Business School

School of Business Papers

The University of Auckland *Year* 2006

Communicating the Message:Translating tasks into queries in a database context

Ananth Srinivasan* Gretchen Irwin†

*University of Auckland, a.srinivasan@auckland.ac.nz

†University of Auckland, gretchen.irwin@colostate.edu

Communicating the Message: Translating Tasks Into Queries in a Database Context

—ANANTH SRINIVASAN AND GRETCHEN IRWIN

***Abstract*—*This paper examines two components of the user–database interface: the data modeling constructs used to represent the database structure and the query language constructs used for data retrieval. From a theoretical perspective, if both the data modeling and query language support high-level abstractions (HLAs), such as generalization and composition, then the "semantic distance" between the user and the interface will be reduced. We used an in-depth verbal protocol study to explore how users were able to effectively complete two tasks: constructing a data model with HLAs and formulating queries against the data model. Results suggest that a successful strategy for modeling with HLAs involves the systematic transition between higher and lower levels of abstraction. In addition, there is some support for the idea that there is a "productivity payoff" to modeling with HLAs, because subsequent query can be simplified.*

***Index Terms*—*Data modeling abstractions, query formulation, user–database interface, verbal protocol.*

In his work on organizational communication, Te'eni focuses on the issue of how designers of computing applications can foster better user communication in an organizational context [1]. While his model of organizational communication looks at a broad characterization of the problem involving the exchange of information among organizational participants, some of the ideas that are explored apply equally well to the communication that takes place during typical user–system interactions. Understanding the problem of how a user translates the requirements of a task to fit a technically constrained space is fundamental to our ability to produce better design products. In this research, we examine the issues that surround a particular type of user–system interaction, namely the composition of problem requirements in a database interaction context.

In the context of data management systems, two important components of the user–database interface are the data model and the query language. The database designers and the end users are affected by both of these aspects of the interface. The DATA MODEL provides a grammar (a set of rules and constructs) for representing the static, deep structure of a database [2]. Each construct in the grammar, such as a table or an entity type, is a data abstraction that captures the meaning of some part of a real-world system. The designer communicates his understanding of the problem domain by creating a representation using the constructs supported by the modeling grammar. This representation becomes the foundation for physical database design and for query formulation. The QUERY LANGUAGE allows the user to manipulate data abstractions in order to accomplish domain-specific tasks. End users pose questions to (and get answers from) the database by using the query language as a communication device to translate their task requirements into useful results.

The usability of the user–database interface depends, among other things, on the "semantic distance" between the abstractions in the users' domain and the abstractions supported by the interface [3], [4]. When the semantic distance is small, the cognitive load on the user is expected to be less than when the semantic distance is large. An interface that narrows the semantic distance allows users to better focus on their tasks and is therefore expected to improve user productivity and system usability.

The semantic distance argument suggests that data models and query languages will be more effective if they support high-level abstractions (HLAs) and user-oriented semantics, rather than lower level abstractions and system-oriented semantics. We use the terms ABSTRACTION or DATA MODELING ABSTRACTION in this paper to refer to a semantic construct that is supported by the user–database interface. For example, an interface that allows a user to manipulate a sales order as a single composite object is a more user-oriented and higher level abstraction than an interface that forces the user to manipulate a sales order as a set of rows spread across two separate tables (the order and the order line) and related

Manuscript received November 9, 2004; revised June 6, 2005.
A. Srinivasan is with the Department of Information Systems and Operations Management, University of Auckland, Auckland, Private Bag 92019, New Zealand (email: a.srinivasan@auckland.ac.nz).
G. Irwin is with the Computer Information Systems Department, College of Business, Colorado State University, Fort Collins, CO 80523-1277 USA (email: gretchen.irwin@colostate.edu).

IEEE DOI 10.1109/TPC.2006.875075

via a foreign key/primary key pair. While this line of reasoning is appealing, empirical studies of user–database interaction have not consistently supported the argument. For example, some studies found that participants using an entity-relationship (ER) representation outperformed participants using a relational representation on query formulation tasks [4], [5], while other studies found the opposite result [6], [7].

We argue that much of the inconsistency in prior research stems from a lack of understanding of the cognitive demands placed on users during data-modeling and query-formulation tasks. The purpose of this paper is threefold. First, we use Te'eni's communication model as a theoretical framework for understanding the cognitive context in which user–database interaction occurs [1]. The model is used to synthesize the extant literature on both data models and query languages, which reveals conflicting evidence regarding the benefits of HLA support in the user–database interface. Second, we provide a theoretically grounded framework for reconciling the conflicting results and present research hypotheses drawn from this framework. Specifically, the framework suggests conditions under which HLAs should lead to a productivity payoff for end users. Third, we describe an exploratory protocol study of users engaged in data modeling and query formulation tasks. The study investigates the ease or difficulty with which participants create data models using HLAs and whether subsequent queries are easier to formulate when based on a model with HLAs.

Results of the study reveal that a successful strategy for modeling with HLAs involves the systematic transition from higher levels of abstraction to lower levels of abstraction and back again. This strategy differentiated the best from the worst performers on the data modeling task. There are also indications that the effort expended in successfully modeling with HLAs does have a payoff in data retrieval tasks, particularly as the query tasks become more complex.

PRIOR RESEARCH

Te'eni presents a theoretically grounded and comprehensive model of organizational communication [1]. The model has three main factors: inputs to the communication process, the communication process itself, and the communication impact or outcome. The model examines both **cognitive** elements of communication, such as the increased complexity when the representation and use of information are incompatible, and **affective** elements, such as the increased complexity when there is mistrust between the parties. Our research focuses on the cognitive, rather than the affective, aspects of communication.

The adaptation of Te'eni's model shown in Fig. 1 is useful for understanding user–database interaction, where the database interface is a communication device with which the user must interact to accomplish his or her task. Each instance of user–database interaction may be viewed as a communicative act where the user is both the sender and receiver of the message and the database interface defines the communication medium and constrains the form of the message. The key arguments that are useful for our own research are outlined below.

- The primary communication outcome of user–database interaction is understanding based on the feedback from or results returned by the system.
- In user–database interaction, the sender is also often the receiver of the message. For example, an analyst may be using a conceptual modeling language to increase his/her own understanding of the users' domain or may be creating the model to increase mutual understanding between the analyst and user.
- Cognitive complexity increases when the representation and use of information are incompatible. This source of complexity is inherent in data modeling and query formulation tasks. Data modeling tasks are cognitively

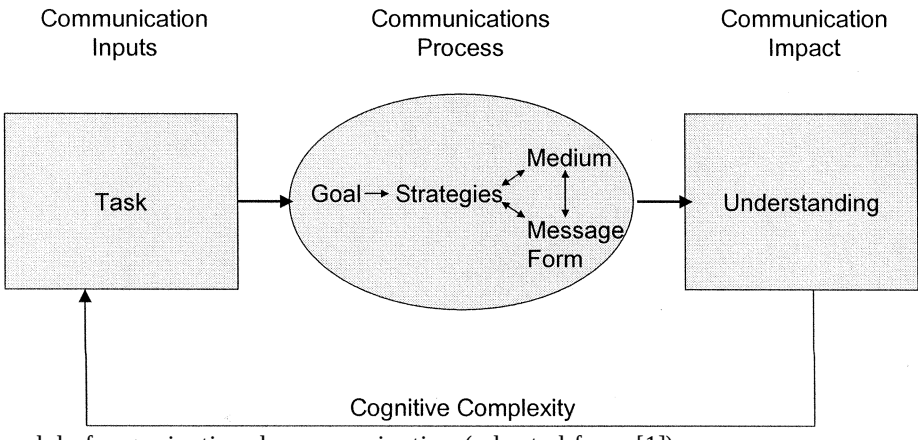


Fig. 1. Cognitive model of organizational communication (adapted from [1]).

complex because the modeler's initial mental representation of the problem domain must be translated into a representation that is constrained by the narrow set of constructs defined in the modeling language. Query formulation tasks are cognitively complex because of the gaps between the user's initial mental representation of a task (e.g., list the total sales, by store, for the last two years), the database query representation (e.g., with SELECT, FROM, WHERE, and GROUP BY clauses), and the representation of the desired information (e.g., the query result set).
- The communication process is goal-driven and constrained by the medium and message. The process consists of an explicit goal and one or more strategies for accomplishing the goal.
- The main input to the communication process is the task, which in a user–database interaction context includes data modeling and query formulation tasks.

