Journal of the Association for Information Systems

Volume 7 | Issue 8 Article 22

8-31-2006

Analysis of Competing Data Structures: Does Ontological Clarity Produce Better End User Query Performance

Paul L. Bowen
The University of Queensland

Robert A. O'Farrell
The University of Queensland

Fiona H. Rohde
The University of Queensland, f.rohde@business.uq.edu.au

Recommended Citation
Bowen, Paul L.; O'Farrell, Robert A.; and Rohde, Fiona H. (2006) "Analysis of Competing Data Structures: Does Ontological Clarity Produce Better End User Query Performance," *Journal of the Association for Information Systems*, 7(8), .
DOI: 10.17705/1jais.00098
Available at: https://aisel.aisnet.org/jais/vol7/iss8/22

Journal of the Association for Information Systems
Special Issue

Analysis of Competing Data Structures: Does Ontological Clarity Produce Better End User Query Performance [1]

Paul L. Bowen, Ph.D. CPA
UQ Business School
The University of Queensland
Brisbane, Queensland, Australia 4072
&
College of Business
Florida State University
Tallahassee, Florida 32306-1110, USA

Robert A. O'Farrell, BCom
UQ Business School
The University of Queensland
Brisbane, Queensland, Australia 4072

Fiona H. Rohde, Ph.D.
UQ Business School
The University of Queensland
Brisbane, Queensland, Australia 4072
f.rohde@business.uq.edu.au

Abstract

End users respond to stakeholders' information requests by using query tools to retrieve information from their organizations' data stores. The structure of these data stores impacts end users' performance, e.g., the accuracy of their responses. Ontologically clearer conceptual models have been shown to facilitate better problem solving within real-world application domains. If, however, ontologically clearer conceptual models are directly transformed into implementation (logical) data models, the differences in the number of entities and relationships may cause cognitive issues for end users that are likely to affect their query performance. This paper reports the results of an experiment that investigated the effect on query performance of more traditional logical models compared to ontologically clearer logical models. Results indicate that end users of the

[1] Jeffery Parsons was the accepting senior editor. This paper was submitted on 28 February, 2005, and went through 2 revisions.

ontologically clearer implementation made fewer semantic errors overall. Thus, the benefits of ontological clarity at the conceptual level may translate into similar benefits when querying ontologically clearer logical models. Unfortunately, an examination of the specific types of errors that were made indicated that the benefits are not clear cut. While the removal of optional attributes and relationships led to an overall reduction in the number of errors, closer analyses show that some types of errors (involving projection and restriction) decreased as expected, while other types of errors (involving joins) increased.

Keywords: ontology, information retrieval, end user performance, logical data model

Introduction

End users increasingly employ SQL-based query tools to provide timely responses to stakeholders' information requests (Owei, 2003; Hayes and Hunton, 2001). Semantically accurate queries produce higher quality information for decision makers and, hence, lead to better decisions (Klein, 2002). Factors that affect the quality of queries include data structure complexity (Borthick et al., 2001b) and the extent to which the information retrieval agent understands the application domain (Burton-Jones and Weber, 1998).

Conceptual data models are used as the basis for the design tasks performed when building or modifying data schemas (Hoffer et al., 2004). Data models developed at the conceptual level have important consequences for the construction and use of information systems. For example, some CASE tools automatically transform conceptual models into logical, i.e., implementation, data structures (Hoffer et al., 2004). If ontologically clearer conceptual models are directly transformed into implementation data models, the differences in the number of tables may cause cognitive issues for end users that are likely to adversely affect their query performance. Conversely, the decreased uncertainty associated with ontologically clearer implementation models may enhance query performance.

Ontological research into conceptual data models indicates that people perform problem solving tasks better when using conceptual Entity Relationship Diagrams (ERDs) that evidence higher degrees of ontological clarity (Burton-Jones and Weber, 1998; Gemino, 1998; Wand et al., 1999; Bodart et al., 2001; Gemino and Wand, 2005). The primary purpose of this research is to determine—for application domains similar in size to those used in prior research on conceptual models—whether greater ontological clarity produces better performance on information retrieval tasks. Because of the detailed coding method used to record query errors, we were able to perform in-depth analyses of how each data structure affected specific aspects of the queries. That is, this research examined, on a clause by clause basis, the errors and problems produced by query developers using the two alternative logical data structures.

This paper extends the research into the effects of the ontological clarity of semantic data models by investigating whether the benefits of greater ontological clarity for problem solving tasks translate to information retrieval tasks. That is, while prior research involved problem solving tasks using conceptual data models, this research examines to what extent ontological clarity impacts query performance using logical (implementation) data models. The specific part of the BWW ontology examined in this

research is Weber's (1997) assertion that Bunge's ontology (1977) implies that optional properties should not be used in semantic modelling. Instead, subtypes with mandatory properties should replace each optional property (e.g., students should be divided into "postgraduate students" and "undergraduate students" where only postgraduate students have the property of degree held).

The next section of this paper reviews the Bunge-Wand-Weber (BWW) model and the implications of the BWW model relative to increasing ontological clarity within logical data models. The section also discusses the issues affecting end user query performance before developing a series of hypotheses linking ontological clarity and end-user query performance. The section following hypothesis development details the research method, including the details of the experiment and the operationalization of the constructs. After the research method, the results are presented and discussed. The final section contains the conclusions, implications of the research, limitations, and directions for future research.

Background, Theory, and Hypotheses

Application of the Bunge-Wand-Weber Model

Bunge's (1977) theory of ontology, applied to information systems by Wand and Weber (1993), has gained widespread attention within both the IS and software engineering conceptual modelling domains (Burton-Jones and Weber, 1998; Green and Rosemann, 2000; Weber, 2003; Green and Rosemann, 2004). This attention has focused on the application of the BWW model to varied application domains (e.g., business and conceptual modelling). The BWW model provides guidelines that researchers are able to use to evaluate modelling grammars for ontological completeness (Wand and Wang, 1996; Shanks et al., 2003).

One of the BWW guidelines concerns optional properties. Bodart et al. (2001) used three experiments to examine whether optional properties should be used in conceptual modelling. Their results indicated that for recall and comprehension, participants using diagrams with optional properties outperformed participants using diagrams with mandatory properties. But for the problem solving tasks (i.e., identifying possible explanations for causes of a situation) participants using diagrams with mandatory properties outperformed participants using diagrams with optional properties. The improved problem solving occurred because the participants receiving the ontologically clearer diagram (i.e., the diagram with mandatory properties) developed a better understanding of the domains. Recently, Gemino and Wand (2005) extended Bodart et al. using a set of experimental tasks that not only confirmed that improved understanding of a conceptual data model could be achieved through the use of mandatory properties but demonstrated that mandatory properties also enhance problem solving.

Within the research reported in this paper, end users were required to formulate queries to satisfy information requests. In so doing, end users must apply their knowledge of the data structure and of the query language to produce a query that satisfies the information request. Thus, formulating queries can be considered a problem solving task.

Increasing Ontological Clarity Using the BWW Model

Optional properties

Optionality is an important feature of many conceptual modelling grammars (Bodart et al., 2001). However, to clearly convey the ontological meaning of the constructs in conceptual models, Wand et al. (1999) and Weber (1997) assert that optionality should be avoided. Within data modelling two "optional" situations can arise. The first occurs when a "thing" may or may not possess an attribute, i.e., an optional attribute. The second occurs when a "thing" may or may not participate in a relationship with another thing, i.e., an optional relationship. These two situations can be avoided, and thus ontological clarity improved, by creating a subclass that represents those "things" that possess the property and another subclass that represents those "things" that do not possess the property.

Consider the following example in relation to optional attributes: a customer places a sales order. When initially creating the sales order, the name of the employee who will fill the order is not known (i.e., picked_by_employee is an optional attribute at this point in the sales and delivery process (see Figure 1-a). The ontologically clearer solution is to model this optional attribute using two subtypes: one for orders that have been picked and are now awaiting shipment and another subtype for orders that have not been picked (see Figure 1-b).

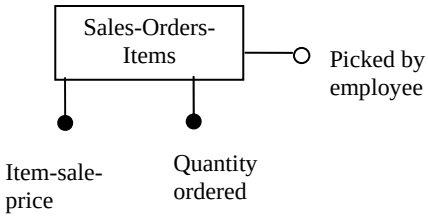


Figure 1-a. Optional attribute in a more traditional model

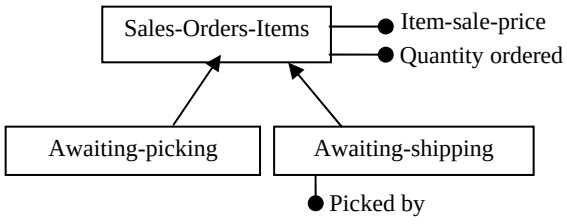


Figure 1-b. Replacing the optional attribute in an Ontologically Clearer Model by the Use of Subtyping.

Similarly, consider the following example pertaining to optional relationships. The example relates to inventory items that have never appeared on a sales order and inventory items that have appeared on at least one sales order. A possible solution is to

represent these two states by an optional relationship between two entities (see Figure 2-a). The ontologically clearer solution is to create one subclass for those items that have had no sales and a second subclass for those items that have had sales. Only those inventory items that have experienced sales can participate in the relationship with sales order items (see Figure 2-b).

