HETEROGENEOUS DATA BASE ACCESS

Thesis by

Alexandros Christou Papachristidis

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1984

(Submitted December 20, 1983)

ACKNOWLEDGEMENTS

I wish to thank my advisor, Professor Frederick B. Thompson, for his guidance and criticism of my ideas during this work. His encouragement and enthusiasm have also been a major contribution to the completion of this work.

Thanks are also due to Caltech, as a whole, that provided an ideal academic environment throughout my stay.

I would like also to thank my parents whose advice, guidance, and encouragement have been and will always be of great value.

Financial support for this thesis was provided by the Hewlett-Packard Company, Desktop Computer Division, under a contract with the California Institute of Technology.

ABSTRACT

This thesis presents a design for accessing commercially available data bases and other data base systems from a given, "local" data base system. Using this design, data from the local data base and from one or more "foreign" data bases will be integrated in framing the response to a single query. The design does not presume any standardization or integration of any kind on the part of a target data base.

An expert system has been designed, to be used by the applications programmer, for acquiring the information about a target data base, the communication path with the target computer, and the protocol for sending a data retrieval request. It also obtains from the applications programmer the nature of the data to be accessed and instructions for converting the incoming data into the "local" system's own format. This information is then associated with each word referring to data residing in the "foreign" data base, and is stored in the lexicon of the "local" system. The design also includes the extension of the underlying, host, system so that this information is properly used at the appropriate points in the processing of a user query.

State-of-the-art techniques used for interfacing heterogeneous data bases and natural language front-ends to data base systems are examined. The unique accomplishments of this thesis are identified. At the present time, there are no other systems, either commercial or research, that provide the heterogeneous access capabilities of the design presented here.

The design of Heterogeneous Data Base Access has been implemented as a part of the ASK System, **A** **S**imple **K**nowledgable System, providing natural language interface with the user.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1: AIM OF THIS THESIS

This thesis presents a system design for integrating the facility to access commercially available and other data base systems into a given data base system. This facility integrates the information contained in the response of these other systems in the processing of a given query. The type of the data base being accessed does not have to be the same as that of the accessing data base and no software level communication protocol restriction applies to that communication. Incorporated in the ASK system [Thompsons 83], this design has been implemented to the point where this feature can be demonstrated.

1.2: FOREIGN DATA BASES

The term *foreign data base* refers to any data base that is not an integral part of a given data base system. Foreign data base systems include, typically, both those where access rights are commercially available over public media, and pre-existing data base systems in the home environment of some data base system. Access to foreign data bases is highly desirable, especially when data both from the home data base itself and from foreign data bases are involved in the processing of a query.

A system's feature of accessing foreign data bases has to be distinguished from the case of front-end systems that can be attached to one data base system at a time and do not support their own data base environment. A system featuring foreign access, in the sense intended here, has to be a complete data base system, related to many other available foreign data bases simultaneously. This way, the answer to a single query may integrate data from the system's resident data base and requisite data from the other foreign data bases into the desired answer. The solution to the former problem, i.e., attaching a front-end to one data base only is, of course, a special case of the latter. A system providing foreign data base access will be called a *local system.*

1.3: NECESSITY FOR ACCESSING FOREIGN DATA BASES

As Adiba, et al., point out [Adiba 78], heterogeneity is a natural consequence of users' requirements implied by the evolution of techniques employed in computer technology, and the new products that come out of that. In our era, characterized by a struggle to acquire and process as much as possible information prior to making a decision, access facilities to data residing in computers around the globe have become a necessity. The business world does not only rely on its own information stored in its own computers, but also has to access other information-providing systems such as the Dow Jones News Data Base and others listed in [CSG 82] as well. That complies with the predictions of [Canning 73a] and [Canning 73b] foreseeing the movement of the program logic from the main computer (foreign computer) to the direction of remote terminals (local computers).

On the other hand, rapid changes in technology led to new systems showing radical changes as far as better usability and/or performance are concerned. The already commercially available data resources and those that will be becoming available in the near future are enormous. These resources neither are standardized nor can they be expected to be standardized. What is expected is quite the opposite. As mentioned in [Peebles 78], different systems could be modified to a standardized form, but few users are willing to do that. So, current systems that have consumed considerable amounts of time and money usually have to be abandoned in favor of new, "better" systems, before the initially scheduled time for withdrawing them from service. On the other hand, a shift to a new system implies considerable adaptation costs for the use of it, including expenses for data base structure changes and/or user training for the new interface. Under these conditions, one can see that there is a clear need for a system that would not make obsolete any current information system employed in a working, decision-making environment; a system that can perform its tasks in cooperation with any existing system, i.e., a system possessing the foreign access feature.

In 1973, Bachman resembled the data base programmer-user to a navigator trying to navigate through a data base by using as vehicles the various procedures of the system [Bachman 73]. In today's plethora of information supporting systems, the user's navigation problem becomes even more difficult, since the person has to go through the oceans of data and has to know all the information pertaining to the unfriendly seas of information written in the various data base manuals. Natural language systems have solved that problem considerably by shifting the burden of mediating between the view of the way the data are stored, and the way the user thinks about them, onto the system [Grosz 83]. But not all existing systems can show such a nice feature. The use of a natural language interface in a data base system provides automatic navigation and processing of the data, since no information on data location and procedures involved in data processing needs to be specified by the user. Bachman had stated that building a data base system may seem similar to restructuring the solar system with the optimal goal of minimizing the travel time between its planets. Here, it is stated that there is also a considerable need to find a system that can travel from one such solar system to another, That is, a system featuring the foreign access facility, providing for data base access to dissimilar data bases.

In some cases, access to a foreign data base may be particularly slow or expensive, e.g., over a long distance phone line, and the data involved sufficiently stable over time that it would be desirable to bring the data into the local system once and for all, rather than make repeated foreign accesses. There is also the case when a using installation recieves a new local system, it will often be necessary to transfer data residing in the installation's previous computer into the new local system's data structures, facilitating the shift to the new system. In these cases, there is a need for a *foreign input* facility, which will bring all of the desired data from a foreign data base and make it, once and for all, a part of the local system. Such a facility is but a slight modification of the foreign access feature.

1.4: OUTLINE OF THE REMAINING PART OF THE THESIS

In the next chapter the current status of accessing foreign data bases will be studied. Since access to a foreign data base implies access to a heterogeneous data base, the survey will use literature material referring to the distributed heterogeneous systems. Natural language systems being used as front-ends to data bases will be examined, and their characteristics and stucture will be analyzed. In both cases a study of existing systems on the subject will be made. The third issue to examine will be the search for logical components of a data base and the investigation of the minimal logical constituents of a data base system. The outcome of this examination will lead to the decision on the interface to be established between a local and a foreign data base. Finally, the investigation of the characteristics of foreign data bases will be dealt with. This four-direction analysis will open the way to a design consisting of (1) an expert dialogue for acquiring information pertaining to foreign data bases, making the system capable of establishing a communication path with the foreign computer, sending a data retrieval request, and converting the incoming data into its own format, and (2) the mechanism of foreign data integration into the local system.

That design will be tailored to fit the ASK system, a system for the storage, manipulation and processing of information that provides a natural language interface with the user.

CHAPTER 2: STATUS

2.1: STANDARDIZATION OF INTERFACES

The standardization of interfaces can be viewed as the adoption of a common communication language between computer systems. As the computer technology advanced, efforts have been made to invent techniques to standardize the interface between computers, in order to facilitate the communication between them.

The first standardization efforts for data base systems can be seen in the work of Codd [Codd 70] who proposed the relational model for data base systems. His initial goal was to protect future users from having to know the internal represenation of the data in a computer. He made the remark that the adoption of the relational model for data bases permits the development of a "Universal Data Sublanguage" based on a form of the predicate calculus, providing, thus, easy transportability of a relational data base system. In his case, the standardization was between the relational system and the underlying computer system.

In the meantime, the Data Base Task Group of CODASYL proposed a model [CODASYL 73] that was believed to provide a general conceptual framework, a basis for a scientific discipline for stored representations [Senko 75]. After this model, the DIAM (Data Independent Access Model) was proposed in 1973 by Senko, et al. [Senko 73]. It aimed to assist the task of facilitating information systems implementation and usage. Both were models for data structure descriptions of data base systems.

Although the scientific community was quite optimistic about the oncoming heterogeneous data base communication [Adiba 77], [Gardarin 77], [Peebles 78] and [Canning 76], it also realized that it is almost impossible to identify a common intermediate level to be shared by a number of different data bases [Adiba 78]. There were two reasons for this:

- the number of existing data bases and the systems that support them were already very large and growing rapidly with no clear mandate that would enforce standardization;
- data base system technology continued to change rapidly in many dimensions, thus making any effort to enforce standardization obsolete too.

It seemed that the entire data base had to be treated as an atomic element. Again, the necessity to use a unique canonical data description according to some principles, for example, the proposed model of the CODASYL Data Base Task Group, was pointed out.

In the late 70's, although the issue of data representation standardization was not at all over (since there was not any such scheme adopted as standard), the set of problems related with data bases was extended to include the task of inter-data base communication. It had been realized that in a distributed system - foreign access is a case of that - commonality could be achieved either through standardization or integration [Deppe 76]. The development of a data base management language had been proposed as a solution to that problem [Deppe 76].That language should be a communication language between data bases and should also have inherent information on the data structures of the communicating data bases. So, the problem was (1) to find a proper communication language to be accepted as standardized language of inter-data base communication, and (2) to embed information on the data structures of the communicating data bases into it. The Data Base Task Group of CODASYL worked on the issue of a common data definition, so that data independence and stability could be achieved in inter-data base communication

[CODASYL 77]. That model, again, has not been widely used in the computer industry.

