1988

SEMANTICS FOR HANDLING QUERIES WITH MISSING INFORMATION

D. G. Shin
University of Connecticut

SEMANTICS FOR HANDLING QUERIES WITH MISSING INFORMATION

D. G. Shin
Computer Science and Engineering Department
University of Connecticut

ABSTRACT

Natural language (NL) queries formed by novice, inexperienced and occasional users tend to be incomplete, mainly because this class of users are not likely to be familiar with the functional or organizational specifications of the underlying database. A subclass of incomplete NL queries is identified, namely *queries with missing information*. The focus of the paper is on data semantics issues involved in handling the NL queries with missing information. In particular, the following issues are addressed: What kinds of semantics about the data are necessary for the system to determine what information is missing in a query? What techniques can the system employ to carry out the decision process? If the user fails to provide the answer to a supplementary information request, how can the system calculate an alternative way of requesting the supplementary information? An approach to solving these problems is also provided.

1. INTRODUCTION

A number of natural language (NL) query systems have been developed in an attempt to allow a wider user community to access the traditionally developed database (DB) systems. One assumption made in these systems is that the NL interface's function is simply to translate a user query into a valid retrieval procedure. When laymen are considered as potential DB system users, this may be a rather naive view. It is not likely that this class of users is familiar with the functional and organizational specifications of the underlying system and, therefore, their NL request may not be able to be directly mapped onto the appropriate system retrieval function.

NL queries become incomplete for a number of reasons. The expressions are vague or ambiguous, contain few related terms or undetermined references, use unknown, incorrectly or inconsistent terms, or are organized in a way that the system can not process efficiently. Among the possible different types of incomplete queries, a particular class of queries is of interest to us: queries which may not be able to be converted into the system's retrieval function due to the lack of information provided in the query itself. This class of NL queries is called *queries with missing information*.

Like other incomplete queries, queries with missing information should be refined at the system's direction, assuming novice, casual NL users are not familiar with the details of the underlying data. Designing such an NL interface entails investigation of research issues that generally fall into two broad research areas and a combination of the two areas: identification and organization of semantics and structural knowledge about the stored data, and development of a computational model that can maintain the coherence of the dialogue exchanged between user and system. In this work, we are only concerned with the issues associated with the former area.

Experimentation with an initial prototype construction has revealed that determining the information that is missing in a query is a problem that lies more in the aspect of the semantics of the underlying database than in the linguistic aspect of natural language understanding (Shin 1987).[1] This point is well illustrated by a typical example dialogue commonly encountered at the reference librarian's desk. Suppose a client asks the librarian about the location of a book. The librarian usually responds with a counter question requesting supplementary information such as "What year was the book published?" or "Is this a periodical?" Generally speaking, the response is not caused by the linguistic inability of the client, but by his/her inability to foresee that the requested information is critical to the question. By understanding the organizational structure of the books, however, the librarian knows that the information is crucial to locate the book.

In particular, we discuss three aspects of handling queries with missing information:

1) What kinds of semantics about the data are necessary for the system to determine what information is missing in a query?

2) What techniques can the system employ to carry out this task?

3) If the user fails to provide the answer to the supplementary information request, how can the system calculate an alternative way of requesting the supplementary information?

This paper is organized as follows. In Section 2, an example is presented which raises various semantic issues that will to be discussed. This is done by investigating a hypothetical dialogue between a novice user and an oceanographic database system. In Section 3, an extension is made to the relational data model. The notion of "F-domain connection" is introduced and, accordingly, the ordinary relational algebra is expanded to include a generalized version of join called "function join." Section 4 contains techniques for determining the missing information in a query based on the semantics known as functional dependencies. The notion of "F-domain connection" is also used in this process. Finally, Section 5 contains a summary and conclusion.

2. EXAMPLE

A dialogue fragment is given below to illustrate what is meant by a query with missing information and how such a query can be handled by an intelligent database interface. The domain of discourse between user and system is "oceanographic data."

Example 2.1: A dialogue fragment between user and system

(1) **User:** What is the pressure at the bottom of the ocean area B25?

(2) **System:** Do you know the ATM (Atmospheric Pressure) value of the region?

(3) **User:** It was 318 LB.

(4) **System:** How about the density of the water of the region?

(5) **User:** I don't know.

