ELSEVIER

Decision Support Systems 32 (2001) 3–25

Decision Support Systems

www.elsevier.com/locate/dsw

The effects of information request ambiguity and construct incongruence on query development

A. Faye Borthick [a,*], Paul L. Bowen [b,1], Donald R. Jones [a,2], Michael Hung Kam Tse [c,3]

[a] *School of Accountancy, Georgia State University, POB 4050, Atlanta GA 30302-4050, USA*
[b] *Department of Commerce, The University of Queensland, Brisbane, Queensland, 4072, Australia*
[c] *Corporate Finance, JP Morgan, Jardine House, 1 Connaught Place, Central, Hong Kong, China*

Abstract

This paper examines the effects of information request ambiguity and construct incongruence on end user's ability to develop SQL queries with an interactive relational database query language. In this experiment, ambiguity in information requests adversely affected accuracy and efficiency. Incongruities among the information request, the query syntax, and the data representation adversely affected accuracy, efficiency, and confidence.

The results for ambiguity suggest that organizations might elicit better query development if end users were sensitized to the nature of ambiguities that could arise in their business contexts. End users could translate natural language queries into pseudo-SQL that could be examined for precision before the queries were developed. The results for incongruence suggest that better query development might ensue if semantic distances could be reduced by giving users data representations and database views that maximize construct congruence for the kinds of queries in typical domains.

Keywords: Query development; Requirements ambiguity; Construct congruence; Web front end

1. Introduction

Even before Web front ends to legacy data on mainframes were created, managers and staff members were experiencing the need to retrieve and analyze data from various sources. They could not rely on IS professionals for ad hoc data retrieval and analysis because of the demands on IS professionals' time for developing organizational systems. With Web front ends, however, data can be accessible to anyone with a Web browser [9,16]. In some organizations, young staff members, writing their own queries, are driving the business with their analyses [10]. Thus, in self-defense, managers and staff members are discovering that their competitiveness depends on their ability to develop database queries.

Web front ends may have made more data accessible and the actual querying easier, but perennial

* Corresponding author. Tel.: +1-404-651-4472; fax: +1-404-651-1033.
E-mail addresses: borthick@gsu.edu, bowen@lorien.commerce.uq.edu.au (P.L. Bowen), donjones@gsu.edu (D.R. Jones), michael.hk.tse@jfleming.com (M. Hung Kam Tse).
URL: http://www.gsu.edu/~accafb/borthick.htm.
[1] Tel.: +61-7-3365-6584; fax: +61-7-3365-6788.
[2] Tel.: +1-404-651-0963; fax: +1-404-651-1033.
[3] Tel.: +852-2978-7613; fax: +852-2810-6558.

0167-9236/01/$ - see front matter
PII: S0167-9236(01)00097-5

problems with database querying remain—the difficulties associated with understanding data structures from textual or graphical representations and mapping the meaning of a query into the interface language [4,5,18,30,32]. This research investigates the query development difficulties associated with varying semantic distance, i.e., the distance between the information users want and the expression in the query interface language that will produce that information [17]. The distance is manipulated in two ways: through the level of ambiguity of information requests and through the extent of congruence between constructs of the data representation and the interface language.

Potential benefits of this research include improved communication, e.g., between management and knowledge workers, and improved query support tools. Communication improvements might result from clearer statements of information requirements, more informed analysis of information requests, and greater use of divide and conquer strategies to ensure that information provided matches the information desired. Improved query support tools might entail enhancements to query front ends, e.g., to highlight potential ambiguities, and the creation of graphical data models and database views that facilitate typical queries by specific groups of end users.

This study extends prior research on end user query performance [4,5,18,30,32]. Prior research typically compared different query languages or different forms of data representations. This research builds and tests a theory to explain why ambiguity and incongruence adversely affect end user query performance. Methodological improvements over prior studies include a more realistic business setting, direct interaction between experimental participants and the computerized information system, complete interactive capture of this interaction, and a detailed analysis of the errors made by participants.

2. Ambiguity and incongruence as impediments to query development

In general, because humans are limited information processors, more effective problem solving results when less cognitive effort is required [22]. In Norman's [23,24] model of user query performance, users exert cognitive effort to bridge the semantic distance between their query objectives and the way they must specify these objectives to the information system. Thus, the cognitive effort required to formulate successful queries increases with increasing semantic distance [17]. Consistent with this model, empirical evidence indicates that information requests with shorter semantic distances require less cognitive effort and lead to better performance in terms of query correctness and query development time [2,30,32]. Two aspects that could affect semantic distance are the ambiguity of the information request and the congruence of the information request with the syntax of the query language and the data representation.

2.1. Model of the query process

Formulating a query requires transforming an information request into query components [19]. It requires knowledge in three domains: knowledge of the information needed, knowledge of the database structure, and knowledge of the query language [20]. Ineptness in one or more of these domains generates user errors [25] that produce erroneous information and lead to inappropriate decisions [8,27].

Fig. 1 illustrates a conceptualization of how end users formulate queries. First, users identify required constructs by filtering out unnecessary components from and adding missing or implied components to the statement of the information request. Second, they examine the data representation to identify the

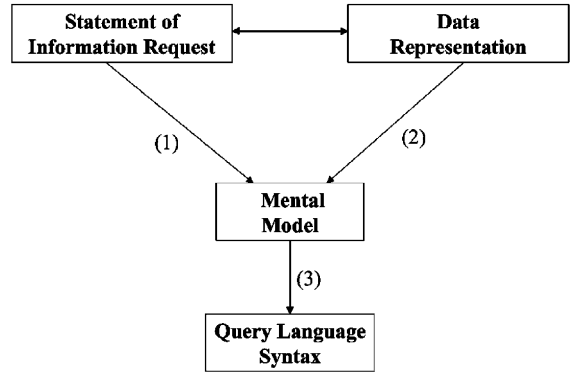

Fig. 1. Model of end users' query formulation processes.

tables and attributes needed to obtain the required constructs. These two processes result in a mental model of the information request in terms of the available data. Third, end users translate their mental models into the query language syntax required to satisfy their mental model of the information requirements.

Difficulties and errors result from discrepancies between the user's mental model and the other representations, i.e., the information request, the data representation, and the query language syntax. These discrepancies can be conceptualized as semantic distances. The semantic distance between query objectives and correct queries has two aspects: the information requirement distance and the data representation distance. The information requirement distance denotes the gap between the statement of the information request and the operations (operators) available in the query language (path (1) and (3) in Fig. 1). The data representation distance signifies the gap between the data representation and the constructs (operands) required to formulate the appropriate query (path (2) and (3) in Fig. 1). Greater cognitive effort is required when either semantic distance increases, and users may or may not be aware of mismatches between the information request and the query syntax and between the information request and the data representation.

2.2. Ambiguity of information requests

One problem in transforming a natural language request to an appropriate statement in the query language, i.e., of traversing the information requirement distance, is resolving the ambiguity of information requests [1,7,26,31]. Information requests that are ambiguous cause end users to be uncertain about the intent of the information request. This uncertainty, i.e., multiple possible desired outcomes of the information request, will lead to one-to-many mappings from words in the natural language to the syntax (operators) in the query language. That is, a natural language information request may have multiple interpretations such that several query statements may appear to be possible solutions. The existence of multiple possibilities is inherent in organizational situations, especially given that requests can come from external and internal stakeholders such as customers, suppliers, regulators, co-workers, and managers.

End users can reduce ambiguity of information requests by translating them into a less ambiguous form, i.e., performing a stepwise refinement of the information request. Because of the smaller semantic distances [23,24], end users are likely to find it easier to resolve ambiguity in their own language than in a computer language. Consider, for example, the natural language information request for a transportation information system:

> Manager-English: Management wants to know the routes that each truck is not permitted to travel.

or the same request in language that is closer to the syntax of the query language:

> Pseudo-SQL: List all truck numbers and, where applicable, the route numbers where the truck violates height or weight constraints.

