The Challenge Of Negation In Searches And Queries

Valerie J. Harvey (E-mail: harvey@rmu.edu), Robert Morris University
Jeanne M. Baugh (E-mail: baugh@rmu.edu), Robert Morris University
Bruce A. Johnston (E-mail: johnston@rmu.edu), Robert Morris University
Constance M. Ruzich (E-mail: ruzich@rmu.edu), Robert Morris University
Arthur J. Grant (E-mail: granta@rmu.edu), Robert Morris University

Abstract

Negation poses certain challenges for queries and searches. This paper deals with exclusionary queries implemented using the ISO database language SQL and a dialog-based interface and with retrieval searches involving negation. This research arose because instructors in database courses noticed a large proportion of students making mistakes on certain queries. The paper explores underlying comprehension issues and makes practical recommendations on identifying potential sources of error and avoiding incorrect or misleading results. Proposed actions include changes in general education and database training and encouraging implementation of the new SQL:1999 standard.

Introduction

Negation poses certain challenges for the process of formulating queries and interpreting query results. Herbert Clark (Stanford University, an expert on comprehension of negation) states in summarizing studies on negation, "The main result of these studies can be summarized succinctly: negation is more difficult to comprehend than affirmation."[1]) Margaret Donaldson (University of Edinburgh, an expert on learning comprehension of negation) commented, "few are likely to be aware of the effects that difficulties with negatives can have."[2]

Michael Chui studied the Boolean query interface used for Web searches and reports that "subjects generally have difficulty expressing Boolean queries, incorrectly expressing almost 40% of their queries in this condition." He notes an effect on correctness from the underlying Boolean forms and from the number of terms. Chui states that "negation makes queries more difficult to express."[3]

This research investigated the rates of student errors in writing exclusionary SQL and other queries and searches and examined psycholinguistic findings with regard to comprehension of negation. The research team included representatives from information systems, computer science, library science, linguistics, biomedical informatics, and communications. Examples are drawn from business and biomedical contexts.

Exclusionary queries in the ISO database language SQL

SQL (from "Structured Query Language") is the international standard database language. All SQL queries have at least two parts:

Readers with comments or questions are encouraged to contact the authors via email.

1) A SELECT clause (in the form of a list of attributes and expressions) which specifies the output of the query.

2) A FROM clause which specifies the database tables (in the form of a list of tables) to be consulted by the query.

Usually there is a third clause, a WHERE clause, which specifies particular conditions (restrictions) which apply to the query and the logical connections among tables in the database. This clause has the form of a condition, normally a compound condition using logical connectives (AND, OR) and may include negation. SQL is based primarily on set theory and formal logic statements.

A relational database schema is a specification of the structure of a database and consists of, for each table in the database, the name of the table followed by a list of attributes in that table). A schema may also document database constraints.

There is a class of exclusionary SQL queries that the majority of students (in and undergraduate database management systems (or even in graduate-level course on databases) will incorrectly implement in SQL. This is a practical teaching concern. These queries are called exclusionary queries because a portion of a set is excluded. Here are examples from texts commonly used in our department, with negation underlined:

- *Find the customer number, last name, and first name for every customer who did not place an order on October 20, 2003.*[4]

- *List the order number and order date for every order that was placed by Ferguson's but does not contain an order line for a gas range.*[5]

- *Show the names and salary of all salespeople who do not have an order with Abernathy Construction…*[6]

The meaning (semantics) of the natural language (English) query is often not correctly understood. Thus the meaning (semantics) of the SQL query implementation most frequently chosen by students does not match the meaning of the natural language query.

In general in logic, as well as in computer-based implementation of logic programming languages such as Prolog, it is recognized that the handling of complements presents certain challenges. Only the minority of database students will have had a course on logic, in which prepositional and syllogistic logic have been treated systematically and thoroughly. Results (from a group of 73) will be provided for student attempts to code four queries of a database consisting of two tables with the following schema:

The schema for the database is:

Customer (cust_id, cust_last_name, cust_first_name, cust_street, cust_city, cust_zip, cust_phone,
cust_credit_limit)

Invoice (inv_id, cust_id, inv_date, inv_amount, inv_status)

The queries are:

1) *List the first and last name of the customers from Pittsburgh*
2) *List the first and last names of the customers whose invoices are paid in full*
3) *List the first and last name of all customers who did not place an order on 1/10/2002*
4) *List the first and last name of the customers who placed an order on 3/2/2002*

This set of four queries was designed with the following in mind:

- There should be one single-table query (not requiring a join or subquery) and there should be one query involving both tables but not involving placing orders; these should provide a basis for comparison.