Figure 2-a. Optional relationships in a more traditional model

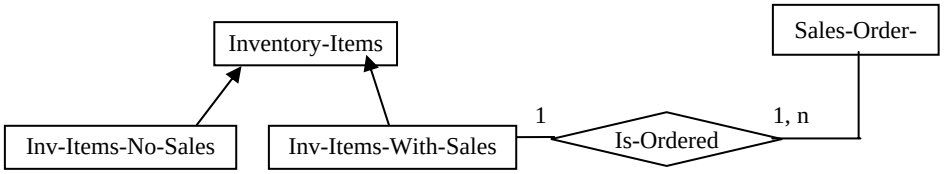


Figure 2-b. Replacing the optional relationship in an ontologically clearer model by the use of subtyping.

Associative entities

Weber (1997) argued that using relationships with attributes will undermine the ontological clarity of conceptual models. He suggests replacing relationships with attributes by mutual properties to improve the ontological clarity of Entity-Relationship Diagrams (ERDs).[2] These mutual properties belong to the entities involved in the association. For example, within a traditional ERD the two entities "inventory items" and "sales orders" participate in a many-to-many association with each other in which details about individual items on each sales order need to be recorded.

In practice, conceptual ERDs typically represent these extra details as attributes of the relationship "order" between two entities "inventory items" and "sales orders" (Chen 1976) (see Figure 3-a). According to Bunge's (1977) ontology, properties do not have properties, thus representing a relationship (i.e., a mutual property, with attributes in conceptual models reduces ontological clarity). Wand et al., (1999) and Weber (1997) suggest that one solution is to think about "sales order items" as a thing that has certain properties. The three things then participate in relationships with each other (see Figure 3-b).

2 Entity-relationship diagrams (ERDs) are the scripts generated by the entity-relationship grammar (Wand and Weber, 2002).

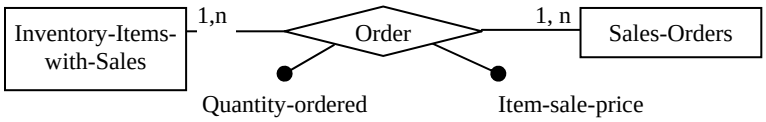


Figure 3-a. Relationships with attributes in a more traditional model

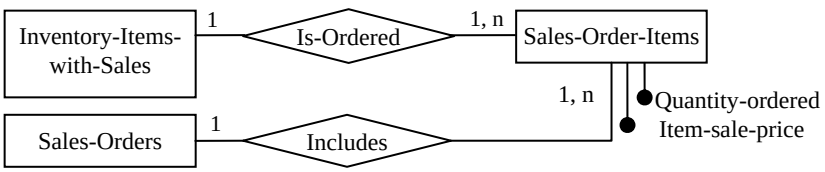


Figure 3-b. Relationships With Attributes Represented As Associative Entity In An Ontologically Clearer Model

End User Query Performance

Many layers of complexity can be found within the context of information processing. Researchers, however, are consistent in asserting a positive relationship between complexity and errors in task performance, i.e., as task complexity increases, performance decreases (March and Simon, 1958; Campbell, 1988; Rho and March, 1997; Borthick et al., 2001a; Borthick et al., 2001b; Chan et al., 1999; Jih et al., 1989).

IS researchers have investigated the effects of task complexity on human-computer interaction (Jacko et al., 1995). Prior research on the relationship between data modeling and human performance has shown that factors such as task type, complexity, experience, and the data modeling formalism can affect the users' performance and attitudes (Topi and Ramesh, 2002).

There is also a large body of research that investigates factors that affect end user query performance (see, e.g., Axelsen et al., 2001; Borthick et al., 2001a; Borthick et al., 2001b; Chan et al., 1993; Chan et al., 2004). Query performance has been shown to be affected by the end users' ability to understand the application domain (Jih et al., 1989; Rho and March, 1997; Siau et al., 2004) and by their ability to translate their understanding of the application domain correctly into a query language (Chan et al., 1993; Chan et al., 1999; Suh and Jenkins, 1992). Rho and March (1997) suggested that semantic overload associated with specific database representations and query languages may degrade end users' ability to access corporate databases. Because of differences in each of the clauses that make up a query, even within simple queries the number of mental comparisons required can vary greatly.

Effects of Ontological Clarity on Complexity (Hypothesis Development)

Effect on Overall Semantic Errors

To develop a query, end users map the constructs in the information request to the attributes in the database. This mapping process requires end users to locate the appropriate constructs within the data model before applying the syntax rules specified by the query language. The data structure (or data model) portrays the logical organization of the database to users. Different data models (e.g., traditional versus ontologically clearer) can result in different logical organizations.

Problem-solving tasks such as query formulation require model users to reason about the domain as part of their solution process (Gemino and Wand, 2003). For new information requests, end users are likely to begin their query formulation with some uncertainty concerning the mapping between the information requests and the data structure. Browne and Ramesh (2002) state that problem solving under conditions of uncertainty produce cognitive biases. Reducing uncertainty should tend to reduce cognitive biases. Increasing ontological clarity (e.g., explicitly identifying lawful subtypes) should decrease the amount of uncertainty associated with the data model. From this perspective, ontologically clearer data models should enhance end users' understanding of the application domain, thereby improving query performance.

More traditional data structures and ontologically clearer data structures yield different numbers of entities and relationships (i.e., the ontologically clearer models typically contain substantially more entities than the corresponding more traditional data models). When formulating queries for data structures with more entities (e.g., more normalized data structures) end users often need to reassemble the fragmented data (Date, 2004). Prior research indicates that this data reassembly results in queries that contain more terms and more complicated logic and thus leads to more errors (Borthick et al., 2001b). Because of the opposing forces of simplicity (parsimony) and greater ontological clarity, H1 is stated in the null form:

Hypothesis 1: End users querying ontologically clearer data structures do not make a different number of semantic errors than end users querying more traditional data structures.

Effects on Specific Types of Semantic Errors

Hypothesis 1 tests the net effect of the different challenges associated with the two types of data structures. Investigating the effects of optionality on different clauses within the SQL queries will provide greater insights into differences resulting from the two alternative types of data structures and will facilitate practitioners' efforts to extract practical benefits from this research.

End-user queries involve projection, join, and restriction (or selection) operations.[3] Typically, end users begin with the projection operation, i.e., they determine the columns

3 The basic query operations in a relational system are projection, join, and selection (also called restriction). A projection is a subset of the columns in a table. A join links the rows in two or more tables by comparing the values in specified fields. Joins are often combined with projections and selections. A restriction is a subset of the rows in a table.

required to satisfy the information request. They place these attributes or formulas in the primary SELECT clause of the SQL query. Users composing the primary SELECT clause are unlikely to find one of the two types of data model significantly easier, as the ontologically clearer data model will require searching more tables but fewer attributes in each table, whereas the traditional data structure will require searching fewer tables but more attributes in each table.

In addition to the primary SELECT clause, SQL queries can also include SELECT clauses in subqueries. Queries of data structures containing optional attributes and relationships (traditional data structures) are more likely to require subqueries than queries of ontologically clearer data structures. For example, using a traditional data structure, a query involving reporting on inventory items with no sales would typically require a subquery to exclude inventory items with sales (see model answers for information requests 7 and 9 in Appendix II).[4] In contrast, using an ontologically clearer data structure, the same query would require a join to a table containing inventory items with no sales but would not necessarily require a subquery.

Because of the increased subtyping, queries of ontologically clearer data structures are, however, more likely to require subqueries, including UNION, INTERSECT, or MINUS set operations. For example, if an information request relates to sales orders that have been paid, in an ontologically clearer data structure such orders can be found in multiple subtypes (see the model answer for information request 6 in Appendix II). In contrast, using a traditional data structure would require a restriction based on an attribute value but would not necessarily require a subquery. .

Because of these opposing forces, H2 is stated in the null form:

Hypothesis 2: End users querying ontologically clearer data structures do not make a different number of SELECT errors than end users querying more traditional data structures.

Once end users determine the columns, they must determine which entities (tables) contain the attributes required to produce those columns. They list these tables in the primary FROM clause of the SQL query and, if necessary, in the subqueries. Relative to ontologically clearer data models, the presence of optional properties in traditional data models should not significantly affect end users' ability to locate and list the appropriate tables.

On the other hand, because of sub-typing, the ontologically clearer data structures exhibit more fragmented schemas, because queries using these structures typically require more tables in the FROM clause than the equivalent queries of traditional data structures (see Appendix II). The increased fragmentation of the ontologically clearer data models requires end users to expend more cognitive effort to locate the same information and thus is likely to lead to more errors. Accordingly:

Hypothesis 3: End users querying ontologically clearer data structures make more FROM errors than end users querying more traditional data structures.

---

4 The example could also have been resolved by the use of an outer join rather than a subquery. The use of an outer join is still more complicated that the alternative using the ontologically clearer model.

After end users determine the tables needed for the query, they must specify how these tables are related. They place these joins in the WHERE or FROM clause of the SQL query.[5] The logic for the impact on performance of these JOINs for the alternative data structures parallels that of the FROM clauses. That is, except for extremely small (e.g., single table) queries, end users querying the ontologically clearer data structures must consider more fragmented schemas. This increased fragmentation of the data models requires end users to join more tables than required by end users of the traditional data models (see Appendix II). The increase in cognitive effort required by users of the ontologically clearer data models increases the likelihood that they will make more errors.