The manager-English version is the more ambiguous of the two statements because it could be mapped to several different pseudo-SQL statements [1,7, 26,31]. To make the lexical transformations from manager-English to pseudo-SQL [28], users would need to understand the query in a business context in which trucks are not permitted to travel on routes for which they violate height or weight constraints. The existence of multiple interpretations increases the information load [3], increasing the cognitive effort users must expend to develop a correct query. For this query, the manager-English request does not explicitly state that determining acceptable routes involves height or weight constraints or both. In this example, users of the manager-English request are more likely to make query errors of omission than users of the pseudo-SQL request.

Because the manager-English request acknowledges the existence of one or more categories of permissions that are required for a truck to travel a route, users must search their memory or consult some other source, e.g., a route table, to determine what the applicable constraints are. Then, if they want to compare their query with the manager-English request, users may create the list of constraints again. These search and compare processes take time, which would decrease a user's efficiency compared to formulating the query for the pseudo-SQL version of the information request.

Because query accuracy, efficiency, and confidence are often related, it is common to assess them jointly [33]. Users working with the more ambiguous manager-English request may identify multiple interpretations of the information requested and of the applicable constraints. Hence, they are likely to be less confident in their queries than users of the pseudo-SQL request. The combined correctness, efficiency, and confidence effects are the first hypothesis:

H1. User query performance (correctness, efficiency, and confidence) will be inversely related to the ambiguity of information requests.

2.3. Incongruence

A second major problem in creating a query that correctly satisfies the information request is identifying the correct data elements, the ways they need to be related to each other, and the restrictions that need to be imposed on them. Successfully negotiating this data representation distance requires correctly mapping between real world constructs and their representation in the information system. The greater the mismatch between the real world constructs and their representation in the information system, the greater will be the cognitive effort that users will have to exert to construct a correct query and the more likely it will be that the queries contain errors.

For a specific query, if the real world constructs perfectly match their representation in the information system, there would be a one-to-one match between the two, i.e., the real world constructs and the information system would be *construct congruent*. Greater construct incongruence implies degraded Task-Technology-Fit [11], which is associated with worse performance [12,34]. As construct incongruence increases, the complexity of a query also increases, which increases the time and cognitive effort required to develop a correct query.

With respect to queries, construct incongruence occurs when the representations in the information system must be reconstituted in some way before they can be incorporated into the query clauses that actually produce the desired result. For example, queries whose successful construction requires users to create temporary views, develop new relationships, or perform outer joins exhibit construct incongruence.

Database administrators typically strive to create data structures that minimize incongruence between the real world constructs and the database by making the data structures match the schemas of the predominate uses of the data. They are, however, constrained to only one physical representation of the database. Thus, construct incongruence may arise in connection with uses of the database that are infrequent or were not anticipated by the database administrator. Because the conditions that prompt new uses of existing data evolve over time in organizations, it is not realistic for the database administrator to be expected to anticipate all possible uses of the database.

Users confronted with construct incongruity are likely to detect cues that alert them to inadequately developed queries. Each time users compare the information request with a newly developed query is an occasion on which they could detect mismatches between the information request and the query syntax and between the information request and the data representation. To the extent this comparison process identifies query errors, users get confirmation that they have not developed a usable query, which ought to undermine their confidence in the correctness of their queries. Thus, user's confidence in their queries should be inversely related to the degree of construct incongruence.

For example, consider two information requests. First, compare:

> Pseudo-SQL: List the driver's name and the trip number for trips where drivers were assigned trucks that violate the height constraint for the prescribed route.

with its corresponding query (see the data structure in Fig. 2),

```
select surname, firstname, trip.trip_no
from driver, trip_driver, trip, route, truck
where driver.license_no = trip_driver.license_no and
        trip_driver.trip_no = trip.trip_no and
    trip.route_no = route.route_no and
    trip.truck_no = truck.truck_no and
    height > height_constraint;
```

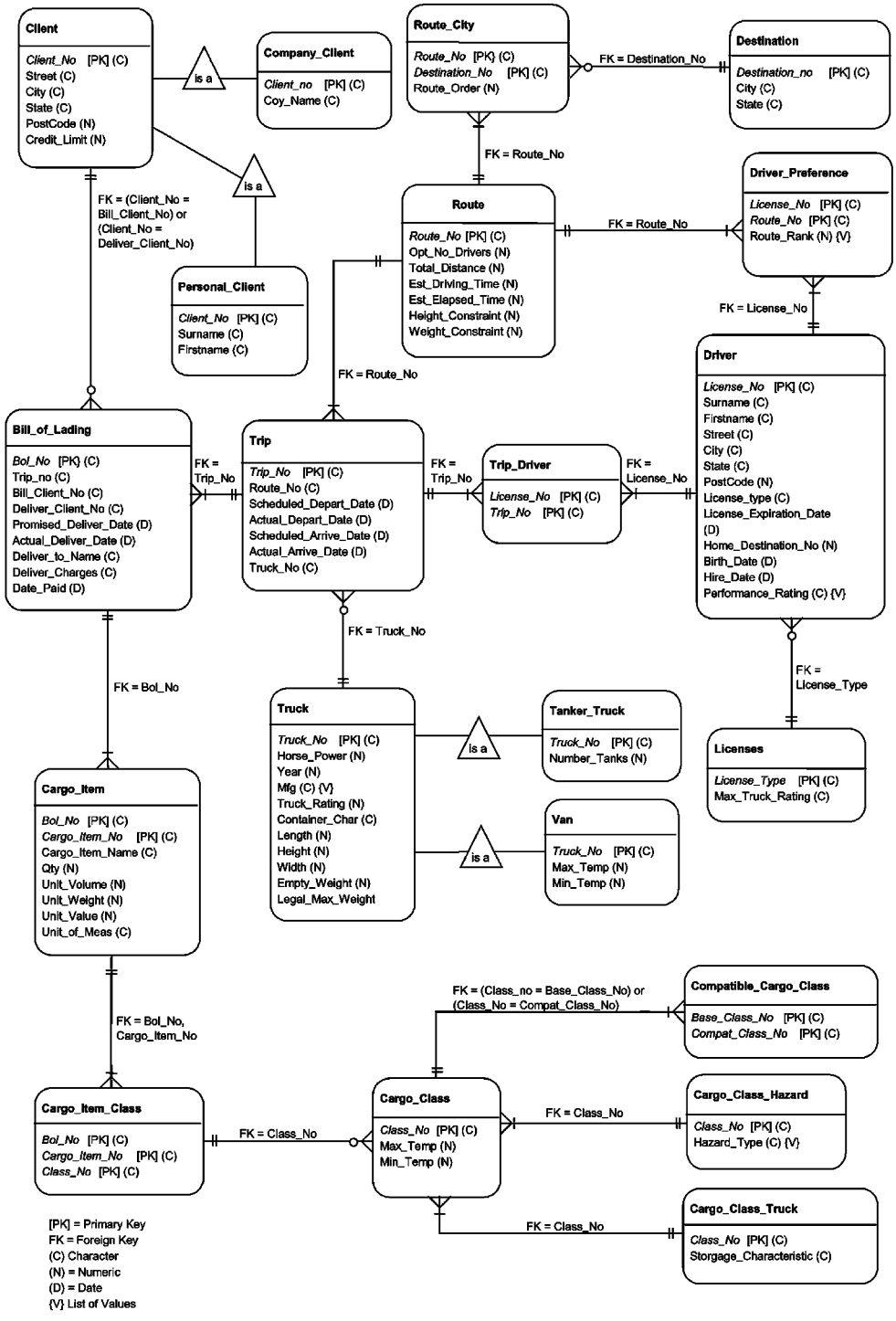

Fig. 2. Entity-relationship diagram.

Second, compare:

Pseudo-SQL: List all truck numbers and, where applicable, the route numbers where the truck violates height constraints.

with its corresponding query (see the data structure in Fig. 2),

```
create view truckviolation as
select truck_no, route_no
from route, truck
where height > height_constraint;

select truck.truck_no, route_no
from truck, truckviolation
where truck.truck_no = truckviolation.truck_no (+);
```

For the first information request, a relatively good task-technology fit exists between the information request, the entity-relationship diagram (ERD), and the required query. For example, the first join of the query involves the tables *driver* and *trip_driver*. On the ERD, these tables are adjacent to each other and the foreign key necessary for a natural join of the two tables, *license_no*, is readily apparent. Hence, end users can relatively easily derive the required join, *driver.license_no = trip_driver.license_no*.