- There should be affirmative and negative queries of the type for which the negative query had been observed to lead to difficulty (in classroom experience with take-home exercises and in-class quizzes).

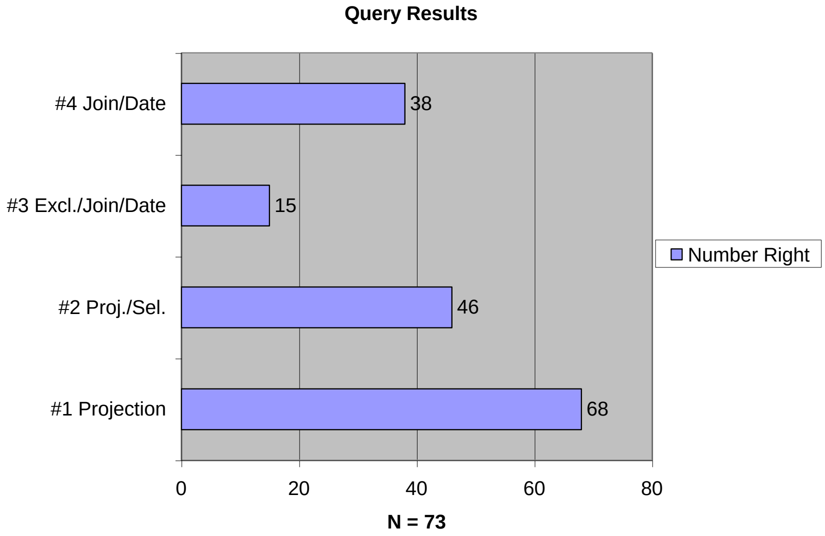
- The negative query should precede the affirmative query to reduce the possibility that the solution chosen for the affirmative query would be adopted inappropriately for the negative query.

73 students (in different sections, taught by different instructors) were assigned to write a set of queries as an in-class exercise. Most had some prior experience with relational databases in a previous course or from work. Some, particularly graduate students, had prior experience with SQL. All 73 students had received at least three prior weeks of instruction in SQL. They had prior practice with SQL.

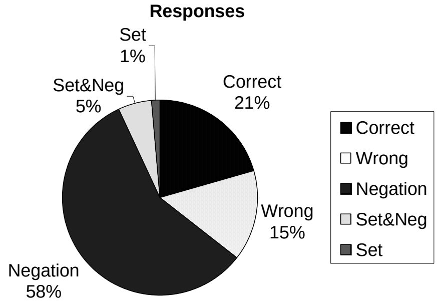
Results were as follows:

	Number Right	
#1 Projection	68	93.15%
#2 Proj./Sel.	46	63.01%
#3 Excl./Join/Date	15	20.55%
#4 Join/Date	38	52.05%
N	73	

Here are the results by query, expressed as a chart:



For query 3, here is a breakdown of the results:



The most common query design for the fourth (affirmative) query was the correct one (or an attempt at it, with some legitimate variations):

```
select cust_first_name, cust_last_name from customer, orders
where inv_date = '1/10/2002'
and customer.cust_id = invoice.cust_id
```

The most common query design for the third (exclusionary) query was an incorrect one (or an attempt at it, with some legitimate variations):

```
select cust_first_name, cust_last_name from customer, orders
where inv_date <> '1/10/2002'
and customer.cust_id= invoice.cust_id
```

The first task faced by learner is: What does this query mean? Possible interpretations are:

- *List customer information for every customer who placed an order on a day other than January 10, 2002.*

- *List customer information for every customer who placed an order on a day other than January 10, 2002 and did not place an order on January 10, 2002*

- *List customer information for every customer who did not place an order on January 10, 2002, including those who did not place any order at all.*

There are two issues here:

1) This is an example of set difference - we want to subtract the set of all customers who placed an order on January 10 from the set of customers we do not wish to include in the result. The set difference operation will assure that we do not include customers who placed an order on a different date and also on January 10.

2) We should be clear what set we are subtracting from: we need to include customers who did not place any order, not just those who placed an order, but on some other date. We should not assume that the

wording includes only customers who placed an order (but not on January 10) rather than “customers” (all customers, customers in general), even if they did not place an order. In systems analysis, if there is any doubt, the IT specialist responsible for the query needs to determine from the Subject Matter Expert (SME) what the precise meaning of the query is.