Finally, in the early 80's, it was realized that it was not necessary to build a unique stand-alone language, but rather a set of data base facilities, which can interact with at least the three major models of data base systems: relational, hierarchical, and network [Polk 81] and [Konolige 81]. The UDL (Unified Data Language) proposed by Date [Date 81] is such a set of data base facilities which can be incorporated into a variety of host languages. However, none of these late techniques have been actually used.

Generally speaking, these facilities depend upon the nature of the interface to be constructed. There can be requests for data or processed data shipments to the accessing data base. That requires design decisions to be made by a data base designer regarding the type of the data interface. The decision maker has to take the structure of the accessing system into account. Certainly, the least possible dependence on the structure of the target data base is desired.

Through all these years, researchers, influenced by the semantic richness of the relational data base model and its ability to be described in the predicate calculus, decided to adopt the relational model either as the data base model of the individual data bases in a distributed heterogeneous network [Gardarin 77], [Deppe 76], or as the data base model of the host data base system when building friendly, natural language front-ends to existing data bases [Hendrix 78], [Harris 77a and 73b] and [Grosz 82 and 83]. In the next sections the heterogeneous access mechanisms of data base systems and the structure of natural language front-end systems for data bases will be examined.

2.2: HETEROGENEOUS SYSTEMS COMMUNICATION

The remark made in [Polk 81] in 1981, that there was no current fully operational distributed data base system with access to heterogeneous data bases, is true for the present time as well. The only system discussed in the literature that planned to provide heterogeneous data base communication was the SIRIUS project conducted at INRIA (Institut National de Recherche en Informatique et Automatique), France [Gardarin 77] and [Le Bihan 80]. The project was planned to take five years - from 1976 to 1981. Although these papers were impressive in what the system was intended to do, no other reference on the continuation of this work was found in the literature. The communication between the dissimilar - heterogeneous - data base systems, as described in the SIRIUS project, is of interest to this thesis. It is therefore described here in some detail.

At first, the notion of a "Virtual Data Base Management System" was developed [Gardarin 77]. It provided common data description and data manipulation primitives throughout the network. That was the standardization effort. In this way, there would be a unified view of the data residing in different computers, and the translation of the data structures as well as the manipulation of the different languages would be facilitated. The relational model had been accepted as the model of the virtual data base system.

After the standardization, the design of the integration followed. Clearly, there were decisions to be made relevant to the type of local data bases to be taken into account and the problem of translation between the common model and the local models. For each computer on the network, a translator had to be built. There was no general purpose facility for building such an integrator. In the initial work of Adiba, et al. [Adiba 77] it was shown that it is possible to transform descriptions of data bases from one model to another. So, the initial design assumed local data bases of any of the three well-established models. But, the implementation phase of the project, the SIRIUS-DELTA system, assumed local data bases of the relational type only.

The query handling consisted of (1) translation of the user's statement into a block of basic primitives, (2) evaluation of that block resulting in an optimum order for the remote actions to be taken and identification of the data that should be retrieved from each site, and (3) communication with the remote sites requesting data. These data could be unprocessed data or results of an internal computation at the remote sites. Of course, a target data base had to look like a "virtual data base." The communication between the various sites had to be conducted in a common language, namely, the language of the virtual data base. The method of maintaining a "directory" so that each site could know what data were available in every other site was not adequately dealt with in the articles. Certainly, there was no information acquisition phase on the location of data in the network that would be as simple as a question- answering dialogue between the respective data base system computer and a data base expert.

The SIRIUS project approach to the heterogeneous system communication is close to the idea behind the ISC/CICS protocols. These are a set of protocols allowing systems complying with them to communicate with each other [Date 83]. In the SIRIUS case, integrators would have been built on top of each system of the network, allowing it to talk to other sites.

From this thesis's point of view, the design of the heterogeneous system communication in project SIRIUS has the following drawbacks:

- an integrator for each communicating system has to be built;
- the topology is rather constant;
- the target data bases are of the relational model only.

The major drawback is that each new system connected to the network had to have an integrator built on top of it, so that it can communicate with the other data base systems. In other words, the target data base had to know about its existence on a network. In the case of foreign access there is no common acceptable scheme for data base intercommunication. Furthermore, a data base owner cannot compel the foreign data base owner to have such an integrator built on the foreign system. The local computer has to access the foreign computer as a subscriber.

The topology of the proposed system was assumed to be constant, whereas, in the case of foreign access, the topology changes rapidly with time. A computer can be connected with any other accessible computer and at the end of the conversation the link is "broken."

Certainly, the assumption of a relational foreign data base is an easy solution to the handling of the complexities that arise on interfacing two communicating data bases. However, most other available data bases are not relational, and many have a different structure that does not comply with any of the other two well-established data base models. There is no provision for such data base interface in the project SIRIUS design.

2.3: NATURAL LANGUAGE FRONT-ENDS

Natural language systems being used as front-ends to data bases have been designed and used in read-only mode. Consequently, there is no problem with data base integrity since the data are not changed through the front-ends. These systems have been designed to be friendly interfaces - the language of communication is English - between a user and a data base system. It is well-known that most of the data base languages are simply another kind of programming language specialized for data retrieval and processing. So, the idea was to let the data base personnel be responsible for updating and feeding the data base with data and to let the user access these data in natural language.

A common characteristic of these front-ends is that they translate the user's query into another language, usually a form of the predicate calculus [Barr 81]. Then the translated query is translated into the target data base language and evaluated against that data base. The attention of designers of data base front-end natural language systems have been more or less directed toward the efficient "translation" of the user's query into the meaning of the request. That interlingua technique has been used in the three natural language front-end systems that will be discussed here: INTELLECT [Johnson 81], LADDER [Hendrix 79], and TEAM [Grosz 82a] [Grosz 83].

- INTELLECT

This system evolved from the ROBOT system [Harris 77a and 77b]. It has been the first commercially available natural language system used as a front-end. Its attachment to a "back-end" data base system requires the development of an extensive software interface to that particular data base system. The system works by mapping English language questions into a language of data base semantics, a form of predicate calculus. Since these semantics are independent of the data base contents, the system can adapt to various data bases by changing its dictionary only, and by using the data base language for parsing and answering the user's query. It can work with one data base only per installation. INTELLECT has been implemented to work on large mainframe computers made by certain manufacturers. It has to be readjusted for each data base.

- LADDER

This front-end natural language system was intended to work with a large distributed data base, the data base being distributed over the ARPANET [Hendrix 79]. The system - it was developed at SRI International - employed the interlingua technique. The interlingua queries were broken down to a set of queries accessing individual files. The computers on the network were accessed and the queries, expressed in query language, were sent. The system then proceeded by combining the incoming responses in order to answer the user's query. LADDER required a homogeneous distributed data base and had to be reprogrammed for each distributed data base system. The directory information on the location of the files, required for query decomposition, was pre-stored in the system. It has not been an easily transportable system, since it intermixed information about the language with information about the domain and the data base [Grosz 83]. Its final implementation was for the Department of Defense Navy data base.

- TEAM

TEAM system [Grosz 82a] and [Grosz 83] has been developed at SRI International as well. It functions as a front-end to relational data bases only and can support small local data bases, but only relational ones. It uses an interlingua as well, translating user's query into the target data base language query.

It consists of the DIALOGIC component [Grosz 82b], translating the English queries into their literal representations in a context expressed in a form of predicate calculus; the data access component, the one responsible for accessing the data base; and the information acquisition component. The latter is responsible for acquiring information on the words the system has to know about and for storing them in its lexicon; on the kind of relations and properties related to these words and their storing in its conceptual schema; and on the mapping of these relations and properties onto the structure of the particular data base and their storing in its data base schema.

TEAM system's innovation is that it provides a dialogue facility with data base experts who are not necessarily familiar with natural language processing techniques. During this dialogue, it acquires information on the target data base structure. This feature provides easy transportability of the system, but it can work with only one data base at a time; each version of the system, created during that dialogue, can work with one data base only. Currently, TEAM system provides transportability to a limited number of data bases.

The discussed natural language systems possess some properties that are not favorable from this thesis' point of view:

- use of interlingua;
- no multiple data base access;
- inefficient use of the front-end's language;
- system reprogramming for the whole target data base.

The first drawback is due to the computational overhead that results from the translation operations, and the fact that if the target data base language is less expressive than the language used in a front-end system, then some uncertainty as far as the response of the front-end system is concerned and some inefficiency in its use are inevitable. On the other hand, the up-to-date systems cannot access more than one data base during the processing of a single query. They have to be reprogrammed from the beginning for each individual target data base. This process is facilitated by a dialogue, in the case of the TEAM system, but that does not solve the problem of accessing more than one foreign data base during the processing of a single query.

CHAPTER 3: DESIGN

3.1: SPECIFICATIONS

So far, the status of the systems that provide heterogeneous data base access and natural language user interface to data bases have been examined and their drawbacks have been pointed out, all in relation to the aim of this dissertation. As a consequence of that analysis the specifications of the system to be designed have become more and more apparent. Clearly, the system has to provide:

- natural language interface with the user;
- support of a local data base;
- data location transparency for accessed data;
- use of local and foreign data in the processing of a single query;
- easy addition of new words associated with foreign data bases;
- facility to consolidate foreign data into its own data base (foreign input);

- communication with most - if not all - the existing commercially available data bases.

We now proceed to the analysis of the issues involved in such a system, i.e., the interface with foreign data bases including issues on the information to be conveyed regarding retrieval and processing of foreign data.