Conversely, the second information request does not exhibit a good task-technology fit between the information request, the ERD, and the required query. The query needs a new view, a Cartesian product, and an outer join, none of which are obvious from the ERD. End users must recognize that a view will facilitate obtaining the information requested. They must then determine what tables and attributes are required to construct the view *truckviolation* including the foreign key necessary to join the view with the *truck* table. This view is atypical in that it does not contain a join restriction but retains the full Cartesian product of the *route* and *truck* tables. Furthermore, to ensure that the query reports all trucks, a left outer join is required between *truck* and *truckviolation*. Hence, relative to the first information request, the second information request exhibits greater construct incongruence, which will require more cognitive effort to represent in the query, which is likely to lead to poorer end user query performance.

The combined effects of accuracy, efficiency, and confidence related to construct congruence comprise the second hypothesis:

H2. User query performance (accuracy, efficiency, and confidence) will be inversely related to construct incongruence.

2.4. Query complexity

The effects of task complexity have been studied extensively [3,35]. More complex tasks, e.g., more complex queries, place greater cognitive demands on persons undertaking the tasks and reduce their performance [3]. For computing tasks including queries, complexity is often measured using Halstead's [15] difficulty measure [18]. In this research, the measure is a function of the number of mental discriminations required to write a query. To confirm the expected relationship between performance as manifested by

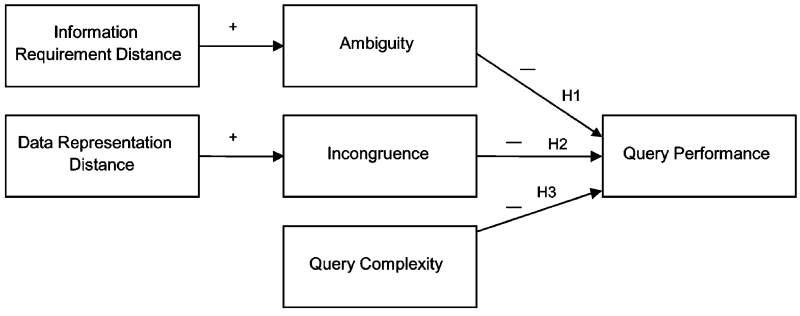

Fig. 3. Query performance model.

accuracy, efficiency, and confidence, the third hypothesis is:

H3. User query performance (accuracy, efficiency, and confidence) will be inversely related to query complexity.

Fig. 3 summarizes the hypothesized relationships of ambiguity, incongruence, and query complexity with performance.

3. Method

3.1. Design

The hypotheses were tested in a two-factor within-subjects laboratory experiment in which participants composed and executed queries in Oracle SQL. The ambiguity factor had two levels: information requests posed in pseudo-SQL (low ambiguity) or manager-English (high ambiguity). The congruence factor had two levels: information requests that were construct congruent or construct incongruent. The traditional approach to testing query performance has been with pencil and paper (with or without an intermediary to return results) or simulated systems that did not reveal results to participants [4,5,13,14,18,21,30,32]. As in Ref. [6], the testing approach in this experiment incorporates a higher level of realism in that participants had the opportunity to work on their queries until they were satisfied with the query results from the database system.

3.2. Participants

Participants were 23 graduate business students enrolled in an information systems course. Five percent of their course grade was based on performance in the experiment. Before the experiment, participants received training in interpreting ERDs and formulating SQL queries. Because they were intelligent, had computing experience, and had training in query development, the participants were appropriate surrogates for casual users that are beginning to develop their own database queries. Table 1 summarizes demographic data for the two groups.

Table 1
Demographic data

Variable	Group 1	Group 2
Gender		
Female	3	5
Male	9	6
Degree		
Graduate information systems	11	11
Graduate business administration	1	0
GPA (7-point scale)		
Mean	5.33	5.20
Standard deviation	1.0112	0.9970

3.3. Procedure and variables

To ensure the equivalence of the two groups needed for the research design [29], participants were divided into two equivalent groups based on a ranking of their information systems competence, which was determined by examining their information systems experience, education, and GPA. An information systems expert ranked the participants. The participant considered to have the most competence in information systems was ranked number 23, the person with the next most competence was ranked 22, etc. The participant with the highest ranking was assigned to group A, the participant with the next highest ranking, to group B, followed by B, A, A, B, B, etc. The groups were then randomly assigned to the two treatments.

At the beginning of the 2-h experimental session, participants received instructions, an ERD[4] representing the database (Fig. 2), and 16 information requests (Appendix A shows the 10 information requests that participants completed, the responses to which are the basis for the analysis here) to satisfy

[4] This study incorporates only one kind of depiction of the entity relationships, unlike [20] in which the depiction varied. The use of the ERD for all conditions corresponds to high representation realism, which is associated with better query performance [4]. Because they were interacting with the database through SQL, a linear-keyword language, participants in this experiment were all actually interacting with the database at a low level of representation realism.

Table 2
Experimental results: summary by information request

Variable	Mean (standard deviation) by condition			
	Congruent/unambiguous	Congruent/ambiguous	Incongruent/unambiguous	Incongruent/ambiguous
Question 1	$N = 11$	$N = 12$	$N = 0$	$N = 0$
Complexity	5.7400 (0.0000)	5.7400 (0.0000)		
Micro	0.0000 (0.0000)	1.0000 (2.3355)		
Macro	0.0000 (0.0000)	0.16667 (0.3892)		
Time	5.4545 (3.4165)	6.5833 (3.5022)		
Attempts	1.3636 (0.6742)	2.1667 (1.7495)		
Confidence	6.7273 (0.4671)	6.5833 (0.6686)		
Question 2	$N = 0$	$N = 0$	$N = 12$	$N = 11$
Complexity			4.4700 (0.0000)	4.4700 (0.0000)
Micro			3.1667 (10.9695)	1.6364 (5.4272)
Macro			0.0833 (0.2887)	0.1818 (0.6030)
Time			6.5000 (7.5978)	7.0909 (7.0207)
Attempts			2.2500 (1.2881)	2.0909 (1.7003)
Confidence			6.5833 (0.6686)	6.0909 (1.2210)
Question 3	$N = 12$	$N = 11$	$N = 0$	$N = 0$
Complexity	10.0500 (0.0000)	10.0500 (0.0000)		
Micro	1.0000 (3.4641)	0.8182 (1.9400)		
Macro	0.0833 (0.2887)	0.1818 (0.4045)		
Time	8.9167 (3.7528)	7.5455 (2.2962)		
Attempts	2.2500 (1.2881)	2.1818 (1.4709)		
Confidence	6.1667 (1.1146)	6.3636 (1.2060)		
Question 4	$N = 0$	$N = 0$	$N = 11$	$N = 11$
Complexity			7.7400 (0.0000)	7.7400 (0.0000)
Micro			18.6364 (14.0519)	22.9091 (10.5305)
Macro			0.8182 (0.4045)	1.1818 (0.4045)
Time			13.9091 (9.4282)	11.0000 (5.4037)
Attempts			3.7273 (2.2843)	5.3636 (2.4606)
Confidence			5.1818 (1.6624)	4.8182 (2.5226)
Question 5	$N = 10$	$N = 11$	$N = 0$	$N = 0$
Complexity	18.5600 (0.0000)	18.5600 (0.0000)		
Micro	2.7000 (4.3218)	11.3636 (9.7188)		
Macro	0.5000 (0.7071)	1.5455 (0.9342)		
Time	11.8000 (6.9570)	14.2727 (12.2400)		
Attempts	4.4000 (2.4585)	7.1818 (4.5347)		
Confidence	5.2000 (1.7512)	5.0909 (1.5136)		
Question 6	$N = 0$	$N = 0$	$N = 9$	$N = 11$
Complexity			14.0000 (0.0000)	14.0000 (0.0000)
Micro			32.0000 (15.2561)	30.1818 (16.3757)
Macro			1.1111 (0.3333)	1.0909 (0.5394)
Time			10.8889 (6.6039)	12.1818 (9.0423)
Attempts			4.2222 (2.6822)	5.6364 (4.8015)
Confidence			3.6667 (2.2913)	5.1818 (1.7787)
Question 7	$N = 8$	$N = 11$	$N = 0$	$N = 0$
Complexity	21.5100 (0.0000)	21.5100 (0.0000)		
Micro	1.0000 (2.8284)	0.7273 (2.4121)		
Macro	0.2500 (0.7071)	0.0909 (0.3015)		
Time	7.3750 (4.1726)	7.5455 (3.8565)		