(6) **System:** Do you know the temperature of the water of the region?

(7) **User:** ...

The above dialogue leads to a brief discussion of three issues: what kinds of semantics about the data are essential to the system; how the system determines what information is missing in a query; and what strategy the system should employ to gather additional relevant information. These issues are discussed in the following subsections.

2.1 Organizational Knowledge about the Stored Data

When natural language is used to express a query, the user need not know the logical schemas of the database. The task of converting NL queries into correct system retrieval procedures belongs to the system. To perform this conversion, the system must contain knowledge about the way the data has been organized. If the logical entities' "relations" are considered to be the conceptual primitive objects of the database (in fact, this work is based on the relational data model), then the organizational knowledge about the data includes information such as what each relation is about, what property each attribute of a relation describes, how the attributes of relations are logically related to each other, what semantics associated with each domain, and what logical relationships exist between the domains.

Among these various types of knowledge, our concern is with the semantics associated with each domain and a particular type of logical relationship called "F-domain connection" ("F" stands for functional) that is to be maintained between a pair of domains. Given a user NL query, the system utilizes this knowledge to produce a low level system retrieval function (e.g., a relational algebraic expression) that is equivalent to the NL query. For example, given an NL query such as (1) in Example 2.1, the system determines which set of relations should be joined and how they should be joined. Such a decision will be possible only if the system contains sufficient information about the involved relations, their attributes and the domains from which the attribute values are drawn. Further discussion on this subject is deferred to Section 3.1, in which other notions of relational data models are presented.

2.2 User's Unawareness of Data Dependencies

One of the common mistakes that a novice user may make is failing to include enough information in his query to allow the system to derive an answer. A typical example of this type has been shown by query (1) of Example 2.1. Here the user is unaware of that the pressure at the bottom of the ocean depends on the ATM value and the density of the water and, therefore, the values for ATM and density should have been provided when inquiring the pressure.

Response (3) of Example 2.1 implies that the user actually knew the value of the ATM of the region. He simply did not make a mental association between the ATM value and the water pressure when formulating the query. When the user fails to notice such dependencies among the data and presents an incomplete query, an intelligent system should be able to identify what is missing and aid the user in completing the query.

2.3 Alternate Method of Processing the Missing Information

Missing information in a query is defined in a narrow sense as the information, when omitted, that makes it impossible for the system to derive the answer to the query. In a broad sense, the definition can also include the

information, when omitted, that leads to an unmanageable size of super set of the intended answer. If the system derives too large a set of answers, refinement of the answer set may be speculated. For this process, the system may require that further information be gathered from the user. The missing information in a broad sense will be illustrated in Section 4, in which Example 4.1 provides an appropriate background for the discussion. An example of the missing information in a narrow sense is given by the ATM value in query (1) of Example 2.1. Without the ATM value, no answer can be derived for the query.

When confronted with a query containing missing information, the system requests supplementary information from the user. The user may not be able to provide the requested information at this point, as was the case in (5) of Example 2.1. Even in such a case, the system may still be provided with the desired information by relying on other means of deriving the information. For example, when the value for density was not available from the user, suppose the system knows that the temperature of water determines its density. As an alternative way of knowing the density value, the system may request the temperature of the region. Such robustness in the system is feasible if other dependencies related to the concerned data are also stored and utilized.

3. SEMANTICS BETWEEN DOMAINS

The data semantic issues briefly mentioned in Section 2 are discussed in detail in this section. We first present the logical schemas of the oceanographic database which are the basis of the dialogue of Example 2.1 and which will be used throughout the rest of this paper. The logical schemas include:

OCEAN(AREA-ID,LATITUDE,LONGITUDE,BOTTOM-O)
OUTCOME(M-ID,DEPTH-C,TEMP-C,SSPD)
THEORY(DEPTH-T,ATM-T,DENSITY-T,PRESSURE)
DENSITY(ATM-D,TEMP-D,DENSITY-D)

The relation OCEAN contains information about the segmented ocean. AREA-ID specifies the identity of the individual ocean segments, LATITUDE and LONGITUDE indicate the location of the segment, and BOTTOM-O denotes its bottom depth. OUTCOME contains the results of measuring various oceanic information such as temperature (TEMP-O), density (DENSITY-O), and the speed of sound (SSPD) at certain depth intervals for each ocean segment. THEORY contains the proven relationship among depth, ATM, density and pressure in a tabular form. Similarly, DENSITY describes another proven relationship among ATM, temperature and density in a tabular form. Although THEORY and DENSITY could have been expressed in algebraic form, similar to the way many other oceanographic theories are expressed, tabular representation is adequate.