One correct form of the query could be:

```
SELECT cust_first_name, cust_last_name
FROM customer
WHERE cust_id not in
        (SELECT customer.cust_id
        FROM customer, invoice
        WHERE Customer.Customer_Number = Orders.Customer_Number
        AND inv_date = '1/10/2002');
```

Oracle's SQL offers a minus operator to implement set difference. In that case a correct query could be:

```
(SELECT cust_first_name, cust_last_name
FROM customer)
MINUS
(SELECT cust_first_name, cust_last_name
FROM customer, invoice
WHERE customer.cust_id = invoice.cust_id)
AND inv_date = '1/10/2002');
```

The meaning of both of these query implementations could be stated as:

List customer information for all the customers except those who placed an order on January 10th.

This “except” wording in English comes closest to the set difference operation of:

List (the set of) all customers minus the (set of) customers who placed an order on the 10th.

```
SELECT cust_first_name, cust_last_name
FROM Customer)                              ...defines the set of all customers

SELECT cust_first_name, cust_last_name
FROM Customer, Orders
WHERE Customer.Customer_Number = Orders.Customer_Number)
AND Order_Date = '1/10/2002'
              ...defines the set of customers who placed an order on January 10.
```

According to the ISO SQL92 and SQL:1999 standards[7], set difference could be expressed using the EXCEPT operator:

```
(SELECT cust_first_name, cust_last_name
FROM customer)
EXCEPT
(SELECT cust_first_name, cust_last_name
FROM customer, invoice
WHERE customer.cust_id = invoice.cust_id)
AND inv_date = '1/10/2002');
```

The following display shows three sets: those who hadn't placed any order, those who placed an order on any day other than the 10th, and those who placed an order on the 10th. The rightmost three columns show the customers included in the result of a correct set difference implementation of the query and those included in the negative join version of the query (syntactically correct but semantically incorrect) and identify spurious extra rows or omitted rows for the incorrect query implementation.

cust_id	cust_ first_ name	cust_ last_ name	Didn't Place Order	Placed Order Other Day	Placed Order Jan 10th	Not Jan on 10th, Set Diff.	Neg Join Version of Query	Errors in Neg. Join
1237	Greg	Smith		●	●		●	Extra Row
1439	John	Atkins		●		●	●	
1509	Gordon	Brown		●		●	●	
1987	Andrew	Ashcroft		●		●	●	
2908	Chris	Redding			●			
3451	Thomas	Jones		●		●	●	
5521	George	Murray	●			●		Omission
6745	Julia	Green		●	●		●	Extra Row
8967	Kevin	Johnson		●		●	●	
9780	Mary	Livingston			●			

If we examine the relation customer ¨ invoice we note that set of customer rows involved in the relation customer is not equal to the source (the entire customer set), while the set of invoice rows involved in the relation is equal to the source (the entire invoice set). These rows will not participate in a join.

During this project an effort was made to identify practical examples of exclusionary queries. Greg Shorr provided the following examples for the biomedical field:

1) Patients with certain findings who do not carry a certain diagnosis

   Find all patients with B/P > 140/80 X3 in the past year who do not carry the diagnosis of hypertension.

2) Patients with certain diagnoses that do not receive certain treatments

   Find all patients with hyperlipidemia who are not currently taking Aspirin.

3) Patients who carry certain diagnoses that do not seem to meet certain diagnostic criteria.

   Find patients who carry the DX of diabetes who did not have a fasting blood glucose in the year prior to the date of 1st diagnosis

4) Patients on certain therapies that do not meet certain diagnostic criterion.

   Find patients taking Synthroid without laboratory evidence of hypothyroidism in the year prior to initiation of therapy

Shorr noted that time constraints usually play a part in exclusionary queries.[8]

Exclusionary queries using a dialog-based query interface

The File Manager database system (often called VA FileMan), now in Version 22, was developed for the health care system of the U.S. Veterans Affairs Department and runs on platforms conforming to the ISO 10756 standard for the M or MUMPS language. File Manager is used in health care and biomedical research in a number of countries. The query interface of this system, unlike SQL, is a dialog invoked from the “Search File Entries” option. Some exclusionary queries have the property of dealing with a composite, such as an order with several lines possible. This example implements the following query:

List all the customers (name, number) who did not place an order containing a line for a widget (A4726)