Table 2 (*continued*)

Variable	Mean (standard deviation) by condition			
	Congruent/unambiguous	Congruent/ambiguous	Incongruent/unambiguous	Incongruent/ambiguous
Question 7	$N = 8$	$N = 11$	$N = 0$	$N = 0$
Attempts	3.8750 (3.8336)	3.8182 (2.9939)		
Confidence	5.5000 (1.6036)	5.7273 (1.4206)		
Question 8	$N = 0$	$N = 0$	$N = 11$	$N = 9$
Complexity			16.5200 (0.0000)	16.5200 (0.0000)
Micro			8.9091 (9.8230)	10.5556 (19.1645)
Macro			0.8182 (0.4045)	1.3333 (0.5000)
Time			7.4545 (3.9080)	9.4444 (3.5395)
Attempts			9.4444 (3.0000)	5.8889 (3.8550)
Confidence			4.0000 (3.0000)	5.8889 (3.8550)
Question 9	$N = 9$	$N = 7$	5.7273 (1.2721)	4.3333 (2.5981)
Complexity	30.3700 (0.0000)	30.3700 (0.0000)	$N = 0$	$N = 0$
Micro	6.8889 (9.5975)	27.5714 (9.9307)		
Macro	0.7778 (0.8333)	2.8571 (0.3780)		
Time	11.2222 (4.2947)	11.4286 (3.9521)		
Attempts	3.7778 (3.0732)	6.2857 (3.1472)		
Confidence	5.4444 (1.3333)	4.5714 (2.0702)		
Question 10	$N = 0$	$N = 0$	$N = 7$	$N = 8$
Complexity			22.5300 (0.0000)	22.5300 (0.0000)
Micro			34.1429 (17.4683)	40.8750 (23.2406)
Macro			1.1429 (0.3780)	(0.0000) (0.0000)
Time			13.7143 (6.7259)	10.3750 (3.5832)
Attempts			7.0000 (6.8069)	4.6250 (3.0208)
Confidence			4.2857 (2.5635)	5.1250 (1.6421)

with SQL queries. Odd-numbered requests had low construct congruence; even-numbered requests had high construct congruence. Each request had two versions: pseudo-SQL and manager-English. Each group received equal numbers of requests posed in each version, but for different requests, i.e., for each request, one group received the pseudo-SQL version and the other group, the manager-English version. After executing a query, participants were permitted to modify the query or move to the next query after

Table 3
Experimental results: summary by variable

Measurement	Mean (standard deviation) by condition			
	$N = 50$	$N = 52$	$N = 50$	$N = 50$
	Congruent/unambiguous	Congruent/ambiguous	Incongruent/unambiguous	Incongruent/ambiguous
Complexity	16.2950 (8.7427)	16.0152 (8.2891)	12.0842 (6.2541)	12.3446 (6.2797)
Micro	2.1800 (5.2980)	6.6731 (10.9754)	17.3600 (17.5415)	20.4800 (20.2154)
Macro	0.3000 (0.6145)	0.8077 (1.1209)	0.7400 (0.5272)	1.1000 (0.7354)
Time	8.9000 (5.0679)	9.2692 (6.8431)	10.1400 (7.5404)	10.0200 (6.3132)
Attempts	3.0200 (2.5674)	4.1346 (3.5260)	3.9800 (3.4905)	4.6800 (3.5135)
Confidence	5.8600 (1.3704)	5.7692 (1.5031)	5.2400 (1.9332)	5.1400 (2.0204)

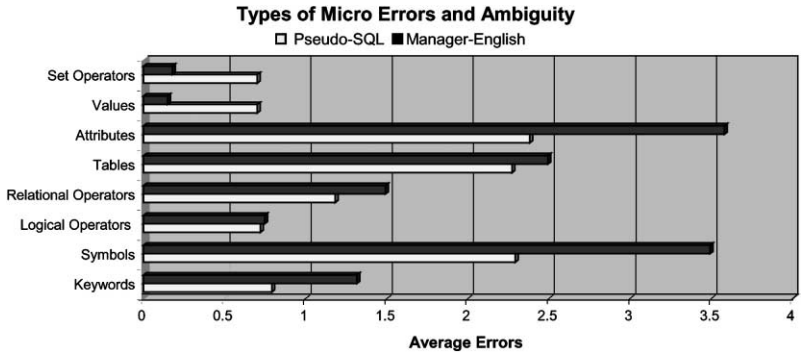

Fig. 4. Types of micro errors (average) and ambiguity.

indicating their level of confidence in the correctness of the query. All computer interactions and time stamps were recorded in log files.

Without knowledge of the identity of the participant or the version of the request the participant received, two researchers independently coded the accuracy of each query (based on the last query attempt) by reference to standard query solutions and resolved differences. Accuracy was assessed in two ways: in terms of the number of macro errors (errors involving row, column, and aggregation errors) and micro errors (the count of the minimum number of changes required to transform an actual query into a correct query). Appendix B contains an example of the accuracy coding for one information request. Efficiency was measured as the total time spent on a query and as the number of query attempts. Confidence was measured as participants' self-report after each query.

A complexity measure for each query was calculated as Halstead's [15] difficulty measure as illustrated in Appendix C, and the information request pairs were arranged so that the difficulty of the congruent request of each pair was always equal to or greater than the difficulty of the incongruent request. The query performance data were analyzed in separate analysis of covariance (ANCOVA) models for macro performance, micro performance, elapsed time, number of query attempts, and confidence. In each case, performance was analyzed as a function of ambiguity (coded 0-1) as a nested component by request, incongruence (coded 0-1), and complexity (covariate). Because of the 2-h time constraint, few participants attempted information re-

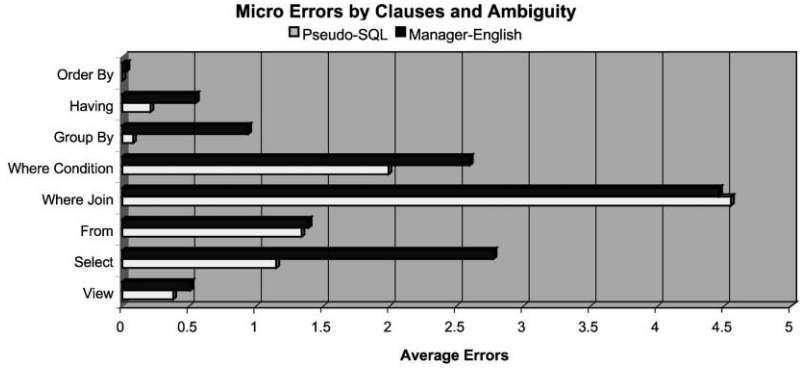

Fig. 5. Micro errors by clauses (average) and ambiguity.