3.2: INFORMATION TO BE CONVEYED

We are specifically concerned here with setting up foreign data base access and the character of the information that must be supplied during this setup phase. That information is to be divided into the (1) hardware related information, and (2) data retrieval and conversion information.

3.2.1: Hardware Interface

In order for two computers to communicate there must be a communication line they communicate over. That line has to have certain characteristics complying with the specifications of the communication modules of the two systems. The hardware characteristics of the line, e.g., baud rate, handshake, etc., have to conform with the expectations of the computers connected at its ends.

Lately, the electronics industry has provided special hardware that can facilitate computer communication. These modules, such as those Hewlett-Packard built HP98628A, can be computer programmable and can be controlled to be set to a particular configuration. Autodial capability modems help in establishing a communication path to another computer through a telephone line. Under these conditions, it is relatively easy to establish a communication path with another computer by using specialized system-driven procedures whose arguments are the configuration parameters of the desired communication line.

The information on the characteristics of the communication line has to be added into the system in an easy way. As a solution to that problem a dialogue has been designed, much in the style of expert systems [Barr 82]. Typical of questions asked by the computer during this dialogue is that concerning the telephone line. Foreign computers can be accessed either through a telephone line connection or through a dedicated line. In the former case the phone number of the target data base must be asked for. The complete dialogue schema for this "configuration" expert system is presented in section 4.3.

3.2.2: Data Retrieval

We are opposed to the idea of using a natural language system as a front-end to any data base. This would involve some sort of interlingua action by translating the system's language into the respective data base language. As has already been mentioned, the result would not be very satisfactory because such systems cannot access more than one data base during the processing of a single query, and they are limited by the linguistic capabilities of the underlying data base system. Hence, if the full linguistic power of a natural language system is desired, the interfacing has to be restricted to the leaves only of the parsing graph. Such a leaf is denoted by a word or idiom of the local system's own language. The translation of this single word or idiom into the query language of the relevant foreign data base system may, of course, be more involved. The leaf evaluation concept has been used in the POLYPHEME prototype developed under the project SIRIUS [Le Bihan 80] at the virtual data base machine level. So, there was still a translation phase from the user's query language into the network-wide standardized language involving an interlingua technique. Moreover, the directory information used did assume that the target data bases were homogeneous ones, since the access was done in the language of the virtual machine.

.Here, there is an assumption to be made regarding the structure of the local data base. As such, the entity- relationship model proposed by Chen [Chen 77] will be used in this discussion (as we shall see, it is closely related to the ASK System data model). Under that scheme, the leaves of the parsing graph are of an individual (atom), a set (entity), or a binary relation (relation, attribute). In the case of individuals, if they are known to the local system, there is no need to retrieve them out of another data base. If they are not known to the local system, it is their foreign data base relationships that are relevant. What is needed, therefore, are the unary and binary predicates to which an individual can be an argument. Clearly, in set theory, a unary relation is a set and coincides with the membership condition of that set. Hence the only foreign data base query formats that are required for foreign data retrieval are those that retrieve a set and a binary relation out of a foreign data base. This conforms with the theoretical results in [Childs 77], examining the set theoretic interface between the information space, data space, and storage space of a data base.

The data structures we will use that correspond to the entity-relationship model are class-attribute-relation, where an attribute is single valued, e.g., "father," and a relation may be multiple valued, e.g., "child." It can be rigorously shown that this data model is closely related to, and subsumes, the relational model. An n-tuple of the relational model corresponds to an entity, the fields of that n-tuple, to attribute values where the attribute identifies the field name.

To illustrate the technique, assume that the establishment of access to a foreign data base is desired and the foreign data base is a relational data base with a "ship" file with fields: <name,destination,length,...>. Then the only queries to be sent to the foreign computer will be of the form:

o ship	- Ship(name)
o destination	- Ship(name,destination)
o length	- Ship(name,length)
o ...	

The translation of a given set or binary relation word of the local system's language into a query of the accessed system's languge need not be as simple as this illustration and indeed may be much more selective. For example, if "ship" refers only to ships of a length of less than 80 feet, then the respective queries can be:

o ship	- (Ship with length<80)Ship(name)
o destination	- (Ship with length<80)Ship(name,destination)
o length	- (Ship with length<80)Ship(name,length)
o ...	

A particular reference from a leaf is looked up by the system in its lexicon. When the system detects from this lexical entry that the leaf is not within the system itself, it will use the respective interfacing routine, available through the lexical entry, to get the information out of the foreign data base stored in another computer system. The same routine will also convert the data into the system's format, either an entity, attribute, or binary relation, and will resume the processing of the respective query. So, by using the leaf interface technique, the system is able to communicate with more than one data base during the processing of a single query.

The issue of response time is one that must be dealt with. For the front-end type systems that use an interlingua and interface to one fixed data base system, an entire query is passed to the data base system in the format of the interlingua and only the completely processed query results are passed back. In the case that concerns this thesis, all leaves relevant to the query are passed to various foreign data base systems and entire classes and binary relations are passed back. This raises the question of whether the front-end systems' communication demands are significantly less than the "leaf" method so as to be superior from the critical point of view of user interface time. Two cases must be considered: (1) if the communication path is fast, such as using high speed coaxial cable or optical fiber, it has been found that the steps to initiate the communication comprise the major communication delay; (2) in the case the communication path is slow, such as using a telephone line and a low baud rate modem, the data transmission delays are comparable to or more than the time spent on the initiation of the communication. An experiment with the Datatrieve Data Base system of DEC on the Computing Center VAX of Caltech showed that at the 9600 baud rate and medium computing load on the VAX, the data transmission took 8 seconds to complete whereas the initiation of the dialogue took 1 minute. The logout sequence was completed in 6 seconds. In the case of a 300 baud rate line to the same computer, the login and logout times remained almost the same, but the data transmission time was increased to 3 minutes and 10 seconds, i.e., twenty times slower than the previous case. In general, the data retrieval in the case of a high speed communication line can be one or more orders of magnitude faster than the communication initiation dialogue between the two computers.

Of course, the interfacing has to be a general one. Certainly, interfaces such as the ICS/CICS of IBM [Date 83] and other similar ones are to be excluded from this design, on the grounds that they can serve a small subset of all existing systems. This thesis assumes an interface by which the foreign computer will be accessed in the same way as by a user-terminal pair. This approach has been considered to be much more general, especially in the case of commercially available data bases.

The information from accessing foreign computer records is extracted and built into the local system's formats, and thus will enter directly into the further processing of the query. This translation process requires knowledge of the structure of the foreign system's response. This knowledge is part of the special information the local system was given during the information acquisition phase on the foreign data base.

3.3: FOREIGN ACCESS

In summing up the analysis and decisions made in the previous sections, the following characteristics of the local system have been decided on:

- interfacing with the foreign computer at the leaves of the parsing graph; only sets and binary relations cross the boundary of the two systems;
- the local system emulates a user-terminal pair when accessing foreign computers;
- the local system has an expert subsystem, in the form of a dialogue, for acquiring information on the communication line characteristics, setting up dialogues with foreign computers and "reading" data out of recorded parts of these dialogues.

Each leaf of a parsing graph referring to data residing in a foreign computer has this information concerning both the hardware interface and data retrieval stored in its lexical entry. Such a word of the local data base language that corresponds to foreign data will be called an *access word*.

3.3.1: Access Words

These can be of the types: class, attribute, or relation. Their use can lead to syntactic or semantic ambiguities within the local system. The way these undesirable effects are bypassed will be described in the section for query handling. For the time being, the focus is on the information necessary for each access word in order for the system to perform its foreign access duties, and more specifically, information on how to:

- establish hardware connection;
- login to the foreign system;
- logout from the foreign system;
- phrase the query statement for this particular word;
- retrieve the data;
- get the data from the recorded parts of the two systems' dialogue and convert them to the specified type (i.e, entity, attribute, or relation).

In the case of access words referring to the same foreign data base, there can be some efficiency gained if the first three elements in the above list of the itemized information are stored as pertinent to a particular data base. In this way, that information can be stored in a specialized foreign data base directory and it can be used upon request during the information acquisition phase for each access word.

3.3.2: Data Base Access

Once the foreign computer has been accessed on the hardware level, the local system probably has to login into the foreign data base, to access the data, and to logout. As it is stated in [Adiba 78], in the case of heterogeneous data base communication there is a need for accommodation of differences in sending and receiving formats, data types, and data representations. On the other hand, the design decision made is to emulate the user-terminal pair on the local computer. Hence the communication with foreign computers has to rely on the kind of protocol used during the user-computer communication.

On examining the computer systems that are available today, one can clearly see that they always prompt the user in a certain predictable way. For example, they can prompt with a prompt sign such as "> ," or "$ " in the beginning of a line, by placing the cursor in a specific place on the screen of a CRT terminal, or by giving another expected - at least to a system programmer's knowledge - response. That is, when an action is initiated, the response of the computer is always such that it denotes that indeed the action has been performed and it is probably expecting the next input command. This is the central design assumption made and is related to the target data base system behavior.

Because of that, the communication with a foreign computer has been designed to be conducted in terms of

- what a user would type in order to perform the same actions, and
- what the foreign computer is expected to prompt.

These are the two primary factors the information acquisition dialogue has to take into account. There is, of course, secondary information required for the foreign computer access, related to aspects of reliability of the foreign system. More specifically, in the case of a foreign computer failure, or if something cannot be performed for any reason, the local system has to be protected from waiting indefinitely for the foreign system to respond. In case something goes wrong, or, more commonly, if the access attempt must be repeated, the system must have the option of exercising certain steps of a dialogue again. The former option calls for a timeout mechanism, whereas the latter, for an appropriate recovery mechanism.

