whether the request was presented with high or low ambiguity—students generally wrote correct queries and were justified in their high levels of confidence. However, when query difficulty increased with longer WHERE clauses and the addition of GROUP BY and HAVING clauses, students' performance decreased, and decreased more when the request was written in manager-English (high ambiguity) than when it was written in pseudo-SQL (low ambiguity). In addition, while students were less confident in their query accuracy as tasks became more complex, they were still overly-confident, and this over-confidence was more

pronounced with manager-English than with pseudo-SQL wording.

The models of query formulation put forth by Reisner (1981) and Borthick et al. (2001) provide a context for interpreting these results and suggesting future teaching and research directions. Query formulation involves: (1) generating the correct query structure (i.e.,) the correct SELECT statement template), and (2) mapping elements of the information request into database elements (i.e., tables, columns, values) to insert into the correct "slots" of the query template. We refer to the latter as lexical transformations, and pseudo-SQL requests simplify these transformations by clarifying which columns and tables are needed in the query (e.g., the phrase "online order flag" corresponds to the OnlineOrderFlag column). We expect students with manager-English requests to exert more mental effort on lexical transformations than the students with pseudo-SQL, but as long as the structure of the query is simple—as with our first experimental task—the extra effort for the manager-English requests was manageable and did not affect query performance. Thus, for simple queries, we recommend using manager-English wording for instructional purposes, since this increases realism without decreasing performance.

However, as the complexity of the query *structure* increased (e.g., adding a GROUP BY clause), students needed to exert significant mental effort on generating the correct query template, regardless of how the query was worded. In this situation, the additional effort needed by students with the manager-English wording was significant and their performance suffered more than their pseudo-SQL counterparts. One way to help students learn to write more difficult queries in a classroom setting may be to reduce the lexical transformation complexity, thru the use of pseudo-SQL task wording, and focus first on the structural complexity. As the students build confidence and skill with generating the correct query structure, we can introduce more ambiguity into the wording of the tasks.

To help with the structural complexity of query formulation, instructors might use a query template that prompts students to think about whether and why each clause in a SELECT statement is needed. For example, we now use the template in Figure 15 during class discussions and encourage students to reference it when solving homework problems. We believe this may help them better understand the purpose and function of each clause and how certain clauses work together, which in turn may reduce the problems with respect to GROUP BY and WHERE versus HAVING clauses that we observed in our more complex experimental tasks.

We also use this template in class to help with lexical transformations, by analyzing the words in the information request to determine which columns and tables from the data model need to be included and in which clause(s). For example, to decide what to specify in a WHERE clause, we ask, "*Which* orders does the user want to see?" The students respond that it is *online* orders only, and then we reference the data model and data dictionary to determine what columns and values will indicate online orders. Essentially, we use the template to help students create a pseudo-SQL plan for the query. With the plan in place, students can then focus on writing the specific SQL syntax, and in this way, better manage the mental effort required of complex tasks.

Task ambiguity and query complexity affected *actual* task performance, as well as students' *confidence* in their task performance.

SELECT	Which columns/expressions should be in the result set?
FROM	Which tables/views provide the source data for this query? What join conditions are needed? INNER or OUTER join?
WHERE	Which rows should be included in the result set (i.e., what criteria should be used to filter rows)?
GROUP BY	How should rows be grouped or aggregated (often so that an aggregate function can be applied to each group)?
HAVING	Which groups (as specified in the GROUP BY clause) should be included in the result set (i.e., what criteria should be used to filter groups)?
ORDER BY	By which column(s) should the resulting rows be sorted? In ascending or descending (DESC) order?

Figure 15. Query Template

Students' confidence decreased as task complexity increased, but did not decrease as much as actual performance did, meaning that students were overly-confident when their query solutions were inaccurate. Two suggestions to help students evaluate their own queries are: (1) to practice interpreting common SQL error messages and modifying query attempts in response, and (2) to teach strategies for confirming query results (e.g., through control checks). The former suggestion addresses queries that do not execute, and may help students distinguish between simple syntactic errors (e.g., a missing apostrophe or a misspelled column name) and major logic errors (e.g., a missing clause). The latter suggestion addresses queries that execute but return incorrect results, and may help students validate the results and thus bring their confidence closer to their actual performance.