Hypothesis 4: End users querying ontologically clearer data structures make more JOIN errors than end users querying more traditional data structures.

In addition to projection (H2) and join (H3 and H4) operations, information requests frequently require restriction (or selection) operations. Along with the JOINs, end users typically place these restrictions (CONDITIONs) in the WHERE clause of the SQL query. Obtaining the desired data from more traditional data structures containing optional attributes often requires the appropriate use of IS NULL or IS NOT NULL constructs in the CONDITION clause. The correct formulation of such CONDITIONS requires greater cognitive effort than the corresponding queries of ontologically clearer data structures that do not require the IS NULL or IS NOT NULL constructs. The increase in cognitive effort required by users of the traditional data models increases the likelihood that they will make more errors. Equivalently:

Hypothesis 5: End users querying ontologically clearer data structures make fewer CONDITION errors than end users querying more traditional data structures.

Research Method

Research Design, Participants, and Data Collection

In a laboratory experiment, participants composed and executed queries in SQL for one of two relational databases. Both data structures satisfied third normal form. The first database exhibited a more traditional data structure. The second equivalent data structure differed only in that, as suggested by Burton-Jones and Weber (1998), Bodart et al. (2001) and Weber (2003), optional properties and relationships were removed, i.e., the second data structure was ontologically clearer (Appendix I). To increase statistical power, the same laboratory experiment was conducted on two separate occasions, one year apart.[6] On the first occasion, forty-six advanced undergraduate and masters-level Commerce students participated in the experiment. On the second occasion, thirty-five advanced undergraduate and masters-level Commerce students participated in the experiment. The two groups were students in equivalent courses being offered in two

5 Joins are accomplished in the "where" clause when using SQL '87 standard. Alternately, the joins could be specified in the "from" clause as per the SQL '92 standard.

6 The data collected during the two different years was examined and no significant differences between the two groups were detected. Thus, the results are reported as a single experiment with two data collection points.

consecutive years. All participants were familiar with general computing concepts and activities and, prior to the experiment, had received identical training in developing SQL queries.[7] Self-reported demographic data indicated that the majority of participants had no significant exposure to either databases or SQL prior to taking the subject. To minimize bias in the experiment, the training session used a "mixed" ERD not related to the scenario, containing aspects from both traditional and ontologically clearer data models. The sample questions and answers covered both the traditional and ontologically clearer aspects of the training diagram. All participants received their training via the same instructor.

All participants received a set of instructions containing the scenario, the appropriate ERD, and the details of tasks to be performed (Appendix II contains the information requests). To control for experience and education effects, participants were assigned to one of two groups according to their GPA. The person with the highest GPA was ranked 1, the next ranked 2, etc. Participants were assigned to groups according to their rank, i.e., 1 to group A, 2 to group B, 3 to group B, 4 to group A, etc. This method of assignment was intended to make the two groups as equivalent as possible. The groups were then randomly assigned to a treatment.

The participants had two hours to construct, as accurately as possible, appropriate queries for as many of the fourteen information requests as they could. Participants received 7% course credit for participating. Participants were informed that each completed query would be marked according to its accuracy. Because the correct query formulations were generally increasing in complexity, participants were encouraged to do their best on each query before moving to the next information request.[8]

Each information request had two model formulations: one for the more traditional data structure and one for the ontologically clearer data structure. The experiment was designed so that for the even-numbered information requests, the correct query of the more traditional data structure was shorter than the correct query of the ontologically clearer data structure. For the odd-numbered information requests, the reverse occurred. This design attempted to minimize any potential bias relative to complexity for either data structure.

Participants used a UNIX shell script that recorded their entire session. Each participant was presented the information requests in the same order. After each query attempt was executed, the system displayed the SQL result, i.e., either the rows returned by the query or a syntax error message. Participants could revise their queries as many times as they wished. Once an information request had been deemed completed by the participants, they could not return to it.

Subject to the constraints imposed by the laboratory setting, the experiment was

---

7 Prior to the experiment, participants had received approximately 10 hours of instruction in ERDs and SQL queries. Participants had also completed one 2 hour quiz composing SQL queries.

8 The grading criteria for the students' results, not the coding for the statistical analysis, was as follows. The students received a base of 50% of the available points if they produced at least four syntactically correct queries that reasonably addressed the corresponding information requests. Essentially all students received this 50 points. Each completed query was graded on a 0 to 5 scale based on its accuracy. Because of the increasing complexity of the queries, obtaining the same score on each successive query became increasingly challenging.

designed to be as realistic as possible. For example, the feedback from the DBMS throughout the experiment allowed the querying experience to approximate that undertaken during organizational database searches. Similarly, subject to time constraints and deadlines, query developers are allowed as many attempts as they wish to obtain the desired information. Organizational query developers only present the results of their final query to the person making the information request. Hence, this research analysed only each participant's last attempt for each information request.

Operationalizing the Variables

Dependent Variables

We determined the number of semantic errors by counting the number of semantic errors in each participant's last query attempt for each information request. Information requests that were not attempted by a particular participant were not included in the scoring. Furthermore, the final question being attempted at the end of the two-hour period may not have been included when it was obvious to both data coders that the SQL query was not complete. Two experienced coders independently determined the minimum number of changes, if any, required to make each query from each participant semantically correct. After the two individuals independently performed this task, they cross-checked their error coding sheets (Appendix III contains a blank coding sheet.) for correctness and consistency and resolved any differences. When the two coders compared their solutions, the possible outcomes were initial total agreement, one coder being deemed correct, or both coders changing their solution. Given the criteria of making the minimum number changes to reach a semantically correct solution, after re-examining each query and each coder's solution, the coders were always able to reach agreement on the number of errors, if any, in each participant's query.

The total number of each type of semantic error was determined by counting the number of that type of semantic error in the last attempt for each information request, i.e., the minimum number of changes to reach a semantically correct solution. The minimum number of semantic errors for a particular query is zero. The maximum varies depending on the query being composed.[9] That is, the number of semantic errors for each SQL clause was determined by counting the number of changes (both items added and items missed out) required to change the participant's query such that it was semantically correct. Although a model query solution was developed, this solution was not necessarily the only semantically correct solution. Each participant's query must be evaluated in terms of its accuracy in producing a semantically correct solution. This possibility of many different "correct" solutions made the use of anchoring error coding schemes (Parsons and Saunders, 2004) difficult to adopt. These error counts are the dependent variables for Hypotheses 2 through 5. These errors were then summed to derive the total number of semantic errors. This total error count was the dependent variable for Hypothesis 1.

Independent Variables

The independent variable was the treatment group. Group was a categorical variable

9 The maximum number of errors for any one query by a parsimonious participant was 39 and for any one query by a ontologically clearer participant was 52.

with the values of more traditional or ontologically clearer. The information requests were generally of increasing complexity. The order of the information requests took into account the "challenges" encountered by participants when composing a query. A query containing a subquery or outer join, for example, was likely to be shorter that a query joining multiple tables. The participant was, however, more likely to find the shorter query more challenging. Information request number (INFO REQ NUMBER) was a variable with values from 1 to 14.[10] Grade Point Average (GPA) was included as a covariate.

Results

Summary Performance

Table 1 summarizes the participants' characteristics and performance by data structure. The table indicates that, in absolute terms, each participant generated fewer errors on average per information request when querying the ontologically clearer data structure than when querying the more traditional data structure.

Table 1: Participant Characteristics and Performance [11]

	More Traditional	Ontologically Clearer
Grade Point Average (7-point scale, 7 high)		
Mean	4.9899	4.8888
Standard deviation	0.9488	0.9789
Gender		
Number of males	27	25
Number of females	13	16
Number of Information Requests completed		
Mean	9.2000	8.6829
Standard deviation	1.4178	1.8767
Number of Attempts per Completed Information Request		
Mean	9.6829	9.5000
Standard deviation	6.8827	5.5667
Number of Information Requests Semantically Correct		
Mean	5.5122	5.5476
Standard deviation	1.8045	2.0387
Semantic Errors per Request Attempted		
Mean	4.2663	3.5899
Standard deviation	7.6528	7.0571

10 The Spearman Correlation coefficient between information request number and number of semantic errors is 0.42698 (p = 0.0001). The Pearson Correlation coefficient between information request number and number of semantic errors is 0.40154 (p = 0.0001). Thus, although information request number is an ordinal variable, we assert that it behaves as an interval variable and treat it accordingly in our data analysis.

11 The table contains statistics related to each of the 81 participants. Each participant receiving the more traditional treatment completed on average 9.2000 information requests, making on average 9.6829 attempts per completed information request, and yielding on average 4.2663 semantic errors per completed information request.

The Effect of Ontological Clarity on Total Semantic Errors

Comparing the queries completed by the more traditional group with the queries completed by the ontologically clearer group, nested ANCOVA results [12] indicate that the number of semantic errors was significantly associated with the level of ontological clarity ($F_{12, 697} = 4.44$, $p = 0.0001$, two-tail test) (Table 2). The means results (Table 3, Panel A) confirm that the end users querying the ontologically clearer data structure made significantly fewer semantic errors than end users querying the more traditional data structure. Thus, Hypothesis 1 (stated in null form) can be rejected in favor of the ontologically clearer data structure.