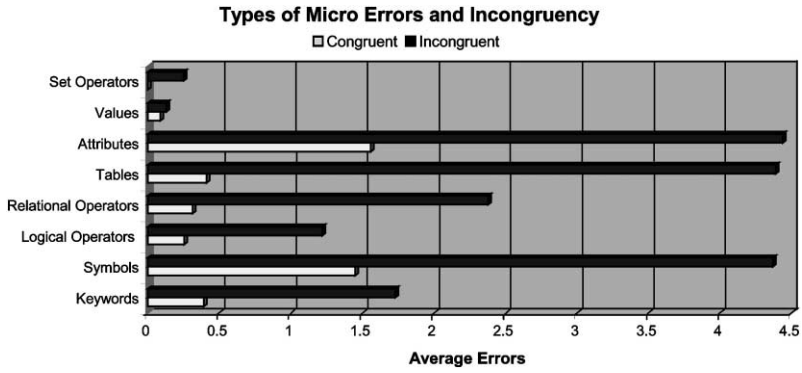

Fig. 6. Types of micro errors (average) and incongruency.

quests 11 to 16. All statistical analyses are based on the 202 responses to information requests one to ten. The information requests and model SQL queries are shown in Appendix A.

4. Results

4.1. Summary statistics

Table 2 shows experimental results summarized by information request, and Table 3 summarizes the results by experimental variable. Table 3 verifies that the average complexity of the congruent information requests is higher than that of the incongruent information requests, i.e., that the information requests are biased against finding the hypothesized inverse relationship between incongruence and performance. Average performance for accuracy (micro and macro errors) shows noticeable degradation when information requests are more ambiguous or incongruent. Average efficiency (time and attempts) and average confidence measures, however, show only slight degradation as ambiguity and incongruence increase.

Figs. 4 and 6 display the types of micro errors made by participants for the different levels of ambiguity and incongruence, respectively. Figs. 5 and 7 present micro errors by clauses for the different levels of ambiguity and incongruence.

Fig. 4 shows that, except for Values and Set Operators, information requests formulated in manager-English, on average, resulted in more micro errors for each category. The difference in accuracy between manager-English and pseudo-SQL is especially pronounced for Symbol and Attribute errors. Fig. 5 reveals that information requests formulated in manager-English, on average, resulted in more micro

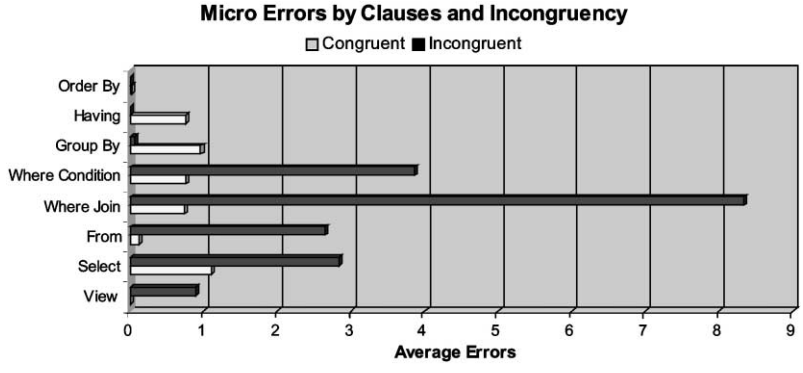

Fig. 7. Micro errors by clauses (average) and incongruency.

errors for seven of eight different classes of SQL clauses. The difference between manager-English and pseudo-SQL is most evident for Select errors. The higher average number of Attribute and Select errors for manager-English formulations indicate that participants experienced difficulty in identifying what information was required from information requests that were more ambiguous. That is, the experimental participants experienced more difficulty identifying the correct columns than recognizing the appropriate row restrictions.

Fig. 6 illustrates the effects of incongruence on types of micro errors. The negative effects of incongruence on participants' performance was especially pronounced for errors related to Attributes, Tables, Relational Operators, Symbols, and Keywords. Fig. 7 shows that queries with greater construct incongruence resulted in more errors for six of eight classes of SQL clauses. Performance differences were most evident for errors in the Where Condition, Where Join, From, and Select clauses.

4.2. Tests of hypotheses

Regression results appear in Table 4 and are summarized in Table 5. For H1, that accuracy, effi-

Table 4
ANCOVA results for performance components
Cells with '*' are ambiguity parameter estimates that are available from the authors by information request.

Source ($n = 202$)	*df*	Mean square	*F* value	*p* Value	Parameter estimate	Standard error of estimate	R^2
Model: accuracy micro errors	12	2232.0830	15.55	0.0001			0.4967
Error	167	143.5789					
Intercept				0.0210	−7.0402	3.0992	
Complexity	1	1812.7555	12.63	0.0005	0.5658	0.1592	
Incongruence	1	7151.2602	49.81	0.0001	17.5626	2.4885	
Ambiguity	10	874.6809	6.09	0.0001	*	*	
Model: accuracy macro errors	12	7.5652	29.60	0.0001			0.6527
Error	167	0.25562					
Intercept				0.0210	−0.3043	0.1308	
Complexity	1	7.7867	30.46	0.0001	0.0371	0.0067	
Incongruence	1	8.2400	32.24	0.0001	0.5962	0.1050	
Ambiguity	10	4.3835	17.15	0.0001	*	*	
Model: efficiency time	12	70.5968	1.76	0.0580			0.1004
Error	167	40.1697					
Intercept				0.0003	6.0487	1.6393	
Complexity	1	173.3585	4.32	0.0391	0.1750	0.0842	
Incongruence	1	90.6014	2.26	0.1348	1.9768	1.3163	
Ambiguity	10	46.0172	1.15	0.3305	*	*	
Model: efficiency number of attempts	12	42.7901	4.71	0.0001			0.2302
Error	167	9.0853					
Intercept				0.3762	0.6915	0.7796	
Complexity	1	115.6110	12.73	0.0005	0.1429	0.0401	
Incongruence	1	56.5460	6.22	0.0135	1.5617	0.6260	
Ambiguity	10	22.3305	2.46	0.0088	*	*	
Model: confidence	12	7.6835	2.80	0.0015			0.1510
Error	167	2.7423					
Intercept				0.0001	7.0094	0.4283	
Complexity	1	28.1709	10.27	0.0016	−0.0705	0.0220	
Incongruence	1	19.4967	7.11	0.0083	−0.9170	0.3439	
Ambiguity	10	1.8109	0.66	0.7601	*	*	

Table 5
ANCOVA results summary

Measurement	Results consistent with hypotheses		
	H1: Ambiguity	H2: Incongruence	H3: Confidence
Accuracy			
Micro errors	Yes	Yes	Yes
Macro errors	Yes	Yes	Yes
Efficiency			
Time	No	No	Yes
Number of attempts	Yes	Yes	Yes
Confidence	No	Yes	Yes

ciency, and confidence are inversely related to the ambiguity of information requests, the results support the hypothesis for accuracy as measured by micro and macro errors and for efficiency as measured by the number of attempts. Confidence was not significantly associated with ambiguity.

For H2, that accuracy, efficiency, and confidence are inversely related to construct congruence, the results support the hypothesis for accuracy as assessed by micro and macro errors, for efficiency as assessed by the number of attempts, and for confidence. Eventhough efficiency, as measured by the number of attempts, was significant for both hypotheses, elapsed time was not significant for either one. This suggests that elapsed time may be driven by factors that are independent of a user's ability to formulate correct queries. As expected, H3, that query performance will be inversely related to query complexity, was supported by all measures of accuracy, efficiency, and confidence.

5. Discussion

The results of this study support the idea that the interaction among information requests, the query language, and the data representation affect individuals' ability to formulate correct queries. Specifically, ambiguity in information requests adversely affects accuracy and efficiency. Incongruence among the information request, the query syntax, and the data representation adversely affects accuracy, efficiency, and confidence. Because these effects are in addition to the complexity effect, it may be appropriate to include ambiguity and incongruence as well as complexity in future research on improving query development.

5.1. Implications arising from ambiguity results

The results for ambiguity suggest that organizations might elicit better query development from casual users if they were sensitized to the nature of the kind of ambiguities that could arise in their business contexts and were trained to translate natural language queries into pseudo-SQL that could be examined for precision before the queries were developed. For example, database professionals and users together could identify archetype ambiguities inherent in specific user's queries. Once sensitized to specific ambiguities, users could clarify the meanings of the information requests.