3.3.3: Data Retrieval

Using the technique proposed in the previous section and recording selected parts of that conversation one can perform the data retrieval actions, no matter how many question-answering cycles will be between the beginning of the actions and the final one giving the desired response. Of course, the information acquisition dialogue has to provide the option of specifying when the recording of communication, that is, the actual data to be retrieved, will start and when it will end. The recording contains not only the retrieved data, but some more information as well, such as redundant characters, headers, control characters, etc. It is, roughly speaking, a printout. But printouts of data base systems follow a particular format when they respond to a user's query. This is the second central assumption made for a foreign data base system.

So, the recorded response of a foreign data base can be considered to be a record structure that consists of logical records of variable length. A logical record is considered to be a sequence of physical records of various types and/or logical records related logically to each other. Consequently, an investigation of the required information for unscrambling the output of a foreign computer and for preparing for its translation to the local formats has to be conducted. As a result, another expert system dialogue has to be built for acquiring and using that information for the various access words. This dialogue has to get the necessary information for imposing a structure over the foreign output, since record structures by themselves "are not complete in representing information nor are they fully describing" [Kent 79].

Each record has a beginning denoted by the appearance of a specific string of characters, by the end of the previous record, by the beginning of a file, etc. It also has an end and, in the case of the logical records, that end is denoted by a specific condition such as the appearance of a specific control character, the end of a file, etc. In the case of foreign access, each logical record can have nested logical records. For example, when accessing a library book directory and trying to print the titles of books in a specific domain, neither the length of the output nor the length of the titles is known a priori. It must be mentioned, though, that physical records can be treated as special cases of logical records. So, if the problem of unscrambling the logical records is solved, the same technique can apply to the physical records as well.

There is also another important issue to be taken into consideration during the investigation of the characteristics of the output as a multi-level logical record, namely, the checking for the consistency of the record. For example, if for some reason the foreign system malfunctions and gives a diagnostic message, then the output structure is likely to have an inconsistent view compared to the one expected. Assuming that error messages issued by foreign systems follow certain patterns known to the data base expert, a set of restrictions for the body of a logical record has been defined. These restrictions consist of checkings for the existence of particular given characters within a record.

So far the following aspects of the output of the foreign computer seen as a logical record structure have been found:

- each record has a beginning;
- there is a condition denoting the end of the record;
- logical records can be nested;

- physical records are special cases of logical ones;
- each entry itself is a variable length logical record;
- there may be restrictions on the size, location, and/or contents of logical records.

When a typical output of a foreign system is examined, one can see multiple pages, headers, etc. Then it becomes apparent that, if each page is considered to be a logical record, it is a repeated one. On the other hand, by treating each individual line in a column table of a printout as an individual data record (logical record), the repetition of that logical record within the larger (table) record is obvious. Consequently, the design has a provision for defining a condition that specifies whether the same record pattern is expected or not. A condition is actually the same as a restriction, the only difference being in the semantics in each case. This decision is extremely helpful in the case of tables, multiple page outputs, and repeated patterns that do not always have the same length but follow a particular pattern that can be understood by their users and can be described in terms of logical records of variable length.

Hence, by supplying the local system with information on the structure of the foreign output, it is possible to "walk through" its characters and locate the various logical records. The design of this particular expert system aims in locating the individual members of the sets or binary relations and pick them so that they can be used by the local system. Consequently, there are two types of records to be used: the "normal" records and the "pick" records. The only distinction between them lies in their semantics, for in the case of the latter a particular procedure is invoked, making its content known to the local system.

So, a logical record can be specified by its:

- record type;
- beginning;
- restricting conditions;
- end of record;
- repetition conditions.

The description of the logical record structure is facilitated by the use of one cursor when a set has been accessed, or two cursors if a binary relation or attribute will be accessed. The cursor is used in defining the beginning and the end of the records and in describing their restriction and repetition conditions. They are supposed to "move" on the printout by commands similar to those used in window editors. In the case of a binary relation or attribute, each cursor is supposed to move through the records whose contents are the arguments of the relation or the attribute. More details will be given in chapter 4, where the implementation of this design is presented.

3.3.4: Data Conversion

The data conversion has to act in two directions:

- storing the foreign data in temporary storage;
- eliminating ambiguities that could possibly arise by the previous action.

The first action of this list aims in speeding up the process of any subsequent query to the local system referring to the same data. But in doing so, each individual member of a set, attribute, or binary relation has to be checked as to whether it appears in the lexicon. If it does, then there is no need to be re-defined; if not, then it must be added into a temporary lexicon that has to work together with the already existing lexicon of the local system. This way, each individual is guaranteed to appear only once in the system lexicons, and thus the ambiguity is avoided.

There is also another issue related to ambiguity: the case when a local system name for a set, attribute, or relation coincides with that of an access word of the same type. As a solution to this problem, the parser checks that this is the case and creates temporary sets, attributes, or relations with subsets, subattributes, or subrelations of the respective ones which are permanent to the system. In any subsequent time it uses these temporary data structures rather than the permanent ones. This technique of integrating the foreign data into the local system is described in detail in section 4.4, where the query evaluation issues of the implementation of this design are presented.

3.4: DOMESTIC ACCESS

Consider the case where the manufacturer of the local system (which includes the foreign access aspects introduced in this thesis) also supports alternative data base systems, data base systems supplied by the manufacturer before the local system was marketed. These other systems will be referred to as *domestic systems*. A prime requirement for such a manufacturer is, on the sale of the local system, not to obsolete the investment of his customer in these domestic systems. Since everything about a domestic system is known by its manufacturer, most aspects of the data structures underlying the foreign access to these domestic data base systems can be built in by the supplier of a local system. These are parts of the foreign access dialogue including the hardware specification of the communication line (except for the "phone number" aspects) and the access and conversation aspects with the domestic data base system, including how to unscramble its response. Only the part in which the local vocabulary to be used and its translation into the query language of the domestic system need to be provided by the user. Thus the expert system dialogue needed in this case is only a small portion of that needed in the full foreign access case.

The implementation of these automatic parts for foreign access and the truncation of the expert system dialogue to that needed for domestic access are left for the manufacturer and not included in this thesis. But since this is only a specialization of the general foreign access capability, this is really only a small exercise. The point here is that domestic access is a much smoother and more direct procedure for the user. Thus it saves his prior investment in a convenient way.

The files for a domestic data base system are, of course, of known structure. If they are physically accessible through the local system, then again an expert subsystem can be built as before, accessing them through specialized procedures. This case is desirable as well for the efficiencies that can be obtained.

3.5: FOREIGN INPUT

There may be cases when data residing in foreign data bases are desired to be brought into the local data base system once and for all, thus becoming part of the local data base. For example, there may be stable data available such as "handbook" data, which for efficiency reasons one wants locally available. Or, during a transition phase, data currently in a data base system about to become obsolete will need to be transported into the new data base system. The same expert system used in the cases of foreign access and data bases or data (files) of known structure can be used with only a small change in its final stages. More specifically, there is no temporary lexicon involved during the processing of a foreign input. All defined *input words* (defined similarly as the access words) and data which have been brought over are integrated into the local system's lexicon. In this way, they become permanent to the local system.

Obviously, just as foreign input corresponds to foreign access, domestic input can be easily implemented once domestic access is available. A judicious use of domestic access and domestic input, over the protracted learning period upon aquisition of the local system, can smooth the transition to the new local system.

CHAPTER 4: IMPLEMENTATION

4.1: ASK SYSTEM

The ASK system [Thompson 83] is a system for the storage, manipulation and processing of information that provides natural language interface with the user. Its structure contains a lexicon and dictionary used by its parser that supports general rewrite rule (type 0) [Barr 81] grammars. It has been implemented on the Hewlett- Packard HP9836/26 desktop computers.

4.1.1: Lexicon

The lexicon of the ASK system associates the words of its vocabulary, i.e., the leaves of the parsing tree, with the semantic category they belong to, the so-called part of speech, and the respective data structures, i.e., their payloads. A payload can be viewed as the data structure that contains all the information that pertains to a particular word in the context of the discourse. For example, the payload of the access words that are stored in the lexicon is the access information acquired during their respective foreign access dialogue. The interest of this dissertation regarding the lexicon lies primarily in the way an entry is associated with its payload, and how ambiguities are handled.

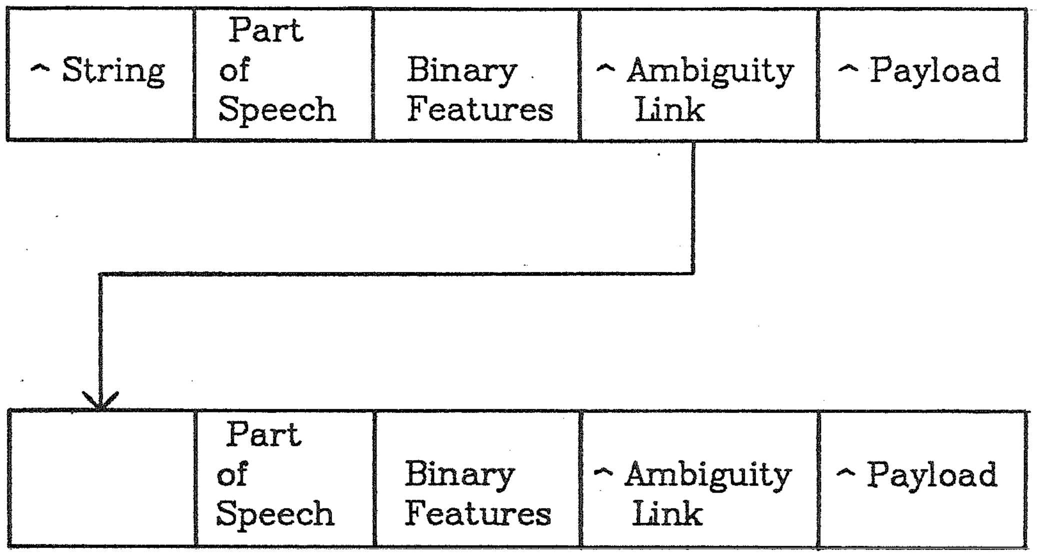

Figure 4.1: A Typical Lexical Entry

A typical entry in the ASK lexicon is shown in figure 4.1. Since the location of a particular entry is found through a hashing technique, its first field includes a pointer to the string of that entry. There is also a field pointing to the payload and, in the case of an ambiguous definition, an ambiguity link pointing to another entry with similar structure. The latter entry is linked to and accessed through the top entry. Consequently, it does not keep any underlying string information. There is also a provision for binary features to be assigned to each lexical entry. These binary features are used in distinguishing entries defined by identical strings and parts of speech.