We organize the discussion of prior research first based on the type of communication task that was studied and second on whether the emphasis of the study was on the process or on the impact (outcome).

Conceptual Modeling Tasks There are many experimental studies on conceptual modeling tasks, where the designer interacts with a modeling language to create a representation of the users' problem domain. In terms of the communication model in Fig. 1, the participant in these studies—a novice or experienced data modeler—was both the sender and receiver of the message and had the goal of understanding the problem domain in terms of conceptual data modeling constructs provided by the modeling language. Table I summarizes a few studies that investigate the modeling process.

The process studies in Table I used in-depth verbal protocols from small samples (usually ten or fewer participants). The data models examined include entity-based models [8], [9], object-oriented (OO) models [10], and comparisons of the ER, relational, and/or OO models [11]. A consistent finding across most of these studies is that participants who produced better models thought about the problem in terms of the abstractions supported by the data model (e.g., entities, objects, relationships). When the modeler was able to bridge the semantic distance between real-world abstractions and abstractions supported by the modeling language, the cognitive complexity of the task was reduced. Another consistent finding was that successful modelers shifted their focus from the high-level problem to the details [9], [10]. Decomposition was a successful strategy for data modeling tasks, although this decomposition rarely proceeded in a clear top-down fashion.

The Vessey and Conger study [11] showed that novices had more difficulty learning the OO model than the ER model. This is surprising given the claims of the "naturalness" of the OO model and its support for higher level abstractions such as generalization and aggregation. However, other studies have explored the difficulties of learning and using OO constructs [12]–[14], which partially offset the naturalness arguments.

Table II summarizes many of the studies that focus on the outcome of the task rather than on the process itself. These studies treat the data modeling process as a black box and examine the effect of user, task, or model characteristics on task performance. Other reviews of data modeling studies can be found in [15].

In these studies, participants were asked to create a representation of the database structure for a given case using a modeling technique in which they had been trained. Performance was typically measured by the quality of the data model, defined as a correctness or completeness score. High-quality data models are important because they provide the basis for problem-domain understanding and user–analyst communication. Task characteristics were held constant in the studies shown in Table II, and user

TABLE I
EMPIRICAL STUDIES OF THE DATA MODELING PROCESS

Study	Research Questions	Results
Batra & Davis [8]	Expert-novice differences in conceptual data modeling	Experts focused on "big picture" first, and were able to extract standard abstractions from a categorization of problem descriptions. Novices had difficulty integrating parts of the problem domain.
Vessey & Conger [11]	Learning difficulties for different modeling techniques	Process-based modeling was easiest for novices to learn; object-oriented modeling was most difficult
Srinivasan & Te'eni [9]	Complexity management activities in conceptual data modeling	Novices used heuristics in data modeling, some of which effectively reduced problem complexity. Transitions between levels of abstraction were key to successful modeling.
Pennington et al. [10]	Expert-novice differences in procedural vs. object-oriented (OO) design	OO experts quickly defined objects and reasoned about (analyzed) the problem space using these objects. Novices spent more time analyzing the situation and postponed object specification until late in the session; transition from analysis to design was time-consuming and problematic.

characteristics were usually controlled through a homogenous group of students (surrogate business users). Some studies examined novice versus expert differences in data modeling performance, where expertise was operationally defined in terms of programming or database experience and coursework (e.g., [3]).

The studies in Table II examine the impact of data model characteristics on task performance. They compare two or more data models, where the models differ in terms of their SEMANTICS and SYNTAX. Semantics refers to the type of abstractions or constructs supported; syntax refers to the set of symbols used to represent these constructs [7]. The semantics and syntax of a particular modeling language dictate the form of the representation and may reduce or expand the cognitive complexity of the task. Table III outlines the semantic and syntactical aspects of the data models studied.

Several studies compare the relational model (with its table-based semantics) to one of the models with entity-based semantics. For data modeling and model-comprehension tasks, these studies have generally found that participants using entity-based semantics outperformed participants using table-based semantics, although the advantage is sometimes limited to certain parts of the task [3], [7], [19]. Other studies compare entity-based models to the object-oriented model. Proponents of the OO model argue that it is semantically richer and closer to the user's world than other data models, which means that an OO modeling language would be expected to reduce the cognitive complexity of the task. However, empirical evidence has not supported this argument and has instead found that the OO model was more difficult to learn than data-oriented models [11] and the OO model resulted in less accurate designs than the ER model [18].

In summary, data models with higher level abstractions have not consistently outperformed models with lower level abstractions. There is some support for entity-based semantics over table-based semantics but little, if any, support for object-oriented semantics over entity-based semantics. However, comparing results across studies is difficult because of variances in the specific modeling techniques used, the extent and nature of training provided, the experience level of the participants, the domain and complexity of the experimental tasks, and the ways in which data model quality is measured. What is clear is that there is still much we do not understand about the data modeling process and the influence of different semantic constructs on that process.

Query Formulation Tasks As is the case with data model usability, most of the prior work on query language usability treats the query formulation process as a black box, opting to study the effect of different user, task, or query language characteristics on task performance. While there are many tasks that could be used to examine query language usability (e.g., question comprehension, query reading), most studies include query writing tasks, since this is the primary task "real" users are faced with in using query languages to retrieve data from a database.

TABLE II
EMPIRICAL STUDIES OF DATA MODELING OUTCOMES

Study	Research Questions	Results
Batra et al. [3]	Compare relational versus EER representations on data model correctness and ease of use	EER models better than relational in some respects, but differences were not significant on more complex aspects of task (e.g., recursive relationships).
Batra & Kirs [16]	For novice designers, compare two different relational representations (data aggregation versus logical relational design) on data model quality	No significant differences in quality between the two representations.
Kim & March [17]	For experienced analysts, compare EER model versus NIAM on data model quality and perceived usefulness	EER group had better quality solutions than NIAM group; EER modeling language perceived as more useful than NIAM
Bock & Ryan [18]	Compare EER versus OO languages on data model quality	EER group performed better than OO group on several aspects of the solution.
Batra & Antony [19]	Compare EER versus OO languages for modeling user views, when characteristics of the view are varied	The EER group performed better than the OO group; and the degree of nesting within a user view had significant effect on data model quality
Palvia et al. [20]	For novices, compare Data Structure Diagram (DSD), ER model, and OO model on comprehension, task efficiency	OO group performed better than DSD or ER groups on most dimensions.
Shanks [21]	Compare novice and experienced designers on ER model quality	Experienced designers were more correct, complete, innovative, and flexible than those of novices
Shoval & Shiran [22]	Compare EER and OO languages on data model quality and time to complete task	EER group performed better on some quality dimensions and took less time to complete task

Table IV summarizes the empirical studies of the usability of query languages in query writing tasks.

Query formulation **task performance** is usually measured in terms of the number of correct queries, the number of errors made, and/or the time to complete the task. Some studies include dependent variables based on attitudinal measures (e.g., confidence in query accuracy, satisfaction) in addition to objective performance measures. A few others differentiate between learning task (e.g., training time required to meet a certain criterion) and experimental task performance [23].