Table 2: Main Model and Effects of Data Structure on Accuracy [13]

Source	R^2	df	Mean Square	F Value	Pr > F
Model	0.3440	26	519.40	14.06	0.0001
Error		697	36.95		
INFO REQ NUMBER		13	873.50	23.64	0.0001
Group (Info Req Number)		12	164.18	4.44	0.0001
GPA		1	511.21	13.84	0.0002

The Effect of Ontological Clarity on Different Types of Semantic Errors

Summary Performance

Table 3, Panel A summarizes the participants' performance in total and for different types of semantic errors by data structure. The table indicates that, in absolute terms, participants querying the more traditional data structure generated more errors in total than participants querying the ontologically clearer data structure. Table 3 Panel B indicates that, in absolute terms, participants querying the ontologically clearer data structure generated fewer errors in total for the queries that favored the ontologically clearer group than participants querying the more traditional data structure. Moreover, in absolute terms, participants querying the ontologically clearer data structure generated fewer errors in total for the queries that favored the more traditional group than participants querying the more traditional data structure.[14]

The table also indicates that, in absolute terms, participants querying the more traditional data structure generated more SELECT, CONDITION, GROUP BY, and HAVING errors than participants querying the ontologically clearer data structure. Table 3, Panel B, also indicates that during both experiments, in absolute terms, participants generated more FROM and JOIN errors when querying the ontologically clearer data structure rather than the more traditional data structure.

12 We analyzed the data, nested by query (thus the 12 degrees of freedom), to ensure that the results we were finding as significant were consistent across the range of information requests.
13 There are no significant interactions among the model components (covariate and predictors).
14 The contrasts for the More Traditional Odd Numbered questions and Ontologically Clearer Odd Numbered questions was significant ($F=4.52$, $p = 0.0338$). The contrasts for the More Traditional Even Numbered questions and Ontologically Clearer Even Numbered questions was not significant ($F=0.55p = 0.4578$).

Table 3: Types of Error Performance

Panel A: Different Error Types for Each Query Completed

	Overall Performance	
	More Traditional	Ontologically Clearer
SELECT Errors		
Mean	0.8995	0.7191
Standard deviation	2.2240	2.0139
FROM Errors		
Mean	0.3315	0.4410
Standard deviation	1.0641	1.3106
JOIN Errors		
Mean	0.9022	1.4944
Standard deviation	2.6721	3.4187
CONDITION Errors		
Mean	1.3913	0.4129
Standard deviation	2.7282	1.5996
GROUP BY Errors		
Mean	0.3179	0.2167
Standard deviation	1.3569	1.5242
HAVING Errors		
Mean	0.3369	0.1798
Standard deviation	1.9505	1.2178
Semantic Errors per query		
Mean	4.2663	3.5899
Standard deviation	7.6528	7.0571

Panel B. Different Error Types on Odd versus Even Information Requests

	Odd Numbered Requests (OC* Favored)		Even Numbered Requests (More Traditional Favored)	
	More Traditional	Ontologically Clearer	More Traditional	Ontologically Clearer
SELECT Errors				
Mean	0.7319	0.4456	1.0862	1.0429
Standard deviation	1.6447	1.3687	2.7219	2.5443
FROM Errors				
Mean	0.5361	0.3523	0.1034	0.5460
Standard deviation	1.3121	1.1041	0.6183	1.5162
JOIN Errors				
Mean	0.8454	1.2642	0.9655	1.7669
Standard deviation	2.3004	3.0580	3.0390	3.7934
CONDITION Errors				
Mean	1.0412	0.1451	1.7816	0.7301
Standard deviation	2.1203	0.9894	3.2379	2.0639
GROUP BY Errors				
Mean	0.1082	0.1452	0.5517	0.2883
Standard deviation	0.4920	0.7215	1.8794	1.3413
HAVING Errors				
Mean	0.3505	0.1709	0.3218	0.1902
Standard deviation	1.9530	1.0292	6.2680	1.4123
SEMANTIC ERRORS per request attempted				
Mean	3.7526	2.6943	4.8391	4.6503
Standard deviation	7.2904	4.8342	8.0200	8.9079

*OC indicates ontologically clearer

The Effect of Ontological Clarity on Select Errors

Comparing the queries completed by the more traditional group with the queries completed by the ontologically clearer group, nested ANCOVA results indicate that the number of SELECT errors was significantly associated with the level of ontological clarity ($F_{12,697} = 3.71$, p = 0.0001, two-tail test) (Table 4). The means results (Table 3, Panel A) confirm that the end users querying the ontologically clearer data structure made significantly fewer SELECT errors than end users querying the more traditional data structure. Table 3, Panel B also indicates that, in absolute terms, participants querying the ontologically clearer data structure generated fewer SELECT errors for the queries that favored the ontologically clearer group than participants querying the more traditional data structure. Moreover, in absolute terms, participants querying the ontologically clearer data structure generated fewer SELECT errors for the queries that favored the more traditional group than participants querying the more traditional data structure. Thus Hypothesis 2 (stated in null form) can be rejected in favor of the ontologically clearer data structure.

Table 4: Effect of Ontological Clarity on Different Types of Errors [15]

Error Type	R^2	F Value	Pr > F
SELECT	0.2599	3.71	0.0001
FROM	0.1906	4.30	0.0001
JOIN	0.2439	6.94	0.0001
CONDITION	0.3019	5.77	0.0001
GROUP BY[16]	0.1492	0.94	0.5050
HAVING	0.1271	0.87	0.5809

The Effect of Ontological Clarity on From Errors

Comparing the more traditional with the ontologically clearer group, nested ANCOVA results indicate that the number of FROM errors was significantly associated with the level of ontological clarity ($F_{12,697} = 4.30$, p = 0.0230, two-tail test) (Table 4). The means results (Table 3, Panel A) confirm that end users querying the ontologically clearer data structure made more FROM errors than end users querying the more traditional data structure. Table 3, Panel B indicates that, in absolute terms, both groups of participants generated fewer FROM errors for the queries that favored their particular data structure, with the difference being more pronounced in the group where the queries favored the more traditional data structure. This finding supports the assertion that participants may have more difficultly selecting the appropriate tables when using the ontologically clearer data structure because the ontological clearer data structure contains more entities (tables). The results support Hypothesis 3.

---

15 Each row of the table represents a different statistical test. Thus, the SELECT row indicates the R-squared, the F Value. and the p Value for the nested ANCOVA results when SELECT errors is taken as a dependent variable; the FROM row indicates the R-squared, the F Value, and the p Value for the nested ANCOVA results when FROM errors is taken as a dependent variable and so on.

16 Comparing the more traditional with the ontologically clearer group, nested ANCOVA results indicate that the number of GROUP BY errors was not significantly associated with the level of ontological clarity. Similarly, comparing the more traditional with the ontologically clearer group, nested ANCOVA results indicate that the number of HAVING errors was not significantly associated with the level of ontological clarity.

The Effect of Ontological Clarity on JOIN Errors

Comparing the more traditional with the ontologically clearer group, nested ANCOVA results indicate that the number of JOIN errors was significantly associated with the level of ontological clarity ($F_{12, 697} = 6.94$, $p = 0.0001$, two-tail test) (Table 4). The means results (Table 3, Panel A) confirm that end users querying the ontologically clearer data structure made significantly more JOIN errors than end users querying the more traditional data structure. Table 3, Panel B also indicates that, in absolute terms, participants generated fewer JOIN errors for the queries that favored the more traditional group, when querying the more traditional data structure rather than the ontologically clearer data structure. Moreover, in absolute terms, participants querying the more traditional data structure generated fewer JOIN errors for the queries that favored the ontologically clearer group than participants querying the ontologically clearer data structure. Because the data in the ontologically clearer data structure are more fragmented, the queries often required additional joins in the WHERE clause. The increased requirement of joins is consistent with a corresponding increase in the likelihood that the query developers make more JOIN errors using the ontologically clearer data structure. Hence, Hypothesis 4 is supported.

The Effect of Ontological Clarity on CONDITION Errors

Comparing the more traditional with the ontologically clearer group, nested ANCOVA results indicate that the number of CONDITION errors was significantly associated with the level of ontological clarity ($F_{12, 697} = 5.77$, $p = 0.0001$, two-tail test) (Table 4). The means results (Table 3, Panel A) confirm that end users querying the ontologically clearer data structure made significantly fewer CONDITION errors than end users querying the more traditional data structure. Table 3, Panel B also indicates that, in absolute terms, participants querying the ontologically clearer data structure generated fewer CONDITION errors for the queries that favored the ontologically clearer group than participants querying the more traditional data structure. Moreover, in absolute terms, participants querying the ontologically clearer data structure generated fewer CONDITION errors for the queries that favored the more traditional group than participants querying the more traditional data structure. The inclusion of optional properties in the more traditional model, and their subsequent removal in the ontological clearer model, was a fundamental difference between the two data structures. This difference required more extensive use of IS (NOT) NULL in the queries formulated for the more traditional data structure, whereas additional joins are required for queries using the ontologically clearer data structure. That the users of the more traditional data structure make more CONDITION errors is, thus, consistent with the expectations asserted by Hypothesis 5.

COVARIATES

GPA and query complexity (proxied by information request number) were significantly associated with the overall number of semantic errors made (Table 2). Regardless of the treatment group, participants with higher GPAs performed the query tasks significantly more accurately. Also, as expected, more complex queries produced significantly more errors.

