JOURNAL OF INFORMATION SYSTEMS
Vol. 19, No. 1
Spring 2005
pp. 43–74

The Effects on End-User Query Performance of Incorporating Object-Oriented Abstractions in Database Accounting Systems

Roger S. Debreceny
University of Hawaii

Paul L. Bowen
University of Queensland

ABSTRACT: Object-oriented (OO) advocates assert that concepts such as generalization-specialization hierarchies (GSHs) and abstract data types (ADTs) make information systems more usable by increasing the level of abstraction of the data structure. This study analyzes the effects of GSHs and ADTs on the performance of end-users of accounting information systems. Two groups of experimental participants interactively developed Structured Query Language (SQL) queries to answer ten business questions. The control group (n = 28) used data stored in a traditional relational schema. The experimental group (n = 31) used the same data stored in an OO schema that included GSHs and ADTs. Both schemas implemented the same database accounting model of the sales cycle of a hypothetical company. Participants using the higher abstraction (OO) schema with GSHs and ADTs made fewer semantic errors than did participants using the traditional relational schema. The OO participants also required less time to formulate their queries. These results have several important implications. First, relational database vendors should continue, if not accelerate, their efforts to incorporate OO features such as GSHs and ADTs into their database systems. Second, users of accounting information systems need to improve their understanding of the implications of various data structures on their interactive queries. Third, research should investigate the effects of other abstraction mechanisms, including classification/instantiation and aggregation/decomposition, on query performance.

Keywords: accounting information systems; database management systems; object-orientation; abstraction; human-computer interaction; query performance; information retrieval.

We thank participants and reviewers at the 2000 American Accounting Association Annual Meeting, the 2004 Midyear Meeting of the Information Systems Section of the American Accounting Association, and workshop participants at the University of Queensland (UQ). We especially appreciate the insightful comments of Peter Clarkson, Cheryl Dunn, Colin Ferguson, Greg Gerard, Jon Heales, Martin Putterill, Ron Weber, the Associate Editor, and two reviewers. Bill McCarthy and Ted Mock provided invaluable assistance and suggestions during the inception of the research. Jim Hansen, David Harvey, Tim Lehmann, Cheryl Lim, and Ray Meservy assisted with the conduct of the research and the coding of the data. The Director of Research at Southern Cross University provided financial support to Professor Debreceny. Professor Bowen received financial support from the UQ Business School Research Fund.

I. INTRODUCTION

In order to better fulfill their goals, a wide range of organizations have made significant investments in complex computerized repositories of organizational knowledge including accounting and enterprise information systems and data warehouses. Increasingly managers and other decision makers are empowered to retrieve and interpret information directly from these data repositories to support their work (Speier and Morris 2003; Wixom and Watson 2001). To increase the effectiveness and efficiency of the information retrieval processes, these organizations have made substantial investments in query interfaces and datamining tools (Cooper et al. 2000). Accurately retrieving information depends, however, not only on the quality of the data extraction tools, but also on the structure of the data the decision makers access (Borthick et al. 2001; Bowen and Rohde 2002; Bowen et al. 2004; Kimball and Ross 2002).

Research in cognition, linguistics, and psycholinguistics shows that humans use abstraction mechanisms it facilitates their classification of knowledge domains, promotes effective problem solving, and aids information retrieval (Lakoff 1987; Langacker 1999; Simon 1996). Humans use abstractions to recognize the similarities between objects, conditions, processes, and actions, and to temporally suppress their differences. Conceptual models of information systems employ abstraction mechanisms to produce more direct and natural representations of the "subject world" (Hammer and McLeod 1981; Jarke et al. 1992; Mylopoulos 1998). The four primary abstraction mechanisms generally recognized in conceptual modeling are (1) classification/instantiation; (2) aggregation/decomposition; (3) generalization/specialization, and (4) grouping/individualization (Mattos 1988; Taivalsaari 1996).

A major thrust in software engineering and computer applications over the last three decades has been the development of the object-oriented (OO, or object) paradigm. In large measure, the object paradigm seeks to reduce design and query complexity by directly employing abstraction mechanisms. For example, in the realm of database management systems (DBMSs), proponents of object databases assert that these databases reduce the mismatch between organizational processes and the implementation of business solutions by providing support for abstraction mechanisms (Cattell 1994; Loomis 1995; Stonebraker et al. 1999). OO databases typically incorporate generalization-specialization hierarchies (GSHs) and abstract data types (ADTs) as implementations of the generalization/specialization and grouping/individualization abstraction mechanisms, respectively.

The dominant model for commercial database management systems over the last several decades has, however, been the relational model (Date 2004). The relational model has a strong theoretical foundation in set theory (Codd 1970, 1990). Relational researchers have used this foundation to enhance and extend the relational database model in many areas such as integrity, security, concurrency control, and distributed processing (see, e.g., Bayer et al. 1980; Bell 1992; Buneman and Clemons 1979; Castano et al. 1995; Ceri et al. 1994; Franaszek et al. 1992). Relational advocates (e.g., Date 2004) argue that providing full support for domains in relational database management systems would align those systems with the aims of proponents of the OO paradigm. For example, domains, also called types, are a set of values. Although domains include primitive data types such as integer and character, they can also include data types such as colors or multiple attributes such as the concatenation of the components of an address (Date 2004, 111). One perspective on ADTs is that they are a subset of domains. Developers of relational DBMSs have increasingly incorporated OO functionality, including support for GSHs and ADTs, into their products. These hybrid databases are referred to as object-relational DBMSs (Loomis 1995; Stonebraker et al. 1999).

A substantial portion of the accounting information systems research into database accounting systems has focused on the use of conceptual modeling techniques and approaches to designing database accounting systems (David et al. 2002; Dunn and Grabski 2002). Little evidence exists, however, on the effects of employing abstraction mechanisms in database accounting systems on end-user query performance (Dunn and Grabski 2002). This research investigates whether abstraction mechanisms benefit information retrieval tasks (Jarke et al. 1992). In this study, we experimentally tested whether incorporating GSHs and ADTs into database accounting information systems improves end-user query performance. Researchers have encouraged the use of experiments to provide richer insights into the general results provided by analytical and design-science research (see, e.g., Dunn and Grabski 2002; Wand and Weber 2002; Weber 2002). To conduct the research, we used a commercially available object-relational database management system (DBMS) to create schemas of the same database accounting system but with different levels of abstraction. The high-abstraction (object) schema included both GSHs and ADTs. The low-abstraction (traditional relational) schema included neither GSHs nor ADTs. System end-users queried the schemas to satisfy a set of ten information requests. We measured performance on two dimensions: effectiveness (query accuracy) and efficiency (time spent). The results showed that users of the higher abstraction (object) schema were more effective, i.e., made fewer errors, than users who queried the lower abstraction (relational) schema. Indeed, compared with users of the relational schema, users of the object schema made just over half the number of semantic errors. Object schema users were also more efficient; i.e., they required less time to compose their queries.

II. HYPOTHESIS DEVELOPMENT

Cognition and Abstraction Mechanisms

The key issue addressed in this study is how to improve the fit between information system design and use in the presence of human cognitive limitations (Jones and Eining 1996; Kim et al. 2000; Rose 2002; Weber 2002). In particular, we focus on end-users' directed information retrieval from an accounting information system. This task is challenging as it requires combining knowledge of accounting structures with information retrieval techniques (Bouwman and Bradley 1997; Libby and Luft 1993). Theories in cognition and reasoning allow us to make predictions on the way problem solvers will interact with information systems that exhibit different characteristics and functionalities. Chunking and categorization theories are closely related theories that make assertions about how experts retrieve information from and process information in short- and long-term memory (Chase and Simon 1973; De Groot 1978; Miller 1956; Newell and Simon 1972). Chunking is a method of organizing information by grouping items together in a meaningful way. Chunking theory asserts that experts organize their mental chunks both hierarchically and semantically.

Chase and Simon (1973) postulated that chunks might be linked together in a hierarchical framework. Subsequent research has shown that experts do indeed maintain a large number of chunks that are indexed by a discrimination net (Gobet 1998; Gobet and Simon 1996). This discrimination net has strong hierarchical characteristics that allow experts to associate a perceived chunk with a particular leaf node in the organization of their memory. Competing theories, including the levels-of-processing theory (Craik and Lockhart 1972) and the connectionistic model (Rumelhart 1994; Rumelhart and Ortony 1977), also emphasize the importance of experts' superior ability to construct efficient semantic networks via hierarchical structures.[1] Evidence for the efficacy of employing hierarchical semantic structures is found in many disciplines including chess and physics (Gobet 1998). Hierarchical models have proven productive in a range of information technologies. For example, in researching computer interfaces, Mynatt (1997) investigated the transformation of graphical user interfaces into auditory interfaces for blind users. She hypothesized that, when comparing spatial, hierarchical, and conversational models, a hierarchical model best captures the underlying structure of the graphical interface. Her experimental investigation confirmed her assertion. Similar superior results for hierarchical structures are also observed in human interaction with hypertext (Simpson and McKnight 1990; van Nimwegen et al. 1999; Wright and Lickorish 1990).

An important component in chunking is categorization where problem solvers recognize the similarities within groups of elements and classify those groups in well-defined semantic structures (Murphy and Medin 1985; Rosch 1973; Wisniewski and Medin 1994). Recent research shows that problems solvers are not only able to learn and apply integrated categories, but are also capable of discerning the causal relationships between categories (Ahn 1998; Rehder 2003). In summary, chunking theory and categorization theory show that grouping together related elements aids learning and problem solving at both the automatic (perceptual) and deliberate (goal-oriented) levels. Hierarchies and categories are both examples of problem solvers employing abstraction mechanisms to overcome the complexity in the problems they face.

