Data Base Management, J. W. Klimbie and K. L. Koffeman, (eds.)
© North-Holland Publishing Company (1974)

SEVEN STEPS TO RENDEZVOUS WITH THE CASUAL USER

by

E. F. Codd
IBM Research Laboratory
San Jose, California

ABSTRACT: If we are to satisfy the needs of casual users of data bases, we must break through the barriers that presently prevent these users from freely employing their native languages (e.g., English) to specify what they want. In this paper we introduce an approach (already partially implemented) that permits a user to engage a relational data base system in a dialog with the objective of attaining agreement between the user and the system as to the user's needs. The system allows this dialog to be in unrestricted English so long as it is able to extract a viable quantum of information from the user's response. Immediately the system finds that the user's response is inadequately decipherable or clearly inadequate, it confronts the user with a multiple choice question. As soon as possible, the conversation reverts to unrestricted English.

1. Introduction

The aim of the work reported in this paper is to bring casual users into effective communication with formatted data bases. A casual user is one whose interactions with the system are irregular in time and not motivated by his job or social role. Such a user cannot be expected to be knowledgeable about computers, programming, logic, or relations. Neither can he be expected to be willing to learn an artificial language (even if it is oriented towards non-programmers) or artificial constraints placed upon a natural language such as English (and English is used as an example throughout this paper). The class of casual users is quite broad -- it includes almost all of institutional management (private or public) and housewives.

The only way to entice such a user to interact with a computerized data base is to permit him free use of his native tongue -- or at least the illusion of free use. The irregularity of his interactions implies that it would be unsafe to assume that he remembers anything he may have learned from his previous interactions with the system. Computer-assisted instruction techniques are therefore inapplicable.

It is very unlikely that any two English-speaking persons understand precisely the same English. In a sense, each of us understands a private (and usually extensible) subset of the language. We enjoy the illusion of using English freely as a consequence of our ability and willingness to participate in clarification dialog: that is, dialog that includes queries about previous utterances in the dialog.

Certain experimental systems (such as the Rapidly Extensible Language System [1] and Converse [2]) have been designed to support one-way communication (from the user to the system) in restricted English. The freedom in their English is orders of magnitude superior to that of the software products that support "English-like" query languages. Nevertheless, this freedom is generally considered to be adequate only for the kind of user whose involvement with the data base is of such a professional or job-oriented nature that he is willing to invest the effort of learning to live with the restrictions.

The few systems that do support dialog in English appear to be oriented towards other kinds of dialog (not the clarification type). To understand the distinction between these types, consider the following utterance by person A:

A: "I went to a downtown dealer yesterday and bought a very expensive car. Afterwards, I felt I should not have done it."

Now consider three alternative responses:

B1: "I am sorry you have feelings of regret. What will you do now?"

B2: "I also bought a car recently and discovered that it has a very high gas consumption"

B3: "A downtown dealer? Which one, and what do you mean by 'very expensive'?"

Response B1 is a stroking response -- it assures A that B1 has been listening to A and invites him to continue his story. The conversational pattern illustrated by A's utterance and B1's response is typical of stroking dialog. This is the kind of dialog supported by Weizenbaum's ELIZA [3]. Response B2 is a contributive response -- entirely new information is introduced. Winograd's SHURDLU [4] supports dialog involving contributive utterances and questions about changes of state in the given environment, but not questions about the meaning of previous utterances. Response B3 is a clarification response, an essential component of clarification dialog. In section 3 we discuss the introduction of a specific kind of clarification dialog as one of seven steps needed to support the casual user. First, however, we must introduce a sample data base and dialog about it for expository reasons.

2. Sample Data Base and Dialog

Consider a somewhat limited data base for expository purposes. Let it contain information about suppliers, parts, projects, and shipments of parts by suppliers to projects. The relation names appearing in the relational schema are SUPPLIER, PART, PROJECT, and SHIP. The attributes associated with each of these relations are shown in Table 1.

RELATION	D	ATTRIBUTE	COMMENT	
supplier	s	snumber	supplier serial no.	key
	x	sname	name	
	l	sloc	city (location)	
	r	rating	integer-coded rating	
part	p	pnumber	part serial no.	key
	y	pname	name of part	
	q	qoh	quantity on hand	
	q	qoo	quantity on order	
	t	ptype	classification of part	
project	j	jnumber	project serial no.	key
	z	jname	project name	
	l	jloc	city (location)	
ship	s	snumber	supplier serial no.	key
	p	pnumber	part serial no.	key
	j	jnumber	project serial no.	key
	d	date	shipdate	key
	q	qship	quantity shipped	

Table 1: Relational Schema

The comment gives a brief interpretation of the corresponding attribute and indicates whether or not it participates in the primary key of the designated relation. The column labelled D indicates the underlying domain from which the designated attribute draws its values. Note, for example, that qoh, qoo, qship are all defined on the same underlying domain, whereas supplier rating is defined

on a distinct domain, even though it too is a non-negative integer. This indicates to the system that any query that entails comparing supplier rating (for equality or inequality) with any of the attributes qoh, qoo, qship is semantically invalid.

Now consider the following sample dialog between a casual user and the data base as described above. We shall assume for this example that the user has an adequate idea of what each relation name and attribute denotes. However, we do not assume he knows what a relation is in the abstract sense. Neither do we assume he knows anything about the concepts of attribute, domain, relational algebra, etc.

SYSTEM: Please proceed

USER: Give me a list of the part numbers, quantities, and suppliers of Houston parts

SYSTEM: You are requesting (among other things) a listing of quantities of parts. Do you want:
1. quantities on hand
2. quantities on order
3. quantities shipped ?

USER: I want quantities on order

SYSTEM: Fine! Now, with regard to "Houston parts", do you mean:
1. parts supplied by Houston suppliers
2. parts supplied to Houston projects
3. parts associated with Houston in some other way ?

USER: I mean item 2

(Note: nothing prevents the user from responding in a manner which is incomprehensible to the system -- for example, "le deuxieme" -- but if he does, the system tightens up the dialog by: "Please reply by means of one of the numerals 1 or 2 or 3". How response 3 would be treated is discussed later.)

SYSTEM: Good! I need one more piece of information. In the listing of part numbers, quantity on order, and suppliers, how do you want suppliers to be identified:
1. by number
2. by name
3. some other way ?

USER: By name and by city

SYSTEM: O.K. This is what I understand your query to be: "Find all combinations of part number, quantity on order, supplier name, and supplier city such that the supplier supplied the part to a project located in Houston". Is this what you intended ?

(We now follow two possibilities. Case 1: the user is satisfied)

USER: Yes

SYSTEM: Do you want the listing to be ordered in any special way?

USER: Yes, by part number and then by supplier number

This ends the dialog in Case 1. The system now proceeds to interpret the relational calculus formula for the user's query -- a formula that it has been

synthesizing in its store, partly from the original statement by the user, and partly from the succeeding dialog.