On the independent variable side, some researchers have examined the influence of **user characteristics**, such as programming experience, on task performance. However, most studies control for individual differences by using a homogenous group of participants, typically students with limited programming or database background. This is a reasonable approach given that query languages are high-level special-purpose languages generally intended to be used by nonprogrammers [23]. In terms of **task characteristics**, most studies have varied the complexity level for the query writing tasks, in order to investigate main and interaction effects related to task complexity. Not surprisingly, several studies found a main effect for complexity, where simpler queries take less time and tend to be more accurate than complex queries, other things equal [6]. More interesting is the interaction effect between complexity and other factors, as discussed below. One of the challenges in comparing task complexity across studies is that there is no commonly accepted definition or measure of complexity for query writing tasks, so that the most complex queries from one study may bear little resemblance to the most complex queries in another study.

Query language characteristics have been examined more than any other factor related to query

TABLE III
COMPARISON OF DATA MODEL SEMANTICS AND SYNTAX

Data Model	Semantic Constructs	Syntactic Representation
Relational	Tables with columns and rows, linked by foreign keys	Tabular or textual
Logical Data Structure (LDS)	Entities with attributes and relationships to other entities	Graphical (different symbols from ER model)
Entity Relationship (ER)	Entities with attributes and relationships to other entities	Graphical (different symbols from LDS)
Extended ER	Same as ER, plus aggregation and generalization	Graphical
Object *Oriented*	Objects with properties and behavior, *aggregation and* generalization	Graphical

TABLE IV
EMPIRICAL STUDIES OF QUERY FORMULATION TASK OUTCOMES

Study	Focus	Main Results
Chan [24]	SQL and KQL	There was a significant correlation between lines of code and query accuracy
Jih et al. [6]	Relational vs. ER	Relational model users had fewer syntax errors but took longer to write queries than ER model users. No difference on semantic errors. Main effect for task complexity, but no interaction effects.
Yen & Scamell [25]	SQL vs. QBE	QBE users had more accurate queries and were more satisfied than SQL users in pencil & paper test. In online testing (with feedback provided and corrections allowed), no difference in query accuracy, but QBE users took less time to write & debug than SQL users.
Leitheiser & March [7]	Entity-based (LDS) vs. Table-based	On query language learning tasks, table groups took less time than entity group, text-only group did better than symbol groups, and implicit relationship groups did better than explicit relationship groups. On query writing tasks, table groups did better than entity group on some questions; and implicit relationship groups did better than explicit relationship groups for some questions.
Chan et al. [4]	Relational vs. ER	ER/KQL users outperformed relational/SQL users (65% less time, 16% more confident, and 38% more accurate). Differences more pronounced with more complex queries.
Chan et al. [5]	Relational vs. ER	ER/KQL users outperformed relational/SQL users (15% more accurate, 58% of the time taken for SQL users). No difference in confidence.
Siau et al. [26]	Relational vs. ER	ER/VKQL users better than relational/QBE users in initial test (less than ½ the time, 13% more accurate and 16% more confident). Similar results for retention/relearning tests (less than ½ the time, more accurate and more confident than QBE users). Differences stronger with more complex queries.
Speier & Morris [27]	Visual vs. text-based query interface	For low-complexity tasks, accuracy was better with text based interface, but less time taken with visual interface. For high-complexity tasks, accuracy was better with visual interface, but less time taken per task with text-based interface.

formulation. Some of these studies compare two languages that support the same semantic constructs using different syntactical approaches, such as SQL's linear keyword approach to QBE's two-dimensional template-filling approach. As shown in Table IV, results from these studies have been mixed. One study found that users had fewer errors, took less time, and were more satisfied with QBE than with SQL [25]; others found that SQL required less training time and was preferred to QBE; and still others found that in more realistic settings, where feedback and error messages were provided, there was no difference in query accuracy but an advantage to QBE in terms of query writing time [25]. Thus, when two query languages support the same abstractions (e.g., tables, columns, keys), it is unclear how differences in form affect task outcome.

Other studies compare languages that use the same form or syntactical approach (e.g., linear keyword) but differ in the abstractions supported (e.g., SQL's table-based constructs versus natural language constructs). The argument here is that natural languages, which reduce the semantic distance between the user's real-world abstractions and the query language's abstractions, should be easier and better to use. However, the results have generally showed no significant difference in test results or query accuracy between SQL and natural languages. Comparisons across these studies are difficult because there is no standard or accepted natural language interface, and the interfaces and vocabulary restrictions and training provided in these studies varied considerably.

Lastly, some prior research has examined the effect of **data model characteristics** on query formulation tasks. These studies can be split into those that varied the data representation while holding the query language constant [6], [7], and those that varied both the representation and the query language [4], [5], [26]. The underlying argument between these two sets of studies is what is the more appropriate or fair comparison—does the query language need to support the same semantics as the data model? In terms of Te'eni's communication model (Fig. 1), does reducing the semantic distance between the data modeling abstractions and the query language abstractions reduce the cognitive complexity of the query formulation task?

When all participants use SQL—a query language with table-based semantics—there is limited support for the benefits of having a conceptual, entity-based model versus a relational or tabular representation [6], [7]. If, on the other hand, the query language is varied to match the semantics of the underlying data model, results consistently show the benefits to using higher level models [4], [5], [26]. These results are consistent with Reisner's description of the query formulation process, which says users "first generate a query template, then…translate (or 'transform') words from the English question into the appropriate query language terms, and then insert these terms into the template" [23, p. 26]. The translation between the real-world task and the query language is facilitated when the constructs in the representation of the database easily map to the constructs in the query template.

In summary, the human factor studies of query languages have not convincingly demonstrated that certain language features or data model features reduce the cognitive complexity or improve user performance on query formulation tasks. There is some support, however, for the argument that when the data model and the query language both support high-level abstractions, productivity is better than when the model and query language both support lower level abstractions. However, this result is complicated by the fact that these studies vary both the query language and the data model, making it difficult to attribute the benefits to differences in the data models or to syntactical or other differences in the query languages. In the present research, we conduct an exploratory study to address the various research questions that arise from this discussion.

RESEARCH MODEL

Our empirical study examines two interrelated tasks in the context of user–database interaction, namely data modeling and query formulation. We explore the extent to which high-level abstractions (HLAs) reduce the cognitive complexity of the tasks and facilitate the successful completion of the tasks.

The first research question opens the "black box" of conceptual modeling. Given the mixed results from prior studies on the effectiveness of higher level data models, we wanted to explore the modeling process when participants were free (not coerced) to design with HLAs and to identify strategies that distinguish successful HLA modelers from unsuccessful ones. It is unclear whether modeling is more or less difficult with HLAs than with lower level abstractions such as tables. On the one hand, HLAs such as aggregation and generalization, are constructs believed to be a "natural" part of human cognition. On the other hand, higher levels of abstraction may be more difficult to process because they require integration of multiple, less-concrete entities. Thus, our first research question explores user behavior to identify successful strategies for modeling with HLAs. In terms of Te'eni's communication model, this question focuses on the task process, where the modeler is communicating via a specific modeling language that includes constructs (HLAs) intended to reduce cognitive complexity, and on the strategies employed to accomplish the task.

RQ1: What strategies do more and less successful data modelers employ?

The second question then investigates how HLAs (or their omission) affects query formulation performance. Fig. 2 shows the research model that motivates this second question and illustrates where the productivity payoff from modeling with HLAs may be realized. A complex, data-oriented task typically means that a user must make reference to multiple constructs and explicitly take into account the interaction among these constructs. The limiting factor here is the ability of the user to apply an arbitrary set of abstractions to collectively reference a representation unit. Even if the typical user is cognitively able to make such references, the issue of whether the software environment allows for such references to be directly translated into database queries needs attention. We argue here that employing a strategy of HLAs as a mechanism to cope with complexity can be leveraged if the software environment provides direct support for such cognitive activity.