Generalization-Specialization Hierarchies

Generalization-specialization hierarchies (GSHs) model class/subclass structures via hierarchical tree structures (Smith and Smith 1977). Classes are groupings of objects with "similar properties, common behavior, and common relationships to other objects and common semantics" (Rumbaugh et al. 1994, 24). Object classes inherit data and behavioral aspects of more general classes and, in turn, provide data and methods to more specialized classes. Where needed, subclasses can add or modify properties or characteristics, i.e., allow specialization (Taivalsaari 1996, 439). GSHs assist human understanding of data structures and database design processes by pushing complexities, e.g., differences, down to appropriate levels in the tree structure (Taivalsaari 1996, 442).

Semantic database models often include capabilities to represent generalization-specialization hierarchies. For example, the Extended Entity-Relationship (EER) model provides techniques for depicting GSHs (Teorey et al. 1986). Object-oriented database systems can implement generalization-specialization hierarchies directly. Traditional relational systems can implement the structural characteristics of GSHs only by creating additional relations (Elmasri and Navathe 2003). Implementing the behavioral characteristics of GSHs within the relational paradigm requires a combination of stored procedures, triggers, and applications.

The economic activities of an entity can be represented by generalization-specialization hierarchies (Adamson and Dilts 1995; Chu 1992). For example, at the highest level, all accounting objects can be categorized as either stocks or flows (Ijiri 1967, 1975; McCarthy 1982). Stocks can be specialized into assets, liabilities, or owners' equity. Assets, for example, can be further specialized into asset classes on the expected cash conversion time cycle.

[1] See Sloman (1998) for a contrary perspective. Sloman argues that most recognized hierarchical semantic structures are better described as particular examples of similarity or categorization.

Figure 1 shows a fragment of a simplified schema that employs a hypothetical generalization-specialization hierarchy, implemented in an object-relational database. The highest level of the hierarchy (`person`) shows generic attributes that describe a person. This is specialized in two "subtables" for employees (`employee`) and sales contacts (`sales_contact`). Each of these two tables automatically inherits the attributes of its parent. The inherited attributes are shown in italics. The employee table adds a link to the department in which employees are working and each employee's title. The sales contact table has added contact information and a link to the employee of the corporation that is their primary contact. This latter attribute is a link from one branch of the GSH to another.

To retrieve information from the child levels of the GSH, the user only needs to include the specialist table in the query. For example, if end-users were asked to "print the full name" of employees that work in the Southeast region, they would merely state "`select given_name, middle_name, family_name from employee where department = 'SE-Sales'`" in their SQL query. If GSHs were not functionally available to the database designer, then the schema would require separate and discrete tables for `person`, `employee`, and `sales_contact`. The resulting SQL query would require a join and

FIGURE 1
Hypothetical Example of a Generalization-Specialization Hierarchy (GSH)

Tables

person

Attribute	**Data Type**
pers_id	char(8) not null primary key
given_name	char(20)
middle_name	char(20)
family_name	char(20)
date_of_birth	date
nationality	char(20)

employee

Attribute	**Data Type**
pers_id	*char(8) not null primary key*
given_name	*char(20)*
middle_name	*char(20)*
family_name	*char(20)*
date_of_birth	*date*
nationality	*char(20)*
department	char(8) references **int_org**
title	char(20)
under person	

sales_contact

Attribute	**Data Type**
pers_id	*char(8) not null primary key*
given_name	*char(20)*
middle_name	*char(20)*
family_name	*char(20)*
date_of_birth	*date*
nationality	*char(20)*
date_of_first_contact	date
date_of_last_contact	date
primary_contact	char(8) references **employee**
under person	

would be formulated as "select given_name, middle_name, family_name from person, employee where person.pers_id=employee.emp_id and department = 'SE-Sales'." This latter query is clearly more complex, requiring the end-user to recall not only the attributes of a person's name, but also the base person table and the associated employee table, and specify how to join those tables together. When end-users query a GSH they do not need to explicitly join parent (person) and child (employee) entities, as they are required to do when querying traditional relational database schemas. Alternatively, each child entity can be viewed as a single chunk rather than two or more distinct entities. Further, the level of complexity of queries increases dramatically as the depth of the GSH increases.

Formal testing of the effect of abstraction mechanisms on database accounting system end-user performance has been limited and the results mixed (Dunn and Grabski 2002). Dunn (1999) tested the ability of two groups of advanced undergraduate accounting students to generate financial statements from an underlying relational database schema. One group used a sequential interface (low abstraction). The other group interacted with a graphical ER representation of the relational schema that included GSHs (high abstraction). Contrary to expectations, users of the high-abstraction interface were less accurate than the users of the low-abstraction interface. Conversely, in two experiments that tested the interaction of end-users with REA-based accounting information systems (McCarthy 1982) and traditional "Debit-Credit Accounting" systems, Dunn and Grabski (2000; 2001) found that the higher semantic expressiveness of the REA model resulted in higher levels of task completion accuracy.

Recall that hierarchical structures are a central feature of theories of cognition and reasoning, in particular, chunking theory. Organizing data structures within a hierarchical semantic network should exhibit higher levels of abstraction and provide better foundations for end-user problem solving. Generalization-specialization hierarchies within the object paradigm are examples of hierarchical semantic networks. Hierarchies allow end-users composing queries to assume that attributes in the parent entity are present in any child entity. We assert that database accounting systems that directly implement generalization-specialization hierarchies will provide a more productive environment for end-user information retrieval than systems that do not implement generalization-specialization hierarchies or implement them in a less elegant fashion. That is, we expect that employing GSHs in database accounting system schema design will result in closer alignment of schemas with underlying knowledge structures. This close alignment will enhance end-user's query performance. This performance can be measured on multiple dimensions including semantic query accuracy (Borthick et al. 2001; Reisner 1977, 1981; Smelcer 1995) and query efficiency (Jones and Eining 1996; Wu et al. 1994). Hence, the first set of hypotheses is:

H1a: End-users querying database accounting systems that incorporate generalization-specialization hierarchies (GSHs) will make fewer semantic errors than end-users querying database accounting systems that do not incorporate GSHs.

H1b: End-users querying database accounting systems that incorporate generalization-specialization hierarchies (GSHs) will take less time than end-users querying database accounting systems that do not incorporate GSHs.

Abstract Data Types (ADTs)

Chunking and categorization theories show that problem solvers visualize categories of elements in their processing of information. In the object paradigm, ADTs provide a grouping/individualization abstraction mechanism for building such categories. ADTs group related information elements in a structured fashion within a single data type. In a full OO system, an ADT may include methods that provide behavioral characteristics. ADTs may incorporate atomic data types as well as other ADTs. Employing ADTs facilitates the creation of elaborate taxonomies and promotes classification schemes (Cattell 1994; Loomis 1995). Figure 2 shows a fragment of a hypothetical object-relational database schema with two ADTS and two tables.

The `address_t` ADT is a contact address structure with columns for physical, telephonic, and electronic addresses. The schema declares the ADT once only as `address_t`. The `shipdetail_t` ADT stores information on typical shipping data including shipping dates and quantities. Tables that require address or shipping information may declare a single column within the schema and declare that column as the appropriate abstract data type, just as the simple `string` or `float` data types might be used for textual or numeric

FIGURE 2
Hypothetical Example of Abstract Data Types (ADTs)

Tables

inwardsinv

Attribute	Data Type
po_id	char(10) references purchorderline
po_line_id	char(10) references purchorderline
unitpurchaseprice	dollar
shipfromdetail	shipdetail_t
shipfromaddress	address_t

outwardsinv

Attribute	Data Type
order_id	char(10) references salesorderline
order_line_id	char(10) references salesorderline
unitsellingprice	dollar
shiptodetails	shipdetail_t
shipaddress	address_t
billingaddress	address_t

Abstract Data Types

shipdetail_t

Attribute	Data Type
item_id	char(6) references inventoryitem
req_qty	integer
min_qty	integer
earliestshipdate	date
prefshipdate	date
lastshipdate	date

address_t

Attribute	Data Type
contactname	char(30)
address1	char(30)
address2	char(30)
city	char(30)
state	char(30)
zipcode	char(10)
country	char(30)
phone	char(20)
fax	char(20)
email	char(30)

data. In this example schema, the inwardsinv table employs the shipdetail_t ADT as the data type of the shipfromdetail column and the address_t ADT as the data type of the shipfromaddress column. The outwardsinv table also uses each of these ADTs.

The benefits of ADTs flow to both the designers and the users of the schema. The schema designers may declare an ADT and then re-use the ADT when appropriate, knowing that they will be using consistent data structures, names, and methods. As far as users are concerned, ADTs operate consistently across all applications within the database system in which they are defined, thereby providing an additional level of abstraction likely to enhance end-user query processes (Cattell 1994; Graham 2001, 18). For example, if end-users were asked to "print the full name and address" of organizations from the schema shown in Figure 2, then they would merely state "select name,address" in their query. If the ADT were not available, then they would need to recall all the attributes of name and address resulting in "select oname, odscrptn, otype, snum, sname, sdetail, city, state, areacode."

Because of enhanced categorization, providing end-users with ADTs that group related elements together should improve problem solving. Hence, we expect that database accounting schemas that incorporate ADTs will enhance system end-users' query performance. As before, we measure end-users' query performance by semantic query accuracy and query efficiency. Hence, the second set of hypotheses is:

H2a: End-users querying database accounting systems that incorporate abstract data types (ADTs) will make fewer semantic errors than end-users querying database accounting systems that do not incorporate ADTs.

H2b: End-users querying database accounting systems that incorporate abstract data types (ADTs) will take less time than end-users querying database accounting systems that do not incorporate ADTs.