From Fig. 4, ambiguity is associated with errors in attributes, keywords, and, to a lesser extent, relational operators and tables. From Fig. 5, ambiguity is associated with errors in select, where condition, group by, and having clauses. Using currently available tools and techniques, organizations can take a number of steps to reduce these ambiguity-induced errors. For example, the user responsible for formulating a query could focus on more clearly identifying the attributes and columns needed to satisfy the request and on the conditions the data must satisfy. Users could work on improving their communications with the people making the information requests and following a more structured stepwise refinement process when converting the information requests to SQL queries. Users could develop pseudo-SQL representations of the information request and discuss that intermediate formulation with the information requestor.

Database owners could improve the data dictionary to enhance definitions and descriptions of both tables and attributes within the tables. Organizations could train users responsible for formulating queries to be more conscientious and insightful when developing queries. For example, information requestors typically want to see the values of the attributes used

in the restrictions even if they do not explicitly identify them when stating the attributes they want. Organizations could also cross train their employees. That is, given the importance to most organizations of exploiting their information system resources, information requestors need to improve their understanding of the data stored in their organization's information systems, and information providers (persons responsible for formulating the queries) need to improve their understanding of the organization.

Using specification languages such as VDM, Z, and B could help reduce ambiguity between information system analysts and programmers. To information requestors, however, these languages are likely to be even less understandable than query languages such as SQL and QBE. Future research could develop and test a specification language to facilitate clearer communications between information requestors and information providers.

5.2. Implications arising from incongruence results

The results for incongruence suggest that better query development might ensue if semantic distances could be reduced by giving users data representations and database views that maximize construct congruence for the kinds of queries in their domains. For example (based on the ERD in Fig. 2), users that often developed queries requiring the Driver and Client tables to be joined on PostCode (question 2 as explained in Appendix A) could be given a view with that join.

From Figs. 6 and 7, incongruence is associated with substantial increases in errors for almost all types and almost all clauses. The typical way of viewing the data structure and its relationships, e.g., via an ERD, creates and reinforces a powerful mental model of an information system. This mental model acts as an anchor that can inhibit an information provider from formulating a query that correctly satisfies the information request. In addition to the improvements recommended to reduce problems arising from ambiguity, organizations could develop or encourage their database management system provider to develop an adaptive data structure interface, e.g., an adaptive ERD interface.

There are several ways that using an adaptive ERD interface might promote improved query performance. For example, ERDs could be tailored to the requirements of specific groups of users. In such settings, users would rarely need to perform joins in their queries, which would make them less complex and thus less error prone.

Because it is not possible to anticipate every information need in advance, tailored ERDs would not be feasible in every situation. Instead, an adaptive interface could show or list the tables comprising an information system rather than displaying an ERD with the typical relationships and foreign keys. Individual tables could be expanded to display their attributes or be designated as part of the subsystem necessary to satisfy the information request. After selecting the tables deemed relevant to the information request, users could then specify the foreign keys between those tables. If the underlying database management system provided domain support, the adaptive interface could inform users of possible foreign keys and issue warnings about attempts to use foreign keys with incompatible domains.

Formulating queries in any query interface would be facilitated by consistent naming practices, i.e., using the same name for the same attribute in different tables and avoiding using the same name for attributes that are actually different. An even better practice would be to use a data dictionary to resolve such inconsistencies.

Future research could investigate the desirable characteristics of a knowledge-based interface that assists users in formulating queries. Such an interface might request a natural language formulation of the information request, help users build their queries, warn users of likely errors, check the queries against natural language formulations for completeness, and suggest ways to enhance queries.

Acknowledgements

The authors are indebted to Jon Heales, Ron Weber, program committee members and participants at The Pacific Asia Conference on Information Systems 2000, and anonymous reviewers for helpful comments.

Appendix A. Information requests and model SQL queries

1. Congruent; Halstead Difficulty = 6.06

1m. Management wants the names of personal clients and their credit limits.

1p. List client names and credit limits where the clients are personal clients.

surname, firstname, credit_limit

```
select surname, firstname, credit_limit
from client, personal_client
where client.client_no = personal_client.client_no;
```

2. Incongruent (missing relationship); **Halstead Difficulty = 5.25**

2m. Management wants to know the names of drivers with the same post code as clients.

2p. List names of drivers where the post code of the driver is the same as the post code of a client.

surname, firstname

```
select surname, firstname
from driver, client
where driver.postcode = client.postcode;
```

The incongruence in question 2 is the missing direct relationship between the two tables in the join. The typical (congruent) relationship between tables is illustrated in question 1, i.e., in Fig. 2, the Client and Personal_Client tables are adjacent with the obvious foreign key of Client_No. In contrast, the two tables in question 2 are *not* adjacent. In fact, the minimum path between the Driver and Client tables involves a minimum of three intermediate tables. Furthermore, because the attribute Postcode is not the primary key of either table, the foreign key between the Driver and Client tables is more difficult to determine than for question 1, where Client_No is the primary key for both of the tables involved in the join.

3. Congruent (aggregation); **Halstead Difficulty = 10.25**

3m. Management wants the names of bill-to clients who are persons and their total delivery charges. Clients with larger charges should be listed first.

3p. List the client numbers of bill-to clients who are persons and their total delivery charges with clients having larger charges listed first.

surname, firstname, sum(deliver_charges)

```
select surname, firstname, sum(deliver_charges)
from personal_client, bill_of_lading
where     personal_client.client_no = bill_of_lading.bill_client_no
group by surname, firstname
order by 3 desc;
```

4. Incongruent (subquery, cartesian product, or, negation, missing relationship); **Halstead Difficulty = 8.27**

4m. Management wants to know the trucks that can travel all routes.

4p. List the truck numbers of trucks where the truck does not exceed either the height or weight constraints for any route.

truck_no

```
select truck_no
from truck
where truck_no not in
                (select truck_no
                  from route, truck
                        where (height > height_constraint or
                  empty_weight > weight_constraint));
```

The incongruencies in question 4 are subquery, Cartesian product, negation, and missing relationship. Although the query for question 4 does not involve identifying a foreign key, the effects of a missing relationship were examined in the discussion of question 2. Contrast the query for question 4 with the query for question 7. While the query for question 7 involves more tables and several joins, it involves adjacent tables, positive relationships, and natural joins. All conditions address restrictions on a single collection of data. The query for question 4, however, involves two collections of data, i.e., one collection for the primary query and another collection for the subquery. The join in the subquery remains a Cartesian product rather than a more typical natural join. Cartesian products are especially difficult to conceptualize because they require matching each and every record in one table with each and every record in a second table. Cartesian products typically produce an extremely large number of records. Furthermore, many people find negative logic, e.g., *not in*, more difficult, i.e., less congruent with the way they normally think than positive logic.

5. Congruent (aggregation); **Halstead Difficulty = 19.35**

5m. Management wants to know the names of clients who are persons that are over their credit limit.

5p. List the names of clients who are persons where the sum of their unpaid delivery charges exceeds their credit limit.

surname, firstname, credit_limit, sum(deliver_charges)

```
select surname, firstname, credit_limit, sum(deliver_charges)
from personal_client, bill_of_lading, client
where client.client_no = bill_of_lading.bill_client_no and
    client.client_no = personal_client.client_no and
    date_paid is null
group by surname, firstname, credit_limit
having sum(deliver_charges) > credit_limit;
```

6. Incongruent (missing relationships, cartesian product, outer join); **Halstead Difficulty = 16.36**

6m. Management wants to know *all* trucks and the routes that each truck is not permitted to travel.

6p. List *all* truck numbers and where applicable the route numbers where the truck violates height or weight constraints.

```
truck_no, route_no

create view truckviolation as
select truck_no, route_no
from route, truck
where (height > height_constraint or
    empty_weight > weight_constraint);

select truck.truck_no, route_no
from truck, truckviolation
where truck.truck_no = truckviolation.truck_no (+);
```

The incongruencies in question 6 are Cartesian product, missing relationship, view, and outer join. The effects of a missing relationship were examined in the discussion of question 2. The effects of a Cartesian product were examined in the discussion of question 4. A view produces at least two mental incongruencies. First, like the subquery examined in question 4, views create an additional collection of data. Second, the user has to mentally add the view to the ERD and determine the foreign keys to the adjacent tables. Outer joins, while not as mentally incongruent as Cartesian products, require more cognitive effort than natural joins. Users must determine which table contains the data that must be retained and which table may not contain tuples that match tuples in the first table.