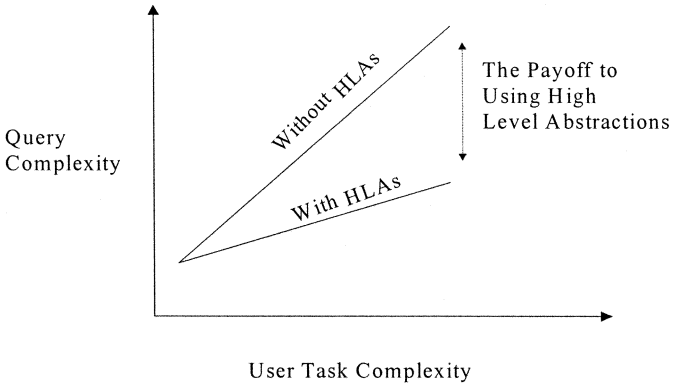


Fig. 2. Productivity payoff to high level abstraction use.

Our main arguments focus on the tradeoff associated with high levels of abstraction in user–database interfaces. First, it does little good to support HLAs in query formulation if the same constructs are not supported in the data model and vice versa. The benefit to using HLAs in one part of the interface will be offset if the user must translate to or from a lower level of abstraction in another part of the interface. It is vital that the semantic distance between data representation and query language be small in order to reduce the cognitive complexity of the user–database interaction. It also improves the analyzability of the query formulation tasks because the process of formulating queries should be simplified if the same HLA constructs are supported in the data model and by the query language.

Second, we believe the benefit of HLAs comes not so much in terms of improved data modeling performance but in less complex queries. Conceptual modeling is a task that is (ideally) done once and remains reasonably stable and static over time. Query formulation, on the other hand, is more dynamic and will happen repeatedly as the information needed from the database changes. Thus, even if modeling with HLAs is more difficult than without HLAs, the payoff from easier, simpler queries based on the HLA model may be significant.

Third, we contend that the payoff from HLAs in query formulation will be more noticeable as the user tasks become more complex. A task that asks for all information from one table with no constraints will be easy regardless of the representation used. However, tasks with many constraints that require joins of multiple tables will be easier with a query language that supports HLAs, because HLAs allow a collection of entities to be manipulated as a single object and hide some of the complexity that would otherwise be needed in the query. This is particularly true for queries with complex predicate clauses involving join conditions. The second set of research questions explores this premise.

RQ2: Is query formulation easier when the data model includes HLAs than when the data model does not include HLAs? Does the ease or difficulty of query formulation vary as the task increases in complexity?

RESEARCH METHOD

Our research questions focus primarily on problem-solving processes. One of the most rigorous and reliable techniques for tracing cognitive processes is the concurrent verbal protocol, which consists of the utterances made by a participant who is “thinking aloud” while solving a problem [28]. Concurrent verbal protocols have a long tradition of use in cognitive psychology to study problem solving and decision making behavior (see [29]–[31] for extensive reviews) and have also been used in software engineering to study a variety of design tasks [9]–[11], [32]–[37]. The concurrent verbal protocol approach is based on the assumption that the verbalizations represent a subset of the contents of short-term memory. Thus, the verbalizations produce a trace of the successive states of information to which a participant is attending.

This section outlines the research method employed. Participants were asked to think aloud and were videotaped as they worked on a data modeling task followed by a set of query formulation tasks.

Participants and Procedure Verbal protocols provide rich qualitative data on cognitive processes of individuals. However, the time- and data-intensive nature of verbal protocol analysis tends to keep sample sizes small, often with ten or fewer participants (e.g., [9]–[11], [34]–[36]). In our study, ten participants were recruited to take part in the experiment from a graduate level course in database management. They were familiar with the concepts of logical database design, normalization, relational

systems, and the structure and syntax of SQL. They had implemented a fairly complex database system as part of the course requirement. Participation in the experiment was entirely voluntary and involved two training sessions of approximately three hours each, followed by the experimental task. The whole process required a time commitment of about ten hours spread over a two-week period. These participants represented a fairly uniform skill level and had a relatively homogeneous background in terms of content knowledge. Therefore, we expect that differences in performance are largely due to cognitive processing differences and/or the use of HLAs rather than to differing degrees of knowledge about the modeling environment.

The Tasks The experimental task required an environment that allowed users to represent a given problem using HLAs. Specifically, we focused on two abstraction mechanisms—generalization and composition [38]. These are commonly discussed extensions to the ER model and also are core OO modeling constructs. These two HLAs allow collections of entities to be viewed at a more abstract level as a single object.

Generalizations allow for the definition of an entity type along with its associated subtypes. While all subtypes share a common set of properties, each one may have a unique property subset that is different from other subtypes of the same parent. The generalization abstraction is constructed as a cluster that contains the supertype with its associated subtypes. Further, subtypes inherit properties from their supertype.

Compositions allow a group of related yet distinct entities to be treated as a molecule. The composite object abstraction captures the relationship concept from ER modeling as well as more complicated situations where the composite object in turn is associated with another entity. A composite object is declared in terms of its component entities. A component entity may be another composite object; in such cases, it is meaningful to refer to a chain of related objects.

Participants completed two tasks of comparable complexity. The first task was used for learning purposes during two training sessions held one week apart. Participants were taught how to construct data models with HLAs and how to use these abstractions in data retrieval tasks.

The second task was the experimental task. Participants were asked to read the narrative and construct a model of the situation using HLAs. The narrative described an organization specializing in hazardous waste disposal and the scheduling of trucks and engineers to pick up containers from client sites. (The case is described in more detail in [9].) The ideal data modeling solution included both generalization and composition HLAs. For example, the solution contained a generalization HLA for different types of employee as well as different types of vehicle. The solution also contained a composite HLA for a work schedule, which was composed of the cargo, site, vehicle, and engineer assigned to a job. Subjects used an ER-type notation to create their model. They had been trained how to use HLAs and how to explicitly represent generalization and composition constructs using the notation. They were asked to "think aloud" as they developed their representation. This session was videotaped for later analysis.

Participants were then asked to construct queries based on their representations for a set of ten tasks, which are listed in the appendix. The tasks varied in complexity from simple to complex. We used an approach that roughly follows the complexity metric proposed by Cardenas to characterize the queries [39]. This metric examines the predicate component of a query and enumerates the number of item value specifications ($M1$), conjunction specifications ($M2$), and join conditions ($M3$). We characterized each of the ten tasks by a vector of three values to indicate the complexity of the task. For each task, we calculated two complexity profiles, one based on a model without HLAs and one based on a model with HLAs. Table V shows the complexity profile for the ten queries under both scenarios.

As Table V shows, queries based on the model without HLAs are generally more complex than their counterparts with HLAs. This observation becomes more pronounced as the tasks become more complex. For example, query #2 is a simple query (no conjunction or join conditions) whose complexity profile is the same for both design alternatives.

TABLE V
COMPLEXITY PROFILES FOR QUERIES WITH RELATIONAL DESIGN AND DESIGN WITH HLAS

Query Task	Model without HLAs <M1,M2,M3>	Model with HLAs <M1,M2,M3>
1	<0,0,0>	<0,0,0>
2	<1,0,0>	<1,0,0>
3	<1,1,1>	<1,0,0>
4	<1,1,1>	<1,0,0>
5	<1,2,2>	<1,0,0>
6	<1,2,2>	<1,0,0>
7	<2,3,2>	<2,1,0>
8	<2,3,2>	<2,1,0>
9	<3,3,1>	<3,2,0>
10	<1,2,2>	<1,1,1>

M1 = Number of item value specifications; M2 = Number of conjunction specifications; M3 = Number of join conditions.