III. METHOD

Introduction

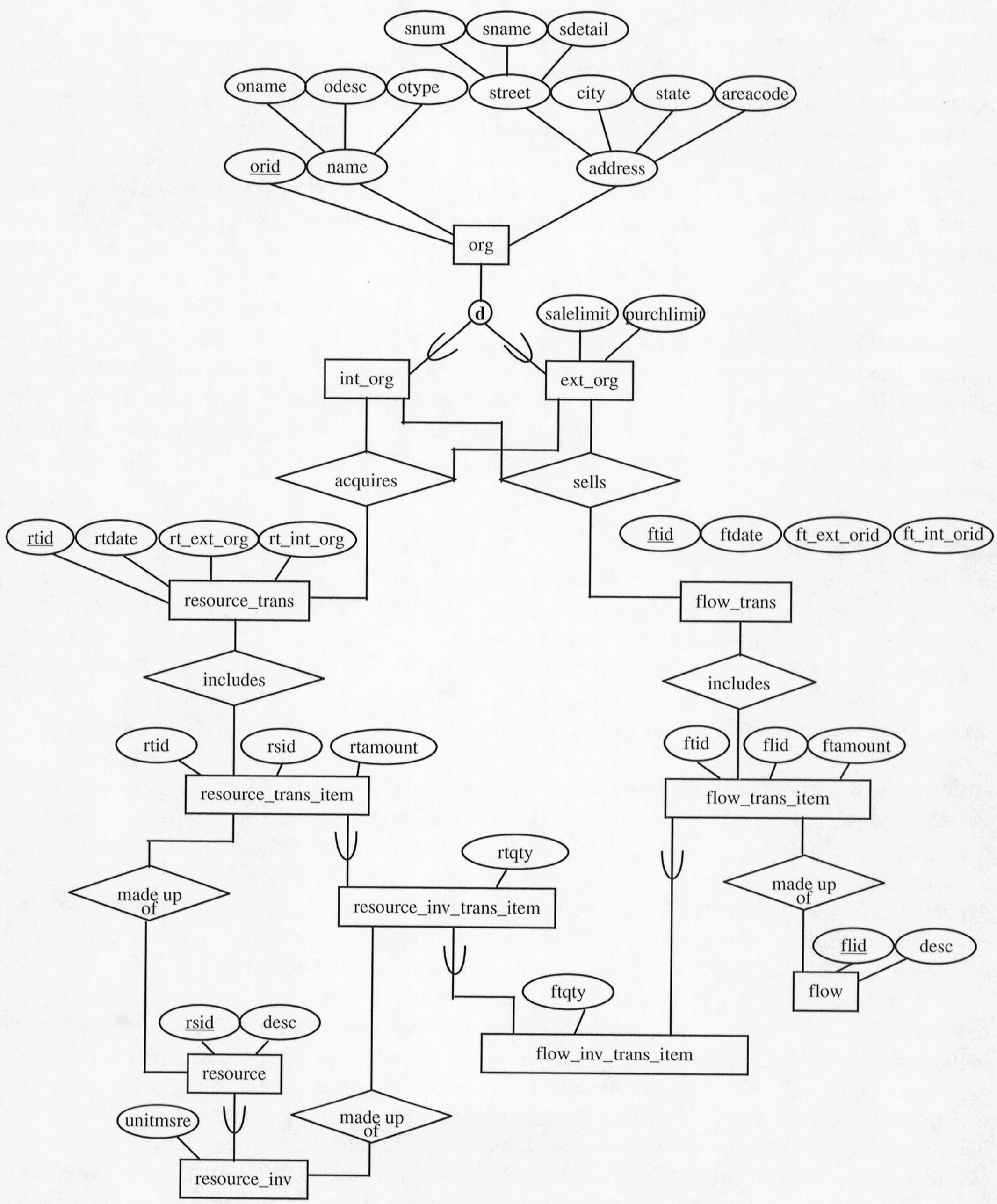
We used a laboratory experiment to test the hypotheses. This section describes the steps taken to design the alternative object and relational schemas, the creation of a set of information requests, selection of participants, the management of the experiment, and the dependent and independent variables.

Object and Relational Database Schema Design

To compare the competing paradigms, we created object and relational instantiations of a small-scale database accounting system. The database accounting system models the inventory acquisition and sales cycle for a hypothetical retailer of computer equipment. Figure 3 shows the Extended Entity-Relation (EER) semantic model of the database accounting system.

Using this EER model, we then constructed object and relational schemas, each in third normal form (3NF). The object schema, shown in Appendix A, incorporates both generalization-specialization hierarchies and abstract data types. We employ GSHs to represent hierarchies of accounting resources and claims (assets and liabilities) and flows (revenues and expenses). These GSH structures are (1) org with specializations ext_org and int_org; (2) resource with specialization resource_inv; (3) resource_trans_item

FIGURE 3
Modified Extended Entity-Relationship Model for Hypothetical Company



with specialization resource_inv_trans_item; and (4) flow_trans_item with specialization flow_inv_trans_item. We employ two relatively simple ADTs in the object schema to represent organizational names (oname_t) and addresses (address_t). These ADTs are reused in org and, by extension, ext_org and int_org.

Semantic constructs such as the generalization-specialization hierarchies in the EER model in Figure 3 were not directly available when designing the traditional relational schema. This necessitated making several design choices in modeling a semantically equivalent schema. We followed the rules established by Elmasri and Navathe (2003) to translate the generalization-specialization hierarchies in the semantic model to the traditional relational schema shown in Appendix B. For example, in the object schema the GSH structure of resource_trans_item and its specialization resource_inv_trans_item represent inventory resource transactions. This structure is implemented by the tables resource_non_inv_trans_item and resource_inv_trans_item to accommodate the bifurcation of resource transactions into inventory and non-inventory.

Although the object and relational schemas represent the same database accounting system, the two competing schemas have different structural characteristics. The object schema has seven general tables and five tables that specialize those general tables. The object schema has 21 distinct attributes. In contrast, the relational schema consists of ten tables and 30 attributes. Hence, the object schema contains two more tables but nine fewer attributes.

Information Requests

Participants in the experiment formulated queries for information requests that are typical of those required of entry-level accountants. The design criteria of the ten information requests were (1) to provide a range of tasks from simple to difficult across both schemas; (2) to focus on differences associated with GSHs and ADTs, and (3) to test the participants' understanding of the database accounting information system. Table 1 shows each of the information requests along with the most efficient object and relational solutions.

Independent Variables—Measuring Complexity

We measured the effect of the object and relational schemas on end-user query performance by observing the effects on accuracy and efficiency of the complexity associated with the most efficient (model) object and relational queries. To assess the effects of the experimental manipulations, we divided the complexity in each query into complexity that arises in the object schema and the differential complexity that arises in the traditional relational model resulting from the absence of object-oriented features.

The complexity of the model object and relational queries to solve the information requests was measured by a program complexity metric. There are more than 100 such metrics (Zuse 1991). Although no one software complexity metric can meet all desired properties (e.g., reliability, comprehensibility, correctness, understandability, ease of implementation) (Fenton 1994; Weyuker 1988), the Halstead (1977) complexity metrics have been shown to be reliable and highly correlated to other well-established complexity metrics (Banker et al. 1993; Lind and Vairavan 1989). The Halstead complexity metrics have been employed in a number of studies on human-computer interaction (e.g., Bowen et al. 2003). We employ the Halstead Difficulty (D) complexity metric to measure complexity of the

TABLE 1
Experimental Queries, Solutions, and Halstead Difficulty Complexity Values

#	Information Request	Object Solution	Relational Solution	*OC*	*GSHC*	*ADTC*
1	Print out a list of the names and address details of all of our customers and suppliers only.	select name, address from ext_org;	select oname, odscrptn, otype, snum, sname, sdetail, city, state, areacode from ext_org;	1.7	4.5	5.1
2	Print out the name and dollar limit of those customers for which a positive purchase limit has been set.	select name,salelimit from ext_org where name.otype='Cust' and salelimit > 0;	select oname,salelimit from org,ext_org where org.orid=ext_org.orid and otype='Cust' and salelimit > 0;	4.6	1.4	1.8
3	Print the full name and address of organizations that are customers or suppliers of the Dennis Company NOT resident in [state].	select name,address from ext_org where name.otype='Cust' and address.state <> '[state]';	select oname, odscrptn, otype, snum, sname, sdetail, city, state, areacode from org, ext_org where org.orid=ext_org.orid and otype='Cust' and state<> '[state]';	4.6	4.7	4.9
4	Print out the name of organizations that are neither departments nor customers or suppliers.	select name from only (org);	select distinct(oname, odscrptn, otype) from org where orid not in (select org.orid from org, int_org,ext_org where org.orid=int_org.orid or org.orid=ext_org.orid);	1.7	11.6	1.5
5	Provide a listing of the descriptions of the distinct resources that have been used in the various resource transactions for all the transactions in the database.	select distinct dscrptn from resource, resource_trans_item where resource_trans_item.rsid = resource.rsid;	select distinct dscrptn from resource, resource_inv_trans_item, resource_non_inv_trans_item where resource.rsid=resource_inv_trans_item.rsid or resource.rsid=resource_non_inv_trans_item.rsid;	4.9	3.6	0.0

(*continued on next page*)

TABLE 1 (continued)