(Case 2: the user is not satisfied)

USER: I have changed my mind

(Note: the system does not understand this idiom, but is fully aware of its inability to understand, so it tightens up the dialog as follows)

SYSTEM: Please select item 1 or 2 below:
1. the re-statement is satisfactory
2. the re-statement is not satisfactory

USER: 2

SYSTEM: If your intent can be expressed by making minor changes in my re-statement, please type it in with your changes incorporated. If your change is major, you may re-state your query in your own terms.

This is where we leave the dialog in Case 2.

We may now ask: is it feasible to support this kind of dialog at this stage of the software art? Can the program that attempts to analyze the user's statements and generate questions for the user be independent of the particular data base description (and hence work just as well for medical applications as for self-service airline reservations)? Can the program be sufficiently language-independent that it could be readily applied to dialog in any Indo-European language? The only way to provide convincing answers to these questions is to implement a system and attempt to apply it in several environments of quite different character and with different languages. We outline below seven steps that we believe will yield a subsystem that does support dialog of the type illustrated above, and that will be readily applicable in a wide variety of environments. It is too early to predict how language-independent the code will be. The author is implementing such a subsystem and it is already in partial operation.

3. The Seven Steps

Step 1: Select a simple data model

The user's view of the data in a formatted data base has a fundamental impact on the way he conceives and formulates queries and other types of transactions. For his sake, this view or data model should have enough structure to enable him to rapidly and concisely identify the part of the data base he is interested in — hence, more structure is needed than that of the Rand Relational Data File [5]. At the same time, the data model clearly should not have a multiplicity of structural alternatives for representing data. Such a multiplicity is incompatible with the casual user's unwillingness to consciously engage in a learning process and with his tendency to forget what he may have learned unconsciously, because of the irregularity of his interactions. The relational model [6], in which the data is structured as a collection of non-hierarchic relations of assorted degrees, appears to provide the right balance between too little structure and too much. It also requires very little in the way of description, as can be seen in the example of a relational schema exhibited in Table 1.

Step 2: Select a high level logic as internal target

When users (especially casual ones) are given freedom of expression in interrogating data bases in their native tongue, we can expect a large proportion

of their queries to be poorly formulated. These queries may be ambiguous, semantically incomplete, semantically incompatible with the data base description, or (perhaps most sinister of all) well-formed, complete, unambiguous, and meaningful, but not what the user really intended. The last of these possibilities is treated in step 4.

To enable the system to detect semantic incompleteness, incompatibility, or ambiguity, the user's source statements should be translated into an internal, precise language that is of the highest possible level. Data sublanguage ALPHA [7] is an example of such a language. It is based on the relational calculus for relations of arbitrary degree [8] and permits simpler tests for the semantic anomalies cited above -- much simpler than a procedural-level target language would permit. An additional advantage of selecting a high level, precise language as internal target is that the translation scheme can be oriented very heavily towards the semantics of the user's query rather than its syntax. For example, the translation scheme in the RENDEZVOUS system in its present implemented form is capable of interpreting English queries expressed in a large number of distinct ways, yet it is unable to distinguish a noun from a verb! By de-emphasizing syntax to a maximal extent, the RENDEZVOUS analyzer is able to interpret discontiguous fragments of a query that are separated by an undecipherable fragment. This capability is extremely important in that it permits the generation of more meaningful dialog for finding out what the undecipherable fragment is all about. This brings us to the third step.

Step 3: Introduce clarification dialog of bounded scope

Having a very limited understanding of English, the system must play the dialog game with finesse. It must keep the dialog closely tied to the data base description and the user's intended query. We call this clarification dialog of bounded scope, and discuss the tactics for it below for cases of incomprehensible words and constructions, semantic ambiguities, incompatibilities with the data base description, and semantic incompleteness. Underlying all these tactics is the following general rule: all questions to the user must be framed in the context of his utterances. Note that, although there is an attempt to limit the semantic range of user responses, there is no need to limit (and it is psychologically advantageous not to limit) the syntactic range, unless and until the user employs obscure modes of expression.

To illustrate tactics for incomprehensible words, suppose the system encounters the word "concerning" in a user's query and discovers that neither "concerning" nor "concern" is in its lexicon. It would be a tactical blunder of the first magnitude to ask the user: "What does 'concerning' mean?". This kind of system response encourages the user to display his talents for poetry or philosophy or linguistic analysis or profanity. Whether the user's response is banal, inarticulate, or erudite, it is extremely likely to be beyond the system's power of comprehension.

A less risky system response is a request to the user to re-phrase his query (or part of it) so as to avoid use of the word "concerning" (and any other words the system failed to understand). The disadvantage of this tactic is that it fails to provide the user with any clues as to what alternative modes of expression the system might be able to understand. Furthermore, it gives the user no reassurance that the system has understood any other part of what he said. The user cannot be expected to tolerate such sensory deprivation!

The most promising tactic for dealing with incomprehensible words appears to be that of temporarily ignoring them, translating the remaining comprehensible fragments into formal fragments, and piecing these together to determine what kinds of things are missing. In the case of incomprehensible constructions, any comprehensible components would naturally be exploited, but the overall construction would be temporarily ignored. If the piecing together of formal

fragments yields a complete query, the system can enter a verification phase to see if it satisfies the user without further embellishment (step 4 reveals how this is accomplished without exposing the user to the formal query itself).

If parts are missing from the collection of formal fragments generated so far, the system attaches priorities to the different kinds of missing parts and proceeds to discover these parts through dialog in order of priority. This problem will be discussed further in step 7.

An example of a simple ambiguity and the way it can be handled was illustrated in the sample dialog of section 2. To be able to detect the ambiguity of "quantities" and list its possible denotations in the context of the given data base, the system must have a table of such synonyms. Such a table can be thought of as an extension to the data base description -- specifically for casual user support, however. While the synonyms for "quantities" were all attributes, in general synonyms may be of various types (consider "number" for example, which may denote the function COUNT or various attributes such as PNUMBER).

To illustrate an incompatibility with the data base description, suppose that serial numbers of parts are required to have at least one digit and one letter. The phrase "part number 417" (or even "part 417") is, in all probability, a loose way of specifying the part with serial number 417. Since "417" contains no letter, it is invalid as a part serial number with this data base description. The system can point out this incompatibility and request the user to supply a proper substitute.

A more interesting case of suspected incompatibility is that of "Houston parts" in the sample dialog of section 2. While the incompatibility above was between an attribute (PNUMBER) and its value (417), the incompatibility here is between a relation name and a domain name, since PART does not have a domain LOCATION (the system can detect that "Houston" belongs in the domain LOCATION through inclusion in the data base description of a table of attribute-implying -- and hence domain-implying -- values for every attribute that has relatively few distinct values). Exploiting these tables, the system can offer to the user possible ways of associating "Houston" with "parts", suggesting first the simplest ways and then, if the user is not satisfied, more complicated ways. A simple metric can be placed on the attributes in the data base description for this purpose. In cases of suspected incompatibility, the system has to be prepared for outright rejection of its suggestions, and for responses of the following kind:

"I meant 'projects' instead of 'parts'"
"I should have said 'heavy' instead of 'Houston'"
"The query is correct as it stands"

In the last case, the system must assume that, for one reason or another, it has made an inadequate analysis of the associations between relation names, attributes, and values occurring both in the phrase under consideration and elsewhere in the user's query. Accordingly, it must now gain this information through further dialog with the user.