T

However, when query #7 is based on a design without HLAs, the predicate clause requires three conjunction conditions and two join conditions. The same query based on a design with HLAs requires one conjunction and no join conditions. This illustrates the motivation for the second research question about the productivity payoff to modeling with HLAs.

Coding of the Data: Data Modeling Task A coding scheme for characterizing the data modeling process was defined and then applied to the videotapes. The primary focus of the coding scheme was to capture the user's cognitive movement across various levels of modeling abstraction. The coding scheme consisted of four levels of abstraction, as described below.

Level 0: Individual properties of an object (the lowest level of abstraction).
Level 1: Basic entities. These were entities that had no subtypes; usually they were components of composite objects (low level abstraction).
Level 2: Supertype/subtypes cluster or a composite object (high level abstractions).
Level 3: Composite object chain (highest level abstraction).

This coding scheme was applied to the videotapes on the basis of the level at which the participant dealt with the problem at a particular point in time. Simple time stamping was used to track the points at which a participant moved from one level of abstraction to another. This coding scheme and time stamping procedure allowed us to generate a transition graph for each participant. The graph shows a process description of the user's data modeling activity over time and clearly shows the shifts between levels of abstraction in the order in which they occurred.

In addition to the process description, we used the quality of the data model as a measure of performance. The strategy that we used to assess data model quality was similar to previous empirical studies of this nature (e.g., [3], [35]). Quality was assessed as deviations of the data model from a complete solution. Counts of the following measures of representation quality were obtained: missing elements, mis-specified elements, and redundant elements. While there may be several alternative correct model formulations, this approach stressed the capture of all relevant linkages between the problem elements.

Coding of the Data: Query Formulation Tasks Participants were asked to represent three essential components for each querying task: the target list (attributes to be displayed in the result), the source list (objects that provide the data), and the predicate condition (restrictions that specify exactly what is to be retrieved). This representation was familiar to participants due to their prior knowledge of SQL and was also flexible enough to support HLAs. Thus, the same representation and syntax was used regardless of whether the data model included HLAs. A data model with HLAs would affect only the **content** of the query. For example, if the source list contained an HLA (e.g, the work schedule composite object), then the join specifications for linking the objects within the list could be ommitted from the predicate condition.

All query construction tasks were videotaped. The data was coded by time stamping segments of the query tasks depending on whether the participant was generating a target list, a source list, or a predicate condition. This enabled us to obtain a distribution of time spent on each component of a query for all participants.

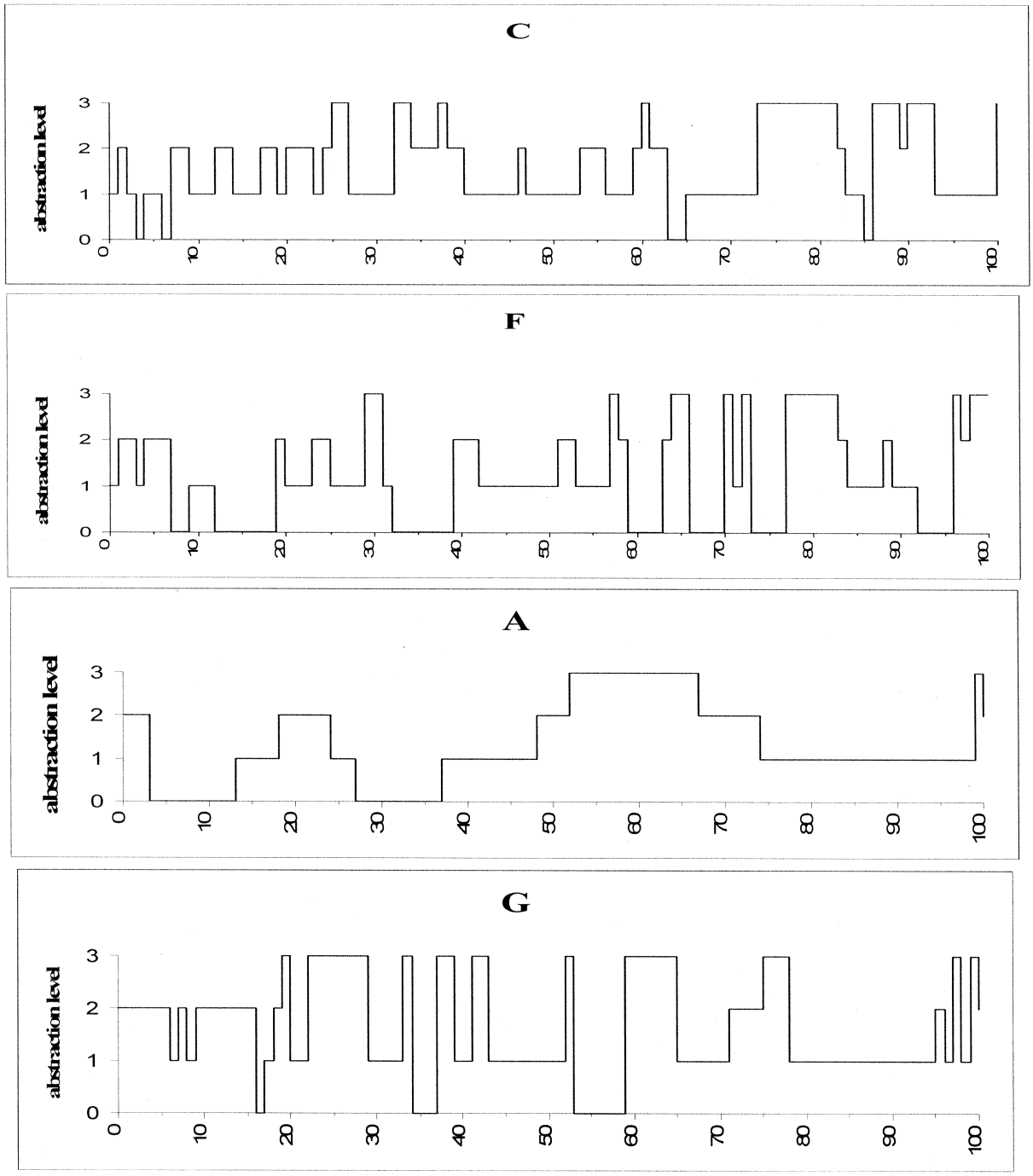
RESULTS

Data Modeling Behavior Table VI shows the raw data for all participants on the data modeling task, including the distribution of time across different

TABLE VI
DATA MODELING RESULTS FOR ALL PARTICIPANTS

			Participant									
			C	F	B	H	D	E	J	I	G	A
Solution Quality (model elements that were)		Missing	0	0	0	0	1	0	0	0	0	2
		Redundant	0	0	0	0	0	0	0	0	1	2
		Mis-specified	0	0	1	1	1	2	2	3	2	3
		Total	0	0	1	1	2	2	2	3	3	7
Level of Abstraction	0	Properties	6%	32%	3%	15%	22%	5%	15%	0%	10%	20%
	1	Entities	49%	35%	31%	32%	36%	28%	39%	36%	46%	44%
	2	Gen. Structures/ Comp. Objects	26%	17%	41%	21%	11%	31%	14%	29%	21%	20%
	3	Composite Chains	19%	16%	25%	32%	31%	36%	32%	35%	23%	16%
Number of Transitions Between Levels of Abstraction			41	37	24	27	23	37	27	17	30	13

levels of abstraction, the number of transitions between levels of abstraction, and model error rates. In terms of the distribution of time spent at different levels of abstraction, Table VI shows that some participants devoted relatively more time to lower levels of abstraction and others did the reverse. The quality of a data model decreases with the number of missing elements, mis-specified elements, and redundant elements. In trying to interpret relationships between the modeling behavior and the performance of the participants, some interesting observations are stated below. We isolated the behavior of Participants A, C, F, and G in order to look at extreme cases in terms of performance. While Participants C and F completely and correctly specified their models, Participants A and G had poorly specified models. Our intention is to study the pairs that are on either end of the performance spectrum to examine the existence of interesting process-related phenomena that might be related to their performance.