7. Congruent; Halstead Difficulty = 21.13

7m. Management wants to know the names of drivers and the trip numbers where the truck exceeded the height constraint.

7p. List the driver's name and the trip numbers where the truck's height exceeded the height constraint for that route.

```
surname, firstname, trip_no

select surname, firstname, trip.trip_no
from route, trip, truck, trip_driver, driver
where     trip.trip_no = trip_driver.trip_no and
    trip_driver.license_no = driver.license_no and
    trip.truck_no = truck.truck_no and
    trip.route_no = route.route_no and
    height > height_constraint;
```

8. Incongruent (outer join, missing relationship, aggregation); **Halstead Difficulty = 16.88**

8m. Management wants to know *all* the destinations (and their city and state) and the number of clients at each of the destinations.

8p. For *all* destinations (same city and state), list the destination number, city, state, and total number of clients at that city and state.

destination_no, city, state, count(client_no)

```
select destination_no, destination.city, destination.state, count(client_no)
from destination, client
where     destination.city = client.city (+) and
    destination.state = client.state (+)
group by destination_no, destination.city, destination.state;
```

The incongruencies in question 6 are missing relationship and outer join. The effects of a missing relationship were examined in the discussion of question 2. The effects of an outer join were examined in the discussion of question 6. The outer join in question 8 is even more incongruent than question 6 because of the concatenated foreign key between Destination and Client. Because of the concatenated foreign key, *two* outer joins are required.

9. Congruent (aggregation, aggregation restriction); **Halstead Difficulty = 30.37**

9m. Management wants to know the trip number, truck number, and maximum and minimum weight of the trucks for the trips where the trucks weighed more than their legal maximum weight.

9p. List the trip number, truck number, and the maximum and minimum weight of trucks for the trips where the cargo weight plus the truck's empty weight exceeds its legal maximum weight.

trip_no, truck_no, legal_max_weight, empty_weight,
sum(unit_weight * qty) + empty_weight

```
select trip.trip_no, trip.truck_no, legal_max_weight, empty_weight,
        sum(unit_weight * qty) + empty_weight
from bill_of_lading, trip, truck, cargo_item
where       trip.truck_no = truck.truck_no and
      trip.trip_no = bill_of_lading.trip_no and
      bill_of_lading.bol_no = cargo_item.bol_no
group by trip.trip_no, trip.truck_no, legal_max_weight, empty_weight
having sum(unit_weight * qty) + empty_weight > legal_max_weight;
```

10. Incongruent (view, missing relationship, cartesian product, outer join); **Halstead Difficulty = 23.19**

10m. Management wants to know *all* drivers and the trucks that each driver is licensed to drive as of 8 Sept 1998.

10p. Assume today is 8 Sept 1998. For *all* drivers, list license number, driver name, and truck number where the driver's license meets or exceeds the requirements for the truck and the license has not expired.

license_no, surname, firstname, truck_no

```
create view legaltruck as
select license_no, truck_no
from driver, truck, licenses
where driver.license_type = licenses.license_type and
      max_truck_rating >= truck_rating and
       license_expiration_date > '8-Sept-1998';

select driver.license_no, surname, firstname, truck_no
from driver, legaltruck
where driver.license_no = legaltruck.license_no (+);
```

The incongruencies in question 10 are Cartesian product, missing relationship, view, and outer join, which have been examined in the discussions of previous questions.

Appendix B. Example of error marking for accuracy in queries

Information request 3: Management wants the names of bill-to clients who are persons and their total delivery charges. Clients with larger charges should be listed first.

Correct solution:

```
select surname, firstname, sum(deliver_charges)
from personal_client, bill_of_lading
where bill_of_lading.bill_client_no = personal_client.client_no
group by surname, firstname
order by 3 desc;
```

Sample actual solution:

```
select surname, sum(deliver_charges)
from bill_of_lading, personal_client
where bill_of_lading.deliver_client_no = personal_client.client_no
  group by surname;
```

The semantic errors in the response are:

- Missing the attribute *firstname* in the *select* clause: this error is recorded at the micro level as one "*Attributes Select*" error and one "*Symbols Select*" error for the extra "," that is required. At the macro level, this error is recorded as a "*Columns*" error.
- Incorrect join the between *bill_of_lading* and the *personal_client* tables: at the micro level, to make the query correct requires the following elements to be deleted and inserted: delete *.deliver_client_no*, which is one "*Symbols Where Join*" error and one "*Attributes Where Join*" error; and insert *.bill_client_no*, which is one "*Symbols Where Join*" error and one "*Attributes Where Join*" error. At the macro level, this is error is recorded as a "*Rows*" error because an incorrect join affects the rows that are retrieved by the query.
- Missing the attribute firstname in the group by clause: this error is recorded at the micro level as one "*Attribute Group by*" error and one "*Symbols Group by*" error for the extra "," that is required. At the macro level, this in not a macro error because this error originated from the error in the *select* clause.
- Missing the order by statement: at the micro level, to make the query correct requires the following elements to be inserted: *order by 3 desc*, which is two "*Keyboards Order by*" errors and one "*Attribute Order by*" error. At the macro level, this error is recorded as a "*Rows*" error because the order of the data retrieved by the query is affected.

Performance, in terms of accuracy, was measured by the total number of micro errors and the total number of macro errors. This example had 11 micro errors and 2 macro errors. The following error counting sheet contains the micro errors and the macro errors for this example query.

Error Counting Sheet		
Name	Information Request Number	Attempts

MICRO ERRORS

Keyboards

View	Select	From	Where Join	Where Cond	Group by	Having	Order by
	1						2

Symbols

View	Select	From	Where Join	Where Cond	Group by	Having	Order by
	1		2		1		

Logical Operators

View	Select	From	Where Join	Where Cond	Group by	Having	Order by

Relational Operators

View	Select	From	Where Join	Where Cond	Group by	Having	Order by

Tables

View	Select	From	Where Join	Where Cond	Group by	Having	Order by

Attributes

View	Select	From	Where Join	Where Cond	Group by	Having	Order by
			2		1		1

Values

View	Select	From	Where Join	Where Cond	Group by	Having	Order by

Set Operators

Where	Union	Intersect	Minus

MACRO ERRORS

Columns	Rows	Aggregation
1	1	

Appendix C. Example calculation of Halstead complexity measure

Halstead's [15] difficulty measure was chosen as the complexity measure for this research. The formula is:

$$D = V/V^* = (N \log_2 n)/(n^* \log_2 n^*)$$

where: N = program length = $N_1 + N_2$; N_1 = total operators; N_2 = total operands; n = vocabulary size = $n_1 + n_2$; n_1 = unique operator count; n_2 = unique operand count; n^* = potential (minimum) vocabulary = $n_1^* + n_2^*$; n_1^* = potential (minimum) operator count; n_2^* = potential (minimum) operand count.

For any SQL query, n^*, the potential (minimum) vocabulary is always 5 because the minimum query is:

```
Select * From table;
```

The solution to information request 3 is:

```
Select surname, firstname, sum(deliver_charges)
From personal_client, bill_of_lading
Where personel_client.client_no = bill_of_lading.client_no
Group by surname, firstname
Order by 3 desc;
```

Operators	Count	Operands	Count
Select	1	surname	2
, (comma)	4	firstname	2
sum	1	deliver_charges	1
( )	1	personal_client	2
From	1	bill_of_lading	2
Where	1	client_no	2
. (period)	2	3	1
= (equal sign)	1		
Group by	1		
Order by	1		
desc	1		
; (semicolon)	1		
$n_1 = 12$		$n_2 = 7$	
$N_1 = 16$		$N_2 = 12$	

$$D = (28\log_2 19)/(5\log_2 5) = 10.25$$

References

[1] H. Almuallim, Y. Akiba, T. Yamazaki, S. Kaneda, Learning verb translation rules from ambiguous examples and a large semantic hierarchy, in: S.J. Hanson, G.A. Drastal, R.L. Rivest (Eds.), Computational Learning Theory and Natural Learning Systems, vol. 4, MIT Press, Boston, 1994, pp. 323–336.