A simple example of incompleteness occurs in the dialog of section 2. In his original query the user asked for suppliers to be listed, but failed to indicate how they were to be identified. This omission was easily rectified by means of a simple question to the user. A more complicated case (and one of suspected rather than known incompleteness) is that in which combinations of things are requested without supplying an adequate specification of the combining criterion. For example, if the sample data base were extended to include information about employees and their assignments to projects, and the user asked for names of people and names of parts in combination without giving a criterion for associating people with parts, it would be reasonable for the system to point

this out and offer the user first the simplest possibilities of doing this, and then, if the user is not satisfied, the more complicated possibilities as discussed earlier in connection with incompatibility.

Step 4: Introduce system re-statement of user's query

Suppose the system has extracted whatever meaning it can from the user's query, and through clarification dialog has rectified suspected or known incompatibilities, ambiguities, and logical gaps. How can we be sure that the system has correctly interpreted the user's intent? It may have correctly interpreted the user's query, although even this is not certain. It is useless to expose the internally generated formal query to the casual user. The most appropriate check appears to be that of synthesizing from the formal query a precise re-statement in system English of the user's query as modified by subsequent dialog. Except for the simplest kinds of queries, the system's re-statement is likely to differ quite a bit in mode and precision of expression from the user's original formulation. This forces the user to consider his intent, and to examine carefully whether the system's version has captured that intent. If the user does not agree that his intent is properly reflected in the system's re-statement, the dialog must be pursued with fresh vigor. It can now be based on the re-statement if the user feels that would be the most expeditious way of getting the system to understand his intent.

Step 5: Separate query formulation from data base search

Notice that the combination of clarification dialog and re-statement of the query in precise English does not entail access to the main body of data in the data base. The actual search for the desired data is commenced only after the user and system are in complete agreement on the user's intent. In this manner the user can be protected from what may be expensive searches for information he does not want, and from the possibly embarrassing consequences of not realizing that either he has mis-formulated his intent or the system has failed to understand his query correctly.

Step 6: Employ multiple choice interrogation as fall-back

A casual user is likely to be impatient and unwilling to tolerate a long string of multiple choice questions concerning his query. Consequently, the use of such stylized dialog should be kept to a minimum. Nevertheless, when a user employs idioms or obscure words, this is the price he can expect to pay.

Because the domain of discourse is restricted to the data base description and because the possible types of components of a formal query are pre-determined, a finite (and more importantly a manageable) number of multiple choice questions can serve to extract a user's query if his only intelligible input is selection numbers, providing that the query does not require for its correct interpretation a function, relation, attribute, or definition not listed in the data base description.

Step 7: Provide a definition capability

In a non-dialog environment, it is the user who must observe the need for a definition and he or some other user acting for him must take the initiative in framing it and supplying it to the system's library of definitions. The Rapidly Extensible Language System [1] has an excellent definition capability for such an environment.

In the dialog environment we are discussing, the system must take on an extra burden: namely, that of detecting the need for a definition and engaging in clarification dialog to extract it from the user. Since definitions can be arbitrarily complex, we cannot expect to provide a complete multiple choice

fall-back for this aspect of understanding a user's query. However, casual user interaction will rarely involve complicated definitions, so the comprehension failure rate on this account will be quite low.

Consider the sample query: "Are any east coast suppliers supplying part 3G25?". Suppose the catalog of definitions does not include any definition for "east coast", "coast", or "east", and the lexicon does not include these words. According to the tactics discussed in Step 3, the system temporarily ignores these words in the query and attempts to decipher the remainder. In this case, the residual query is: "Are any suppliers supplying part 3G25?" and, let us assume, this is comprehensible. Accordingly, the system makes a re-statement and the user immediately expresses his dissatisfaction, noting that "east coast" has been left out (to guard against user oversight of this omission, the system can easily list words and phrases that it finds incomprehensible). At this point, the system may assume (with very little risk) that the phrase "east coast" is not only information bearing, but that it somehow qualifies "suppliers" in the original query. Drawing upon the data base definition, the system can now ask the user whether "east coast" has anything to do with a supplier's rating, location, etc. (all the non-identifying attributes of SUPPLIER), serial number, name (all the identifying attributes), or what he is supplying or capable of supplying (all the relations other than SUPPLIER that have the SUPPLIER key SNUMBER as an attribute). The user would select "location", and the system would then ask the user to tell it which supplier locations are "east coast".

Note that the system avoids asking the user point blank: "What is the meaning of east coast?", because a definition of "east coast" outside of the specific context of supplier locations would be too abstract for the system to comprehend and, in all probability, would cause the casual user a good deal of unnecessary mental anguish in trying to frame such a definition.

Now consider detection of the need for a definition when the residual query (after removal of incomprehensible words) is incomplete. We take the query: "Find the suppliers whose rating exceeds 5" and suppose that "exceeds" is not in the system's lexicon. In this case, the residual query clearly lacks a relational connective (e.g., =) to associate "rating" and "5" adequately. It can therefore offer the user the following initial options: "rating is (greater than) (equal to) (less than) 5". The user will always have to be given an escape option: "something else, if so what?". For example, a user who is a bit eccentric in his use of "exceeds" might respond: "I mean rating is much greater than 5". We now encounter two serious problems:

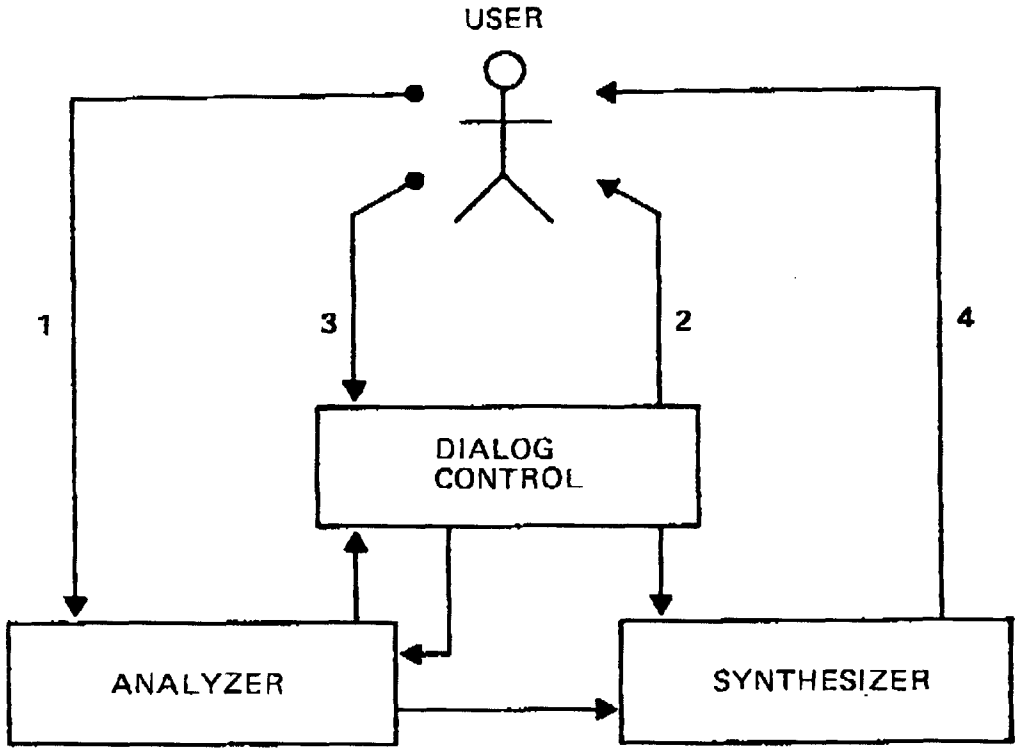
1. "Much greater than" is a fuzzy concept in the sense of Zadeh [9];
2. User eccentricity implies that a definition supplied by one user cannot be assumed to reflect the intent of another user -- it can, however, be used as an option for another user to select if he so desires.

Introduction of support for fuzzy denotations presents interesting and challanging problems that will not be discussed here.

4. The RENDEZVOUS Implementation

We shall only provide an outline of the implementation of the RENDEZVOUS query formulation subsystem. It should nevertheless be adequate for the reader to observe the marked differences between this and other systems that handle English input.

In Fig.1 we show the three principal components -- the analyzer, synthesizer, and dialog control -- and the four types of communication with the user -- statement of query by user, interrogation of user by system, response by the user to this interrogation, and re-statement by the system of its understanding of the query. Let us examine the operation of these components one by one.



1. User makes initial statement of his query (unrestricted English)
2. System interrogates user *about* his query (to obtain information which is missing or hidden in language the system does not understand, and to resolve ambiguities)
3. User responds to system interrogation
4. System provides a re-statement of user's query in system English (in a very precise way, based on the n-ary relational calculus)

Figure 1. RENDEZVOUS Subsystem

4.1 The Analyzer

The principal function of the analyzer is that of translating as much as possible of the user's original statement of his query into data sublanguage ALPHA. Other functions include the discovery and analysis of translation difficulties together with the generation of sufficient dialog parameters to enable the dialog control to piece together clarification dialog appropriate to the state of translation and the state of user interaction. The analyzer also provides interpretation service on the user's responses.

The analyzer has as its components the word transformer, the phrase transformer, and the diagnostic component. We now consider each of these.

4.1.1 Word Transformer

The word transformer is the first component of the analyzer to be invoked. Its first act is to store a copy of the user's query on one side, so that, as the need arises for dialog with the user, questions can be generated using as far as possible the very same terms he used. The word transformer then proceeds to reduce each word (independently of other words) to a normalized form, partly by means of standard suffix transformations and partly by replacing words that have a single canonical synonym by that synonym. Noise words, such as "please", are dropped altogether.

4.1.2 Phrase Transformer

The second component of the analyzer to be activated is the phrase transformer. It attempts, by means of phrase transformation rules, to recognize certain types of phrases in the text, and replace these by other phrases that are syntactically closer to the form required by data sublanguage ALPHA. The syntax of ALPHA is tabulated in the appendix. For an introductory description of ALPHA, see [8]; and for its foundation, see [9]. To facilitate the step-by-step transition from ENGLISH text to ALPHA and the handling of discontiguous decipherable fragments separated by an undecipherable fragment, we introduce the concept of a transition language INTER-ALPHA, whose syntactic types include all those of ALPHA plus certain additional ones that are extremely convenient as temporary "working" types. For example, INTER-ALPHA permits relation names to be used in many cases where ALPHA would require dummy variables having these relations as their ranges. We shall say that an English phrase has been inter-alpha-typed if its INTER-ALPHA type has been recognized, and alpha-typed if its ALPHA type has been recognized. Note that a very large number of English phrases are not alpha-typable, and need not be for this system to operate effectively.

The phrase transformation rules of RENDEZVOUS are multi-purpose. They always specify replacement of one kind of phrase (if it occurs in the text being examined) by another. However, in RENDEZVOUS many of the rules have either one or both of the following additional capabilities: first, the discovery in the text being examined of ambiguities, apparent incompatibilities, etc. that need to be resolved by dialog together with the generation of appropriate linguistic fragments for the dialog control to piece together into questions for the user; second, the storing of inter-alpha-typed phrases in a phrase dump, accompanied by their replacement in the text itself by surrogate symbols that identify their types and positions in the dump. Phrase dumping is reversible.

To understand the RENDEZVOUS phrase transformation rules it is necessary to observe that, after word normalization, the system assigns two lexical classes to every text word that it encounters. If the word is already in the lexicon, the appropriate pair of classes is found there. Otherwise the lexical class pair is computed and reflects only elementary properties of the word (such as whether it is composed of nothing but letters, nothing but digits, or a mixture of both). Most of the lexical classification reflects semantic, rather than English syntactic, properties of words. For example, lexical class 21 consists of names of linking relations -- a linking relation is one that has two or more attributes, each of which is a key of some other relation. A list of the presently existing lexical classes is included in the appendix.

Various kinds of values may appear in a query: for example, 'JONES' as the name of a supplier, 'X3' as a part number, and '15' as a quantity on order. Values that contain digits are easily distinguished from English words and are therefore easily recognized as values. On the other hand, values such as 'JONES' and 'BOSTON' will normally appear without quotes or other distinctive marks. This makes it difficult for the system to distinguish them from English words that do not happen to be included in the lexicon. Why not search the data base? With

the large data bases that we are assuming, query formulation would become unnecessarily expensive if a data base search had to be conducted to recognize every purely alphabetic value cited in the query. Now, certain domains (such as sex, cities in which suppliers and projects are located, etc.) have relatively few distinct values. For these domains we can afford a search at query formulation time, and accordingly we shall treat these values as belonging to the data base description — they are called cataloged values and appear in the lexicon as lexical class 2, while non-cataloged values have a computed lexical class of 1. For each domain the catalog indicates the known syntactic properties of its values, regardless of whether these values are cataloged or not. Nevertheless, there still remains a problem when the system encounters a non-cataloged, purely alphabetic value -- as in the phrase 'SUPPLIER JONES'. When such values occur in contexts that indicate a reasonable probability that they are values of this kind, the phrase transformation rules call (via dialog control) for dialog with the user to clarify the situation -- as in "IS JONES THE NAME OF A SUPPLIER?".