Note: Horizontal axis shows the percent of modeling task completed.

Fig. 3. Data modeling transition graphs for high-performing participants (C and F) and low-performing participants (A and G).

Fig. 3 shows the transition graphs for four participants working on the data modeling task. These graphs demonstrate the participants' dynamic behavior in terms of shifts in the levels of abstraction. Shifts to a higher level of abstraction occur when the participant gets lost in detail and needs the broader context to understand the details or to decide how to proceed. Shifts to a lower level of abstraction occur when the participant feels that the higher level can no longer guide him on how to proceed or understand the problem. In general, the higher levels of abstraction are more difficult to process because they require the participant to think of less concrete entities and integrate such entities into a composite object.

For example, Participant A spent more time on the lower abstraction levels (i.e., the attributes and objects) and deferred treatment of the higher levels until late in the solution process. In contrast, Participant C's graph shows treatment of higher levels much earlier in the process. This apparently contributed to the quality of the model. Working early at relatively high levels of abstractions appears to be beneficial.

Further, in examining the transition graphs of all the participants, it appears that the orderly transition from higher to lower levels of abstraction is far more crucial in determining performance than merely looking at the proportion of time spent at each level (see Fig. 4). For example, Participant F, whose performance was excellent, did in fact spend a considerable amount of time at lower levels. However, he started at high levels of abstraction and built clarifications as he went along at lower levels, thus yielding a good model. Finally, Participant G's graph yields some interesting observations. This participant provided a relatively poor model in spite of spending a fair proportion of solution time at the highest abstraction level. However, the focus on the high level abstractions seemed to have occurred primarily during the earlier part of the model building process. The latter stages of the process show an emphasis on the lower levels. This is in contrast with Participants C and F where the focus on higher levels occurred throughout the process with the participants coming back to these levels constantly till the model was completed.

When one examines the transitions across the various levels, the intensity of transitions is revealing. The transition counts (number of transitions from any level to another) for the four participants are shown in Table VI. Participants A and G have 13 and 30 transitions, respectively; Participants C and F have 41 and 37 transitions, respectively. The difference in these counts between the better and poorer performers suggests that it is vital for users to relate higher level abstract objects to the details and to sustain this back-and-forth pattern over time (see Fig. 5). The intensity of transition is also visually obvious from an examination of the transition graphs. The combination of working with high-level abstractions along with relating them with details appears to contribute significantly toward the production of a high-quality model.

Relationship With Query Formulation Behavior The second research question addresses the relationship between model construction and query construction. Specifically, is there a productivity payoff in incorporating HLAs in the model? As was shown theoretically earlier, queries based on models with HLAs should be less complex queries than queries based on models with lower level abstractions such as relations. Specifically, since the primary impact of using HLAs is on the complexity of the predicate condition, we focus our analysis on this component of the query. We examined the query construction behavior of the four Participants A, C, F, and G. Participants C and F performed well on the data modeling task and correctly specified HLAs in their models. Participants A and G performed poorly on the data modeling task and did not incorporate HLAs in their models. Thus, we would expect Participants C and F to formulate less complex

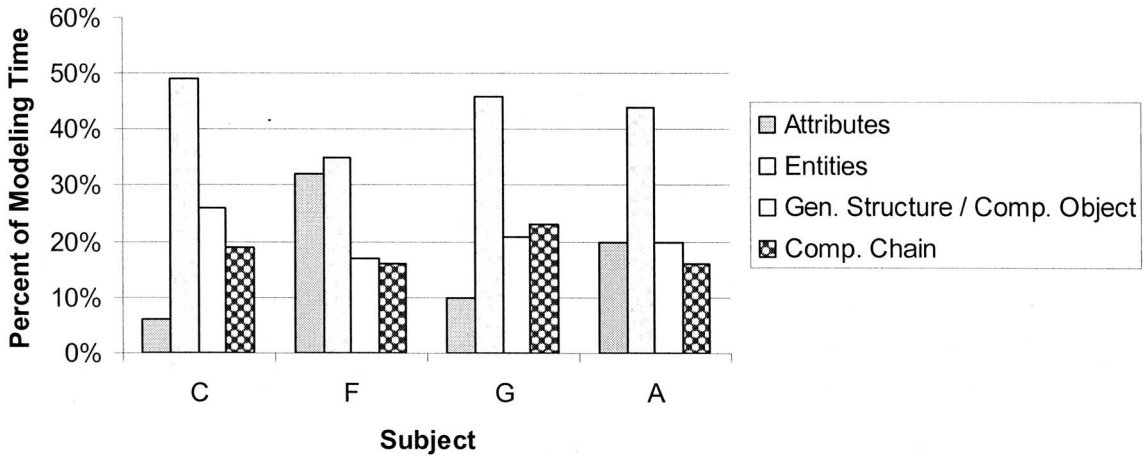


Fig. 4. Percentage of modeling time spent at each abstraction level for best performers (C and F) and worst performers (G and A).

queries and to have fewer errors than Participants A and G.

Tables VII and VIII show the complexity profile for each of the participant's queries, using the same complexity measures described earlier. The complexity profile shows the number of item value specifications, conjunctive conditions, and join conditions in each query's predicate. The complexity profile for each query a participant wrote can be compared to the ideal profile for each query. For Participants A and G, the ideal profile is based on a relational data model, and for Participants C and F, the ideal profile is based on a model with HLAs.

The complexity profiles for the queries of the HLA participants (C and F) corresponded fairly closely to the ideal. Both participants referred to their HLAs in their queries and treated them as single composite entities, as we intended. This enabled them to eliminate many of the join conditions that would be necessary if the HLA were decomposed into elementary relations. In contrast, Participants A and G could not rely on HLAs and had to construct queries that often required joining two or more tables. Specifying join conditions was a consistent problem for these participants, as was identifying the tables needed to process the query correctly. This was also true for the HLA participants, but only task #10 required them to join two HLAs (composite objects).

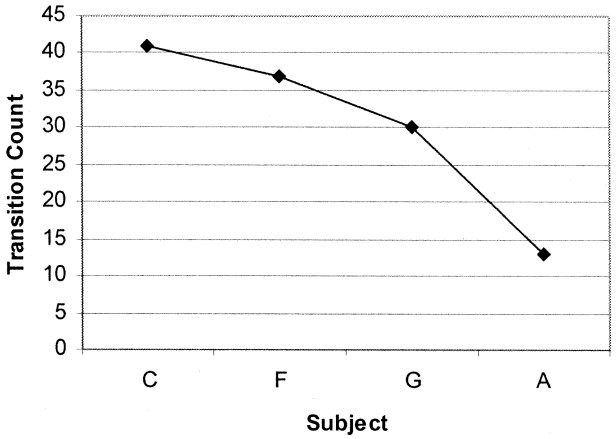


Fig. 5. Number of transitions between abstraction levels for best performers (C and F) and worst performers (G and A).

We also compared the relative amount of time each participant spent on the predicate portion of the queries to see whether a productivity payoff of using HLAs would be evident. Fig. 6 shows the percentage of querying time spent on the predicate component for the participants. Participants C and F spent less of their time on the predicate component than did Participants A and G, on average. What is more striking is that for complex queries (e.g., queries 7–10), the percentage of time spent by A and G on the predicate component is almost uniformly higher. A notable exception to this is Participant A's small amount of time on the most complex query, #10. However, Participant A gave up on this query and was unable to complete it, which actually supports our argument (although it is not obvious from the diagram). This data provides some evidence for the payoff obtained by using HLAs in model construction: queries become easier to construct.