3.1 Explicit Domain Declaration

In the relational data model by Codd (1970), domains are abstract objects. Recently, Osborn and Heaven (1986) proposed a way of arbitrarily defining domains as abstract data types. Once domains are defined as data types, operations can be freely defined on the data types. Another way of specifying domains has been given by Qian and Wiederhold (1986) who define domains as a combination of data type and constraint declaration. What Qian and Wiederhold refer to as "domain constraints" are a means of representing the membership condition of each domain and also of specifying "instance of" abstraction or classification.

Recent trends reflect the need for specifying explicitly the semantics associated with domains. Once stored in the database, these semantics can be used to cause the system to behave in an intelligent way concerning various aspects of data manipulation and maintenance.

Domain declaration is a way of storing some general properties of an entity in an abstract form. What we propose here is an additional type of semantics that needs to be stored along with the domain declarations. These semantics describe a relationship that is observed between two sets of properties and that may be lost during the process of abstraction unless specified explicitly. This relationship is called a "F-domain connection" between a pair of domains (strictly speaking, from one domain to the other). Before we formally introduce the notion of F-domain connection, we illustrate by an example how the domains can be defined in an abstract form. Conceptual languages developed elsewhere (Albano, Cardelli and Orsini 1985; Borgida 1985; Osborn and Heaven 1986) can be used for this purpose and the following way of specifying domains illustrates simply an another way of defining the domain.

Example 2

Consider the relations OCEAN and THEORY. Let BOTTOM and DEPTH be two domains from which the values of the attributes BOTTOM-O of OCEAN and DEPTH-T of THEORY are drawn, respectively. During this abstraction process, there is some semantic connection implicit in the relationship from BOTTOM to DEPTH (this will be clarified shortly in terms of F-domain connection), yet each domain is defined individually. First, BOTTOM is defined as a set of integers with its unit being "feet" as follows:

Define Domain BOTTOM as
    type: integer
    unit: feet
    value: $OCEAN[BOTTOM-O]
    constraints include
        upper limit: 10,000

The argument of "value" virtually specifies the membership condition of the set BOTTOM in terms of a relational algebraic expression involving an existing relation. For example, $OCEAN[BOTTOM-O] means that the actual values for BOTTOM are the result of the projection of OCEAN over BOTTOM-T. When tuples are inserted into the relation OCEAN, the elements of BOTTOM are virtually instantiated. Another way of specifying the membership condition is simply to enumerate the values. For example, the values for a domain, BASIC-COLOR, are "red, blue, yellow." Integrity constraints can also be specified in the domain declaration.

The domain DEPTH is a set of integer intervals of the form $[n,m]$ which stands for an integer interval greater than or equal to n and less than m. The unit "feet" is also attached to the domain. The domain DEPTH is defined below along with some global variables n, m and k.

Define Variable n, m, k: integer

Define Domain DEPTH as
type: interval [n,m]
where $n = k \times 10$, $k \geq 0$, and $m = n + 10$
unit: feet
value: $THEORY[DEPTH]
constraints include
upper limit: max($THEORY[DEPTH-T])

Integer interval representation for DEPTH-T of THEORY is appropriate since otherwise the size of THEORY would be too large to manage (e.g., if the values for the depth are expressed in terms of real numbers).

3.2 F-domain Connection

F-domain connection is a way of representing a functional mapping between a pair of domains in an abstract form. This concept is formally introduced below. Some conventions are given first. A relational database schema is an ordered pair $<D,R>$ where D represents a set of domains and R represents a collection of relations such that for each $R \in R$, $R \subseteq D_1 \times ... \times D_n$, where $D_i \in D$, $1 \leq i \leq n$. A set of corresponding attributes is assumed to be implicit in each relation.

Definition 3.1 F-Domain Connection

For two domains D_1, $D_2 \in D$, if there exits a function f from D_1 to D_2, then it is said that D_1 is *F-functionally connected to* D_2, denoted by $D_1 \xrightarrow{f} D_2$, or there exists a *F-domain connection* (FDC) f from D_1 to D_2.