#	Information Request	Object Solution	Relational Solution	*OC*	*GSHC*	*ADTC*
6	What was the total revenue for the period from 12 January to 15 January, inclusive?	select sum(ftamount) from flow_trans_item,flow_trans where ftdate between 'XXXX-01-12' and "XXXX-01-15' and flow_trans_item.ftid=flow_trans.ftid;	select sum(*) from (select sum(ftamount) from flow_inv_trans_item,flow_trans where flow_trans.ftid=flow_inv_trans_item.ftid and ftdate between 'XXXX-01-12' and 'XXXX-01-15' union select sum(ftamount) from flow_non_inv_trans_item,flow_trans where flow_trans.ftid=flow_non_inv_trans_item.ftid and ftdate between 'XXXX-01-12' and 'XXXX-01-15');	9.2	12.0	0.0
7	What were the total sales of inventory and provision of services to the "East-West Community College" during the period from 2 to 15 February inclusive?	select sum(ftamount) from flow_trans_item, flow_trans, org where org.orid=flow_trans.ft_ext_org and name.oname='East-West Community College' and flow_trans.ftid=flow_trans_item.ftid and ftdate between 'XXXX-02-02" and 'XXXX-02-15';	select sum(a.ftamount) from (select ftamount,ftid from flow_inv_trans_item union all select ftamount,ftid from flow_non_inv_trans_item) a, flow_trans,org where org.oname='East-West Community College' and org.orid=flow_trans.ft_ext_org and a.ftid=flow_trans.ftid and ftdate between 'XXXX-02-02' and 'XXXX-02-15';	16.4	6.8	0.0
8	Provide a list of the total resources brought IN to the company by each department for the month of February?	select sum(rtamount),orid,name.oname from resource_trans_item,resource_trans,int_org where int_org.orid=resource_trans.rt_int_org and resource_trans.rtid=resource_trans_item.rtid and rtdate between 'XXXX-02-01' and 'XXXX-02-28' and rtamount>0 group by orid,name.oname;	select sum(a.rtamount),a.orid,a.oname from (select rtamount,orid,oname from resource_trans,resource_non_inv_trans_item,org where org.orid=resource_trans.rt_int_org and resource_trans.rtid=resource_non_inv_trans_item.rtid and rtdate between 'XXXX-02-01' and 'XXXX-02-28' and rtamount>0 union all select rtamount,orid,oname from resource_trans,resource_inv_trans_item,org where org.orid=resource_trans.rt_int_org and resource_trans.rtid=resource_inv_trans_item.rtid and rtdate between 'XXXX-02-01' and 'XXXX-02-28' and rtamount>0) a group by a.orid,a.oname;	15.2	28.2	1.4

(*continued on next page*)

TABLE 1 (continued)

9	What was the net profit for the period from 1 to 15 January?	select sum(*) from (select sum(ftamount) from flow_trans_item,flow_trans where flow_trans_item.ftid=flow_trans.ftid and ftdate between 'XXXX-01-01' and 'XXXX-01-15' union select sum(rtamount) from flow_inv_trans_item,flow_trans where flow_inv_trans_item.rtid=flow_trans.ftid and ftdate between 'XXXX-01-01' and 'XXXX-01-15');	select sum(*) from (select sum(ftamount) from flow_inv_trans_item,flow_trans where flow_inv_trans_item.ftid=flow_trans.ftid and ftdate between 'XXXX-01-01' and 'XXXX-01-15' union select sum(ftamount) from flow_non_inv_trans_item,flow_trans where flow_non_inv_trans_item.ftid=flow_trans.ftid and ftdate between 'XXXX-01-01' and 'XXXX-01-15' union select sum(rtamount) from resource_inv_trans_item,resource_trans where resource_inv_trans_item.rtid=resource_trans.rtid and rtamount < 0 and rtdate between 'XXXX-01-01' and 'XXXX-01-15');	21.8	13.4	0.0
10	By how much did the amount of inventory that we acquired during the period from 1 January to 15 January from the supplier called "Mega PC" exceed the amount that we paid to them in cash?	select sum(rtamount) from resource_trans,resource_trans_item,ext_org where name.oname='Mega PC' and resource_trans.rt_ext_org=ext_org.orid and resource_trans.rtid=resource_trans_item.rtid and rtdate between 'XXXX-01-01' and 'XXXX-01-15';	select sum(a.rtamount) from (select rtamount,rtid from resource_inv_trans_item union all select rtamount,rtid from resource_non_inv_trans_item) a, resource_trans,org where org.oname='Mega PC' and org.orid=resource_trans.rt_ext_org and a.rtid=resource_trans.rtid and rtdate between 'XXXX-01-01' and 'XXXX-01-15';	17.0	5.9	0.0

model solutions to each information request.[2] The calculation of the Halstead Difficulty (D) complexity metric is described in Appendix C.

We take the level of the complexity of the object query (Object Complexity [*OC*]) to represent the underlying complexity of each information request. The difference between the complexity of the object query and the complexity of the relational query is the Marginal Relational Complexity (*MRC*). *MRC* is decomposed into the effect of added complexity in the relational schema arising from the absence of abstract data types (ADTs) and from the absence of generalization/specialization hierarchies (GSHs). ADT Complexity (*ADTC*) is that part of *MRC* that arises from the absence of ADTs in the relational schema. GSH Complexity (*GSHC*) measures the remaining component of *MRC* attributable to the absence of generalization-specialization hierarchies in the relational schema. Equations (1)–(6) define these relationships:

OC_i = Difficulty of the model OO query for information request i ($i = 1$ to 10); (1)

RC_i = Difficulty of the model traditional relational query for information request i ($i = 1$ to 10); (2)

$MRC_i = RC_i - OC_i$; (3)

$OCGSH_i$ = Difficulty of an OO query for information request i using only generalization-specialization hierarchy functionality (i.e., using no ADTs) i ($i = 1$ to 10); (4)

$ADTC_i = OCGSH_i - OC_i$ ($i = 1$ to 10); and (5)

$GSHC_i = MRC_i - ADTC_i$ ($i = 1$ to 10). (6)

GSHC is always greater than or equal to zero because queries on a schema that employs GSHs are either identical to queries on an equivalent schema under the traditional relational model or reduce the number of tables to be joined, thereby reducing query complexity. *ADTC* is always greater than or equal to zero because queries on a schema that employs ADTs are either identical to queries on an equivalent schema under the traditional relational model or retrieve a collection of attributes, thereby reducing query complexity. The model queries for each information request using the traditional relational and the object schema shown in Table 1 confirm these relationships.

For example, the third information request asks users to retrieve name and address information for specified classes of external organizations. The most efficient solution for this information request made under an object schema that employs both GSHCs and ADTs is:

```
select name,address
from ext_org where name.otype='Cust' and address.state <>
'[state]';
```

Here the user extracts the information directly from the child table "ext_org" that specializes the parent "org" table. This is the most efficient of the three queries (object, object without ADT and traditional relational) ($OC = 4.6$). If the ADT "address_t" were not available in the schema, then the user would need to set out the full elements of the name and address attributes subsumed in the ADT. The resulting query would be:

[2] In this study the pair-wise correlation of the Halstead Difficulty and Lines of Code complexity metrics for the object queries shown in Table 1 was 0.880 ($p < .001$). The corresponding correlation for the relational queries was 0.843 ($p < 0.01$). We used the Lines of Code counting rules for query languages developed by Chan (1999).

```
select oname, odscrptn, otype, snum, sname, sdetail, city, state,
areacode
from ext_org where otype='Cust' and state <> '[state]';
```

The loss of the ADT increased the complexity of the query (*ADTC* = 4.9). In the traditional relational schema, neither GSHs nor ADTs are available. The user must spell out the name and address attributes and additionally join the parent (`org`) and child (`ext_org`) tables.

```
select oname, odscrptn, otype, snum, sname, sdetail, city, state,
areacode
from org, ext_org
where org.orid=ext_org.orid
and otype='Cust'
and state<> '[state]';
```

This query was the most complex, as a result of the additional table and the required join resulting from the absence of the GSH (*GSHC* = 4.7).

Dependent Variables

Each set of hypotheses examines two dependent variables: accuracy, defined as the number of semantic errors made by the participants (*ERRORS*), and efficiency, defined as the number of minutes spent on each information request (*TIME*). The unit of analysis was each participant's final attempt on each of their completed information requests.

When end-users interact with a query processor, they may make either syntactic or semantic errors. Although analyzing syntactic errors is of interest relative to end-user query efficiency, the DBMS detects and reports those errors. Conversely, the query processor does not automatically detect semantic errors. Because end-users may base decisions on output generated by queries containing semantic errors, the level of these errors is the best measure of how well each end-user interpreted the information request. Direct measurement of semantic errors made by end-users in their interaction with an information system is a well accepted method in research on human-computer interaction in general and on the effects of levels of abstraction in particular (Jih et al. 1989; Reisner 1977, 1981; Smelcer 1995; Wu et al. 1994).

Two researchers determined and recorded the semantic errors made on each participant's final attempt for each information request (*ERRORS*). The researchers employed a set of nine coding rules to determine the nature and class of the errors. The errors were categorized as affecting an element of the SQL language, including the SELECT, FROM, WHERE JOIN, WHERE CONDITION, GROUP BY, and HAVING clauses. For example, the final query of one participant in response to the second information request (see Table 1) was:

```
select name,salelimit
from ext_org
where org.purchlimit < 0;
```

As compared with the most efficient (model) solution shown in Table 1, the participant: (1) failed to include '`name.otype='Cust`'' thereby generating a missing WHERE CONDITION, (2) used '`purchlimit`' instead of '`salelimit`' employing a misspecified table in the WHERE CONDITION and (3) used '<' instead of '>' employing a misspecified operator in the WHERE CONDITION. After coding each query, the researchers compared their results and resolved any differences thereby ensuring complete inter-rater reliability. The time-stamps recorded by the automated computer logs were the source of time taken for each information request (*TIME*).