The meaning of this query was agreed to be: The result should list all customers who did not place any order containing a line for a widget, no matter what else the order may have contained and no matter what other orders not containing a line for a widget they may have placed, and the customers who placed no order at all.

```
Select OPTION: SEARCH FILE ENTRIES
OUTPUT FROM WHAT FILE: CUSTOMER//
  -A- SEARCH FOR CUSTOMER FIELD: SALESORDER:
   By 'SALESORDER', do you mean the SALESORDER File,
       Pointing via its 'CUSTOMER' Field (“C” Cross-reference)? Yes// Yes
    -A- SEARCH FOR SALESORDER FIELD: PARTORDERED     (multiple)
    -A- CONDITION: EQUALS
    -A- EQUALS PARTLIST: A4726      WIDGET
    -B- SEARCH FOR SALESORDER PARTORDER SUB-FIELD:
  -B- SEARCH FOR SALESORDER FIELD:
-B- SEARCH FOR CUSTOMER FIELD:

IF: A// 'A  not SALESORDER PARTORDERED EQUALS 10 (A4726)

DO YOU WANT THIS SEARCH SPECIFICATION TO BE CONSIDERED TRUE FOR CONDITION -A-
        1) WHEN AT LEAST ONE OF THE 'SALESORDER' ENTRIES SATISFIES IT
        2) WHEN ALL OF THE 'SALESORDER' ENTRIES SATISFY IT
        3) WHEN ALL OF THE 'SALESORDER' ENTRIES SATISFY IT,
                OR WHEN THERE ARE NO 'SALESORDER' ENTRIES
    CHOOSE 1-3: 1// 3

DO YOU WANT THIS SEARCH SPECIFICATION TO BE CONSIDERED TRUE FOR CONDITION -A-
        1) WHEN AT LEAST ONE OF THE 'PARTORDERED' MULTIPLES SATISFIES IT
        2) WHEN ALL OF THE 'PARTORDERED' MULTIPLES SATISFY IT
        3) WHEN ALL OF THE 'PARTORDERED' MULTIPLES SATISFY IT,
                OR WHEN THERE ARE NO 'PARTORDERED' MULTIPLES
    CHOOSE 1-3: 1// 3
```

Notes on the search dialog: ' (apostrophe) means “not”; the colon (:) after SALESORDER is an instruction to navigate to/from another file referenced/referencing (or pointed to/pointing from) a field in the other file; in the A condition statement, 10 happens to be the unique ID (IEN or Internal Entry Number) for part #A4726; successively pressing the enter key without any entry backs out of the search dialog and completes it; // means default to be accepted by pressing enter. George Timson, the major designer of the File Manager system, was consulted in reviewing File Manager’s capabilities for correct implementation of exclusionary queries.[9]

What is important about this dialog-based approach is the protection offered by the questions as to when the search specification should be considered true and requiring the searcher to make a decision. This part of the dialog reflects whether the search is negative or not and whether the search involves multiples, since the hierarchical database model of File Manager systems supports multiples which are similar to relation-valued attributes in non-1NF relational database systems.[10]

Even with this protection, if there are errors in comprehension, as discussed above with regard to SQL, or errors in entering selections in the search dialog or in selecting the correct search specification conditions, the search will not produce correct results. In File Manager, it is important to choose carefully the file in which to begin a search. Jay Smith emphasizes, with regard to database queries and modeling in general, that it is vital to consider focal point and direction to assure correctness.[11]

Examples of false-positive retrieval searches of narratives and detection algorithms used in biomedical informatics

Full-text retrieval searches can be affected by the presence of negatives in conjunction with the target term.

In the clinical patient record, the medical narrative, as dictated by the health care professional, comprises an important element in the overall patient evaluation and treatment strategy. This narrative feature can be found in diagnostic observations, surgical notes, and discharge summaries – all of which are integral aspects of the patient record and which serve to augment laboratory test results. Even objective diagnostic results, such as imaging (X-ray, CT, MRI) outcomes, are subject to extensive narrative by the interpreting physician.

In recent years, a substantial portion of these narrative reports have been integrated into computerized patient record databases not only for individual retrieval functions, but also for aggregate clinical research purposes. Medical informatics researchers have been challenged in their efforts to implement exacting information retrieval search mechanisms due to several unique aspects of the traditional medical narrative. For example, as Mutalik notes, "for a medical document, the presence of a concept does not necessarily make the document relevant for that concept. The concept may refer to a finding that was looked for, but found to be absent or that occurred in the remote past." [12]

Simple automated concept indexing of medical narratives can also lead to many "false-positive" retrievals since health care personnel typically heavily record negatives in their narratives. For example, a simple search on the concept of "glaucoma" in a patient database would also yield a substantial number of reports stating "no evidence of glaucoma" or similar phrase. Furthermore, the ambiguities of syntax and subtleties of natural language exert an additional impact on the identification of negation in medical narratives.

Medical informatics researchers at both Yale University School of Medicine and at the University of Pittsburgh Center for Biomedical Informatics have developed and refined computer programs to execute negation detection algorithms on various types of narrative clinical medical reports.