4.1.2: Data Types

The data types of the ASK system are organized in terms of

- classes;
- objects;
- attributes;
- relations.

Objects are the basic atoms of the data base system and they can be numbers, individuals, texts, etc. Classes are defined as arbitrary sets of objects, and both attributes and relations associate objects with other objects. This design assumes that the data to be retrieved from a foreign data base are organized in classes (sets), and attributes or relations (binary relations) of one type of objects per access, for example, class of numbers, relation of individual with number(s), etc. The reason is that under no circumstances should the accessing system restrict its communication with the foreign data base by assuming a more complicated foreign data structure. Clearly, if a set consists of different types of objects, the information crossing the boundary of the two computers does not consist only of the membership condition, but also of other relations standing among the members of the set. This fact contrasts with the technique used in this design.

The present design allows classes, texts, attributes, and relations of numbers, individuals, or texts to cross the boundary of two communicating computers. In the case of number attributes, the system asks for the units of the attribute and any associated adjectives. The type of the object to be retrieved from a foreign data base is passed into the system in terms of binary features assigned to each word of the lexicon.

4.1.3: Parser and Parsing Tree

The parsing process is performed in two steps. The first step is the lexicon look-up phase and consists of finding the semantic categories the various words belong to, and adding the respective nodes to the parsing tree. The second step uses the dictionary and applies all the available syntax rules in order to obtain the final parsing tree. This step of the parser is similar to the powerful bottom-up Kay parser [Kay 67] and supports handling of semantic and syntactic ambiguity [Randall 70]. The good parses obtained at the end of this process are processed for evaluation by the semantic processor.

Each node in the parsing tree has its type, e.g., noun, noun phrase, access word, etc. Each node has either a pointer to its payload, or a pointer to a list of its constituent structures, and it is associated with a semantic procedure. This semantic procedure is the one to be called when the evaluation of that node takes place. The leaves of a parsing tree lack, of course, any constituent structures, but they may be associated with semantic procedures. For the purpose of this thesis, they can be viewed as quintuples of the form

(<part of speech>, [<binary features>],
<payload> or <constituent list>,
<semantic procedure>, <link>)

The last element is a pointer to another node of the same type or nil. In the case of access words, a node is of the form:

(access word, [ ], payload, - , nil).

Since there is no associated semantic procedure in access word nodes, it can be evaluated from higher level nodes only. The type of the node that evaluates an access leaf is a *noun* node associated with the *foreign access procedure*. So, an access word will appear in the parsing tree under the structure shown in figure 4.2. The binary feature "+access" denotes that at least one immediate constituent of this node is an access word.

$$\begin{tikzcd} \text{(noun, [+access, ...], - , foreign access procedure, .)} \arrow[d] \\ \text{(access word, [<features>], payload, - , nil).} \end{tikzcd}$$

Figure 4.2: Access Word and Noun Nodes in the Parsing Tree.

Finally, the rest of the system knows how to "bubble up" the data that were obtained during the evaluation of this data structure. The evaluation of the leaves at the right time is guaranteed by the semantic processor of the ASK system.

4.2: FOREIGN ACCESS DIALOGUE

The first part of the dialogue is the phase for acquisition of general information on the foreign data base. This information includes the data base name, communication line characteristics, and login and logout conversation sequences. Since this information is the same for all foreign words referring to the same foreign data base, there is a provision for storing it separately or making it available in any subsequent use of the foreign access dialogue.

```
BEGIN
   Data Base Name?
   IF it is a known foreign data base THEN
      use  existing information as default?
   IF yes THEN do nothing
   ELSE
      Hardware Characteristics of the communication line?
      Login Sequence?
      Logout Sequence?
   WHILE there are access words to be defined DO
      Access Word & Type?
      Data Retrieval Sequence?
      Printout Uncrambling Information?
END
```

Figure 4.3: Outline of the Foreign Access Dialogue.

The second part of the dialogue has to do with the retrieval of the data from the foreign data base and with the unscrambling of the foreign computer printout. The exercise of this part is necessary for each access word defined in the system. The outline of the foreign access dialogue is given in figure 4.3 and recorded sessions of foreign access dialogues are included in Appendix A.

This division of a foreign access dialogue into two parts is based on the system programmer's point of view. Functionally, the division is: hardware connection, conversation with the foreign computer, and unscrambling of the printout. This separation in parts has been adopted in developing this section.

4.2.1: Hardware Connection

This design has used the DATACOMM 98628A board of Hewlett-Packard, fitting its HP9836/26 desktop computers, together with a 300 baud autodial modem (13265A of Hewlett-Packard). Using this board, both the hardware configuration of the communication line and the control of the modem are obtained through statements of PASCAL programs invoking dedicated system procedures.

The dialogue acquires the necessary information on the arguments of these procedures. The values obtained depend upon the mode of the communication, either asynchronous or data link (used in Hewlett-Packard's HP1000 or HP3000 Data Link network applications). Thus the dialogue, after having requested the name of the communication mode, proceeds by asking the user the values of the necessary parameters of the respective communication configuration. Default values and help messages are also provided for convenience. These default values are easily set by the system programmer when the system is set up in a new environment.

The only things that have to be redone in order to implement the system on another computer are the parts relevant to the respective communication board and autodial modem used in the new system. If the function of the modem and the communication board are program controllable, then the interface of controlling them has to be integrated into the system; otherwise, they are set to the desired configuration manually. In the former case, some modifications in the part of the dialogue that acquires the hardware characteristics of the communication line have to be done; in the latter case, the whole part of the dialogue related to these hardware characteristics can be eliminated. Of course, there can be possibilities for solutions between these two extremes, e.g., for the dialogue to ask for the baud rate, echo, and the phone number of the foreign data base only.

4.2.2: Foreign Computer Access

The design decision made dictates that the communication between the local and the foreign computer will be conducted in terms of what a user would type in order to perform the same actions, and what the foreign computer is expected to prompt with a reasonable time delay. There is also the provision for specifying to the local computer when to start recording the conversation with a foreign computer, and a recovery mechanism in case something goes wrong. Because of that, the dialogue has been designed to request the actions to be taken by the local computer, during each step of a conversation with a foreign computer. To this end, the following actions have been defined:

send

sendln

expect

collect

end collect

try

end try

end login

end logout

end data retrieval

The last three commands are for the dialogue driver and denote the end of the respective phase of the information acquisition process. The semantics of the rest of these actions are:

send	It sends a string over the communication line. The string is entered when the computer

	prompts for that, immediately after one types "send."
sendln	Same as "send," but it sends a carriage return at the end of the string as well.
expect	It sets the computer in waiting for a string state. The expected string is entered immediately after one types "expect" and after the computer has prompted for that. The computer will also prompt for a timeout when expecting a string.
collect	It sets the computer in the state of storing all the characters returned from the foreign computer in temporary storage.
end collect	It terminates the collect status.
try	It pushes the current point of the dialogue between the two computers in a stack, together with the number of trials the system will be allowed to perform if something goes wrong, for recovery purposes. Each time a recovery takes place, the dialogue is exercised from the same point, and the number of allowed trials in the

stack is decreased by one. The number of trials is requested by the system immediately after "try" is typed.

end try It pops the stack used for recovery purposes.

The outline of the foreign computer conversation part of the foreign access dialogue is given in figure 4.4.

```
BEGIN
   Action?
   WHILE action is not nil DO
       IF action in [ send, sendln, expect, try] THEN
          CASE on Action:
             send,
             sendln: What to send?
             expect: What to expect?
                       Timeout?
             try:    How many times to try?
       Store action.
       Action?
END
```

Figure 4.4: Outline of the Access Part of the Foreign Access Dialogue.

4.2.3: Foreign Output Processing

According to the design of the foreign data base access facility, the output of a foreign computer is viewed as a structure consisting of variable length logical records. Each such record is specified by its record type, beginning, restricting conditions, end, and repetition conditions as explained in section 3.3.3. Consequently, part of the foreign access dialogue relevant to the processing of the foreign output has to acquire and store all that information into the local system. The outline of this dialogue is given in figure 4.5.

```
BEGIN
   Record type?
   {beginning of record}
   Action?
   WHILE non nil action DO
       Action?
   Restricting condition?
   WHILE non nil restricting condition DO
       Restricting condition?
   Nested record type?
   IF non nil entry THEN
       re-exercise this outline for nested record
   {end of record}
   Action?
   WHILE non nil action DO
       Action?
   End of record condition?
   WHILE non nil End of record condition DO
       End of record condition?
   Same level record type?
   IF non nil THEN
       re-exercise this outline
   ELSE
       pop at the higher level record
END
```

Figure 4.5: Outline of the Printout Structure Part of the Foreign Access Dialogue.