IMPLICATIONS, LIMITATIONS, AND CONCLUSIONS

Understanding how users cope with cognitively complex tasks is important in the design of interfaces to support those tasks. Results of this study

TABLE VII
QUERY COMPLEXITY AND ERROR INFORMATION FROM THE TWO BEST-PERFORMING PARTICIPANTS (WITH HLAs), C AND F

Query Task #	Query Complexity with HLA Model	Participant C			Participant F		
		Actual Query Complexity	Refer to HLAs?	Error(s) in Query?	Actual Query Complexity	Refer to HLAs?	Error(s) in Query?
1	0, 0, 0	0, 0, 0	NA		1, 0, 0	NA	1
2	1, 0, 0	1, 0, 0	NA		1, 0, 0	NA	
3	1, 0, 0	1, 0, 0	Yes		1, 0, 0	Yes	
4	1, 0, 0	1, 0, 0	Yes	1	1, 0, 0	Yes	
5	1, 0, 0	1, 0, 0	Yes		1, 0, 0	Yes	
6	1, 0, 0	1, 0, 0	Yes		1, 0, 0	Yes	
7	2, 1, 0	2, 1, 0	Yes		2, 1, 0	Yes	
8	2, 1, 0	2, 1, 0	Yes		2, 1, 0	Yes	1
9	3, 2, 0	3, 2, 0	Yes		2, 1, 0	Yes	1
10	1, 1, 1	1, 0, 0	Yes	2	1, 0, 0	Yes	2

suggest several avenues for improving user–database interfaces. It is clear, for example, that in data modeling tasks, users ought to be encouraged to work at high levels of abstraction until a broad picture of the problem is at hand and to systematically transition back and forth between levels of abstraction. The interface should provide mechanisms along these directions by which users are inclined to proceed with their modeling activity.

There is some evidence of a productivity payoff to modeling with HLAs. We recognize that the accurate use of these abstractions during model construction is cognitively demanding. However, model construction is typically done once, followed by incremental changes as domain specifications change. The payoff occurs during subsequent use of the model. Use of the model for query formulation is a frequent activity that is performed against a relatively stable representation. Thus, the additional effort invested in constructing a model with HLAs should be quickly offset by reduced effort in query formulation tasks. However, interfaces at the conceptual level must support HLAs for this opportunity to be feasible. Currently, the dominant query language is SQL, which does not support HLAs.

This study examined the productivity payoff associated with HLAs for users who both develop the data model and formulate queries against their model. The two tasks were inter-related in our study because we wanted to explore the impact of HLA use (or lack thereof) in data modeling on subsequent

TABLE VIII
QUERY FORMULATION TASK RESULTS FOR THE TWO WORST-PERFORMING PARTICIPANTS, A AND G

Query Task #	Query Complexity with Relational Model	Participant A		Participant G	
		Actual Query Complexity	Error(s) in Query?	Actual Query Complexity	Error(s) in Query?
1	0, 0, 0	1, 0, 0	1	1, 0, 0	1
2	1, 0, 0	1, 0, 0		1, 0, 0	
3	1, 1, 1	2, 1, 0	2	1, 0, 0	1
4	1, 1, 1	1, 0, 0	2	1, 0, 0	2
5	1, 2, 2	1, 0, 0	2	1, 0, 0	2
6	1, 2, 2	1, 0, 0	1	1, 0, 0	2
7	2, 3, 2	2, 1, 0	1	3, 2, 0	2
8	2, 3, 2	1, 0, 0	3	2, 1, 0	2
9	3, 3, 1	3, 2, 0	1	2, 1, 0	2
10	1, 2, 2	NA	gave up	1, 0, 0	2

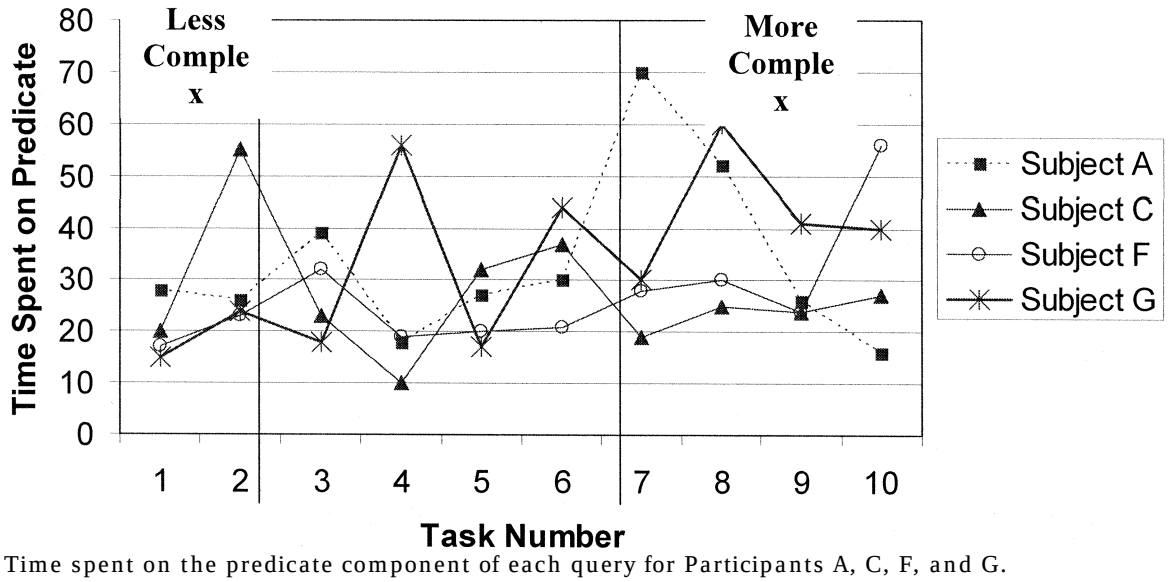


Fig. 6. Time spent on the predicate component of each query for Participants A, C, F, and G.

query formulation. This is one valid setting for investigating the questions of interest, but clearly there are other settings that warrant consideration as well. For example, in many situations, users formulate queries based on a data model created by someone else. In these cases, the issue changes from data model **creation** to data model **comprehension**. It is reasonable to expect that the nature of our findings would not change for a comprehension versus a modeling task, since model creation, in some sense, requires comprehension of the model. However, future studies are needed that examine the usability and productivity payoff of HLAs in different user-task settings.

Given the exploratory nature of this work, the findings should be interpreted as inputs to a reasoned program of research that addresses general issues of usability and interface design in the context of database communication with a user. Te'eni's communication model provides a framework for organizing such a research program. The research approach that we used relies on a methodology that seeks to uncover details about cognitive processes that might highlight important areas of focus. The outcome of this research is biased toward sharpening the nature of the questions that need to be investigated rather than on providing conclusive answers.

REFERENCES

[1] D. Te'eni, "Review: A cognitive-affective model of organizational communication for designing IT," *Manage. Inf. Syst. Quart.*, vol. 25, no. 2, pp. 251–312, 2001.

[2] Y. Wand and R. Weber, "On the deep structure of information systems," *Inf. Syst. J.*, vol. 5, pp. 203–223, 2005.

[3] D. Batra, J. A. Hoffer, and R. P. Bostrom, "Comparing representations with relational and EER models," *Commun. ACM*, vol. 33, no. 2, pp. 126–139, 1990.

[4] H. C. Chan, K. K. Wei, and K. L. Siau, "User-database interface: The effect of abstraction levels on query performance," *Manage. Inf. Syst. Quart.*, vol. 17, no. 4, pp. 441–464, 1993.