Conclusions, Limitations, and Future Research

This study examined relationships between the level of ontological clarity of data structures and query performance. Table 5 presents a summary of the hypotheses and the findings of the study. The table also presents the impact, on average, of incorporating mandatory attributes.

Table 5: Summary of hypotheses and findings

Hypothesis (Predicted)	Overall (Results)	Odd Numbered Requests: OC Favored (Results)	Even Numbered Requests: More Traditional Favored (Results)
H1: Overall Semantic Errors (same: stated in null)	OC* less errors (16% less)	OC less errors (28% less)	OC less errors (4% less)
H2: Select Errors (same: stated in null)	OC less errors (10% less)	OC less errors (39% less)	OC less errors (4% less)
H3: From Errors (OC more errors)	OC more errors (33% more)	OC less errors (34% less)	OC more errors (428% more)
H4: Join Errors (OC more errors)	OC more errors (66% more)	OC more errors (50% more)	OC more errors (83% more)
H5:Condition Errors (OC less errors)	OC less errors (70% less)	OC less errors (99% less)	OC less errors (59% less)

*OC indicates ontologically clearer

The results indicate that end users querying the ontologically clearer data structure made significantly fewer semantic errors when composing their queries than end users querying the more traditional data structure, i.e., ontologically clearer data structures were associated with lower rates of semantic errors (ontologically clearer participants made on average 16% fewer semantic errors). Thus, the null hypothesis (Hypothesis 1) can be rejected. This result held for queries that favored the more traditional group and for queries that favored the ontologically clearer group. The effect of removing optional properties for the queries that favored the ontologically clearer group resulted in participants querying the ontologically clearer data structure making, on average, 28% fewer errors than participants querying the more traditional data structure. The effect of removing optional properties for the queries that favored the more traditional group still led to a small improvement, i.e., a 4% reduction in errors.

End users querying the ontologically clearer data structure made significantly fewer SELECT (allowing us to reject the null version of Hypothesis 2) and CONDITION (supporting Hypothesis 5) errors than end users querying the more traditional data structure. The ontologically clearer group made, on average, 10% fewer SELECT errors and 70% fewer CONDITION errors. Again, this result held for queries that favored the more traditional group and for queries that favored the ontologically clearer group. The effect of removing optional properties was more pronounced for the queries that favored the ontologically clearer group, e.g., these queries, on average, produced 39% fewer SELECT errors and 99% fewer CONDITION errors. Notably, removing optional properties for even-numbered requests, i.e., for requests that favored the more traditional group, led to a substantial reduction in CONDITION errors (59%) and even a modest improvement in SELECT errors (4%). These results are important because, in all

cases (whether the bias was in favour of the query or not), the ontologically clearer participants made fewer errors of both types.

An examination of the FROM and JOIN errors revealed an opposite trend. End users querying the ontologically clearer data structure made significantly more FROM and JOIN errors than end users querying the more traditional data structure. Thus, Hypotheses 3 and 4 are supported. Ontologically clearer participants made, on average, 10% more FROM errors and 70% more JOIN errors. The effect of removing optional properties for the queries that favored the ontologically clearer group led to queries with, on average, 39% fewer FROM errors and 99% more JOIN errors. The effect of removing optional properties for the queries that favored the more traditional group led to a 428% increase in FROM errors and a 59% increase in JOIN errors by the ontologically clearer participants.

Implications of these results for practitioners include exercising caution when striving for greater ontological clarity for logical data models. At this stage it appears that the benefits of ontological clarity at the conceptual level may transfer to the querying of ontologically clearer logical models. An examination of the specific types of errors that were made, however, indicated that the benefits are not pervasive. While the removal of optional attributes and optional relationships led to an overall reduction in the number of errors, as expected, particular types of errors decreased but other types of errors increased. These results are important, as changes in data structures require training that targets the particular types of errors that are likely to increase. Practitioners should also improve metadata, including data dictionaries, for traditional parsimonious data structures. Explaining the meaning of optional properties (both attributes and relationships) is likely to improve the performance of end users querying traditional data models.

This paper makes several contributions to research into the effects of ontological clarity and human-computer interactions via its examination of the influence of ontological clarity on end user query performance (effectiveness). First, the study extends Bodart et al.,'s (2001) and Gemino and Wand's (2005) research relative to problem solving by confirming that end users make fewer semantic errors when retrieving data from an ontologically clearer logical data structure. Second, the study contributes to extant research by identifying the sections of queries most affected by the different data structures.

This study has several limitations. First, the usual caveats associated with laboratory experiments limit the generalizability of the results. Second, the research used students as participants. These participants had, however, received training in information technology (IT) and business-related subjects. Their level of query proficiency was likely to be typical of end users in many organizations. Third, the time pressures in the experiment may be different from those faced by query developers in a business environment. Fourth, the information content of the two models may be perceived to be different. For example, the use of subtyping in the ontologically clearer models made explicit some of the content that was only implicit in the traditional data models. Furthermore, the optionality of attributes was depicted to participants within their attached data dictionary. Fifth, the results are dependent on the particular information requests presented to the participants and the relative levels of complexity of the information requests.

Future research is needed to improve end users' abilities to extract the information they need. Four examples of such research are noted here. First, this experiment needs to be replicated to ensure the robustness of the findings and to enhance the generalizability of the results. These replications need to take a number of forms. The study should be replicated using different sets of information requests within an experimental setting with an increased number of participants. Such research will allow us to better determine the types of errors that are likely to decrease and the types of errors that are likely to increase, which should promote development of training targeted at the type of data model presented. Research could also examine which types of errors (e.g., FROM, JOIN, CONDITION) are more serious and which can be decreased the most through targeted training. The study also needs to be replicated within organizational settings using non-student participants and a set of information requests deemed most applicable to the organization. Second, this research could be extended to examine the effects of ontological clarity on larger, more realistically-sized, data structures. Third, the effects of potentially different content between the models needs to be examined. Fourth, the use of mixed models and the use of views to create the most appropriate models can be examined. This type of research would allow us to examine effects of removing optional attributes versus the removal of optional relationships. This additional research will allow us to determine which changes to the model assist in the comprehension of the model, and thus in the development of more accurate queries (even though the model may increase in size).

Acknowledgements

We wish to thank Ron Weber for his comments, suggestions, and encouragement throughout the project. We appreciate the comments and suggestions provided an ICIS 2004 review team headed by senior editors Ritu Agarwal and Laurie Kirsch, and by ICIS 2004 participants on an earlier version of the manuscript based on the first of two year's experimental data. We are grateful for the funding provided for this research by the UQ Business School and for the data coding assistance provided by Alice Wu.

References

Axelsen, M., A.F. Borthick, and P. L. Bowen (2001) "A Model for and the Effects of Information Request Ambiguity and End User Query Performance", *Proceedings of the International Conference on Information Systems*, December, pp. 537-542.

Bodart, F., M. Sim, A. Patel, and R. Weber (2001) "Should Optional Properties Be Used in Conceptual Modelling? A Theory and Three Empirical Tests," *Information Systems Research,* (12)4, pp.384-405.

Borthick, A.F., P.L. Bowen, D.R. Jones, and M.H.K. Tse (2001a) "The Effects of Information Request Ambiguity and Construct Incongruence on Query Development," *Decision Support Systems*, (32)1, pp. 3-25.

Borthick A. F., P.L. Bowen, S. Liew, and F.H. Rohde (2001b) "The Effects of Normalization on Query developer Query Errors: An Experimental Evaluation," *International Journal of Accounting Information Systems* (2)4, pp. 195-223.

Browne, G. J. and V. Ramesh (2002)" Theory and Techniques for Improving Information Requirements Determination: A Cognitive Perspective", *Information & Management,* (39)8, pp 625-645.

Bunge, M. (1977) *Treatise on Basic Philosophy: Volume 3: Ontology I: The Furniture of the World.*, Boston, MA: Reidel.

Burton-Jones, A. and R. Weber (1998) "Understanding Relationships with Attributes in Entity-Relationship Diagrams". *International Conference of Information Systems*, pp. 214-228.

Campbell D. J. (1988) "Task Complexity: A Review and Analysis." *Academy of Management Review,* (13)1, pp. 40-52.

Chan, H.C., K.K. Wei, and K.L. Siau (1993) "User-Database Interface: the Effect of Abstraction Level on Query Performance: A Field Experiment". *MIS Quarterly,* (17)4, pp. 441-464.

Chan, H.C., B.C.Y. Tan, and K.K. Wei (1999) "Three Important Determinants of User Performance for Database Retrieval", *International Journal of Human Computer Studies* (51)1, pp. 895-918.

Chen, P. S. (1976) "The Entity-Relationship Model - Toward a Unified View of Data" *ACM Transactions on Database Systems,* (1)1, pp. 9-36.

Date, C. J. (2004) *An Introduction to Database Systems* 8th ed. Reading, Massachusetts: Addison Wesley Longman.

Gardner, W.B., and M. Serra (1997) "An Object-Oriented Layered Approach to Interfaces for Hardware / Software Codesign of Embedded Systems." *Thirty-First Annual Hawaii International Conference on System Sciences- 7, Software Technology,* January, pp. 197-203.

Gemino, A. (1998) "Empirical Comparison of System Analysis Techniques", Ph.D. Thesis, Faculty of Commerce and Business Administration, University of British Columbia.