[2] D.A. Boehm-Davis, R.W. Holt, M. Koll, G. Yastrop, R. Peters, Effects of different data base formats on information retrieval. Human Factors 31 (5) (1989) 579–592.

[3] D.J. Campbell, Task complexity: a review and analysis. Academy of Management Review 13 (1) (1988) 40–52.

[4] H.C. Chan, B.C.Y. Tan, K.K. Wei, Three important determinants of user performance for database retrieval, International Journal of Human-Computer Studies 51 (1999) 895–918.

[5] H.C. Chan, K.K. Wei, K.L. Siau, User–database interface: the effect of abstraction levels on query performance, MIS Quarterly 17 (4) (1993) 441–464.

[6] H.C. Chan, K.K. Wei, K.L. Siau, The effect of a database feedback system on user performance, Behaviour and Information Technology 14 (3) (1995) 152–162.

[7] J. Davidson, S.J. Kaplan, Natural language access to data bases: interpreting update requests, American Journal of Computational Linguistics 9 (2) (1983) 57–68.

[8] J.S. Davis, Experimental investigation of the utility of data structure and E-R diagrams in database query, International Journal of Man-Machine Studies 32 (4) (1990) 449–459.
[9] S. Deck, Are You Weary of Warehouses? Computerworld, 1999 (May 31), 61, http://www.computerworld.com/home/print.nsf/all/990531AADE.
[10] J. Gantz, The New World of Enterprise Reporting Is Here, Computerworld, 1999 (Feb. 1), 34, http://www.computerworld.com/home/print.nsf/all/9902018D36.
[11] D.L. Goodhue, Understanding user evaluation of information systems, Management Science 41 (12) (1995) 1827–1844.
[12] D.L. Goodhue, R.L. Thompson, Task-technology fit and individual performance, MIS Quarterly 19 (2) (1995) 213–236.
[13] S.L. Greene, L.M. Gomez, S.J. Devlin, A cognition analysis of database query production, Proceedings of the Human Factors Society 30th Annual Meeting, vol. 2, 1986, pp. 9–13.
[14] S.L. Greene, S.J. Devlin, P.E. Cannata, L.M. Gomez, No Ifs, ANDs, or Ors: a study of database querying, International Journal of Man-Machine Studies 32 (3) (1990) 303–326.
[15] M.H. Halstead, Elements of Software Science. Elsevier, Amsterdam, 1977.
[16] E. Horwitt, Webifying the Mainframe, Computerworld, 1999 (Jan. 25), 76–79, http://www.computerworld.com/home/print.nsf/all/9901258B66.
[17] E. Hutchins, J.D. Hollan, D.A. Norman, Direct manipulation interfaces. in: D.A. Norman, S.W. Draper (Eds.), User Centered System Design: New Perspectives on Human-Computer Interaction, Erlbaum, Hillsdale, NJ, 1985.
[18] J.W.K. Jih, D.A. Braford, C.A. Snyder, N.G.A. Thompson, The effects of relational and entity-relationship data models on query performance of end-users, International Journal of Man-Machine Studies 31 (3) (1989) 257–267.
[19] C. Katzeff, System demands on mental models for a fulltext database, International Journal of Man-Machine Studies 32 (1990) 483–509.
[20] R.L. Leitheiser, S.T. March, The influence of database structure representation on database system learning and use 12 (4) (1996) 187–213.
[21] F.H. Lochovsky, D.C. Tsichritzis, User performance considerations in DBMS selection, Proceedings of ACM SIGMOD. 1977, pp. 128–134.
[22] D.B. Mitchell, R.R. Hunt, How much effort should be devoted to memory? Memory and Cognition 17 (3) (1989) 337–348.
[23] D.A. Norman, Cognitive engineering. in: D.A. Norman, S.W. Draper (Eds.), User Centered System Design, Erlbaum, Hillsdale, NJ, 1986, pp. 31–61.
[24] D.A. Norman, The Psychology of Everyday Things, Basic Books, New York, 1988.
[25] W.D. Ogden, R. Korenstein, J.B. Smelcer, An Intelligent Front-End for SQL, IBM, San Jose, CA, 1986.
[26] R.C. Parkison, K.M. Colby, W.S. Faught, Conversational language comprehension using integrated pattern-matching and parsing, Artificial Intelligence 9 (2) (1977) 111–134.
[27] R.E. Podhorn, H.J. Pikner, The trend to end-user computing, Internal Auditing 7 (1) (1991) 80–83.
[28] P. Reisner, Use of psychological experimentation as an aid to development of a query language, IEEE Transactions on Software Engineering 3 (3) (1977) 218–229.
[29] P. Reisner, Human factors studies of database query languages: a survey and assessment, Computing Surveys 13 (1) (1981) 13–31.
[30] S. Rho, S.T. March, An analysis of semantic overload in database access systems using multi-table query formulation, Journal of Database Management 8 (2) (1997) 3–14.
[31] S. Sekine, J.J. Carroll, S. Ananiadou, J. Tsujii, Automatic learning for semantic collocation, Proceedings of the Third Conference on Applied Natural Language Processing, 1992, pp. 104–110.
[32] K.S. Suh, A.M. Jenkins, A comparison of linear keyword and restricted natural language database interfaces for novice users, Information Systems Research 3 (3) (1992) 252–272.
[33] I. Vessey, Cognitive fit: a theory-based analysis of the graphs versus tables literature, Decision Sciences 22 (1991) 219–241.
[34] I. Vessey, D. Galletta, Cognitive fit: an empirical study of information acquisition, Information Systems Research 2 (1) (1991) 63–84.
[35] R.E. Wood, Task compexity: definition of the construct, Organizational Behavior and Human Decision Processes 37 (1986) 60–82.

A. Faye Borthick, DBA, CMA, CPA, CISA, is Professor of Accountancy and Director of the Teaching and Learning with Technology Center, Georgia State University, Atlanta, GA, USA. For several decades, she has been working at the fault line of technology shifts in business and education. For courses in accounting and information systems, she has been developing new approaches to improving technology-enabled learning experiences, prompting students to solve problems with technology tools, and engaging students in their learning. She has been teaching completely online courses since 1997.

Paul L. Bowen, PhD, is Senior Lecturer in Information Systems, University of Queensland, Brisbane, Australia, where he teaches information systems and auditing. His primary research includes developing analytical models of data quality, measuring the impact of data errors on decision quality, and applying artificial intelligence techniques to the identification and classification of data errors. He also has research interests in database design, internal control, and software reliability. He has been a systems analyst and project manager at Oak Ridge National Laboratory and has taught at The University of Tennessee and Auburn University. http://www.commerce.uq.edu.au/staff/bowen.html.

Donald L. Jones, PhD, is Assistant Professor of Accountancy, Georgia State University, Atlanta, GA, USA. His PhD is in information systems from the University of Texas at Austin. His principal research interest is how information technology affects decision making. He has published in journals such as *Organizational Behavior and Human Decision Processes*, *Journal of Information Systems*, and *Auditing: A Journal of Practice and Theory*. He is currently on the editorial board of the *Journal of Information Systems*. http://www.gsu.edu/~accdrj/.

Michael H.K. Tse, BCom(Hons), BArts, works in the Communication, Medias and Internet team, JP Morgan Corporate Finance, Hong Kong. He has worked on cellular M&A transactions in the Asia-Pacific region. He has also worked in the Chase JF Financial Institutions Group. He graduated from The University of Queensland with a Bachelor of Commerce (First Class Honours) and a Bachelor of Arts, where he won The Queensland Investment Corporation Prize and The Thomas Brown and Sons Award for research in SQL query performance.