Model

To test the hypotheses we analyzed the effects of *GSHC* and *ADTC* on the two metrics for end-user performance, i.e., *ERRORS* and *TIME* (see Equation (7)). The values of *GSHC* and *ADTC* are zero for each of the ten object queries, but take on the values shown in Table 1 for the ten relational queries. In other words, the added complexity of the relational queries is decomposed into those elements arising from the absence in the relational schema of generalization-specialization hierarchies and abstract data types. The model includes *OC* as a covariate.

$$ERRORS \text{ or } TIME = \beta_0 + \beta_1(GSHC) + \beta_2(ADTC) + \beta_3(OC) + \varepsilon. \tag{7}$$

Participants

Participants were 59 students in an advanced accounting information systems (AIS) course at a major research university. The pre-requisites for the subject included basic accounting subjects and an introductory information systems or equivalent computer science subject. The AIS course strongly emphasized database design and the relational model. In addition, the course covered material on control systems and accounting cycles. All participants received extensive training in SQL including over 2.5 hours of instruction over several in-class lectures. Furthermore, the participants had completed two two-hour SQL laboratory quizzes before attempting this experiment. The participants each received $20.00 for taking part in the experiment.

An information systems expert ranked the participants according to their information systems experience, education, and GPA. These rankings were used to create two equivalent groups, i.e., the highest ranked student was assigned to group A, the next highest to group B, followed by B, A, A, B, B, etc. A coin toss determined the assignment of the two groups to either the OO or relational group.

The primary requirement for choice of participants in this experiment was twofold. First, participants were required to have an understanding of database principles and, in particular, practical knowledge of SQL. Second, participants had to have knowledge of fundamental accounting relationships. The probability of finding a sufficient number of accessible individuals with such a combination of skills in the business or professional realms is low. Further, as the queries did not require participants to exercise judgment requiring significant experience in the accounting profession or commercial activities, students were appropriate surrogates.

Experimental Task

For two hours, the participants interactively queried either the object schema or the relational schema and performed the same information retrieval tasks based on the same application domain. At the beginning of the experimental session, participants received a brief description of the hypothetical company, an EER diagram of the database schema, a text specification of the tables and the attributes in each table, and the same ten information requests. The materials also included brief reminders about EER graphical techniques, database accounting systems, and SQL query syntax. The participants' interaction with the database was logged both at the workstation and at the server.[3]

The interface exhibited high realism in that participants received actual query results from the system, e.g., syntax errors, number of records found, and the content of those records.[4] Furthermore, participants could revise and resubmit their queries as many times as they wished.

IV. RESULTS

Demographic Influences

The participants in the study had an average (standard deviation) of 5.4 (27.6) months of relevant work experience. The participants had completed an average (standard deviation) of 2.2 (0.8) computing subjects prior to undertaking this course. There was no statistical difference between the treatment groups on these demographic variables or on other demographic variables such as GPA, age, or gender.

Descriptive Statistics

As there was a fixed time for the experiment, the number of queries attempted by participants varied. The object and relational group completed, on average, the same number of queries (6.8). The dispersion for the object group (SD = 2.2) was higher than that of the relational group (SD = 1.4). Panel A of Table 2 shows the experimental results for semantic errors (*ERRORS*) and time taken (*TIME*). As expected, the object group made fewer errors (42 percent fewer) and took less time per question (7 percent less time) than the traditional relational group.

Panel B of Table 2 breaks down these overall results to show the mean and standard deviation of the *ERRORS* and *TIME* for each of the information requests. The object group had lower errors on eight of the ten information requests.[5] In general, as the information requests became more complex, the relational group took increasingly more time than the object group in completing each information request. Panel C of Table 2 lists the mean and standard deviation of the semantic errors by SQL clause. The object group made significantly fewer errors on six of eight clauses.

Test of Hypotheses

We tested the model (Equation (7)) set out in the previous section with OLS linear regression. Panel A of Table 3 shows the descriptive statistics for the dependent and independent variables. We hypothesized that employing GSHs in the object database schema would improve subjects' query accuracy (H1a) and efficiency (H1b). Panel B of Table 3

---

[3] Both object and relational groups interacted with the Illustra object-relational database management system (Stonebraker et al. 1999).

[4] Many previous tests of query performance used pencil and paper or simulations that did not provide actual query results to participants (see, e.g., Chan et al. 1993; Jih et al. 1989; Rho and March 1997; Suh and Jenkins 1992).

[5] Similarly, the means were significantly different for eight of the ten information requests, as measured by two-tailed t-tests.

TABLE 2
Descriptive Statistics

Panel A: Test of Differences of Means

Dependent Variable	Object n	Object Mean	Object Std. Deviation	Relational n	Relational Mean	Relational Std. Deviation	p[a]
ERRORS	213	3.45	4.13	191	5.93	4.85	.000
TIME	199	8.84	7.18	184	9.54	6.94	.334

Panel B: Errors and Time by Information Request

Info. Request	Stat.	Object *ERRORS*	Object *TIME*	Object n	Relational *ERRORS*	Relational *TIME*	Relational n	Errors t-test p[a]	Time t-test p[a]
1	Mean	0.59	6.86	29	2.20	5.76	25	0.000	0.538
	SD	0.91	7.96		1.32	4.23			
2	Mean	2.07	4.10	29	1.71	4.04	28	0.374	0.969
	SD	1.49	3.32		1.33	2.10			
3	Mean	1.07	9.20	30	2.71	8.42	28	0.000	0.673
	SD	1.01	7.53		2.12	5.93			
4	Mean	2.92	12.21	25	6.36	11.19	28	0.000	0.694
	SD	2.16	10.88		2.47	7.41			
5	Mean	3.13	9.05	23	4.35	10.04	23	0.020	0.522
	SD	1.66	3.97		1.77	5.97			
6	Mean	3.11	8.92	27	9.26	14.92	27	0.000	0.003
	SD	2.91	5.75		3.24	7.69			
7	Mean	5.82	11.84	22	11.94	9.40	16	0.000	0.251
	SD	4.49	7.10		3.60	4.32			
8	Mean	8.13	10.69	15	15.08	14.92	12	0.005	0.123
	SD	6.72	3.75		4.58	8.71			
9	Mean	11.09	11.29	11	14.67	17.00	3	0.290	0.332
	SD	5.34	6.18		2.31	9.90			
10	Mean	12.00	8.00	3	9.00	9.00	1	NA	NA
	SD	7.07	NA		NA	NA			

Panel C: Semantic Errors by SQL Clause

SQL Clause	Object Mean	Object Std. Deviation	Relational Mean	Relational Std. Deviation	t-test p[a]
Select errors	0.73	1.07	1.32	1.59	0.000
From errors	0.97	1.53	1.57	1.59	0.000
Where join errors	0.67	1.02	1.18	1.22	0.000
Logical operator errors	—	—	0.02	0.18	0.084
Where condition errors	0.85	1.15	1.10	1.33	0.040
Group by errors	0.12	0.64	0.15	0.40	0.581
Having errors	—	—	0.01	0.07	0.291
Minus errors	—	—	—	—	NA
View errors	0.06	0.23	0.24	0.45	0.000
Union errors	0.04	0.20	0.28	0.50	0.000
Order by errors	0.01	0.15	0.10	0.45	0.010
Total	3.45	4.13	5.97	4.85	0.000

[a] Significance of two-tailed t-test.

TABLE 3
Regression Results

Panel A: Descriptive Statistics—Dependent and Independent Variables

	Mean	Std. Deviation	Min.	Max.
ERRORS	4.619	4.645	0.00	25.00
TIME	9.183	7.061	1.00	49.00
GSHC	3.727	5.969	0.00	21.81
ADTC	0.930	1.709	0.00	5.19
OC	6.797	5.605	1.69	28.82

Panel B: The Effects of Generalization/Specialization Complexity (*GSHC*) and Abstract Data types Complexity (*ADTC*) on Semantic Errors and Time Taken

	ERRORS		*TIME*	
	Std. Beta Coefficient	p	Std. Beta Coefficient	p
GSHC	0.487	0.000	0.258	0.000
ADTC	−0.091	0.010	−0.156	0.004
OC	0.501	0.000	0.094	0.076
n	404		383	
Adj. R^2	0.587		0.096	

Dependent Variables
ERRORS = number of semantic errors; and
TIME = number of minutes taken to answer question.
Independent Variables
GSHC = higher query complexity in relational queries arising from absence of Generalization-Specialization Hierarchies (GSH).
ADTC = higher query complexity in relational queries arising from absence of Abstract Data Types (ADTs); and
OC = complexity of object query;

shows that the use of GSHs significantly reduced semantic errors, i.e., improved effectiveness and reduced time per information request, i.e., improved efficiency.[6]

We predicted that the presence of ADTs in the object schema would decrease semantic errors (H2a) and reduce the time used by participants (H2b) in completing their queries. Panel B of Table 3 shows that the results of this experiment did not support either of these hypotheses.[7]

[6] Additional tests were conducted on the results shown in Table 3, *viz.*: (1) there are no indications of multicollinearity (mean VIF = 1.18); (2) analysis of the residuals indicates that the OLS is robust; (2) *ERRORS* is measured in these regression analyses for the final attempt on each query. The reported results were confirmed when the regression was run with errors made on all attempts on each query; (4) when separate regressions that individually analyze the effect of *GSHC* and *ADTC* were run, the effect of *GSHC* was almost identical to that shown in Table 3 but the effect of *ADTC* was not significant; (5) an analysis of the relationship between the dependent variables and *MRC* (where *MRC* = *GSHC* + *ADTC*) showed significant and positive relationship with both errors and time, and (6) a repeated measures GLM analysis undertaken on a subset of the experimental results also confirmed the results.

[7] Essentially identical results were observed when the regression was run only for those information requests that used ADTs.