At Yale University, P. Mutalik and colleagues have developed software called *Negfinder*, which is designed to recognize negated concepts when documents are concept indexed using the National Library of Medicine's (NLM) Unified Medical Language System (UMLS) Metathesaurus. Sixty documents containing 8,358 UMLS concepts were evaluated by Negfinder with 544 negations detected, of which only 13 were incorrectly flagged and 27 were missed. A specificity rate of 97% was achieved with 92.5% of all negations found with just four words ("no", "not", "without", and "denies/denied"). A careful analysis of the errors and missed negations reveals some of the subtleties of current detection algorithms, such as the impact of double negatives, non-standard language usage, and intervening phrases beyond the three word limit threshold to invalidate a negation hit.

Further refinements in the Negfinder algorithm include improved "noun-verb" differentiation and enhanced interpretation of antonyms (i.e. "non-smoker" or "anti-rejection") to reduce false retrievals.

W. Chapman and colleagues at the University of Pittsburgh center for Biomedical Informatics utilized a negation algorithm called *NegEx* applied to UMLS phrases in 42,160 dictated clinical reports in the MARS (Medical Archival System) database to analyze the effects of negation. Ten types of narratives from MARS, ranging from Emergency Department dictations to mammography reports to surgical pathology notes to history and physical exams, were included from reports logged by more than two thousand different physicians. While 60 negation phrases were triggered by NegEx, 90% were accounted for by only seven phrases ("no", "denies", "without", "not", "no evidence", "with no", and "negative for".) Precision was 97% when negation identified by NegEx was compared with manual physician review of 813 UMLS phrases.

The analysis concludes that ”because a large portion of all clinical findings mentioned in textual reports are negated, accurately identifying whether clinical observations are present or absent is critical to accurately extracting information from the reports”. [13]

Several medical informatics studies have demonstrated that the application of negation to various types of narrative medical reports requires an understanding of the ramifications of the concept, as well as the development of computerized negation detection algorithms which are sophisticated enough to flag variances in syntax, negation word placement, and semantics. While free-text medical documents present distinct challenges, the accurate identification of negation phrases is crucial to any automated retrieval system design in order to analyze these narrative reports for clinical and research purposes.

Problems of comprehension of negation in natural language

Patricia Wright states that instructions “can also be difficult to understand if they involve negation.” And that “[c]omprehension problems seem inevitable when negation is combined with frequency information.”[14] Wright and Bernard cite prior research showing “that instructions phrased with a negative component are more difficult to understand than their affirmative counterparts” and demonstrate that “the phrasing of a decision rule can have a substantial effect on performance with numerical tables.” Wright and P. Bernard speculate that the latency and accuracy differences of subjects in their research may have reflected internal “recasting” of a decision rule.[15]

The presence of ambiguities related to negation is acknowledged in communications skills textbooks and is the focus of exercises. Clark states that “negation is found in vastly different guises” and is “an extremely heterogeneous phenomenon.”[16]

In addition to the extensive research on comprehension of negation in natural language, consideration should be given to the particular properties of the use of negation in SQL. When negation is used in SQL queries involving NOT EXISTS or implied by a false result for a query involving EXISTS, there is an implicit Closed World Assumption necessary for statements about (truth or falsehood of) existence to make sense.[17] The assertion that some statement is false means simply that the particular statement is not now stored in the database being used. In the treatment of sets a “Universe of Discourse” must first be posited before, for example, the concept of a complement makes sense.

Recommendations on how to avoid errors and misinterpretations due to negation and exclusion in SQL

Jim Melton, editor of the ISO SQL standard, points out that the SQL:1999 standard provides the capability to write a positive Boolean expression, enclose it in parentheses, and append “IS FALSE” or “IS NOT TRUE” at the end, for example:

```
WHERE (some- Boolean-expression) IS FALSE
```

Melton and Alan Simon document this capability in *SQL:1999*.[18] They suggest that if a search condition “sounds funny” when read aloud, the phrase “this test is False” or “this one is true” could “sound better.” The implication is that such an exercise could help with comprehension. Melton and Simon acknowledge that the “more complex a search condition is, the more likely you will want to consider using the IS TRUE, IS FALSE, or IS UNKNOWN variations.” Melton and Simon define negation in SQL as “removing rows” as with EXCEPT, INTERSECT, DISTINCT, NOT EXISTS.[19]

H.-J. Klein describes concerns about the use of negation in SQL due to a “mixture of set based and logic based techniques used for SQL semantics” since three-valued logic is used to define the result of search conditions in WHERE clauses and Boolean logic in determining the value of a predicate including a subquery. He states, “Together with negation (NOT) or quantification (ALL) this may result in a critical loss of information.”[20] Klein makes recommendations on how to understand the meaning of a subquery or nested query.