Proficiency in SQL is recognized as a critical and marketable skill for students majoring in information systems. But helping students learn to write complex queries is a challenge. This study examines two factors that make query formulation difficult and proposes teaching techniques that may help students recognize, manage, and reduce the difficulties. Future studies should evaluate the effectiveness of these techniques and identify other ways to facilitate students' acquisition of this important skill.

7. REFERENCES

Allen, G.N. and March, S.T. (2006), "The Effects of State-Based and Event-Based Data Representation on User Performance in Query Formulation Tasks," *Management Information Systems Quarterly*, 30(2), pp. 269-290.

Allen, G.N. and Parsons, J. (2010), "Is Query Reuse Potentially Harmful? Anchoring and Adjustment in

Adapting Existing Database Queries,"*Information Systems Research*, 21(1), pp. 56-77.

Ashkanasy, N., Bowen, P.L., Rohde, F.H., and 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), pp. 53-82.

Borthick, A.F., Bowen, P.L., Jones, D.R., and Tse, M.H.K. (2001) "The effects of information request ambiguity and construct incongruence on query development," *Decision Support Systems*, 32, pp. 3-25.

Bowen, P.L., O'Farrell, R.A., and 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 the *International Conference on Information Systems (ICIS) 2004 Proceedings*, pp. 77-90.

Bowen, P.L., O'Farrell, R.A., and Rohde, F.H. (2006) "Analysis of competing data structures: Does ontological clarity produce better end user query performance," *Journal of the AIS*, 7(8), pp. 514-544.

Bowen, P.L., O'Farrell, R.A., and Rohde, F.H. (2009) "An Empirical Investigation of End-User Query Development: The Effects of Improved Model Expressiveness vs. Complexity," *Information Systems Research*, 20(4), pp. 565-584.

Campbell, D.J. (1988) "Task complexity and strategy development: A review and conceptual analysis," *Academy of Management Review*, 13, pp. 126-139.

Chan, H.C. (1999) "The relationship between user query accuracy and line of code," *International Journal of Human-Computer Studies*, 51, pp. 851-864.

Chan, H.C., Teo, H.H., and Zeng, X.H. (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(8), pp. 843-853.

Halstead, M. H. (1977) *Elements of Software Science*. Elsevier, Amsterdam.

Reisner, P. (1977) "Use of psychological experimentation as an aid to development of a query language," *IEEE Transactions on Software Engineering*, SE-3 (3), pp. 218-229.

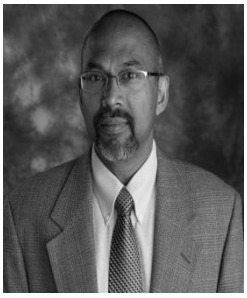
Reisner, P. (1981) "Human factors studies of database query languages: A survey and assessment," *Computing Surveys*, 13(1), pp. 13-31.

Rho, S. and March, S.T. (1997) "An analysis of semantic overload in database access systems using multi-table query formulation," *Journal of Database Management*, 8(2), pp. 3-14.

Yates, J. (1990). Judgment and Decision Making. Prentice Hall, Englewood Cliffs, NJ.

AUTHOR BIOGRAPHIES

Gretchen Irwin Casterella is an associate professor in the Computer Information Systems Department at Colorado State University. Gretchen holds a PhD and a Masters of Science in Information Systems from the University of Colorado. Gretchen's research interests are in systems development, specifically in understanding how individuals learn and master tools, technologies, and approaches for systems analysis and design. Gretchen's research has appeared in the *Communications of the ACM*, the *Journal of MIS*, the *Journal of the AIS*, *IEEE Transactions on Professional Communication*, and *Human-Computer Interaction*.



Leo R. Vijayasarathy is Associate Professor of Computer Information Systems in the College of Business at Colorado State University. He earned an MBA from Marquette University and his Ph.D. from Florida International University. His research on the development, use and consequences of information systems has been published in *Electronic Markets*, *European Journal of Information Systems*, *IEEE Transactions on Professional Communications*, *Information & Management*, *Internet Research*, *International Journal of Production Economics*, and the *Journal of Management Information Systems*. He serves on the editorial advisory board of *Internet Research*

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096