Manipulation Checks

This study tested the manipulation of the complexity of the schemas by obtaining the experimental participants' views on particular aspects of the schemas, including the number of tables and the ease of use in querying the schemas. After reading the instructions but before completing the experimental tasks, participants completed a pre-test questionnaire. After completing the experiment, the participants completed a post-test questionnaire. The opinions of the participants about the quality of their schema, the number of tables, and the ease of querying provide a manipulation check on the effectiveness of the alternative technologies. Table 4 reports the means and standard deviations of the questions in the post-experimental questionnaire.

Table 4 does not reveal a statistically significant difference between the experimental groups for any of the post-test questions, i.e., the manipulation checks do not provide evidence that the participants perceived that the relational and OO schemas were significantly different. Operationalizing concepts such as performance and complexity is difficult. The post-test questions can provide only an indirect check of the manipulation of schema complexity. Furthermore, because they only saw the schema to which they had been randomly assigned, experimental participants had little context by which to judge if the schemas they used were complex, well-designed, or had too many or too few tables.

Limitations

The study is subject to a number of limitations. First, in common with a number of other similar studies in human computer interaction and chunking, we did not directly measure the effect of chunk size or chunk schema structure on the human cognitive processes at work during the subjects' completion of the experimental tasks. To do so, we would have had to employ a procedure such as verbal protocol analysis (Ericsson and Simon 1980, 1993) to assess the manner by which the subjects interacted with the query language and database schema. Indeed, undertaking such a procedure would be appropriate research to follow this study. Second, the subjects may not be representative of end-users, as they are likely to have acquired higher skills in database technology in general and in the SQL language in particular than is typical of end-users. This limits the generalizability of the results. Third, the experimental test is only as good as the design of the two schemas. To allow direct comparison between the two database schemas, the relational schema must reflect each feature in the object schema.

Furthermore, the implementation of the object features of generalization/specialization hierarchies and abstract data types must be sufficiently straightforward for the experimental participants to understand and be able to use the schema productively in a relatively short time. Simplifications imposed by constraints such as these necessarily limit the external validity of the experiment.

V. DISCUSSION AND IMPLICATIONS

This study has several important implications. First, it provides clear evidence that increasing abstraction by incorporating the key abstraction concept of generalization/specialization hierarchies (GSHs) in database accounting schema design reduced the complexity of system end-user queries. The reduced complexity leads end-users to make fewer semantic errors and enhances efficiency. This is contrary to the results found by Dunn (1999), but is in accord with her predictions and with the results of a broad set of studies in the Human-Computer Interaction (HCI) research domain.

Contrary to our predictions, the classification/instantiation abstraction concept, implemented as ADTs, did not result in more effective or efficient end-user querying patterns.

TABLE 4
Participant Views on Aspects of the Experimental Task

Aspect	Object Mean	Object Std. Dev.	Relational Mean	Relational Std. Dev.	t-test p[a]
Efficiency of querying the database with select statements 1 = Extremely inefficient, 7 = Extremely efficient	4.200	1.096	4.036	1.551	0.433
Effectiveness of querying the database with select statements 1 = Extremely ineffective, 7 = Extremely effective	4.185	0.833	4.240	1.300	0.856
Difficulty of constructing select statements 1 = Extremely difficult, 7 = Extremely easy	3.690	1.391	3.678	1.090	0.974
Frustration of querying the database with select statements 1 = Very frustrating, 7 = Not at all frustrating	3.241	1.023	3.393	1.423	0.645
Number of tables in the database 1 = Too few, 7 = Too many	4.931	1.066	5.035	0.922	0.694
Overall quality of the database schema 1 = Very poorly designed, 7 = Very well designed	4.107	1.257	4.071	1.438	0.922

[a] Two-tailed t-test.

Recall that, prior to the experiment, the exclusive focus of the participants' training was on the traditional relational model. The lack of prior exposure to ADTs is likely to have produced the lack of positive effects of ADTs observed in this experiment. Additional research will be necessary to investigate the benefits arising from the implementation of the classification/instantiation abstraction concept in database accounting schema design and on end-user query performance.

Second, the design of database accounting systems should facilitate improved mental mappings. The evidence from this research suggests that providing GSHs in the schema did provide a better mapping that resulted in enhanced query effectiveness and efficiency. The picture for ADTs is not as clear. Developing a clearer understanding of the role of ADTs will require more research, perhaps with ADTs that are more central to the accounting domain. Understanding the way in which end-users bring together their knowledge of data structures, query languages, and accounting and business ontological structures will require research that looks inside the black box of human decision making. Making a formal assessment of the number, size, and inter-relationships of the chunks manipulated by the end-users as they interact with alternative schemas will be an important item on the research agenda that flows from this study. Further, as noted in the "Limitations" subsection, employing verbal protocol analysis would seem to be a desirable component of such a research agenda. Employing this technique would allow us to measure directly the effect of abstraction on information retrieval from database accounting systems, be they organized under object or traditional relational paradigms. Of necessity, such studies would involve relatively small numbers of subjects, but would still provide important insights into end-user interaction with database accounting systems.

Third, given the results of this study, research is needed on employing other types of abstraction in database accounting systems, e.g., classification/instantiation and aggregation/decomposition (Taivalsaari 1996). Employing these abstractions may push the boundaries of current object and object-relational database accounting systems. Careful theoretical and practical assessment of the costs and benefits of these abstractions will be necessary.

VI. SUMMARY AND CONCLUSION

Over the last two decades, research into database accounting systems focused on the application of semantic modeling techniques to core enterprise business processes and the implementation of the resulting semantic models in database management systems. These database accounting systems employ a variety of abstraction mechanisms to increase semantic expressiveness, enhance database accounting schema design, and empower system end-users. Research in cognition and learning, including chunking and category theories, provides support for the positive effects of increasing abstraction on human problem solving. However, there has been only limited empirical testing of the effect of increasing abstraction levels on the performance of end-users' interaction with database accounting systems.

At the same time, there has been an ongoing debate on the ability of databases to employ abstraction mechanisms. The dominant database paradigm is the relational model. This class of databases draws on a strong theoretical foundation in set theory and, over the years, has generated significant implementation improvements in areas such as performance and security. An alternative perspective comes from object-orientation (OO). The proponents of this paradigm point particularly to the direct support in object databases for key abstraction mechanisms.

This paper reports the results of an experiment that tested the effects of including two abstraction mechanisms in database accounting systems. We tested the inclusion of the aggregation/decomposition abstraction with generalization/specialization hierarchies (GSHs) and the grouping/individualization via abstract data types (ADTs). As hypothesized, we found that employing GSHs improved end-user performance as measured by the number of semantic errors made and time taken. Contrary to our predictions, employing ADTs did not improve end-user performance in the experimental setting described in this paper.

A considerable number of interesting research questions flow from this study. Further research will help us better understand the application of abstraction mechanisms to database accounting systems. In addition to the opportunities identified by Dunn and McCarthy (1997), Dunn and Grabski (2002), and David et al. (2002), researchers and practitioners are likely to gain benefits from research that investigates enhancements to modeling techniques, relational, object-relational and "pure" object database systems, and more effective query construction using these systems. One possible research project would be to examine the relationship between modeling business processes and employing ADTs in accounting information systems schema design. Another research task could seek to increase our understanding of and enhance end-users' abilities to assess correctly the accuracy of their queries as they interact with accounting information systems.

A further strand of research that flows from this study is the effect of alternate query interfaces and languages, more sophisticated user-defined data types, and the application of methods and views. For example, there is evidence that query languages that use higher levels of semantic representation (Chan et al. 1994; Chan 1995) or offer graphical interfaces, e.g., QBE (Yen and Scamell 1993), are more productive than SQL. What query interface and language characteristics would best facilitate end-user queries of database accounting systems? Abstract data types can include temporal, text, graphics, and multimedia datatypes. When and how should database accounting systems incorporate these ADTs? The role of views in relational, object, and object-relational data models is an important research question (Kim and Kelley 1995; Kotz-Dittrich and Dittrich 1995; Kung 1990). Because views can dramatically alter the complexity of end-user queries, such research is likely to provide substantial benefits to designers of database accounting systems.

APPENDIX A
Object Schema

Abstract Data Types

oname_t

Attribute	Data Type
oname	char(30)
odscrptn	char(30)
otype	char(8)

address_t

Attribute	Data Type
snum	char(5)
sname	char(30)
sdetail	char(20)
city	char(30)
state	char(30)
zipcode	char(10)

Tables[8]

org

Attribute	Data Type
orid	char(6) not null primary key
name	oname_t
address	address_t

ext_org

Attribute	Data Type
orid	*char(6) not null primary key*
name	*oname_t*
address	*address_t*
purchlimit	dollar
salelimit	dollar
under org	

int_org

Attribute	Data Type
orid	*char(6) not null primary key*

[8] Rows that have been inherited from parent tables are shown in italics.