Jalal Kawash, points out that SQL lacks a universal quantifier construct and therefore must express universal quantification through negating existential quantifiers and proposes a normal form for SQL which eliminates “undesired” forms of negation and universal quantification by applying well-known logical rules.[21] Kawash gives examples using a simple forestry database which he attributes to Bradley.[22] Rick Chow covers set difference in conjunction with set membership, which he calls the “membership test.” In his example of a query involving set difference “List the names of the faculty members who did not teach IS320 at all,” Chow cautions that care must be used in expressing set difference as a subquery, because of partial information, and gives examples of incorrect and correct implementations of the query.[23] Andrew Pletch also makes practical recommendations to students on handling set difference correctly in SQL queries.[24] Gerd Wagner covers a range of computational environments (programming languages, database query languages, modeling languages, production rule systems, and logic programming languages and shows that “negation in all these computational systems is, from a logical point of view, not a clean concept, but combines classical (Boolean) negation with negation-as-failure and the strong negation of three-valued logic (also called Kleene negation).” Wagner distinguishes “two kinds of negation: a weak negation expressing non-truth … and a strong negation expressing explicit falsity” and in Section 2.2, dealing with negation in SQL, he gives examples of SQL queries to show the behavior of negation in SQL with regard to weak and strong negation.[25]

Production environment database specialists were asked for input in order to gain a sense of production practice. Sharon Croke-Allsup confirmed negating an affirmative condition is a practical strategy in assuring comprehension and designing the query implementation: “It's a lot easier for someone to envision a query that picks something, rather than excludes something. So, I tell them, write the query that picks what you DON'T want, and then use that to reject records from another query.” [26] Mary Hoffman reported a strategy of using the outer join (or one-way or match-not-required join) of, for example, the set of all customers and the set of customers placing an order on the day in question. This can be a convenient way to isolate the complement in relational database systems that do not support the set difference operator.[27] Crystal Sloan and Rita Thissen recommended a similar approach.

Edward Barkmeyer of the U.S. National Institute of Standards and Technology indicated that the approach of Object Constraint Language (OCL) was relevant.[28] Jos Warmer of Klasse Objecten commented that OCL provides “select” and “reject” operations for the handling of subsets of collections.[29] According to the OCL language description (section 2.6.1), the select operation, given a Boolean expression, “specifies a subset of a collection” and “results in a collection that contains all the elements from *collection* for which the *boolean-expression* evaluates to true” while “with reject we get the subset of all the elements of the collection for which the expression evaluates to False.” The OCL language description in 2.6.1 further states that each “reject can be restated as a select with the negated expression.”

Recommendations regarding SQL products

The following recommendations are offered regarding database products incorporating SQL, in order to assure the most current database language capabilities with regard to exclusionary queries:

- The SQL set difference operator should be supported, preferably using the EXCEPT syntax of the ISO SQL92 and SQL:1999 standards

- The SQL:1999 standard should be supported; it includes the IS TRUE, IS FALSE, IS UNKNOWN syntax options.

Recommendations on the education of current and future users of information technology

Database education and training seeks to help information technology specialists (aspiring or current) learn how to use the syntax of SQL correctly so that meaning of an SQL query will accurately represent the corresponding natural language query of interest to an individual or organization.

1) In order to improve accuracy in the types of SQL queries of interest here:

a) Within the scope of computer education and training, draw attention to problematic queries and help learners recognize these queries (this will include publicizing the nature of the problem within the discipline in educational and production settings) and describe actions to take to assure accuracy, such as:

   i) students should use the particular SQL syntax needed (MINUS, EXCEPT, alternatives)

   ii) students should learn to ask for more information if the meaning is not clear

   iii) students should analyze the database design and query specification carefully

b) In view of the following, students should receive a balanced treatment of join and subquery implementations of multiple-table queries:

   i) some SQL products do not include the set difference operator,

   ii) as shown above, the most common errors in handling exclusionary queries come from simply negating the join,

   iii) the alternative to "misusing" the join in all systems, and the only option in systems lacking the set difference operator, is the subquery

2) In order to improve the fundamental capability of individuals to recognize and understand the logical issues associated with negation and exclusion:

   a) Within the scope of communications and natural language instruction (in English, Spanish, Chinese, etc.), modules in communication skills courses can identify and emphasize the logical and semantic properties of certain queries so that individuals learn to interpret meaning correctly or to recognize when more information is needed to determine meaning. Assumptions should be discussed for such queries as "*List the first and last name of all customers who did not place an order on 1/10/2002.*" The meaning of queries can be explored without any use of computers. Where such modules are in place, it may productive to show that the skills being acquired are of use not only in general discourse, but even in technical work environments, such as database programming.

   b) High school mathematics programs and college core curricula can include an introduction to logic (possibly within the context of a communication skills program) so that students become acquainted with the concepts of "set difference" and complements and with set operations in general. In some countries all students receive logic instruction covering sets, set operations, and predicate logic.

   c) Students should be shown examples of "false-positive" retrievals in searches of text databases (in library orientation, web search training, or communication skills courses), so they will understand how retrieval searches can be affected by negative expressions.