The following example illustrates this notion.

Example 3.2

The semantics attached to BOTTOM describe collectively all the values that are possible as the bottom depth of various locations of the segmented ocean areas. The domain DEPTH is a collection of integer intervals, each of which corresponds to a segment indicating a depth interval of the water in general. There is a FDC from BOTTOM to DEPTH such that given an integer value in BOTTOM it has a unique corresponding integer interval in DEPTH. For example, for a given value of BOTTOM, say 245, it is mapped onto the integer interval [240,250] in DEPTH. This FDC from BOTTOM to DEPTH is explicitly specified by the function, say CONV, as follows.

Define Variable a, b: integer

Define Variable h: BOTTOM

Define Variable k: DEPTH

Define FDC CONV as
from BOTTOM to DEPTH such that
if CONV(h) = k, where k is of the form $[a,b]$,
then $a = \lfloor h \text{ div } 10 \rfloor \times 10$ and $b = \lceil h \text{ div } 10 \rceil \times 10$

The above definition illustrates that variables are defined not only in terms of data types but also in terms of domain types. Data types may be considered generic domain types. A schematic illustration of CONV is given in Figure 1.

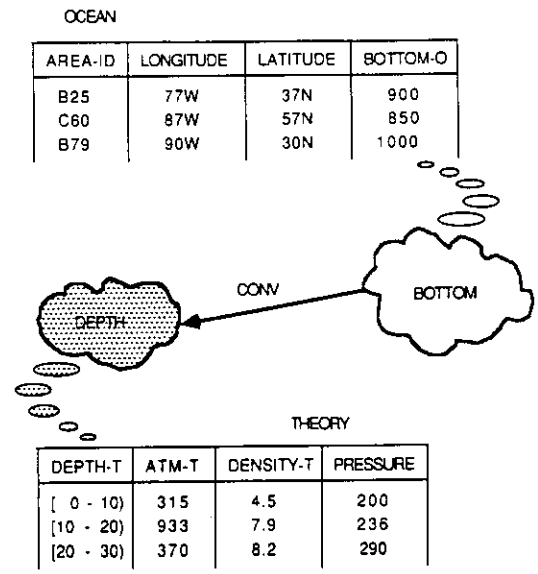

Figure 1. F-domain Connection CONV between BOTTOM and DEPTH

3.3 Function Join

Describing FDC explicitly allows joining two relations over some attributes which are not necessarily defined in an identical domain. This idea is formalized by extending the ordinary relational algebra (Ullman 1982) with an operation called a "function join." This concept is formally described below.

Definition 3.2 Function Join

Given R_1, $R_2 \in R$, let A_1 and A_2 be attributes of R_1 and R_2, respectively. Let D_1, $D_2 \in D$ be the domains of A_1 and A_2, respectively, and let $D_1 \not\perp D_2$. The function join of R_1 and R_2 over A_1 and A_2, denoted by $R_1 \overset{F}{\bowtie} R_2$, where F is of the form $\theta(f(A_1),A_2)$ with θ being some comparison operator, is the set of tuples in $R_1 \times R_2$ such that $f(A_1)$ for the values of A_1 in R_1 is θ-related to the values of A_2 in R_2.

Let F in the above definition be called the "condition phrase." The ordinary θ-join is a special case of a function join in which the function f in the condition phrase is the identity function. The above definition can be easily generalized into the case in which more than two attributes are involved in the join. Example 3.3 illustrates this idea.

Example 3.3

Consider again query (1) of Example 2.1, "What is the pressure at the bottom of the ocean area B25." This query can be translated into the following relational algebraic form:

((OCEAN{where AREA-ID = B25}) $\overset{F}{\bowtie}$ THEORY) [PRESSURE]
where F = EQUEL(CONV(BOTTOM-O),DEPTH-T)

where the brackets "{where ... }" and "[...]" are used to express "restriction" and "projection" operations, respectively. The F-domain connection CONV allows joining over the two attributes BOTTOM-O and DEPTH-T which are defined, respectively, on the two separate domains BOTTOM and DEPTH.

The discussion of how the NL query is actually translated into relational algebraic form is beyond the scope of this work. Such a topic should be discussed in the context of syntactic and semantic parsing of an NL query for the database.