name	*oname_t*
address	*address_t*
under org	

resource

Attribute	Data Type
rsid	char(6) not null primary key
dscrptn	char(30) not null

resource_inv

Attribute	Data Type
rsid	*char(6) not null primary key*
dscrptn	*char(30) not null*
unitmsre	char(8) not null
under resource	

resource_trans

Attribute	Data Type
rtid	char(6) not null primary key
rtdate	date
rt_ext_org	char(6) references ext_org
rt_int_org	char(6) references int_org

resource_trans_item

Attribute	Data Type
rtid	char(6) references resource_trans
rtamount	dollar
rsid	char(6) references resource

resource_inv_trans_item

Attribute	Data Type
rtid	*char(6) references resource_trans*
rtamount	*dollar*
rtqty	integer
under resource_trans_item	

flow

Attribute	Data Type
flid	char(6) not null primary key
dscrptn	char(30) not null

flow_trans

Attribute	Data Type
ftid	char(6) not null primary key
ftdate	date
ft_ext_org	char(6) references ext_org
ft_int_org	char(6) references int_org

flow_trans_item

Attribute	Data Type
ftid	char(6) references flow_trans
ftamount	dollar
flid	char(6) references flow

flow_inv_trans_item

Attribute	Data Type
ftid	*char(6) references flow_trans*
ftamount	*dollar*
flid	*char(6) references flow*
ftqty	integer
rtid	*char(6) references resource_trans*
rtamount	*dollar*
rtqty	*integer*

under resource_inv_trans_item, flow_trans_item

APPENDIX B
Relational Schema

org

Attribute	Data Type
orid	char(6) not null primary key
oname	char(30)
odscrptn	char(30)
otype	char(8)
snum	char(5)
sname	char(30)
sdetail	char(20)
city	char(30)
state	char(30)
zipcode	char(10)

ext_org

Attribute	Data Type
orid	char(6) not null references org
buylimit	numeric(12,2)
salelimit	numeric(12,2)

primary key (orid)

int_org

Attribute	Data Type
orid	char(6) not null references org

primary key (orid)

resource

Attribute	Data Type
rsid	char(6) not null primary key
dscrptn	char(30) not null
isinvflag	char(1)
unitomsre	char(8)

resource_trans

Attribute	Data Type
rtid	char(6) not null primary key
rtdate	date
rt_ext_org	char(6) references ext_org
rt_int_org	char(6) references int_org

resource_non_inv_trans_item

Attribute	Data Type
rtid	char(6) references resource_trans

rtamount	numeric(12,2)
rsid	char(6) references resource

resource_inv_trans_item

Attribute	Data Type
rtid	char(6) not null references resource_trans
rtamount	numeric(12,2)
rsid	char(6) references resource
rtqty	integer

flow

Attribute	Data Type
flid	char(6) not null primary key
dscrptn	char(30) not null

flow_trans

Attribute	Data Type
ftid	char(6) not null primary key
ftdate	date
ft_ext_org	char(6) references ext_org
ft_int_org	char(6) references int_org

flow_non_inv_trans_item

Attribute	Data Type
ftid	char(6) not null references flowtrans
ftamount	numeric(12,2)
flid	char(6) not null references flow

flow_inv_trans_item

Attribute	Data Type
ftid	char(6) not null references flow_trans
ftamount	numeric(12,2)
flid	char(6) not null references flow
ftqty	integer
rtid	char(6) not null references
resource_trans	

APPENDIX C
Halstead Difficulty (D) Complexity Metric

The Halstead Difficulty (D) complexity metric is based on four measures derived directly from the query:

n1 = the number of distinct operators (e.g. 'and' 'or' '>');
n2 = the number of distinct operands (e.g. 'dscrptn' 'resource_trans');
N1 = the total number of operators; and
N2 = the total number of operands.

From this are derived the following measures:

Program vocabulary	$n = n1 + n2$
Program length	$N = N1 + N2$
Program volume	$V = N \log_2 n$

Potential (minimum) program length	$\acute{\eta} = 5$
Potential volume	$V^* = \acute{\eta} \log_2 \acute{\eta}$
Program level	L
Difficulty	$D = 1/L = V/V^*$

For example, the Halstead Difficulty complexity metric for the object and relational solutions to the fifth information request, shown in Table 1, is calculated as follows:

Calculation of the Halstead Difficulty Complexity Metric

Item	**Notation**	**Definition**	**Relational**	**Object**
Distinct Operators	n1		9	8
Distinct Operands	n2		5	4
Total Operators	N1		14	9
Total Operands	N2		12	7
Program vocabulary	n	n1 + n2	14.0	12.0
Program length	N	N1 + N2	26.0	16.0
Program volume	V	$N \log_2 n$	99.0	57.4
Potential (minimum) program length	$\dot{\eta}$	5	5.0	5.0
Potential volume	V*	$\dot{\eta} \log_2 \dot{\eta}$	11.6	11.6
Program level	L	V*/V	0.1	0.2
Difficulty	D	1/L	8.5	4.9

APPENDIX D
Semantic Errors Counting Form

Subject:												
Question:												
Text and Server Feedback:												
Attempt		1	2	3	4	5	6	7	8	9	10	Total
Select	NSL											
	ECI											
	ECO											
	MCL											
	MSE											
	DTM											
	DTE											
	ADT											
From	NFM											
	ETL											
	MTL											
	IHY											
Where Join	LOJ											
	NJN											
	JAT											
	JOP											

Subject:												
Question:												
Text and Server Feedback:												
Attempt		1	2	3	4	5	6	7	8	9	10	Total
	JTB											
	EJN											
	MJN											
Where Condition	NJNC											
	JATC											
	JOPC											
	JTBC											
	EJNC											
	MJNC											
Logical Operator	LOP											
Group By	NGB											
	GAT											
	GOP											
	GTB											
	GEA											
	GMA											
	GLO											
Having	NHV											
	HAT											
	HOP											
	HTB											
	HEA											
	HMA											
	HLO											
Order By	ODR											
	NOR											
	OAM											
	OEX											
	WAO											
	WDN											
View	EV											
	MV											
Union	EU											
	MU											

REFERENCES

Adamson, I. L., and D. M. Dilts. 1995. Development of an accounting object model from accounting transactions. *Journal of Information Systems* 9 (1): 43–64.

Ahn, W. K. 1998. Why are different features central for natural kinds and artifacts? The role of causal status in determining feature centrality. *Cognition* 69 (2): 135–178.

Banker, R. D., S. M. Datar, C. F. Kemerer, and D. Zweig. 1993. Software complexity and maintenance costs. *Communications of the ACM* 36 (11): 81–94.

Bayer, R., M. Heller, and A. Reiser. 1980. Parallelism and recovery in database systems. *ACM Transactions on Database Systems* 5 (2): 139–156.

Bell, D. 1992. *Distributed Database Systems*. Reading, MA: Addison-Wesley.

Borthick, A. F., P. L. Bowen, S. T. Liew, and F. H. Rohde. 2001. The effects of normalization on end-user query errors: An experimental evaluation. *International Journal of Accounting Information Systems* 2 (4): 195–221.

Bouwman, M. J., and W. E. Bradley. 1997. Judgment and decision making, part II: Expertise, consensus and accuracy. In *Behavioral Accounting Research: Foundations and Frontiers*, edited by V. Arnold, and S. G. Sutton, 89–133. Sarasota, FL: American Accounting Association.

Bowen, P. L., and F. H. Rohde. 2002. Further evidence of the effects of normalization on end-user query errors: An experimental evaluation. *International Journal of Accounting Information Systems* 3 (4): 255–290.

———, C. B. Ferguson, T. H. Lehman, and F. H. Rohde. 2003. Cognitive style factors affecting database query performance. *International Journal of Accounting Information Systems* 4: 251–273.

———, F. H. Rohde, and J. Basford. 2004. Ex ante evaluations of alternate data structures for end-user queries: Theory and experimental test. *Journal of Database Management* 15 (4): 45–70.

Buneman, O. P., and E. K. Clemons. 1979. Efficiently monitoring relational databases. *ACM Transactions on Database Systems* 4 (3): 368–382.

Castano, S., M. Fugini, G. Martella, and P. Samarati. 1995. *Database Security*. New York, NY: ACM Press.

Cattell, R. 1994. *Object-Data Management: Object Oriented and Extended Relational Database Systems*. Revised edition. Reading, MA: Addison-Wesley.

Ceri, S., P. Fraternali, S. Paraboschi, and L. Tanca. 1994. Automatic generation of production rules for integrity maintenance. *ACM Transactions on Database Systems* 19 (3): 367–422.

Chan, H. C., K. K. Wei, and K. L. Siau. 1993. User-database interface: The effect of abstraction levels on query performance. *MIS Quarterly* 17 (4): 441–464.

———, ———, and ———. 1994. An empirical study on end-users' update performance for different abstraction levels. *International Journal of Human-Computer Studies* 41 (3): 309–328.

———. 1995. Naturalness of graphical queries based upon the entity relationship model. *Journal of Database Management* 6 (3): 3–13.

———. 1999. The relationship between user query accuracy and lines of code. *International Journal of Human-Computer Studies* 51: 851–864.

Chase, W. G., and H. A. Simon. 1973. Perception in chess. *Cognitive Psychology* 4: 55–81.

Chu, P. C. 1992. An object-oriented approach to modeling financial accounting systems. *Accounting, Management and Information Technology* 2 (1): 39–56.

Codd, E. F. 1970. A relational model of data for large shared data banks. *Communications of the ACM* 13 (6): 377–387.

———. 1990. *The Relational Model for Database Management*. Reading, MA: Addison-Wesley.

Cooper, B. L., H. J. Watson, B. H. Wixom, and D. L. Goodhue. 2000. Data warehousing supports corporate strategy at First American Corporation. *MIS Quarterly* 24 (4): 547–567.

Craik, F. I. M., and R. S. Lockhart. 1972. Levels of processing. A framework for memory research. *Journal of Verbal Learning and Verbal Behavior* 11: 671–684.

Date, C. J. 2004. *An Introduction to Database Systems*. 8th edition. Reading, MA: Addison-Wesley.

David, J. S., G. J. Gerard, and W. E. McCarthy. 2002. Design science: An REA perspective on the future of AIS. In *Research Accounting as an Information Systems Discipline*, edited by V. Arnold and S. G. Sutton. Sarasota, FL: American Accounting Association.

De Groot, A. D. 1978. *Thought and Choice in Chess*. 2nd edition. The Hague, The Netherlands: Mouton de Gruyter.

Dunn, C. L., and W. E. McCarthy. 1997. The REA accounting model: Intellectual heritage and prospects for progress. *Journal of Information Systems* 11 (1): 31–52.

———. 1999. An experimental investigation of an abstraction hierarchy as a financial database interface. Working paper, Florida State University, Tallahassee, FL.