Recommendations on textbook database examples and exercises

Each textbook covering SQL queries should describe the SQL set difference operator and alternative syntax for expressing set difference when using SQL products that do not include a MINUS or EXCEPT operator. Although the EXCEPT version of the set difference operator was included in the 1992 SQL standard, it is not commonly implemented and only explicitly referenced in a minority of database textbooks.[30] Exercise sets should include exclusionary queries and the textbook databases should support query results which will clearly demonstrate the potential for error if an exclusionary query is not implemented correctly.

Even where the SQL:1999 IS FALSE syntax is not available, a statement form of that type can be used to analyze the meaning of a query and should be recommended. Statement forms using “except” can be used in analyzing meaning even when an SQL set difference operator is not available in a given database product.

Recommendations for production environments using databases

Exclusionary queries should be highlighted in staff development so that staff will notice them and give them special attention. The meaning of the natural language query should be determined and reviewed before beginning implementation. A particular effort should be made to review exclusionary query syntax and subject it to testing. Employees should be aware that errors in handling exclusionary queries are due to a convergence of comprehension and technical factors.

Conclusion

Observation of student performance in college/university database courses led to the recognition that certain types of queries, involving negation and exclusion, were associated with very high rates of error. In reviewing retrieval searches, it was found that negative expressions incorporating a target term could affect search strategies. Psycholinguists have been aware of particular problems with comprehension of statements involving negation. General comprehension challenges as well as the properties of database query interfaces contribute to errors with exclusionary queries and negation. Errors can arise from a failure to comprehend the original natural-language statement of a query or a failure to comprehend the meaning of a particular database language statement intended to implement the query or from both in the same instance. After an exploration of how queries can be formulated incorrectly and correctly and of what tools are available for formulating queries, recommendations have been proposed, addressing every opportunity for beneficial intervention, from general language skill and logic instruction to database instruction and production approaches. Given the possible adverse consequences of undetected errors in query implementation, this multifaceted approach seems prudent.

The valuable assistance of those whose e-mail correspondence is documented in the references is gratefully acknowledged.

References

[1] Herbert H. Clark, *Semantics and Comprehension* (*Janua Linguarum,* Series Minor, 187; Mouton, 1976), p. 19. Also, e-mail correspondence, January 13, 2002. Prof. Clark (Stanford University) is known for his work on the comprehension of negation.

[2] Margaret Donaldson (professor of linguistics, University of Edinburgh), e-mail correspondence, January 10, 2002. Prof. Donaldson is known for her focus on comprehension of negation.

[3] Michael Chui, “Pattern, Procedurality & Pictures: Factors Affecting Boolean Query Interface Design for the Web,” URL: http://www.cs.indiana.edu/~mchui/sigir99/ (January 12, 2002).

[4] Philip J. Pratt, *A Guide to SQL*, 6th ed. (Course Technology/Thomson Learning, 2003), p. 106

[5] Philip J. Pratt, *A Guide to SQL*, 6th ed. (Course Technology/Thomson Learning, 2003), p. 106

[6] David M. Kroenke, *Database Processing: Fundamentals*, Design, and Implementation, 8th ed. (Prentice Hall, 2002), p. 252

[7] Jim Melton and Alan R. Simon, *Understanding the New SQL: A Complete Guide* (Morgan Kaufman, 1993), pp. 171-173, and Jim Melton and Alan R. Simon, *SQL:1999: Understanding Relational Language Components* (Morgan Kaufman, 2002), pp. 256-259.

[8] Gregory I. Shorr, M.D. (president of Medical Informatics Consulting, Park City, UT), e-mail correspondence, July 23, 2002.

[9] George Timson (Consultant, U.S. Veterans Affairs Health Care System), e-mail correspondence, January 27, 2002.

[10] Valerie Harvey, "Implications of Non-1NF Extensions to the Relational Database Model for the MUMPS Standard and MUMPS Databases," *Proceedings of the 15th MUMPS Users' Group of Japan Meeting* (Supplement), 1988.