In the present implementation, there are eight classes of phrase transformation rules: the series 100 rules are the first to be candidates for application; when a state is reached in which no more series 100 rules are applicable, the series 200 rules become candidates; and so on, until no more of the series 800 are applicable. Roughly speaking, progression from one class of rules to its successor class corresponds to an advance of one level in the ALPHA syntax tree (see appendix). By this we mean that the goal of the successor class of rules is recognition of a higher level phrase type in INTER-ALPHA and, finally, in ALPHA. We have to say "roughly speaking", because the great variety of permissible word orderings in English is the source of some of the separation of rule classes, and also because the series 100 rules are solely concerned with ascertaining the query type (find set, test existence, or test truth other than existence) and the quota (if any).

Some of the phrase transformation rules entail checks of the semantic compatibility of phrase components with one another (using the data base description as the guide). As will be observed from the example below, apparent incompatibility can give rise to dialog with the user. Because of their strong semantic orientation, we call the left-hand sides of transformation rules semantic templates.

The example is intended to illustrate several features (but not all) of the phrase transformer. It is actual input to and print-out from an execution of RENDEZVOUS on November 1, 1973. The sample of dialog is not representative of the full dialog capability implemented at that time; also, it is only for expository purposes that we set the print control to provide intermediate print-out of rule number, text position (or origin) at which the rule is applied, the rule itself, and the new text after successful application.

Lines 1-2
The user types in his query. The word transformer converts the plural forms NAMES, SUPPLIERS to singular, and replaces SUPPLY by its canonical synonym SHIP (a relation name peculiar to this data base). A terminating symbol ω is appended, and the origin for template matching is set to word # 1 in the text. The phrase transformer is invoked with the series 100 rules activated.

Line 3
The word at the origin (GIVE) is used to find an applicable rule, the search time being kept small by appropriate indexing of the rules. Rule 128 is found to be applicable because GIVE is in lexical class 8, ME is in lexical class 19, THE is in lexical class 25. Note that, if THE had been omitted, a match would still have been obtained due to the presence of the null symbol '*' as an option in the template term '(25,*)'.

```
HSCAN Q08

GIVE ME THE NAMES OF SUPPLIERS WHO SUPPLY PART J5 TO A PROJECT
IN DETROIT
128 01    8 19 (25,*) → F
F NAME OF SUPPLIER WHO SHIP PART J5 TO A PROJECT IN DETROIT ω
132 01    8 → F ↓
F NAME OF SUPPLIER WHO SHIP PART J5 TO A PROJECT IN DETROIT ω
216 07    4 *(15,*) *1 → 1 ( B13↑E631H31 = 3 )
F NAME OF SUPPLIER WHO SHIP PART ( PNUMBER = J5 ) TO PROJECT IN
DETROIT ω
+220 PROJECT IN DETROIT?
IGNORE? NO
NEW PHRASE: PROJECT IN FRESNO
220 14    4^20 *(15,*) *2 → 1 ( B13↑E2311V31F31A3R3 = 3 )
F NAME OF SUPPLIER WHO SHIP PART ( PNUMBER = J5 ) TO PROJECT (
JLOC = FRESNO ) ω
405 02    6 OF (25,*) 20T1 → 4 B41↑E341R1
F SUPPLIER SNAME WHO SHIP PART ( PNUMBER = J5 ) TO PROJECT (
JLOC = FRESNO ) ω
508 05    21 20 ( U ) (15,*) 20L1 ( U ) → 1 ( 2 ( 4 ) ∧ ( 7 ( 9 ) ) ρ
F SUPPLIER SNAME WHO SHIP ( PART ( PNUMBER = J5 ) ∧ ( PROJECT (
JLOC = FRESNO ) ) ω
604 02    4 3 → 1 [ 2 ] ρ
F SUPPLIER [ SNAME ] WHO SHIP ( PART ( PNUMBER = J5 ) ∧ (
PROJECT ( JLOC = FRESNO ) ) ω
603 05    ] (17,*) (10,*) 21 → 1 [] 4 ρ
F SUPPLIER [ SNAME ] ∩ SHIP ( PART ( PNUMBER = J5 ) ∧ ( PROJECT
( JLOC = FRESNO ) ) ω
702 02    20 [ U ] → α62
 F SUPPLIER [ SNAME ] [] SHIP ( PART ( PNUMBER = J5 ) ∧ (
PROJECT ( JLOC = FRESNO ) ) ω
704 04    21 ( U ) → α61
 F SUPPLIER [ SNAME ] ∩ SHIP ( PART ( PNUMBER = J5 ) ∧ (
PROJECT ( JLOC = FRESNO ) ) ω
701 07    20 ( U ) → α60
 F SUPPLIER [ SNAME ] [] SHIP ( PART ( PNUMBER = J5 ) ∧ (
PROJECT ( JLOC = FRESNO ) ) ω
 F SUPPLIER [ SNAME ] [] SHIP ( PART ( PNUMBER = J5 ) ∧ (
PROJECT ( JLOC = FRESNO ) ) ω
```

Sample Input to and Print-out from RENDEZVOUS

November 1, 1973

Line 4
Application of rule 128 has resulted in the replacement of GIVE ME THE by F (a system-generated, non-English word denoting a find type query). The origin is kept at position 1 and the series 100 indexes are searched again.

Lines 5-6
Application of rule 132 has no effect on the text, but does terminate the series 100 rules with the origin still at position 1.

Line 7
The phrase transformer is invoked again, but now with the series 200 rules activated. Finding no matching rule for position 1, the system advances the

origin to 2, and so on, until at position 7 the template of rule 216 matches the phrase PART J5. We can account for this match as follows: PART is a relation name (lexical class 4); J5 is a non-cataloged value (lexical class 1). Note that no preposition (lexical class 15) is discovered in the text, but the null symbol '*' in the template permits a match in spite of this.

The right hand side of rule 216 specifies how the replacing phrase (P say) is to be constructed. The first RHS term '1' specifies as the first word in P the text word which matched the first term in the template (in this case, it was PART in the text which matched lexical class 4 specified by the first template term). The second RHS term '(' calls for a left parend to be appended to P. The third RHS term has two parts: B13 and E631H31. The first part B13 invokes a broadly applicable compatibility checking function B on arguments that, in this case, are the text words matching the first and third template terms -- the relation name PART and the value J5 respectively. Referring to the data base description, the function B finds that PART has at least one attribute for which J5 is a legal value (as evidenced by the syntax of this value), and that PNUMBER is the most likely attribute of the several with this property. Accordingly, PNUMBER is appended to the replacing phrase P.

If the value J5 had been invalid as a part number or as a part name, generation of the replacing phrase P would have been suspended, the code E631H31 would have been interpreted, and information compiled for communication to the dialog control (we shall consider an E code in more detail in section 4.3).

Since J5 was found to be valid, generation of P continues. The fourth RHS term calls for the symbol '=' to be appended to P, while the fifth and final RHS term calls for the text word (in this case J5) which matched template term 3 to be appended to P.