———, and S. V. Grabski. 2000. Perceived semantic expressiveness of accounting systems and task accuracy effects. *International Journal of Accounting Information Systems* 1 (2): 79–87.

———, and ———. 2001. An investigation of localization as an element of cognitive fit in accounting model representations. *Decision Sciences* 32 (1): 55–94.

———, and ———. 2002. Empirical research in semantically modeled accounting systems. In *Research accounting as an information systems discipline*, edited by V. Arnold and S. G. Sutton. 157–180. Sarasota, FL: American Accounting Association.

Elmasri, R., and S. Navathe. 2003. *Fundamentals of Database Systems*. 3rd edition. Boston, MA: Pearson Addison Wesley.

Ericsson, K., and H. Simon. 1980. Verbal reports as data. *Psychological Review* (May): 215–251.

———, and ———. 1993. *Protocol Analysis: Verbal Reports as Data*. 2nd edition. Cambridge, MA: MIT Press.

Fenton, N. 1994. Software measurement: A necessary scientific basis. *IEEE Transactions on Software Engineering* 20 (3): 199–206.

Franaszek, P. A., J. T. Robinson, and A. Thomasian. 1992. Concurrency control for high contention environments. *ACM Transactions on Database Systems* 17 (2): 304–345.

Gobet, F., and H. A. Simon. 1996. Recall of random and distorted positions: Implications for the theory of expertise. *Memory and Cognition* 24: 493–503.

———. 1998. Expert memory: A comparison of four theories. *Cognition* 66: 115–152.

Graham, I. 2001. *Object-Oriented Methods: Principles and Practice*. 3rd edition. Harlow, U.K.: Addison-Wesley.

Halstead, M. 1977. *Elements of Software Science*. New York, NY: North Holland.

Hammer, M., and D. McLeod. 1981. Database description with SDM: A semantic database model. *ACM Transactions on Database System* 6 (3): 351–386.

Ijiri, Y. 1967. *The Foundations of Accounting Measurement*. Englewood Cliffs, NJ: Prentice Hall.

———. 1975. *Theory of Accounting Measurement*. Studies in Accounting Research No. 10. Sarasota, FL: American Accounting Association.

Jarke, M., J. Mylopoulos, J. Schmidt, and Y. Vassiliou. 1992. DAIDA: An environment for evolving information systems. *ACM Transactions on Information Systems* 10 (1): 1–50.

Jih, W. K., D. A. Bradbard, C. A. Snyder, and N. G. Thompson. 1989. The effects of relational and entity–relationship data models on query performance of end-users. *International Journal of Man–Machine Studies* 31: 257–267.

Jones, D. R., and M. M. Eining. 1996. Articulating accounting database queries: An analysis of actual and perceived effort. *Advances in Accounting Information Systems* 4: 157–180.

Kim, J., J. Hahn, and H. Hahn. 2000. How do we understand a system with (so) many diagrams? Cognitive integration processes in diagrammatic reasoning. *Information Systems Research* 11 (3): 284–303.

Kim, W., and W. Kelley. 1995. On view support in object-oriented database systems. In *Modern Database Systems: The Object Model, Interoperability and Beyond*, edited by W. Kim, 108–129. New York: ACM Press.

Kimball, R., and M. Ross. 2002. *The data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. 2nd edition. New York, NY: John Wiley and Sons.

Kotz-Dittrich, A., and K. R. Dittrich. 1995. Where object-oriented DBMSs should do better: A critique based on early experiences. In *Modern Database Systems: The Object Model, Interoperability and Beyond*, edited by W. Kim, 238–280. New York, NY: ACM Press.

Kung, C. 1990. Object subclass hierarchy in SQL: A simple approach. *Communications of the ACM* 33 (7): 117–128.

Lakoff, G. 1987. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind.* Chicago, IL: University of Chicago Press.

Langacker, R. W. 1999. *Foundations of Cognitive Grammar: Theoretical Prerequisites.* Palo Alto, CA: Stanford University Press.

Libby, R., and J. Luft. 1993. Determinants of judgment performance in accounting settings: Ability, knowledge, motivation, and environment. *Accounting, Organizations & Society* 18 (5): 425–450.

Lind, R. K., and K. Vairavan. 1989. An experimental investigation of software metrics and their relationship to software development effort. *IEEE Transactions on Software Engineering* 15 (5): 649–653.

Loomis, M. E. S. 1995. *Object Databases: The Essentials*. Reading, MA: Addison-Wesley.

Mattos, N. 1988. Abstraction concepts: The basis for knowledge modeling. In *Proceedings of the Seventh International Conference on the Entity-Relationship Approach*, edited by C. Batini, 331–350. Berlin, Germany: Springer-Verlag.

McCarthy, W. 1982. The REA accounting model: A generalized framework for accounting systems in a shared data environment. *The Accounting Review* 57 (3): 554–578.

Miller, G. A. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review* 63: 81–97.

Murphy, G. L., and D. L. Medin. 1985. The role of theories in conceptual coherence. *Psychological Review* 92: 289–316.

Mylopoulos, J. 1998. Information modeling in the time of the revolution. *Information Systems* 23 (2): 127–155.

Mynatt, E. D. 1997. Transforming graphical interfaces into auditory interfaces for blind users. *Human-Computer Interaction* 12: 7–45.

Newell, A., and H. A. Simon. 1972. *Human Problem Solving*. Englewood Cliffs, NJ: Prentice Hall.

Rehder, B. 2003. Categorization as causal reasoning. *Cognitive Science* 27: 709–748.

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

———. 1981. Human factors studies of database query languages: A survey and assessment. *ACM Computing Surveys* 13 (1): 13–31.

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

Rosch, E. H. 1973. Natural categories. *Cognitive Psychology* 4: 328–350.

Rose, J. M. 2002. Behavioral decision aid research: Decision aid use and effects. In *Research Accounting as an information Systems Discipline*, edited by V. Arnold and S. G. Sutton, 83–110. Sarasota, FL: American Accounting Association.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorenson. 1994. *Object-Oriented Modeling and Design*. 2nd edition. Englewood Cliffs, NJ: Prentice Hall.

Rumelhart, D. E., and A. Ortony. 1977. The representation of knowledge in memory. In *Schooling and the Acquisition of Knowledge*, edited by R. C. Anderson, R. J. Spiro, and W. E. Montague, 99–136. Hillsdale, NJ: Erlbaum.

———. 1994. Toward an interactive model of reading. In *Theoretical Models and Processes of Reading*, edited by R. Ruddell, M. Ruddell, and H. Singer, 864–894. Newark, DE: International Reading Association.

Simon, H. A. 1996. *The Sciences of the Artificial*. 3rd edition. Cambridge, MA: MIT Press.

Simpson, A., and C. McKnight. 1990. Navigation in hypertext: Structural cues and mental maps. In *Hypertext: State of the Art*, edited by R. McAleese, and C. Green, 73–83. Oxford, U.K.: Intellect.

Sloman, S. A. 1998. Categorical inference is not a tree: The myth of inheritance hierarchies. *Cognitive Psychology* 35 (1): 1–33.

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

Smith, J., and D. Smith. 1977. Database abstractions: Aggregation and generalization. *ACM Transactions on Database Systems* 2 (2): 105–133.

Speier, C., and M. G. Morris. 2003. The influence of query interface design on decision-making performance. *MIS Quarterly* 27 (3): 397–423.

Stonebraker, M. R., D. Moore, and P. Brown. 1999. *Object–Relational DBMSs: Tracking the Next Great Wave*. San Francisco, CA: Morgan Kaufmann.

Suh, K. S., and A. M. Jenkins. 1992. A comparison of linear keyword and restricted natural language database interfaces for novice users. *Information Systems Research* 3 (3): 252–272.

Taivalsaari, A. 1996. On the notion of inheritance. *ACM Computing Surveys* 28 (3): 438–479.

Teorey, T., D. Yang, and J. Fry. 1986. A logical design methodology for relational databases using the extended entity-relationship model. *ACM Computing Surveys* 18 (2): 197–222.

van Nimwegen, C., M. Pouw, and H. van Oostendorp. 1999. The influence of structure and reading-manipulation on usability of hypertexts. *Interacting with Computers* 12 (1): 7–21.

Wand, Y., and R. A. Weber. 2002. Research commentary: Information systems and conceptual modeling—A research agenda. *Information Systems Research* 13 (4): 363–376.

Weber, R. A. 2002. Ontological issues in accounting information systems. In *Research Accounting as an Information Systems Discipline*, edited by V. Arnold, and S. G. Sutton, 13–33. Sarasota, FL: American Accounting Association.

Weyuker, E. 1988. Evaluating software complexity metrics. *IEEE Transactions on Software Engineering* 14 (9): 1357–1365.

Wisniewski, E. J., and D. L. Medin. 1994. On the interaction of theory and data in concept learning. *Cognitive Science* 18: 221–282.

Wixom, B. H., and H. J. Watson. 2001. An empirical investigation of the success factors for data warehousing. *MIS Quarterly* 25 (1): 17–41.

Wright, P., and A. Lickorish. 1990. An empirical comparison of two navigation systems for two hypertexts. In *Hypertext: State of the Art*, edited by R. McAleese, and C. Green, 84–93. Oxford, U.K.: Ablex Publishing Corporation.

Wu, C. Z., H. C. Chan, H. H. Teo, and K. K. Wei. 1994. An experimental study of object–oriented query language and relational query language for novice users. *Journal of Database Management* 5 (4): 16–27.

Yen, M. Y. M., and Scamell, R. W. 1993. A human factors experimental comparison of SQL and QBE. *IEEE Transactions on Software Engineering* 19 (4): 390–410.

Zuse, H. 1991. *Software Complexity: Measures and Methods*. Berlin, Germany: Walter de Gruyter.