[11] Jay Smith, D.Sc., e-mail correspondence, Jan. 11, 2002.

[12] W. W. Chapman; W. Bridewell; P. Hanbury; G. F. Cooper; and B. G. Buchanan, “Evaluation of Negation Phrases in Narrative Clinical Reports.” *Proceedings of the American Medical Informatics Association Symposium* 2001: pages 105-109.

[13] P. G. Mutalik; A. Deshpands; and P. M. Nadkarni, “Use of General-purpose Negation Detection to Augment Concept Indexing of Medical Documents: A Quantitative Study Using the UMLS.” *Proceedings of the American Medical Informatics Association* 2001 (volume 8): pages 598-609.

[14] Patricia Wright, “’The instructions clearly state…’ Can’t people read?” *Applied Ergonomics* 12.3 (1981): 131-141.

[15] P. Wright and P. Bernard, “Effects of ‘More Than’ and ‘Less Than’ Decisions on the Use of Numerical Tables,” *Journal of Applied Psychology* 60, 5 (1975): 606-611. See also P. Wright and A. J. Hull, “Answering Questions about Negative Conditionals,” *Journal of Memory and Language* 25 (1986): 691-709.

[16] Herbert H. Clark, *Semantics and Comprehension* (*Janua Linguarum*, Series Minor, 187; Mouton, 1976), p. 19.

[17] Jeffrey D. Ullman, *Principles of Database and Knowledgebase Systems* (Computer Science Press, 1988), Section 3.10, The Closed World Assumption, pp. 161-164.

[18] Jim Melton (Oracle Corporation), e-mail correspondence, January 11, 2002. See Jim Melton and Alan R. Simon, *SQL:1999: Understanding Relational Language Components* (Morgan Kaufman, 2002), pp. 219-220.

[19] Jim Melton and Alan R. Simon, *SQL:1999: Understanding Relational Language Components* (Morgan Kaufman, 2002), p. 342.

[20] H.-J. Klein (Computer Science, University of Kiel, Germany), “How to Modify SQL Queries in Order to Guarantee Sure Answers,” URL: http://www.acm.org/sigmod/record/issues/9409/sql.ps. (January 12, 2002)

[21] Jalal Kawash (University of Calgary), in “Writing Complex SQL Queries that Require Universal Quantifiers,” URL: http://pages.cpsc.ucalgary.ca/~kawash/papers/wccce00.html (August 19, 2003)

[22] J. Bradley, *File and Database Techniques* (Holt, Rinehart, and Winston, 1982).

[23] Rick Chow (Division of Math and Computer Science, University of South Carolina Spartanburg), URL: http://www.uscs.edu/~rchow/sims421/Review/SQLReview.ppt, slides 20-22 (August 15, 2003).

[24] Andrew Pletsch (Computer Science, SUNY New Paltz), URL: http://www.mcs.newpaltz.edu/~pletch/libsqlnotes.pdf (August 15, 2003).

[25] Gerd Wagner (Eindhoven University of Technology, Faculty of Technology Management), “The Semantic Web Needs Two Kinds of Negation,” technical report at URL: http://tmitwww.tm.tue.nl/staff/gwagner/myruleml/SemWebNeg.pdf (August 19, 2002). See also G. Wagner, “A database needs two kinds of negation,” in B. Thalheim and H.-D. Gerhardt, editors, *Proc. of the 3rd. Symp. on Mathematical Fundamentals of Database and Knowledge Base Systems*, volume 495 of Lecture Notes in Computer Science (Springer-Verlag, 1991), pages 357–371.

[26] Sharon Croke-Allsup, e-mail correspondence, November 6, 2002; Crystal Sloan, e-mail correspondence, February 2, 2002; Rita Thissen, e-mail correspondence, February 2, 2002.

[27] Mary Hoffman, e-mail correspondence, February 4, 2002.

[28] Edward J. Barkmeyer (Manufacturing Systems Integration Division, NIST, Gaithersburg, MD), e-mail correspondence, August 20, 2002.

[29] Jos Warmer (Klasse Objecten, Soest, Netherlands), e-mail correspondence, September 2, 2002. See “Response to the UML 2.0 OCL RfP (ad/2000-09-03), Revised Submission, Version 1.5 (June 3, 2002)”, OMG Document ad/2002-05-09, Section 2.6 Collection Operations, p. 2-19.

[30] The EXCEPT SQL operator is presented in Abraham Silberschatz, Henry F. Korth, and S. Sudershan, *Database System Concepts*, 4th ed. (McGraw-Hill, 2002), pp. 145-146; Remez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems*, 3rd ed. (Addison-Wesley, 2000), pp. 258, 265.