4. FUNCTIONAL DEPENDENCIES AND MISSING INFORMATION

Functional dependencies are a well known concept that plays a central role in determining the missing information in a query. When a query involves more than one relation, the process of determining missing information requires the knowledge of the semantics associated with F-domain connections in addition to knowing functional dependencies. This process is discussed in detail below.

4.1 Determining Missing Information

A few notational conventions are given first. The relational algebraic language augmented with the operation "function join" is denoted by L_F. The L_F is used to internally represent input NL queries. A user provided NL query is denoted by Q_N and, when it is translated into L_F, the resulting expression will be denoted by Q.

We first consider when Q involves only one relation. The case when Q involves more than one relation will be discussed later. Let $R(Q)$ be the set of relations involved in Q and for each $R \in R(Q)$, let $F(R)$ denote the set of FDs which hold in R. For some FD, $X \rightarrow Y \in F(R)$, where X and Y can be composite, if Y appears in the projection list of Q, then X is called *qualification attributes of R with respect to the FD $X \rightarrow Y \in F(R)$* in Q. An attribute among the qualification attributes is said to be *restricted* if a restriction operation is made on the attribute in Q. For a fraction of X, say $X_Q \in X$ (possibly X_Q being empty), if X_Q are the only attributes restricted in Q, then it is said that the values for the attributes $X - X_Q$ are missing in the formal query Q. The attributes $X - X_Q$, denoted by $\dot{X}_Q[R]$, are called the *missing value attributes (MVA) of R with respect to the FD $X \rightarrow Y \in F(R)$* in Q. This notion is illustrated in Example 4.1.

Example 4.1

Let Q_N be "What was the temperature for the measurement id AX419?" This query involves the relation OUTCOME alone. Its translation into L_F is then

Q = (OUTCOME{where M-ID = A4X19}) [TEMP-C]

Figure 2 shows the FDs involved in the logical schemas presented in Section 3. The FD (M-ID, DEPTH-C) → TEMP-C holds in OUTCOME as it is shown in (b) of Figure 2. This FD means that the measurement id, in conjunction with the depth, determines the temperature, since for each measurement several temperatures are taken at different depth intervals (e.g., every 100 feet). The query Q shows that restriction is made on M-ID and projection is made over TEMP-C, yet no restriction has been made on DEPTH-C. With respect to the FD (M-ID, DEPTH-C) → TEMP-C, DEPTH-C is a MVA of the relation OUTCOME.

It should be noticed that MVAs in Q are not necessarily the missing information in Q_N. This issue is discussed briefly, although this subject is more relevant to the area of discussing the system's NL query translation into algebraic forms and the system's response calculation procedure. Suppose, given an NL query, its translation into an algebraic expression has been successful, revealing its corresponding MVSs. The identified MVSs are partial resources for the system to make a decision on whether any additional information is needed to answer the query. The decision process uses other criterion, such as the context of the query, the user's intention, the user's prior knowledge about the data dependencies and the user's questioning pattern. This point is illustrated below by using the query Q_N of Example 4.1, "What is the temperature for the measurement id AX419?" Two cases are analyzed: *CASE 1* for when the values for the MVAs in

Q are interpreted as missing values in Q_N, and *CASE 2* for when the values for the MVA's in Q may not be interpreted as missing values in Q_N.