On examining the outline, one can see clearly that things requiring further specifications are

- cursor moving actions, and
- conditions.

The former are the actions used in placing each cursor in the beginning or the end of a record, whereas the latter are used in evaluating the restrictions and the end of record conditions of each record. As far as the cursor moving actions are concerned, the following ones were used:

right [n]

left [n]

up [n]

down [n]

search+ [n]

search- [n]

beginning of string

end of string

*

#

where "[n]" means optional use of an integer denoted by n. The semantics of these actions during the unscrambling of a foreign output are:

right [n]	the cursor moves n positions to the right or 1 position if n does not appear. If the end of a line is encountered the cursor remains in the rightmost position of the line.
left [n]	similar to the previous case. The cursor moves to the left.
up [n]	similar to the first case. The cursor moves up.
down [n]	similar to the first case. The cursor moves down.
search+ [n]	the cursor moves forward searching for the first or nth occurrence of a string that is entered immediately after this action has been entered. The computer prompts for that. The cursor is left at the beginning of the matched string, if any. Otherwise it returns to

	its initial position.
search- [n]	same as the search+, but backwards.
beginning of string	moves to the first non-blank character to the left of the position where the cursor points to. If the cursor points to a blank character nothing happens.
end of string	same as before, but to the right.
*	sets the current cursor equal to cursor 1.
#	sets the current cursor equal to cursor 2.

The conditions used in the specification of the restricting and end of record conditions follow the simple pattern

<elementary condition>

or

not <elementary condition>

A set of conditions is treated as a single condition consisting of a conjunction of all the conditions entered. In that scheme, an elementary condition stands for any one of the following:

match string

blank line

blank field

alphanumeric string

alphameric string

numeric string

cursor expression in field

string in field

string type in field

true

false

The format of the elementary conditions are as follows:

match string	True if the string that the cursor is pointing to matches with a given string. The string to match is entered immediately after this condition has been entered and the computer has prompted

	accordingly. The cursor can be pointing at any character of the string.
blank line	True if the line the cursor is pointing to is a blank line.
alphanumeric string	True if the string (if any) the cursor is pointing to is an alphanumeric string, i.e., consists of any readable character except blank.
alphameric string	True if the string (if any) the cursor is pointing to is an alphameric string, i.e., consists of any readable character except blank and digits.
numeric string	True if the string (if any) the cursor is pointing to is a numeric string, i.e., consists of digits, ".," and ",."
blank field	True if an area of the foreign output, defined as a field through

	two positions of the cursor, is blank. The computer will prompt for the field information.
cursor expression in field	True if for a particular series of cursor moving actions, the cursor is located within the bound of a field. The computer will prompt for the field information.
string in field	True if a particular string appears within a field. The string is entered after this condition has been entered and the computer has prompted for that. The computer will prompt for the field information.
string type in field	True if a particular string type (alphanumeric, alphameric, or numeric) appears within a field. The computer will prompt asking for the string type and field information.

true	True, always.
false	False, always.

Now, as far as the field information is concerned, this has two parts:

- field type: this is of the line, column, or window type;
- field defining cursor actions. These are two series of cursor moving actions defining the limits of a field.

A final remark is that the actions that really move the cursors on a foreign output are the actions relevant to the definition of the beginning and the end of a logical record. All the other cases of using cursor moving actions, i.e., when specifying a condition, are used in facilitating the expression of these conditions. They are the so called pseudo actions.

4.3: QUERY EVALUATION

After a successful parsing has been completed, the evaluation of the parsing tree takes place. The semantic processor of the ASK system is applied on the constituents of each node of the parsing tree recursively. When an access word leaf is encountered, the foreign access procedure takes action. It uses the payload of the respective access word in identifying the data base to be accessed, and obtaining the information on the characteristics of the communication line to be established. It obtains also information on how to login and logout from that data base. It checks as to whether it is already logged in the same data base; otherwise it logs out the previous one and logs in the current foreign data base to be accessed. The login procedure includes the resetting of the characteristics of the communication line and the transmission of the necessary entries to login.

Going on the same way, the foreign access procedure requests the data and finally "reads" the response, i.e. gets the necessary data out of a printout by using the information acquired during the last part of the respective foreign access dialogue. The data then are translated to ASK system format. At the end of each foreign access, the system does not necessarily log out the foreign data base system, thus reducing probable time delays for logging into the same foreign data base again. If an error happens during the whole foreign access process, the system recovers gracefully through the recovery mechanism of the ASK system. This recovery mechanism includes closing of the foreign data base and resetting the local parameters involved in this communication.

The translation of the data into the ASK system's data structure takes into account whether or not the atoms of the foreign data that were brought over are already known to the ASK system. If they are known they are not added; otherwise, they are added into a temporary lexicon identical to the permanent lexicon of the ASK system. That temporary lexicon is called the ***foreign access lexicon***. Each access word, once instantiated, is added into the foreign access lexicon with the payload being its instantiation. The purpose of its existence is to make all the data brought over from a foreign computer temporary, so that the processing of any subsequent query referring to them does not imply access to the foreign computer again. The temporary lexicon has been designed to exist during a single session. At the end of the session it is lost. Of course, a session can consist of any number of queries to the data base.

At the end of a foreign data base access, the system resumes its normal processing and gives the answer to the query that initiated the access. There are also some other structural changes that happen to the system as well, namely, the addition of new data into the foreign access lexicon.

4.3.1: Parser, Ambiguity, and Foreign Access Lexicon

The foreign access lexicon contains foreign access words that have been instantiated with data converted into the ASK system's format, as well as foreign objects in general that are part of the instantiation of the classes, attributes, or relations that were brought over.

According to the design, if an individual already exists in the ASK system, it is not added into the foreign access lexicon. This decision prevents the creation of ambiguities in the individual atom level of the data base system. In the case of classes, attributes or binary relations, however, it can be the case that the same word can be both a foreign access word and a normal word, i.e., a word both designating data in a foreign data base and designating data in the local data base. In this case, such an ambiguous word would have two or more entries in the lexicon. Let us say, the local meaning is given by the local payload PL. Under these conditions, when the payload of the foreign lexical entry is instantiated, i.e., becomes a temporary payload, say PF, the permanent data structure PL becomes a substructure of the foreign lexical entry data structure PF. In this way, new temporary data are integrated into the ASK system. This technique of integrating data from one data base into another data base was proposed in [Yu 81] for communication between data bases. A schematic diagram of the way the permanent and temporary payloads are linked with each other is shown in figure 4.6.

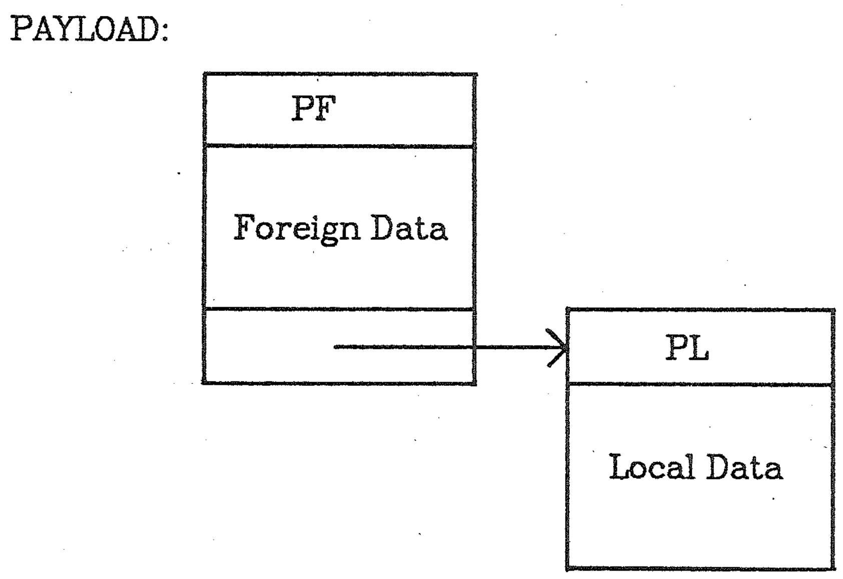

Figure 4.6: Integration Scheme of Foreign Data into the ASK System.

In order to clarify the whole technique of using two lexicons for the foreign data integration, the problem of the worst case, as far as ambiguity is concerned, will be dealt with.

Consider the case of the same string being defined ambiguously with different parts of speech and/or with different type (denoted by certain binary features) in the lexicon. Consider also that certain of these words are defined both as nouns and also as access words defined with the same type as these respective nouns. Under these conditions the following grouping of these entries is considered:

- each noun is associated with the existing access words of the same type;
- remaining access words are grouped separately, each group having access words of the same type;
- remaining nouns of each type (there is only one for each type) are treated individually;
- words that are neither nouns nor access words are treated in the way dictated by the semantic category they belong to.

Under these conditions, each group belongs in a different syntactic category and leads to syntactic ambiguity. ASK system's parser and semantic processor can handle that ambiguity and respond to the user according to the various parsings it obtained. Hence it only remains to describe the processing of each of the groups defined above.

Clearly, the last group, i.e., of words that are neither nouns nor access words, does not concern this thesis. Similarly, the case of a single noun is handled by the system itself and is not related in any way to the foreign access mechanism. Hence, the following cases will be dealt with:

- one or more access words of the same type;
- one or more access words and one noun of the same type.