Lines 8-9

Application of rule 216 has resulted in the replacement of PART J5 by PART(PNUMBER=J5).

Lines 10-11

The template of rule 220 matches the text phrase beginning with PROJECT (word number 14, since each parend in the text counts as a word). In attempting to construct the replacing phrase, the function B discovered an incompatibility between PROJECT and DETROIT. Now, DETROIT is a cataloged value and the catalog indicates that it is a valid supplier location, but not a valid project location. Accordingly, information is passed to dialog control, which in turn emits the message to the user: PROJECT IN DETROIT? IGNORE? The user responds NO, because he discovers that he did make a mistake. In the fully supportive mode, dialog control provides significantly more information to the user (this will be discussed in section 4.3).

Line 12

Dialog control now requests the user to supply a new phrase, and the user responds with PROJECT IN FRESNO (once again, the fully supportive dialog mode would offer options).

Line 13

Rule 220 can now be successfully applied, because the catalog indicates that FRESNO is a valid project location.

Lines 14-15

PROJECT IN FRESNO has been replaced by PROJECT(JLOC='FRESNO').

Line 16

The phrase transformer discovers that no further rules in the 200 series or in the 300 series are applicable. The template of rule 405 matches the text phrase NAME OF SUPPLIER, since NAME is a synonym for at least one catalog item, OF in the text matches OF in the template, lexical class 25 is not matched, but the null alternative * permits matching to proceed, and 20T1 calls for the name of a non-linking relation (which SUPPLIER is) compatible with the text word matching the first template term (namely NAME).

The RHS of rule 405 generates first the word SUPPLIER (the text word matching the fourth template term) and then, by invoking B, the word SNAME (the most likely attribute of SUPPLIER synonymous with NAME, the text word matching the first template term).

Lines 17-18

NAME OF SUPPLIER has been replaced by SUPPLIER SNAME.

Line 19

The next rule found to be applicable is rule 508. The template term U matches the shortest phrase between the text word matching the template term immediately preceding the U and the text word matching the template term immediately succeeding the U. The template of rule 508 accordingly calls for a linking relation name (lexical class 21) followed by a non-linking relation name (lexical class 20) followed by a phrase enclosed in parends followed by a preposition or null followed by a non-linking relation name having a key attribute that is an attribute of the first-mentioned relation followed by a phrase enclosed in parends. The text phrase beginning with SHIP and terminating with the right parend immediately preceding the symbol ω fulfills these requirements.

The RHS of rule 508 generates a replacing phrase which places PART and PROJECT in the scope of the linking relation SHIP (by introducing extra parends) and which introduces logical AND to replace the preposition TO.

Lines 20-21

The substitution specified by rule 508 has now been effected.

Lines 22-24

The next applicable rule 604 now introduces some structure into the target list.

Lines 25-27

At this stage, rule 603 causes WHO to be replaced by a symbol that separates the target list from the qualification expression in a find type query. The resulting text is now completely converted to INTER-ALPHA. The extra steps needed to make the final conversion to data sublanguage ALPHA are quite straightforward, but were not yet implemented as of November 1973.

4.1.3 The Diagnostic Component

In the example described above, dialog was generated solely as a result of the invocation of compatibility-checking functions cited in the phrase transformation rules. Fortuitously, dialog generated in this way proved to be adequate to obtain a complete translation from the English source to INTER-ALPHA, and therefore to the ALPHA target language. A fundamental assumption in the RENDEZVOUS system is that, no matter how hard the designer of the lexicon and transformation rules may try to make these components complete with respect to all possible queries, he is going to fail. There will always be the possibility of strange words or constructions that contain information significant to the precise formulation of the user's query. Accordingly, we associate with certain classes of rules a recognition goal (for example, the separation of the target list from the qualification expression in a find type query). At the end of application of such a class of rules, the diagnostic component of the analyzer is invoked. It checks whether the corresponding goal has been attained and, if not, supplies appropriate parameters to the dialog control for the generation of dialog with the user.

Consider the following example: "Which parts have quantity on hand greater than 100, if we ignore parts of type X3?". Suppose the word and phrase transformers are incapable of deciphering the phrase "if we ignore", but can interpret the remaining fragments. Although translation of the first fragment by itself would yield a complete and sensible query, and although the phrase "if we ignore" contains no catalog items ("if" and "we" are, however, in the lexicon), this undecipherable phrase cannot be ignored by the analyzer, because of the isolated, dependent, and clearly information-bearing phrase at the end of the query. Detection of this situation comes about when the phrase transformer has completed the application of the series 600 rules. At this point, the diagnostic component

discovers that the present (partially translated query) consists of the following phrases and types:

Phrase	INTER-ALPHA Type
F	query type designator
PART	inter target list
:	separator
QOH > 100	qualification expression
IF WE IGNORE	undetermined
PART(PTYPE='K3')	qualified relation expression

Through table look-up the diagnostic component determines the various permissible ways of uniting the first contiguous sequence of recognized INTER-ALPHA expressions with the second (in this example there are only two such sequences). The options for unification sites are reduced to one, namely that the qualified relation expression be absorbed into the qualification expression, because the relation name PART is common to the two sequences (had this not been the case, absorption of the relation name in the second sequence into the inter target list of the first sequence would have been a second option). The options for unification linkages, in this example, are the 16 possible boolean connectives (keep in mind that the phrase "if we ignore" is completely unintelligible to the system). These options are obviously not exposed to the user all at once — the most likely options are exposed first, and a reasonably painless English phrasing is adopted rather than the more concise logical symbols.

More generally, when two INTER-ALPHA expressions are being united, they may be linked up as peers or one expression may be made a subexpression of the other. The linking may entail concatenation of (sub)expressions with or without the accompanying insertion of a boolean connective, relational connective, comma, period, or colon.

4.2 The Synthesizer

When the analyzer and dialog control have generated a complete query in data sublanguage ALPHA, the synthesizer takes over to translate back from the ALPHA statement to a version in English that is (hopefully) equally precise. To accomplish this, it employs additional sets of transformation rules, but essentially the same machinery as the analyzer.

No dialog is needed to complete this translation. The major problem encountered is that of generating reasonably fluent English, while not sacrificing precision. A minor problem is that of avoiding the generation of ambiguous English. In the case of intermixed ANDs and ORs (and possibly NOTs also) the very weak indications of scope that English affords can be augmented to advantage by judicious indentation of phrases on the output display.

When the RENDEZVOUS implementation is completed, it will be possible to type in a query in user-chosen English and observe on the display terminal the gradual step-by-step modification (a word or a phrase at a time) as the query is first translated into ALPHA and then back into English. The combination of these two translations represents a normalization in English that is not merely many-to-one — it is very-many-to-one.

4.3 Dialog Control

Dialog control provides a base for experimentation with dialog styles and tactics. Style includes such items as:

1. how fully the user is kept informed of the system's progress in understanding his inputs;
2. the extent to which a user is constrained in the way he can express his responses;

3. the amount of stroking or encouragement given to the user.