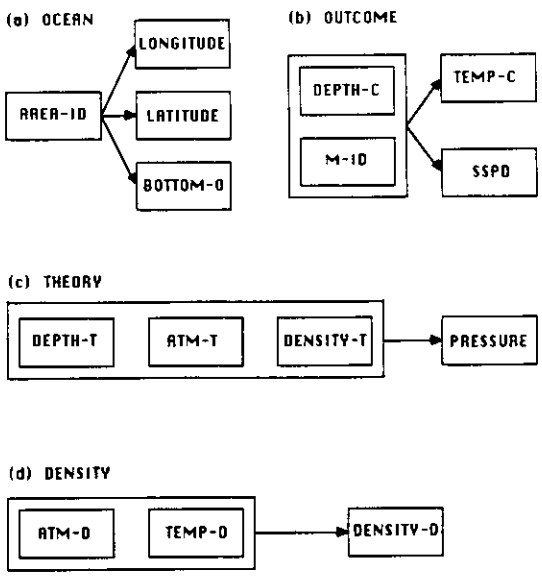

Figure 2. FDs for the Oceanographic Database

CASE 1: Suppose the system is sure that the user does not know the FD, (M-ID, DEPTH-C) → TEMP-C, holds. Since the number of the main verb of the query Q_N of Example 4.1 is singular, the value for the MVA, DEPTH-C, is indeed considered to be the missing information in the query Q_N. This is because otherwise (i.e., if the system is sure that the user knows the dependency) the system should have requested plural responses such as "What are the temperature values for the measurement id = AX419?" In the latter case, the values for the MVA, DEPTH-C, may no longer be considered as the missing information in Q_N, assuming the user may be interested in multiple conditional answers such as

At depth 100 ft, the temperature is ...
At depth 200 ft, the temperature is ...
At depth 300 ft, the temperature is ...

Even in this case, the values for DEPTH-C may possibly be considered as missing information. For example, if the answer set contains hundreds of screensize records, the system may need to engage in an answer refinement dialogue with the user to verify the user's intention based on the derived MVAs.

CASE 2: Suppose the system is sure that the user knows the FD, (M-ID, DEPTH-C) → TEMP-C, holds. Since the number of the main verb of the query Q_N is singular, the system may guess the user's query is based on his knowledge about a peculiar state of the underlying database. That is, only one DEPTH-C value may happen to correspond to that particular measurement id, AX419, in spite of the fact that, in general, there is more than one corresponding DEPTH-C value for each measurement id. If the user has no such prior knowledge about the database, the system should have been asked by a fully qualified query, such as "What is the temperature for the measurement id = AX419 at depth 150 feet?" Whether the value for DEPTH-C should be considered missing can only be determined after the system's actual evaluation of the query. If a unique value is returned, no information is missing in the query. Otherwise, two possibilities arise: either the system incorrectly assumed that the user was aware of the dependency, or the user might have incorrect knowledge about the state of the database and the query should have been asked in a plural form.

We now consider the case when Q involves more than one relation. In this instance, the notion of MVAs needs to be extended and the relations must be pairwise joined. Let two relations, R_1 and R_2, be function joined in Q where its corresponding condition phrase is of the form $F = \theta(f(A_2), A_1)$ in which A_1 and A_2 are the attributes of R_1 and R_2, respectively. Let X_1 be the qualification attributes of R_1 with respect to some FD $X_1 \rightarrow Y_1 \in F(R_1)$. For an attribute, $A_f \in X_1$, if $A_f = A_1$, then A_f is said to be *restricted by the condition phrase F*. For a fraction of X_1, $X_Q^1 \in X_1$ (possibly X_Q^1 being empty), if X_Q^1 is the only attribute restricted in Q, either by a direct restriction operation in Q or by a condition phrase, then $X_1 - X_Q^1$ are called the MVAs of R_1 with respect to the FD $X_1 \rightarrow Y_1 \in F(R_1)$. We illustrate with an example of how the MVAs are determined when a query involves more than one relation.

Example 4.2

Consider the following expression which was previously shown in Example 3.3.

((OCEAN\(where AREA-ID = B25\)) $\overset{F}{\triangleleft}$ THEORY) [PRESSURE]
where F = EQUEL(CONV(BOTTOM-O),DEPTH-T)

This expression shows that projection is made on the PRESSURE attribute of the THEORY relation. In the relation THEORY, FD (DEPTH-T, ATM-T, DENSITY-T) → PRESSURE holds as it is shown in (c) of Figure 2. Hence, DEPTH-T, ATM-T, and DENSITY-T are the qualification attributes with respect to the FD (DEPTH-T, ATM-T, DENSITY-T) → PRESSURE. The attribute DEPTH-T is restricted by the condition phrase F. Therefore, ATM-T and DENSITY-T are the only MVAs of THEORY with respect to the FD (DEPTH-T, ATM-T, DENSITY-T) → PRESSURE.

Once the MVAs are determined, the system uses them to determine the supplementary information to be requested from the user. In the above example, the system can request the values for ATM-T and DENSITY-T, as illustrated respectively by system responses (2) and (4) of Example 2.1.