Gemino, A. and Y. Wand (2003) "Evaluating Modeling Techniques Based on Models of Learning", *Communications of the ACM*, (46)10 pp. 79-84.

Gemino, A. and Y. Wand (2005) "Complexity and Clarity in Conceptual Modeling: Comparison of Mandatory and Optional Properties", *Data and Knowledge Engineering*, (55) pp 301-326.

Green, P. and M. Rosemann, (2000) "Integrated Process Modeling: An Ontological Evaluation" *Information Systems*, (25)2, pp. 73-87.

Green, P. and M. Rosemann, M. (2004) "Applying Ontologies to Business and Systems Modelling Techiques and Perspectives: Lessons Learned", *Journal of Database Management*, (15)2, pp. 105-117.

Hayes, D.C. and J.E. Hunton (2001) "When querying databases, you've got to ask the right question", *Journal of Accountancy,* (191)2, pp. 35-45.

Hoffer, J.A., J.F. George and J. S. Valacich (2004) *Modern Systems Analysis and Design* 4th Edit, Reading, Massachusetts: Addison Wesley Longman.

Jih., K., D. Bradbard, C. Snyder, and N. Thompson (1989) "The Effects of Relational and Entity-Relationship Data Models on Query Performance of Query Developers," *International Journal of Man-Machine Studies*, (31)3, pp. 257-267.

March, J., and H.A. Simon (1958) *Organizations*, New York: John Wiley.

Owei, V. (2003) "Development of a conceptual query language: Adopting the user centered methodology", *The Computer Journal* (46)6, pp. 602-624.

Parsons, J. and C. Saunders (2004) "Cognitive Heuristics in Software Engineering: Applying and Extending Anchoring and Adjustment to Artifact Reuse," *IEEE Transactions on Software Engineering*, (30)12, pp. 873- 888.

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, pp. 3-14.

Siau, K.L., H. C. Chan, and K.K. Wei (2004) "Effects of Query Complexity and Learning on Novice User Query Performance with Conceptual and Logical Database

Interfaces". *IEEE Transactions on Systems, Man and Cybernetics, Part A (Systems and Humans),* (34)2, pp. 276-281.

Shanks. G., E. Tansley and R. Weber (2003) "Using Ontology to Validate Conceptual Models," *Communications of the ACM,* (46)10, pp. 85 – 89.

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, pp. 252-272.

Topi, H. and V. Ramesh (2002) "Human Factors Research on Data Modeling: A Review of Prior Work, An Extended Framework and Future Research Directions", *Journal of Database Management*, (13)2, pp. 3-19.

Wand, Y., and R. Wang (1996) "Anchoring Data Quality Dimensions in Ontological Foundations**"** Communications of the ACM, (39)11, pp 86-95.

Wand, Y., and R. Weber (1993) "On the Ontological Expressiveness of Information System Analysis and Design Grammars". *Journal of Information Systems,* (4)4, pp. 299-330.

Wand, Y., and R. Weber (2002) "Research Commentary: Information Systems and Conceptual Modeling – A Research Agenda", *Information Systems Research*, (13)4, pp 363-378.

Wand, Y., and R. Weber (2004) "Reflection: Ontology in Information Systems" *Journal of Database Management*, (15)2, pp 3 – 7.

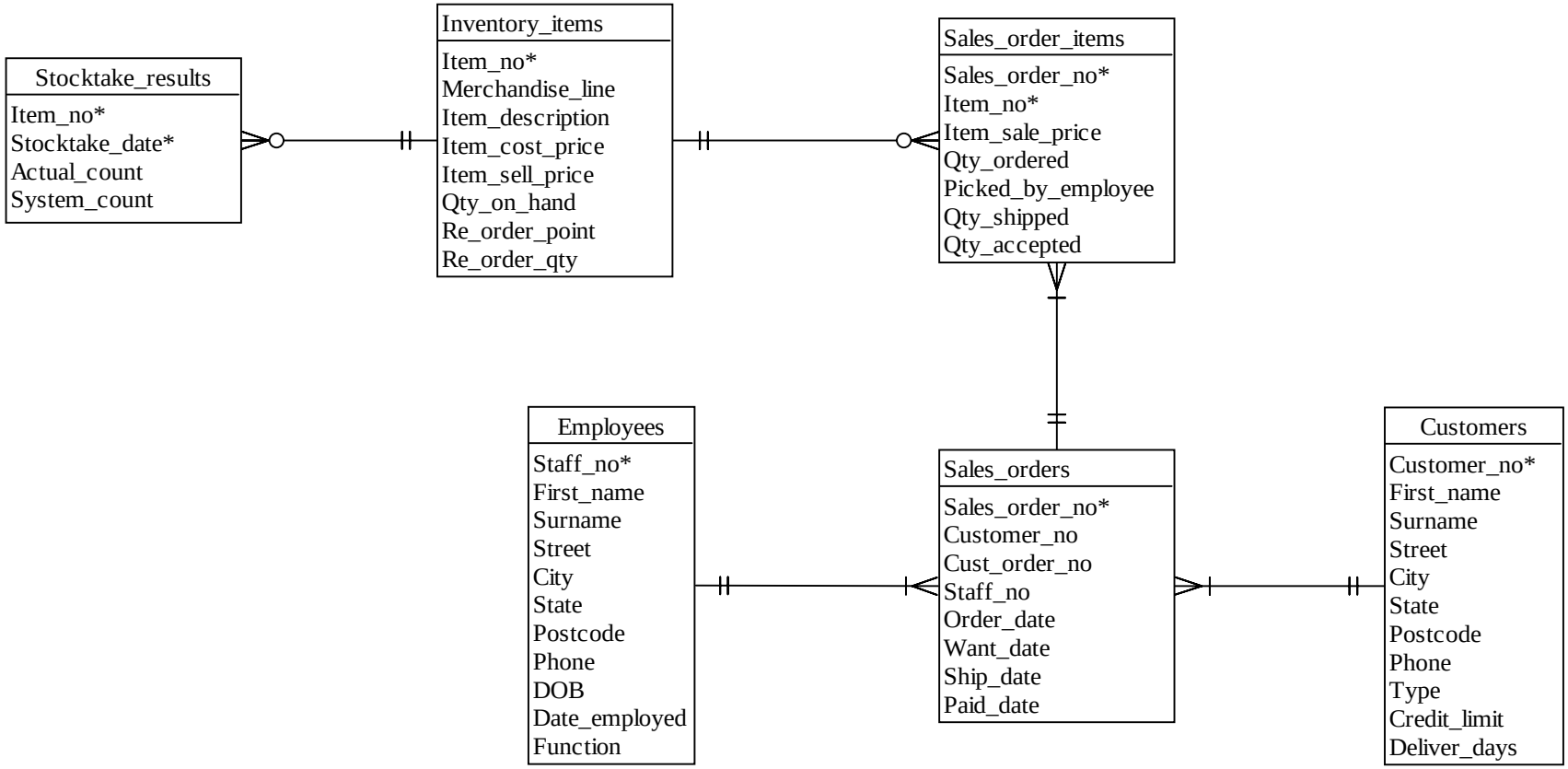
Wand, Y., V. C. Storey and R. Weber (1999) "An Ontological Analysis of the Relationship Construct in Conceptual Modelling", *ACM Transactions on Database Systems,* 24, pp. 494-528.

Weber, R. (1997) Ontological Foundations of Information Systems, *Coopers and Lybrand Accounting Research Methodology Monograph.* Melbourne.