Users who have gained confidence in the system will often prefer the system to adopt a very brief style and omit stroking altogether.

Tactics include when to switch from one style to another, and whether to postpone posing a question to the user until more content has been examined. Development of alternative styles and tactics is still in a rudimentary state.

Now we propose to illustrate how the analyzer and dialog control work together to question the user about his query. We shall take the same query as cited in section 3.1.1. However, we select a more supportive dialog mode.

The phrase PROJECT IN DETROIT matches the template of rule 220. In evaluating the RHS of rule 220, function B is invoked with the relation name PROJECT and cataloged value DETROIT as arguments. A low-level subroutine B24 (where the digits 2, 4 are the primary lexical class numbers of the two arguments) constructs the internal message:

B24A DETROIT PROJECT

where B24 identifies the routine originating the message, A is a code indicating that the relation under consideration (PROJECT) does have an attribute (JLOC) defined on the domain (LOCATION) to which the cataloged value (DETROIT) belongs.

Upon receiving this message, a higher level routine in the analyzer uses the code A in B24A to select the first (E2311V31) of the two E-codes in the current RHS term of the current rule (220). The first part (E2) of this E-code is left undecoded at this stage. The second part (311V31) is decoded as follows:

'3' refers to the text word DETROIT matching the third term in the template;
'1' similarly refers to the text word PROJECT;
'V31' invokes a function that generates the set of all cataloged values in the domain LOCATION (the domain to which DETROIT belongs) that are admissible as values for the LOCATION attribute (JLOC) of PROJECT.

The resulting parameter list is accordingly:

3 DETROIT
1 PROJECT
1 PROJECT
V31 BOSTON, SEATTLE, FRESNO, HOUSTON

These parameters together with the rule number (220) and the E-code prefix (E2) are passed to dialog control.

In the supportive mode, dialog control uses the prefix E2 to select the message template:

WE HAVE NO INFORMATION ON U1 U2S, BUT WE DO HAVE U3S IN U4.

The symbols U1,U2,U3,U4 in this template are "slots" to be replaced by the four parameters from the analyzer. Replacement yields:

WE HAVE NO INFORMATION ON DETROIT PROJECTS, BUT WE DO HAVE PROJECTS IN BOSTON, SEATTLE, FRESNO, HOUSTON.

Dialog control now completes the message by concatenating the one above with:

SELECT ONE OR SUPPLY AN ALTERNATIVE PHRASE OR ASK FOR HELP.

This message is then displayed at the user's terminal.

5. Conclusion

Natural languages have evolved so far in an environment that places very little survival value on precision of expression. Consequently, we have to agree with Montgomery's position (which she stated with impressive erudition in [10]) that "natural language is not a natural query language". Nevertheless, for many (and probably the vast majority) of future users of computers, there is no alternative at this stage of evolution.

Previous work in interpreting queries stated in English -- at least, the work of which this author is aware -- has been based on two unstated assumptions:

1. whenever a user conceives a query, he is able to formulate it accurately in English right away -- that is, he will be able to convey his intent to the system faithfully and precisely at his first attempt;
2. if the user's English is beyond the restricted English understood by the system, it is the responsibility of the user alone to re-state his query in system-comprehensible English, whatever that is!

The first assumption might be tolerable in an application field such as document retrieval where the data retrieved in response to a query normally provides adequate feedback to the user for him to determine whether he has asked the right question. Unfortunately, the formatted, operating data bases of large institutions and enterprises do not have this reassuring property. When casual users are brought into contact with these data bases, we must decisively reject this first assumption.

The second assumption must also be discarded, because of the large psychological impact on the casual user when his query is rejected for what appears to him to be arbitrary reasons. If these two assumptions were applied together to the casual user, it would be tantamount to asking him to engage in blind flying without instruments!

In this paper we have outlined seven steps -- and described an implementation -- intended to bring casual users into effective interaction with large, formatted databases. The seven steps are:

1. Select a simple data model
2. Select a high level logic as internal target
3. Introduce clarification dialog of bounded scope
4. Introduce system re-statement of user's query
5. Separate query formulation from data base search
6. Employ multiple choice interrogation as fall-back
7. Provide a definitional capability.

These steps, particularly the clarification dialog and re-statement steps executed prior to accessing the main body of data, should protect users from the many pitfalls of applying a non-precision-oriented language to the precision-demanding task of query formulation.

6. Acknowledgment

The author is indebted to Professors Frederick Thompson and Bozena Thompson of the California Institute of Technology and to Mr. Charles Kellogg of the System Development Corporation, Santa Monica for explaining their respective systems (REL and CONVERSE) and for providing demonstrations of them in operation.

REFERENCES

1. F. P. Thompson, P. Lockeman, B. H. Dostert, R. Deverill, "REL: A Rapidly Extensible Language System", Proc. 24th ACM National Conference, New York, ACM 1969 pp 399-417.
2. C. H. Kellogg, J. Burger, T. Diller, K. Fogt, "The CONVERSE Natural Language Data Management System: Current Status and Plans", Proc. ACM Symposium on Information Storage and Retrieval, Univ. of Maryland, College Park 1971 pp 33-46.
3. J. Weizenbaum, "ELIZA -- A Computer Program for the Study of Natural Language Communication between Man and Machine", Comm. ACM 9, No. 1, January 1966 pp 36-45.
4. T. Winograd, "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language", MIT Project MAC, Cambridge, Mass., MAC TR-84 1971.

5. R. E. Levien, M. E. Maron, "A Computer System for Inference Execution and Data Retrieval", Comm. ACM 10, No. 11, November 1967.
6. E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", Comm. ACM 13, No. 6, June 1970 pp 377-387.
7. E. F. Codd, "A Data Base Sublanguage founded on the Relational Calculus", Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access, and Control, San Diego, available from ACM New York.
8. E. F. Codd, "Relational Completeness of Data Base Sublanguages", Courant Computer Science Symposia 6, "Data Base Systems", New York City, May 24-25, 1971, Prentice Hall.
9. L. A. Zadeh, "A Fuzzy-Set-Theoretic Interpretation of Linguistic Hedges", Journal of Cybernetics 2, 1972 pp 4-34
10. C. A. Montgomery, "Is Natural Language an Unnatural Query Language?", Proc. ACM National Conference, New York, ACM 1972 pp 1075-1078