4.2 Determining Attribute Value from Other Dependencies

When supplementary information is requested, it is possible that the user will be unable to provide the information. In this case, the system may request other information from which the desired information can be indirectly determined. The following demonstrates how this is done.

Given some MVAs $\dot{X}_Q[R]$, let X_D be a fraction of $\dot{X}_Q[R]$. For some relation R´, R´ $\neq$ R, let $Z \rightarrow W \in F(R')$ and $W = X_D$. If Z does not contain any MVA of R´ with respect to $Z \rightarrow W$ for the query Q, then it is said that the value for X_D in $\dot{X}_Q[R]$ is *derivable* from the value of Z. If Z contains an MVA, say $\dot{Z}$ of R´ with respect to $Z \rightarrow W$ for the query Q, then $\dot{Z}$ is called the *surrogate missing value attributes (SMVA) for* X_D.

Example 4.3

As was the case in user responses (3) and (5) of Example 2.1, at the system's request, the user was able to provide the value for ATM-T, but not the value for DENSITY-T. Although the value for DENSITY-T is not available directly from the user, the system can still derive this quantity if the values for both ATM-D and TEMP-D are known. This possibility is known to the system by the FD (ATM-D, TEMP-D) → DENSITY-D shown in (d) of Figure 2. Since the value for ATM-T is known, so is ATM-D. The system can conclude that the value for DENSITY-D is derivable indirectly once the value for TEMP-D is known. Therefore, system should request the value for TEMP-D. This point has been illustrated by the system response (6) of Example 2.1.

It is possible that the user may still not be able to provide the value for TEMP-D. In this case, TEMP-D becomes a SMVA for DENSITY-T. The system may now try to discover which attributes derive the value for TEMP-D.

5. CONCLUSION

A particular class of incomplete NL queries has been identified as "queries with missing information." This class of queries should be refined at the system's direction, assuming novice, casual NL users are not familiar with the details of the underlying database. Semantic issues which would be necessary to build a system capable of handling queries with missing information have been explored.

An extension has been made to the relational model by introducing the notion of "F-domain connection." Accordingly, the ordinary relational algebra has been expanded by including a generalized version of join called "function join." Techniques to determine the missing information in a query have been presented which utilize the semantics known as functional dependencies. When a query involves more than one relation, in addition to the functional dependencies, the technique requires the F-domain connection relationships which are embedded in the function joins. Once the system determines the information that is missing in a query, it can request supplementary information from the user.

6. ACKNOWLEDGEMENT

This work was partially supported by the National Science Foundation, grant number ECS-8401487.

7. REFERENCES

Albano, A.; Cardelli, L.; and Orsini, R. "Galileo: A Strongly-typed, Interactive Conceptual Language." *ACM Transactions on Database Systems*, Vol 10, No 2, 1985, pp 230-260.

Borgida, A. "Language Features for Flexible Handling of Exceptions in Information Systems." *ACM Transactions on Database Systems*, Vol. 10, No. 4, 1985, pp. 565-603.

Codd, E. F. "A Relational Model for Large Shared Data Banks." *Communications of ACM*, Vol 13, No 6, 1970, pp 377-387.

Osborn, S. L., and Heaven, T. E. "The Design of a Relational Database System with Abstract Data Types for Domains." *ACM Transactions on Database Systems*, Vol 11, No 3, 1986, pp 357-373.

Qian, X., and Wiederhold, G. "Knowledge-Based Integrity Constraint Validation." *Proceedings of VLDB*, 1986, pp 3-12.

Shin, D. G. "Discourse Pattern Analysis for Conversational Data-Retrieval." Technical Report CSE-TR-87-45, The University of Connecticut, Storrs, Connecticut, December 1987.

Templeton, M., and Burger, J. "Considerations for the Development of Natural-Language Interfaces to Database Management Systems." In L. Bloc and M. Jarke (eds.), *Cooperative Interfaces to Information Systems*, Berlin/Heidelberg: Springer-Verlag, 1986, pp. 67-99.

Ullman, J. *Principles of Database Systems*. Rockville, MD: Computer Science Press, 1982.

8. ENDNOTES

1. This observation is generally accepted in the Natural Language Interface development community. Templeton and Burger (1986) state "We believe that current [natural language interface] system development is limited by the need for good semantic modelling techniques."