In the former case, i.e., of one or more access words of the same type, the parser adds the following node in the parsing tree:

```
          ( noun, [+access, ... ], L(1) , foreign access procedure, - )
L(1) :    ( access word, [<feat>], payload(1), - , L(2) )
L(2) :    ( access word, [<feat>], payload(2), - , L(3) )
...
L(n) :    ( access word, [<feat>], payload(n), - , L(n+1) ).
```

This addition happens during the first pass of the parser, when it looks up words in the lexicon.

In the latter case, the node has the structure:

```
        ( noun, [+access, ... ], L(1) , foreign access procedure, - )
L(1) :  ( access word, [<feat>], payload(1), - , L(2) )
L(2) :  ( access word, [<feat>], payload(2), - , L(3) )
...
L(n) :  ( access word, [<feat>], payload(n), - , L(n+1) ).
L(n+1) : ( noun, [<feat>], payload(n+1), - , - ),
```

with the noun linked in as the last element of the constituent list of the node. The common types of all the access words and noun are denoted by "<feat>." Both structures are quite similar, the only difference being the addition of a noun constituent node in the second one.

In case an access to the data base is to be performed, when the top node of this structure is evaluated, the foreign access procedure is called to work on the constituent node list. The following algorithm is used for the evaluation case:

```
Begin
  get constituent node list;
  if first access word not in foreign access lexicon then begin
    {recurse on constituent list}
    while the current element
          of constituent list is not nil do begin
      if it is an access word then begin
        access respective data base;
        if it is the first access word in the list then
          add it in the foreign access lexicon;
        add retrieved data into the foreign access lexicon
            entry's payload;
        end;
      if it is a noun then begin
        add it as a substructure into the foreign access lexicon
            entry's payload;
      end;
  end;
  instantiate the top noun node using the foreign access lexicon;
end.
```

At the end of this process the correct instantiation of the top "noun" node is guaranteed, since either it is instantiated at the time the algorithm is exercised, or it was instantiated.

When data are to be added to an ambiguously defined data structure, then (1) the access does not take place, (2) the data are added into the permanent data structure of the ASK system, and (3) if no such permanent data structure exists in the ASK system, a diagnostic message is issued. This procedure considers the nodes pointing to the data to be added and also the nature of the data structures these data will be added to. So, prior to this evaluation, it checks as to whether the constituent noun node to be evaluated has the feature "access" on, in which case it gets the constituent list of the noun node and tries to find the only noun that possibly is linked in. The payload of this noun is the permanent data structure where the data are to be added. If there is no such noun, then a diagnostic message is printed to the user denoting that data cannot be added to non- local data structures.

The case of structural changes in the data base related to access words has to be taken into account. In this case, if a new data structure is created within the local system, then it is checked as to whether it exists in the foreign access lexicon (with the same type). If it is found, it is made a permanent data structure of the local (ASK) system and it is added as a substructure into the payload of the respective foreign access lexicon entry. Since these data structures are treated as nouns in the ASK system, there is no conflict created in any subsequent query that refers to them. On the other hand, ASK system prevents the creation of a second data structure of the same type. This guarantees the uniqueness of the noun pertaining to a group of function words.

The case of the creation of a new access word conflicting, i.e., leading to ambiguity, with an entry of the foreign access lexicon, is treated differently. Since the content of this entry depends upon the instantiation of the whole list of constituents of a noun node (with the "access" feature set on) then this addition implies structural changes in that list as well. Consequently, the foreign access lexicon entry is deleted, and, the next time an access has to be performed, the respective foreign lexicon entry has to be reinstantiated.

This whole design that involves the parser, cases of ambiguity, and the foreign access lexicon guarantees the availability and integration, into the ASK System, of whatever data exist in a foreign system that is accessed.

4.3.2: Example

Certainly, it is possible to go on by giving numerous examples of the foreign access facility, illustrating all the cases the previous section referred to. Instead, a simple example is given, showing the use of the foreign access in the processing of a query combining data from both local and foreign data bases.

In figure 4.7, a small ASK data base is shown in relation to another data base stored in a foreign computer. The ASK data base

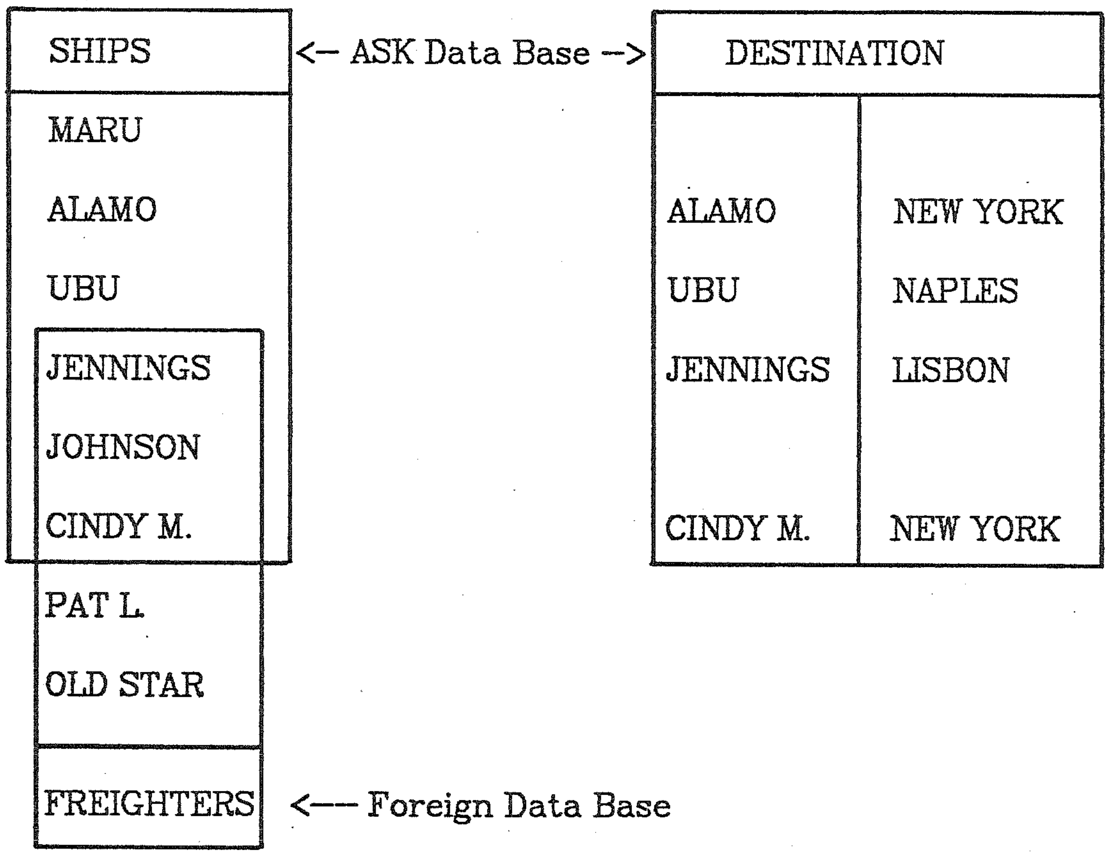

Figure 4.7: Schematic Structure of the Data of ASK and the Foreign Data Base.

consists of a class of ships, an attribute, and an individual who does not belong to any class, but does have an attribute. The foreign data base consists of a class of freighters some of which are under the class "ship" in the ASK data base and one of which is an individual in the ASK system. The word freighter has been defined as an access word in the ASK system. Here is an example of the behavior of the ASK system under these conditions:

```
>What are ships?
  MARU
  ALAMO
  UBU
  JENNINGS
  JOHNSON
  CINDY M.
>What are freighters?
  JENNINGS
  JOHNSON
  CINDY M.
  PAT L.
  OLD STAR
>What is the destination of each ship?
  ship        destination
  ALAMO       NEW YORK
  UBU         NAPLES
  JENNINGS    LISBON
  CINDY M.    NEW YORK
>What is the destination of each freighter?
  freighter   destination
  JENNINGS    LISBON
  CINDY M.    NEW YORK
```

More examples can be found in Appendix B, where the access words defined in the dialogues given in Appendix A are used in queries to the ASK system.

4.4: FOREIGN INPUT

The foreign input facility does not have the need for a separate lexicon. It takes place when the respective "foreign input" dialogue is completed and does not get the parser involved at all. After the completion of a foreign input, all the data are made permanent to the ASK system. The technique of using a subclass is not appropriate in this case, since data brought over are used in instantiating existing or newly created data structures, or simply in extending pre-existing ones. All new objects and data types created are stored in the permanent ASK system lexicon.

CHAPTER 5: CONCLUSIONS

The outcome of this dissertation has been a general design and implementation, in the ASK system, of a foreign access facility. The foreign access feature is highly desirable when data from both the data base itself and foreign data bases are involved in the processing of a query. This way, the informational content of many - if not all - of the currently existing commercially available data bases becomes available for local computation and decision making. Prior to this design, no system providing heterogeneous data base access that placed no requirement for additional programming or particular formats and conventions existed, certainly not with natural language user interface. The only system that uses an expert system type dialogue to establish access is limited to putting a natural language front- end onto a single, relational data base.

The interface between a local and a foreign data base has been decided to be on the leaves of a parsing tree, thus allowing access to more than one data base during the processing of a single query. Only sets and binary relations have been allowed to cross the boundary between the two computers, but provision for texts has also been made. Examining the communication protocols between a user-terminal pair and a local computer, as well as the characteristics of foreign outputs as variable length logical records, an expert system has been built. This expert system is responsible for acquiring information on the characteristics of the communication line to a foreign computer, on the dialogue to be conducted between the two computers (in order for the local computer to login, retrieve the data, and logout), and on unscrambling the foreign output and converting the data into the ASK system's format.