ADDITIONAL BIBLIOGRAPHY

11. Randall Rustin (ed.), "Natural Language Processing", Courant Computer Science Symposia 8, New York City, December 20-21, 1971, Prentice Hall
12 R. F. Simmons, "Natural Language Question-Answering Systems: 1969", Comm. ACM 13, No. 1, January 1970 pp 15-30
13. S. R. Petrick, "Semantic Interpretation in the REQUEST System", IBM Research Report RC4457, IBM Research Center, Yorktown Heights, New York
14. F. P. Palermo, "A Data Base Search Problem", Fourth International Symposium on Computer and Information Science, Miami Beach, December 1972; also available as IBM San Jose Research Report RJ1072
15. J. B. Rothnie, "The Design of Generalized Data Management Systems", Ph.D. Dissertation, Dept. of Civil Engineering, MIT, September 1972
16. R. F. Boyce, D. D. Chamberlin, W. F. King III, M. M. Hammer, "Specifying Queries as Relational Expressions: SQUARE", Proc. ACM SIGPLAN-SIGIR Interface Meeting, Gaithersburg, Maryland, November 4-6, 1973; also available as IBM San Jose Research Report RJ1291

APPENDIX 1

ALPHA Syntax

Terminal Symbol Class	Class Abbreviation	Class Members
function names	F	
relation names	R	
tuple variables	T	
values	x	
attributes	A	
relational connectives	r	= ≠ < ≤ > ≥
logical connectives	I	∧ ∨ ⟶
quantifiers	Q	∀ ∃
query type symbols		f e t
punctuation		: . , ( )
negation		¬

attribute term — term = T . A

argument list — arglist = term or arglist,term

function term — fterm = term or F(arglist)

join term — jterm = (fterm r x) or (fterm r fterm)

boolfunction expression — bfexp = F(arglist)
where F denotes a truth-valued function

boolean expression — boolexp = jterm or bfexp or boolexp or (boolexp) I (boolexp)

qualification expression — qexp = boolexp or Q T (qexp)

target list — tlist = fterm or tlist,fterm

alpha expression — alphexp = tlist : qexp
where tuple variables appearing in target list are those which are free in qualification exp

range expression — rgexp = RANGE R T

range specification — rgspec = rgexp or rgspec,rgexp

quota — quota = integer

ordering expression — ordexp = arglist

find type query — fquery = f quota rgspec:alphexp:ordexp
where tuple variables in ordering expression must appear in target list of alpha expression

exist type query — equery = e r quota T:rgspec:qexp

truth test query — tquery = t rgspec:qexp

alpha query — alphaquery = fquery or equery or tquery

Note 1: all tuple variables appearing in the alpha expression of a find type query must also appear in the corresponding range specification

Note 2: similarly, all tuple variables appearing in the qualification expression of an exist type or truth test query must also appear in the corresponding range specification

Note 3: workspace names have been intentionally omitted from this syntax, because of their irrelevance to the topic of this paper

APPENDIX 2

RENDEZVOUS Lexical Classes

01 non-cataloged value
02 cataloged value
03 attribute
04 relation name
05 function name
06 cataloged synonym
07 do type word
08 find type word
09 units word (e.g., dollars, meters)
10 is type word
11 boolean word
12 comparator
13 location word
14 noise
15 preposition
16 quantifier
17 target-qualification separator
18 time word
19 pronoun
20 non-linking relation name
21 linking relation name
22 general synonym
23 fuzzy word
24 case type word
25 article

31 all-digits word
32 all-letters word
33 alphanumeric word

41 less type word
42 equal type word
43 greater type word
44 parends
45 system word (non-English)

Lexical classes 50 and up are the syntactic classes of INTER-ALPHA (these include the syntactic classes of ALPHA).

APPENDIX 3

Sample Queries

```
      QPRINT 'Q'

01  FIND THE NUMBER OF TYPES OF PARTS SUPPLIED BY TORONTO
SUPPLIERS WHO ARE CURRENTLY SUPPLYING OAKLAND PROJECTS
02  FIND OUT WHETHER PART X4 IS ON ORDER FROM DETROIT
03  FIND THE PART NUMBERS AND QUANTITIES ON HAND OF HOUSTON
PARTS
04  OF ALL THE PARTS SUPPLIED FROM DETROIT, WHICH ONES ARE
SHIPPED TO PROJECT HOPE?
05  IN WHAT QUANTITIES IS PART NUMBER 3 SUPPLIED?
06  DISPLAY SUPPLIER NAMES AND PROJECT NAMES INVOLVING THE
SUPPLY OF PART NUMBER 400
07  WHEN DID SUPPLIER JONES SUPPLY PART NUMBER 5 AND TO WHICH
PROJECTS?
08  DID SUPPLIER JONES SUPPLY PART 5 TO A PROJECT IN DETROIT?
09  IS IT TRUE THAT ALL SUPPLIERS WHO SUPPLY PART 2 ARE LOCATED
IN HOUSTON?
10  GIVE ME A LIST OF THE PARTS SUPPLIED IN 1971 ALONG WITH
THEIR SUPPLIERS
11  WHAT WAS THE TOTAL QUANTITY OF PART 4 SHIPPED BETWEEN APRIL
1972 AND JANUARY 1973?
12  PART NUMBERS, PTYPE, QUANTITIES SUPPLIED TO HOUSTON PROJECTS
13  HOW MANY SUPPLIERS SUPPLY PART 6?
14  WHERE IS PROJECT 8 LOCATED?
15  DOES THERE EXIST A SUPPLIER WHO IS SUPPLYING MORE THAN 50
KINDS OF PARTS?
16  FIND ALL THOSE CASES IN WHICH HOUSTON SUPPLIERS SUPPLY
TORONTO PROJECTS
17  GIVE ME A LIST OF PARTS SHIPPED TO PROJECT 8
18  ARE THERE NOT SUPPLIERS WHO SHIP PART 5 TO PROJECT 8?
19  IS IT THE CASE THAT SUPPLIER JONES SUPPLIES PROJECT 5?
20  WHO SUPPLIES PARTS OF TYPE 9?
21  HOW MANY PARTS NUMBERED 3 ARE SUPPLIED FROM HOUSTON?
22  FIND OUT HOW MANY OF PART X4 ARE ON ORDER
23  HOW MANY SUPPLIERS SUPPLYING PART M6 ARE LOCATED IN HOUSTON?
24  FIND THE PARTS HAVING QOH IN EXCESS OF 10 AND NONE ON ORDER
25  I WANT AT MOST 5 INSTANCES OF A TORONTO SUPPLIER SUPPLYING
A HOUSTON PROJECT
```

DISCUSSION

Delobel : In your example how can your system know that HOUSTON is a city name?

Codd : For those domains which have few distinct values, these values are stored with the Data Base Description along with the relevant attribute so that the system is able to recognize them as values.

Delobel : Is it possible to include dynamically new words in the Data Base Description?

Codd : No, not at present.

Tsichritzis : Can you make a comparison between your system and several other systems developed by Artificial Intelligence People?

Codd : This system is much less ambitious than the A.I. projects in that its domain of discourse is quite narrow and well-defined. One does not need much knowledge to interact with a formatted Data Base. On the other hand I am taking pains to avoid user frustration. the system takes very careful steps to keep the user there.

Langefors : How is your system helping the user understand what he wants?

Codd : There are two kinds of assistance : first the restatement mechanism and second simple display of informal explanation.