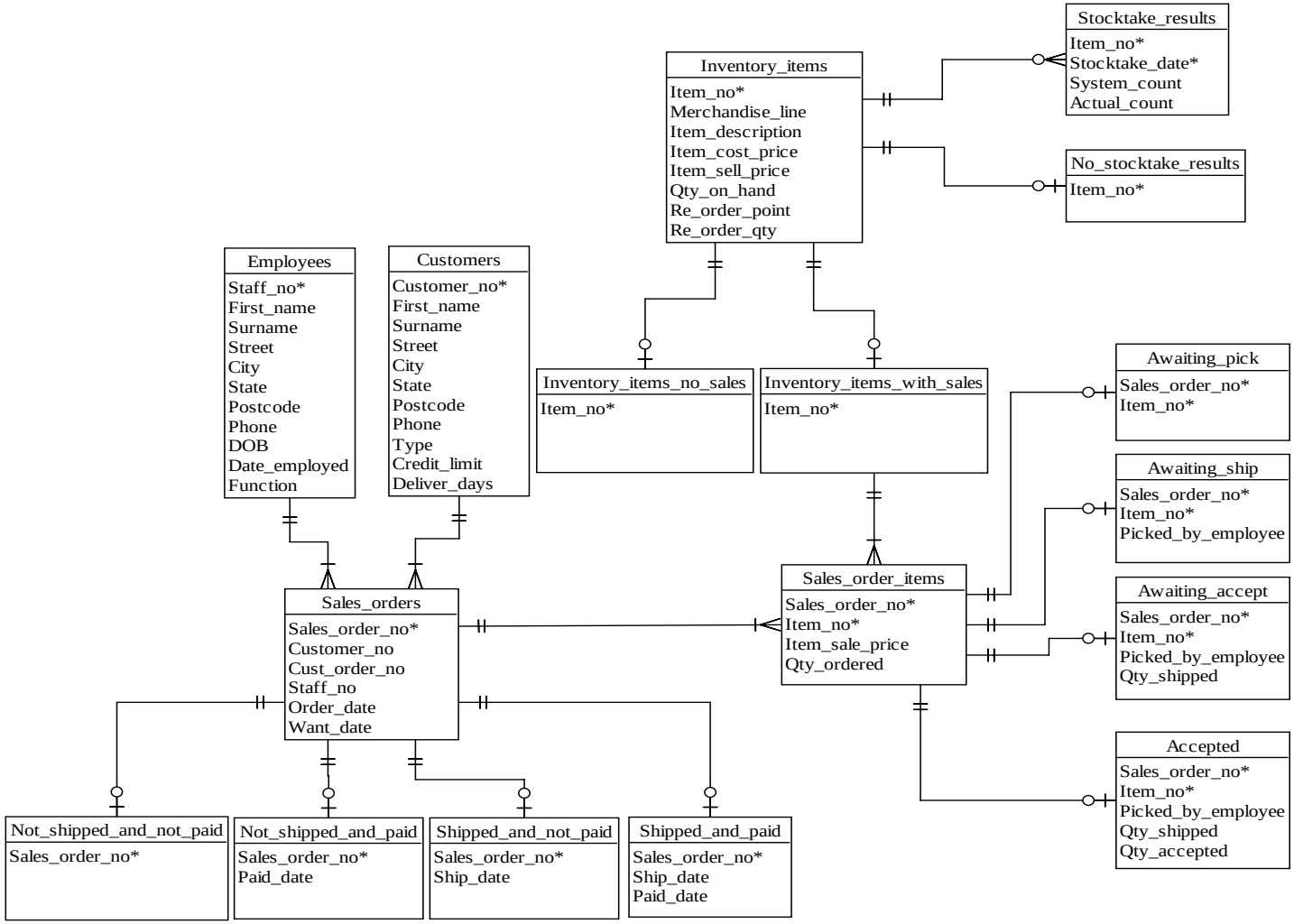
Weber, R. (2003) "Conceptual Modelling and Ontology: Possibilities and Pitfalls, Journal of Database Management, (14)3, pp 1

Weber, R. and Y. Zhang (1996) "An Analytical Evaluation of NIAM's Grammar for Conceptual Schema Diagrams" *Information Systems Journal,* (6)2, pp. 147-70.

APPENDIX I - DATA STRUCTURES
More traditional



Ontologically Clearer



APPENDIX II - INFORMATION REQUESTS & MODEL ANSWERS

Information Request	More Traditional	Ontologically Clearer
1. List staff number, firstname, and surname for employees employed after 10 August 2002.	SELECT staff_no, first_name, surname FROM employees WHERE date_employed > '10-aug-2002';	SELECT staff_no, first_name, surname FROM employees WHERE date_employed > '10-aug-2002';
2. List the sales order number, customer number, want date, and ship date for orders that are shipped and not paid	SELECT sales_order_no, customer_no, want_date, ship_date FROM sales_orders WHERE ship_date IS NOT NULL AND paid_date IS NULL;	SELECT sales_orders.sales_order_no, customer_no, want_date, ship_date FROM sales_orders, shipped_and_not_paid WHERE sales_orders.sales_order_no = shipped_and_not_paid.sales_order_no;
3. List the sales order number and the number of sales order items for items that are yet to be picked.	SELECT sales_order_no, COUNT(item_no) FROM sales_order_items WHERE picked_by_employee IS NULL GROUP BY sales_order_no;	SELECT sales_order_no, COUNT(item_no) FROM awaiting_pick GROUP BY sales_order_no;
4. List employee first name, surname, and the number of sales orders for sales orders that are paid but not shipped.	SELECT first_name, surname, COUNT(sales_order_no) FROM employees, sales_orders WHERE employees.staff_no = sales_orders.staff_no AND ship_date IS NULL AND paid_date IS NOT NULL GROUP BY first_name, surname;	SELECT first_name, surname, COUNT(sales_orders.sales_order_no) FROM employees, sales_orders, not_shipped_and_paid WHERE employees.staff_no = sales_orders.staff_no AND sales_orders.sales_order_no = not_shipped_and_paid.sales_order_no GROUP BY first_name, surname;
5. For items that have been accepted, list item number, sum of quantity shipped, and sum of quantity accepted.	SELECT item_no, SUM(qty_shipped), SUM(qty_accepted) FROM sales_order_items WHERE qty_accepted IS NOT NULL GROUP BY item_no;	SELECT item_no, SUM(qty_shipped), SUM(qty_accepted) FROM accepted GROUP BY item_no;
6. For sales orders that have not yet been shipped, list sales order number, customer number, customer first name, and customer surname.	SELECT sales_order_no, customers.customer_no, first_name, surname FROM sales_orders, customers WHERE customers.customer_no = sales_orders.customer_no AND ship_date IS NULL;	SELECT sales_order_no, customers.customer_no, first_name, surname FROM customers, sales_orders WHERE customers.customer_no = sales_orders.customer_no AND sales_order_no IN (SELECT sales_order_no

		FROM not_shipped_and_not_paid UNION SELECT sales_order_no FROM not_shipped_and_paid);
7. For inventory items that have not been picked and have never been included in a stock take, list inventory item number, item description, and quantity on hand.	SELECT inventory_items.item_no, item_description, qty_on_hand FROM sales_order_items, inventory_items WHERE sales_order_items.item_no = inventory_items.item_no AND picked_by_employee IS NULL AND sales_order_items.item_no NOT IN (SELECT item_no FROM stocktake_results);	SELECT inventory_items.item_no, item_description, qty_on_hand FROM inventory_items, awaiting_pick, no_stocktake_results WHERE awaiting_pick.item_no = inventory_items.item_no AND awaiting_pick.item_no = no_stocktake_results.item_no;
8. For orders placed after 1 August 2003 that have been accepted, list customer number, customer names, and the number of distinct orders where the quantity shipped of one or more items was more than 100 units less than the quantity ordered.	SELECT customers.customer_no, first_name, surname, COUNT(DISTINCT sales_orders.sales_order_no) FROM customers, sales_orders, sales_order_items WHERE customers.customer_no = sales_orders.customer_no AND sales_orders.sales_order_no = sales_order_items.sales_order_no AND order_date > '1-aug-2003' AND qty_accepted IS NOT NULL AND qty_ordered – qty_shipped > 100 GROUP BY customers.customer_no, first_name, surname;	SELECT customers.customer_no, first_name, surname, COUNT(DISTINCT sales_orders.sales_order_no) FROM customers, sales_orders, sales_order_items, accepted WHERE customers.customer_no = sales_orders.customer_no AND sales_orders.sales_order_no = sales_order_items.sales_order_no AND sales_order_items.sales_order_no = accepted.sales_order_no AND sales_order_items.item_no = accepted.item_no AND order_date > '1-aug-2003' AND qty_ordered – qty_shipped > 100 GROUP BY customers.customer_no, first_name, surname;
9. List the item number, item description and number of orders for items that have never been included in a stocktake but have been ordered more than 5 times and are on	SELECT inventory_items.item_no, item_description, COUNT(sales_order_items.sales_order_no) FROM inventory_items, sales_order_items, sales_orders WHERE inventory_items.item_no = sales_order_items.item_no AND sales_order_items.sales_order_no =	SELECT sales_order_items.item_no, item_description, COUNT(sales_order_items.sales_order_no) FROM sales_order_items, no_stocktake_results, inventory_items, shipped_and_paid WHERE sales_order_items.item_no = no_stocktake_results.item_no AND sales_order_items.item_no =

orders that have been both shipped and paid.	sales_orders.sales_order_no AND ship_date IS NOT NULL AND paid_date IS NOT NULL AND inventory_items.item_no NOT IN (SELECT item_no FROM stocktake_results) GROUP BY inventory_items.item_no, item_description HAVING COUNT(sales_order_items.sales_order_no) > 5;	inventory_items.item_no AND sales_order_items.sales_order_no = shipped_and_paid.sales_order_no GROUP BY sales_order_items.item_no, item_description HAVING COUNT(sales_order_items.sales_order_no) > 5;
10. List sales order number, item number, percent of the item rejected, and the first name and surname of the employee who picked the item for sales orders that were paid before they were shipped and 50% or more of the item was rejected.	SELECT sales_orders.sales_order_no, item_no, first_name, surname, 100 * (1 – (qty_accepted/qty_shipped)) FROM sales_orders, sales_order_items, employees WHERE sales_orders.sales_order_no = sales_order_items.sales_order_no AND sales_order_items.picked_by_employee = employees.staff_no AND 1 – (qty_accepted/qty_shipped) >= 0.5 AND ship_date > paid_date;	SELECT sales_orders.sales_order_no, item_no, first_name, surname, 100 * (1 – (qty_accepted/qty_shipped)) FROM shipped_and_paid, sales_orders, employees, accepted WHERE shipped_and_paid.sales_order_no = sales_orders.sales_order_no AND sales_orders.sales_order_no = accepted.sales_order_no AND accepted.picked_by_employee = employees.staff_no AND 1 – (qty_accepted/qty_shipped) >= 0.5 AND ship_date > paid_date;
11. List customer number, sales order number, the number of items not yet picked, and the number of items not yet shipped for sales orders with number of items not yet picked less than 3 and the number of items not yet shipped greater than 10.	CREATE VIEW no_ap AS SELECT customer_no, sales_orders.sales_order_no, COUNT(item_no) AS await_pick FROM sales_orders, sales_order_items WHERE sales_orders.sales_order_no = sales_order_items.sales_order_no AND picked_by_employee IS NULL GROUP BY customer_no, sales_orders.sales_order_no HAVING COUNT(item_no) < 3; SELECT sales_orders.customer_no, sales_orders.sales_order_no, await_pick,	CREATE VIEW no_ap AS SELECT customer_no, awaiting_pick.sales_order_no, COUNT(item_no) AS await_pick FROM sales_orders, awaiting_pick WHERE sales_orders.sales_order_no = awaiting_pick.sales_order_no GROUP BY customer_no, awaiting_pick.sales_order_no HAVING COUNT(item_no) < 3; SELECT sales_orders.customer_no, awaiting_ship.sales_order_no, await_pick, COUNT(item_no) AS await_ship FROM sales_orders, awaiting_ship, no_ap