[5] ——, "An empirical study on end-users' update performance for different abstraction levels," *Int. J. Man-Mach. Stud.*, vol. 41, pp. 309–328, 1994.

[6] W. J. K. Jih, D. A. Bradbard, C. A. Snyder, and N. G. Thompson, "The effects of relational and entity-relationship data models on query performance of end users," *Int. J. Man-Mach. Stud.*, vol. 31, pp. 257–267, 1989.

[7] R. L. Leitheiser and S. T. March, "The influence of database structure representation on database system learning and use," *J. Manage. Inf. Syst.*, vol. 12, no. 4, pp. 187–213, 1996.

[8] D. Batra and J. G. Davis, "Conceptual data modeling in database design: Similarities and differences between expert and novice designers," *Int. J. Man-Mach. Stud.*, vol. 37, pp. 83–101, 1992.

[9] A. Srinivasan and D. Te'eni, "Modeling as constrained problem solving: An empirical study of the data modeling process," *Manage. Sci.*, vol. 41, no. 3, pp. 419–434, 1995.

[10] N. A. Pennington, A. Lee, and B. Rehder, "Cognitive activities and levels of abstraction in procedural and object-oriented design," *Hum.-Comput. Interaction*, vol. 10, pp. 171–226, 1995.

[11] I. Vessey and S. A. Conger, "Requirements specification: Learning object, process, and data methodologies," *Commun. ACM*, vol. 37, no. 5, pp. 102–113, 1994.

[12] J. M. Carroll, M. B. Rosson, and M. K. Singley, "The collaboration thread: A formative evaluation of object-oriented education," in *Empirical Studies of Programmers: 5th Workshop*, C. Cook, J. Scholtz, and J. Spohrer, Eds., Palo Alto, CA, 1993, pp. 26–41.

[13] H. J. Nelson, G. Irwin, and D. Monarchi, "Journeys up the mountain: Different paths to learning object-oriented programming," *Accounting, Manage., Inf. Tech.*, vol. 7, no. 2, pp. 53–85, 1997.

[14] S. D. Sheetz, G. Irwin, D. P. Tegarden, H. J. Nelson, and D. Monarchi, "Exploring the difficulties of learning object-oriented techniques," *J. Manage. Inf. Syst.*, vol. 14, no. 2, pp. 103–132, 1997.

[15] D. Batra and A. Srinivasan, "A review and analysis of the usability of data management environments," *Int. J. Man-Mach. Stud.*, vol. 36, pp. 395–417, 1992.

[16] D. Batra and P. J. Kirs, "The quality of data representations developed by nonexpert designers: An experimental study," *J. Database Manage.*, vol. 4, pp. 17–29, Fall 1993.

[17] Y.-G. Kim and S. T. March, "Comparing data modeling formalisms," *Commun. ACM*, vol. 38, no. 6, pp. 103–115, 1995.

[18] D. B. Bock and T. Ryan, "Accuracy in modeling with extended entity relationship and object-oriented data models," *J. Database Manage.*, vol. 4, no. 4, pp. 30–39, 1993.

[19] D. Batra and S. R. Antony, "Effects of data model and task characteristics on designer performance: A laboratory study," *Int. J. Hum.-Comput. Stud.*, vol. 41, pp. 481–508, 1994.

[20] P. C. Palvia, C. Liao, and P.-L. To, "The impact of conceptual data models on end-user performance," *J. Database Manage.*, vol. 3, no. 4, pp. 4–15, 1992.

[21] G. Shanks, "Conceptual data modeling: An emprical study of expert and novice data modellers," *Australian J. Inf. Syst.*, vol. 4, no. 2, pp. 63–73, 1997.

[22] P. Shoval and S. Shiran, "Entity-relationship and object-oriented data modeling–An experimental comparison of design quality," *Data Knowl. Eng.*, vol. 21, pp. 297–315, 1997.

[23] P. Reisner, "Human factors studies of database query languages: A survey and assessment," *ACM Comput. Surv.*, vol. 13, no. 1, pp. 13–31, 1981.

[24] H. C. Chan, "The relationship between user query accuracy and lines of code," *Int. J. Hum.-Comput. Stud.*, vol. 51, pp. 851–864, 1999.
[25] M. Y.-M. Yen and R. W. Scamell, "A human factors experimental comparison of SQL and QBE," *IEEE Trans. Softw. Eng.*, vol. 19, no. 4, pp. 390–409, Apr. 1993.
[26] K. L. Siau, C. Chan, and K. K. Wei, "Effect of query complexity and learning on novice user query performance with conceptual and logical database interface," *IEEE Trans. Syst., Man & Cybern. A, Syst. Humans*, vol. 34, no. 2, pp. 276–282, Mar. 2004.
[27] C. Speier and M. G. Morris, "The influence of query interface design on decision-making performance," *Manage. Inf. Syst. Quart.*, vol. 27, no. 3, pp. 397–423, 2003.
[28] K. A. Ericsson and H. A. Simon, *Protocol Analysis: Verbal Reports as Data*. Cambridge, MA: MIT Press, 1993.
[29] P. Todd and I. Benbasat, "Process tracing methods in decision support systems research: Exploring the black box," *Manage. Inf. Syst. Quart.*, vol. 11, pp. 493–512, Dec. 1987.
[30] J. K. Ford, N. Schmitt, S. Schechtman, B. Hults, and M. Doherty, "Process tracing methods: Contributions, problems, and neglected research questions," *Org. Behavior Human Decision Processes*, vol. 43, pp. 75–117, 1989.
[31] J. E. Russo, et al., "The validity of verbal protocols," *Memory & Cognition*, vol. 17, no. 6, pp. 759–769, 1989.
[32] V. Arunachalam and W. Sasso, "Cognitive processes in program comprehension: An empirical analysis in the context of software reengineering," *J. Syst. Softw.*, vol. 34, no. 3, pp. 177–189, 1996.
[33] V. Goel and P. Pirolli, "The structure of design problem spaces," *Cogn. Sci.*, vol. 16, pp. 395–429, 1992.
[34] R. Guindon, "Designing the design process: Exploiting opportunistic thoughts," *Hum.-Comput. Interaction*, vol. 5, pp. 303–344, 1990.
[35] B. C. Hungerford, A. R. Hevner, and R. Collins, "Reviewing software diagrams: A cognitive study," *IEEE Trans. Softw. Eng.*, vol. 30, no. 2, pp. 82–96, Feb. 2004.
[36] G. Irwin, "The role of similarity in the reuse of object-oriented models," *J. Manage. Inf. Syst.*, vol. 19, no. 2, pp. 219–248, 2002.
[37] K. D. Schenk, N. P. Vitalari, and K. S. Davis, "Differences between novice and expert systems analysts: What do we know and what do we do?," *J. Manage. Inf. Syst.*, vol. 15, no. 1, pp. 9–50, 1998.
[38] R. Elmasri and S. Navathe, *Fundamentals of Database Systems*. Redwood City, CA: Benjamin Cummings, 1994.
[39] A. F. Cardenas, *Data Base Management Systems*. Boston, MA: Allyn & Bacon, 1985.

Ananth Srinivasan received the Ph.D. degree from the University of Pittsburgh, PA. He is a Professor of Information Systems and Digital Commerce at the University of Auckland Business School, Auckland, New Zealand. His research interests are in the areas of information modeling, empirical evaluation of systems, and digital technology implementation.

Gretchen Irwin received the Ph.D. degree from the University of Colorado, Boulder. She is an Assistant Professor of Computer Information Systems at Colorado State University, Fort Collins, CO. Her research focuses on the usability and productivity claims associated with complex software tools, such as those that support database design, database manipulation, and software reuse.