	COUNT(item_no) AS await_ship FROM sales_orders, sales_order_items, no_ap WHERE sales_orders.sales_order_no = sales_order_items.sales_order_no AND sales_orders.customer_no = no_ap.customer_no AND sales_orders.sales_order_no = no_ap.sales_order_no AND picked_by_employee IS NOT NULL AND qty_shipped IS NULL GROUP BY sales_orders.customer_no, sales_orders.sales_order_no, await_pick HAVING COUNT(item_no) > 10;	WHERE sales_orders.sales_order_no = awaiting_ship.sales_order_no AND sales_orders.sales_order_no = no_ap.sales_order_no AND sales_orders.customer_no = no_ap.customer_no GROUP BY sales_orders.customer_no, awaiting_ship.sales_order_no, await_pick HAVING COUNT(item_no) > 10;
12. List customer number, customer first name, customer surname, and percent of sales orders shipped after the want date for customers with more than 10 sales orders.	CREATE VIEW active_customers AS SELECT customers.customer_no, first_name, surname, COUNT(sales_order_no) AS numsalesorders FROM sales_orders, customers WHERE sales_orders.customer_no = customers.customer_no GROUP BY customers.customer_no, first_name, surname HAVING COUNT(sales_order_no) > 10; SELECT active_customers.customer_no, active_customers.first_name, active_customers.surname, (COUNT(sales_order_no)/numsalesorders) * 100 AS percentage_late FROM sales_orders, active_customers WHERE active_customers.customer_no = sales_orders.customer_no AND ship_date > want_date GROUP BY active_customers.customer_no, active_customers.first_name, active_customers.surname, numsalesorders;	CREATE VIEW active_customers AS SELECT customers.customer_no, first_name, surname, COUNT(sales_order_no) AS numsalesorders FROM sales_orders, customers WHERE sales_orders.customer_no = customers.customer_no GROUP BY customers.customer_no, first_name, surname HAVING COUNT(sales_order_no) > 10; SELECT active_customers.customer_no, active_customers.first_name, active_customers.surname, (COUNT(sales_order_no)/numsalesorders) * 100 AS percentage_late FROM sales_orders, active_customers WHERE sales_orders.sales_order_no IN (SELECT sales_orders.sales_order_no FROM shipped_and_paid, sales_orders WHERE shipped_and_paid.sales_order_no = sales_orders.sales_order_no AND ship_date > want_date UNION SELECT sales_orders.sales_order_no

		FROM shipped_and_not_paid, sales_orders WHERE shipped_and_not_paid.sales_order_no = sales_orders.sales_order_no AND ship_date > want_date) AND active_customers.customer_no = sales_orders.customer_no GROUP BY active_customers.customer_no, active_customers.first_name, active_customers.surname, numsalesorders;
13. List merchandise line, the percent of items in that merchandise line that have never been sold, and the percent of inventory cost of the items that have not been sold relative to the total inventory cost for that merchandise line. List only those merchandise lines with either percent of items that have never been sold greater than 10% or the percent of dollar cost of items that have never been sold greater than 20%.	CREATE VIEW totals AS SELECT merchandise_line, COUNT(*) ctot, SUM(item_cost_price * qty_on_hand) doltot FROM inventory_items GROUP BY merchandise_line; CREATE VIEW notsold AS SELECT merchandise_line, count(*) cns, SUM(item_cost_price * qty_on_hand) dolns FROM inventory_items WHERE item_no NOT IN (SELECT item_no FROM sales_order_items GROUP BY merchandise_line); SELECT totals.merchandise_line, 100*(cns/ctot), 100*(dolns/doltot) FROM totals, notsold WHERE totals.merchandise_line = notsold.merchandise_line AND ((cns/ctot) > 0.1 OR (dolns/doltot) > 0.2);	CREATE VIEW totals AS SELECT merchandise_line, COUNT(*) ctot, SUM(item_cost_price * qty_on_hand) doltot FROM inventory_items GROUP BY merchandise_line; CREATE VIEW notsold AS SELECT merchandise_line, count(*) cns, SUM(item_cost_price * qty_on_hand) dolns FROM inventory_items, inventory_items_no_sales WHERE inventory_items.item_no = inventory_items_no_sales.item_no; GROUP BY merchandise_line SELECT totals.merchandise_line, 100*(cns/ctot), 100*(dolns/doltot) FROM totals, notsold WHERE totals.merchandise_line = notsold.merchandise_line AND ((cns/ctot) > 0.1 OR (dolns/doltot) > 0.2);
14. For all customers list the customers first name and surname and, if they placed orders in September 2003, also include the total amount they have paid for	CREATE VIEW paid AS SELECT customer_no, SUM(qty_accepted * item_sale_price) dolpaid FROM sales_orders, sales_order_items WHERE sales_orders.sales_order_no = sales_order_items.sales_order_no	CREATE VIEW paid AS SELECT customer_no, SUM(qty_accepted * item_sale_price) dolpaid FROM sales_orders, sales_order_items, accepted, shipped_and_paid WHERE shipped_and_paid.sales_order_no =

orders placed during September 2003, and the total amount they owe for items they have accepted but have not paid for for orders placed during September 2003.	AND order_date BETWEEN '1-sep-2003' AND '30-sep-2003' AND ship_date IS NOT NULL AND paid_date IS NOT NULL GROUP BY customer_no; CREATE VIEW owe AS SELECT customer_no, SUM(qty_accepted * item_sale_price) dolowe FROM sales_orders, sales_order_items WHERE sales_orders.sales_order_no = sales_order_items.sales_order_no AND order_date BETWEEN '1-sep-2003' AND '30-sep-2003' AND ship_date IS NOT NULL AND paid_date IS NULL GROUP BY customer_no; SELECT first_name, surname, dolpaid, dolowe FROM customers, paid, owe WHERE customers.customer_no = paid.customer_no (+) AND customers.customer_no = owe.customer_no (+);	sales_orders.sales_order_no AND sales_orders.sales_order_no = sales_order_items.sales_order_no AND sales_order_items.sales_order_no = accepted.sales_order_no AND sales_order_items.item_no = accepted.item_no AND order_date BETWEEN '1-sep-2003' AND '30-sep-2003' GROUP BY customer_no; CREATE VIEW owe AS SELECT customer_no, SUM(qty_accepted * item_sale_price) dolowe FROM sales_orders, sales_order_items, accepted, shipped_and_not_paid WHERE shipped_and_not_paid.sales_order_no = sales_orders.sales_order_no AND sales_orders.sales_order_no = sales_order_items.sales_order_no AND sales_order_items.sales_order_no = accepted.sales_order_no AND sales_order_items.item_no = accepted.item_no AND order_date BETWEEN '1-sep-2003' AND '30-sep-2003' GROUP BY customer_no; SELECT first_name, surname, dolpaid, dolowe FROM customers, paid, owe WHERE customers.customer_no = paid.customer_no (+) AND customers.customer_no = owe.customer_no (+);

APPENDIX III - ERROR COUNTING FORM

Student ID	Question Number	Attempts

SEMANTIC

Keywords and Logical Operators

View	Select	From	Where	Group by	Having	Order by

Set Operators

Where	Union	Intersect	Minus

Symbols and Relational Operators

View	Select	From	Where	Group by	Having	Order by

Tables

View	Select	From	Where	Group by	Having	Order by

Attributes

View	Select	From	Where	Group by	Having	Order by

Values

View	Select	From	Where	Group by	Having	Order by

About the Authors

Paul L. Bowen, PhD, CPA, is an Associate Professor in the Business School at Florida State University. His primary research activities involve the measurement, control, and improvement of information quality. This research encompasses investigating strategies for improving end-user information retrieval performance. Paul's previous experience includes thirteen years at the University of Queensland and eight years as a systems analyst and project manager at the Oak Ridge National Laboratory.

Robert O'Farrell has recently completed a Masters Degree specializing in the area of information systems. Prior to undertaking his Masters he completed a Bachelor of Information Technology. Robert is currently employed by a local IT firm while still maintaining close ties to the University of Queensland via tutoring and research duties within the Business School.

Fiona H. Rohde, PhD, is an Associate Professor within the UQ Business School, The University of Queensland. Her primary research activities involve the area of outsourcing, and also the area of information management and its effect of information retrieval. Fiona worked for KPMG in the Computer Audit Division before joining the school approximately 15 years ago.

Journal of the Association for Information Systems
ISSN: 1536-9323