Data brought over are made temporary to the ASK system by being stored in a temporary lexicon. The action of storing the data in a temporary lexicon provides computing efficiency, temporary integration of the foreign data into the ASK system, and elimination of semantic ambiguities in the system.

The access to other data bases can cause delays since the login and data retrieval to other data bases is likely to be a slow process. There are two reasons for not considering this a negative point. The first one is that with relatively fast communication line the major delay cause is the login and logout sequence to the foreign data base, which would not be avoided anyway. The feature of not logging out of a logged in foreign data base is likely to reduce this delay considerably. The second reason is that decision making, requiring prior processing of data residing in different computers scattered around the world, is much more efficient by using the foreign access method, because (1) the foreign data bases can have different query languages that the user is not expected to know; (2) if the data are not integrated - at least temporarily - in the local system, their processing is much more time consuming since manual techniques have to be used.

Finally, variations of the foreign access have been discussed, namely, domestic access and foreign input, providing built-in compatibility to supported data base systems and permanent integration of foreign data into the local system,

BIBLIOGRAPHY

Adiba, M., J.C. Chupin, R. Demolombe, G. Gardarin, and J. Le Bihan, "Issues in Distributed Data Base Management Systems: A Technical Overview," in **Proceedings of the 4th International Conference on Very Large Data Bases**, Paris, 1978.

Adiba, M., and C. Delobel, "The Problem of Cooperation Between Different D.B.M.S.," in **Architecture and Models in Data Base Management Systems**, G.M. Nijssen (editor), North Holland Publishing Company, 1977.

Bachman, Charles W.,"The Programmer as Navigator," **Communications of the ACM**, vol.16, no. 11, November 1973.

Bachman, Charles W., "The Evolution of Storage Structures," **Communications of the ACM**, vol.15, no. 7, July 1972.

Barr, Avron, and E.A. Feigenbaum (editors), **The Handbook of Artificial Intelligence**, vol. 1, HeurisTech Press, Stanford, CA, 1981.

Barr, Avron, and E.A. Feigenbaum (editors), **The Handbook of Artificial Intelligence**, vol. 2, HeurisTech Press, Stanford, CA, 1982.

Bing Yao, S., "Optimization of Query Evaluation Algorithms," **ACM Transactions on Database Systems**, vol. 4, no. 2, June 1979.

Canaday, R.H., R.D. Harrison, E.L. Ivie, J.L. Ryder and L.A. Wehr, "A Back-end Computer for Data Management," **Communications of the ACM**, vol. 17, no. 10, October 1974.

Canning, Richard G., "Distributed Data Systems," **EDP Analyzer**, vol. 14, no. 6, June 1976.

Canning, Richard G., "The Emerging Computer Networks," **EDP Analyzer**, vol. 11, no. 1, January 1973(a).

Canning, Richard G., "Distributed Intelligence in Data Communications," **EDP Analyzer**, vol. 11, no. 2, February 1973(b).

Chen, Peter P., and S. Bing Yao, "Design and Performance Tools for Data Base Systems," in **Proceedings of the 3rd International Conference on Very Large Data Bases**, Tokyo, 1977.

Chen, Peter P., "The Entity-Relationship Model-Toward a Unified View of Data," **ACM Transactions on Database Systems**, vol. 1, no. 1, March 1976.

Childs, D.L., "Extended Set Theory: A General Model for Very Large, Distributed, Backend Information Systems," in **Proceedings of the 3rd International Conference on Very Large Data Bases**, Tokyo, 1977.

CODASYL Data Base Task Group, "Stored-Data Description and Data Translation: A Model and Language," **Information Systems**, vol. 2, 1977.

CODASYL Data Base Task Group, "CODASYL Data Description Language," **Journal of Development**, National Bureau of Standards Handbook 113, U.S. Government Printing Office, Washington, D.C., 1973.

Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," **Communications of the ACM**, vol. 13, no. 6, June 1970.

CSG, **Directory of Online Information Resources**, 10th edition, CSG Press, Kensington, MD, 1982.

Date, C.J., **An Introduction to Database Systems**, vol I, 3rd edition, Addison-Wesley Publishing Co., Reading, MA, 1983.

Date, C.J., **An Introduction to Database Systems**, vol II, Addison-Wesley Publishing Co., Reading, MA, 1983.

Deppe, Mark E., and James P. Fry, "Distributed Data bases - A Summary of Research," **Computer Networks**, vol. 1, 1976.

Gardarin, G., and J. Le Bihan, "An Approach towards a Virtual Data Base Protocol for Computer Networks," in **Proceedings of the AICA-77**, Italy, 1977.

Grosz, Barbara J., "TEAM: A Transportable Natural-Language Interface System," in **Proceedings of the Conference on Applied Natural Language Processing**, Santa Monica, CA, 1983.

Grosz, Barbara J., "Transportable Natural-Language Interfaces: Problems and Techniques," in **Proceedings of the 20th Annual Meeting of the Association for the Computational Linguistics**, Toronto, 1982(a).

Grosz, Barbara J., Norman Haas, Gary Hendrix, Jerry Hobbs, Paul Martin, Robert Moore, Jane Robinson, and Stanley Rosenschein, "DIALOGIC: A Core Natural-Language Processing System," in **Proceedings of the COLING 82**, 1982(b).

Harris, L.R., "ROBOT: A High Performance Natural Language Processor for Data Base Query," **SIGART Newsletter**, no. 61, February 1977(a).

Harris, L.R., "User Oriented Data Base Query with ROBOT Natural Language Query System," in **Proceedings of the 3rd International Conference on Very Large Data Bases**, Tokyo, 1977(b).

Hendrix, Gary G., Earl D. Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum, "Developing a Natural Language Interface to Complex Data," **ACM Transactions on Database Systems**, vol. 3, no. 2, June 1978.

Johnson, Jan, "INTELLECT on Demand," **Datamation**, vol. 27, no. 12, November 1981.

Kay, M., **Experiments with a Powerful Parser**, Technical Report, RM- 5452-PR, the RAND Corporation, Santa Monica, CA, 1967.

Kent, W., "Entities and Relationships in Information," in **Architecture and Models in Data Base Management Systems**, G.M. Nijssen (editor), North Holland Publishing Company, 1977.

Kent, W., "Limitations of Record Based Information Systems," **ACM Transactions on Database Systems**, vol. 4, no. 1, March 1979.

Konolige, Kurt, **A Framework for Portable Natural Language Interface to Large Data Bases**, Technical note 197, SRI International, Menlo Park, California, 1979.

Konolige, Kurt, "A Metalanguage Representation of Relational Databases for Deductive Question-Answering Systems," in **Proceedings of the 7th International Joint Conference on Artificial Intelligence**, Vancouver, 1981.

Le Bihan, J., C. Esculier, G. Le Lann, W. Litwin, G. Gardarin, S. Sedillot, and L. Treille, "SIRIUS: A French Nationwide Project on Distributed Data Bases," in **Proceedings of the 6th International Conference on Very Large Data Bases**, Montreal, 1980.

Peebles, Richard, and Eric Manning, "System Architecture for Distributed Data Management," **Computer**, vol. 11, no. 1, January 1978.

Polk, Warren J., and Kerry Byrd, "Managing the Very Large Database," **Datamation**, September 1981.

Randall, David Lawrence, **Formal Methods in the Foundations of Science**, Ph.D. thesis, Information Science Department, California Institute of Technology, Pasadena, Ca, 1970.

Senko, Michael E., "Information Systems: Records, Relations, Sets, Entities, and Things," **Information Systems**, vol. 1, 1975.

Senko, Michael E., E.B. Altman, M.M. Astrahan and P.L. Fehder, "Data Structures and Accessing in Data-Base Systems: I. Evolution of Information Systems," **IBM Systems Journal**, vol. 12, no. 1, 1973.

Thompson, Bozena H., and Frederick B. Thompson, "Introducing ASK, a Simple Knowledgeable System," in **Proceedings of the Conference on Applied Natural Language Processing**, Santa Monica, CA, 1983.

Thompson, Bozena H., and Frederick B. Thompson, **ASK: A Simple Knowledgeable System**, forthcoming.

Thompson, Bozena H., and Frederick B. Thompson, "Rapidly Extendable Natural Language," in **Proceedings of the Annual Conference of the Association for Computing Machinery**, 1978.

Thompson, Bozena H., and Frederick B. Thompson, "Shifting to a Higher Gear in a Natural Language System," in **Proceedings of the National Computer Conference**, 1981.

Warren, David H.D., "Efficient Processing of Interactive Relational Database Queries Expressed in Logic," in **Proceedings of the 7th International Conference on Very Large Data Bases**, Cannes, 1981.

Yu, Kwang-I, **Communicative Databases**, Ph.D. thesis, Computer Science Department, California Institute of Technology, Pasadena, Ca, 1981.

APPENDIX A: EXAMPLES OF FOREIGN ACCESS DIALOGUES

The following three dialogues with the ASK system aim at establishing foreign data base access to a data base residing on the VAX computer of the Computing Center of the California Institute of Technology. This data base system, named Datatrieve (built by DEC), was a relational one and provided a demonstration file with data on yachts. The first of the following dialogues gives the communication line characteristics of the connection to Datatrieve system and the login and logout sequences, whereas the second and third dialogues introduce the access words "yacht" (class) and "rig" (relation) into the vocabulary of the ASK system. The user's responses to this dialogue are shown in boldface letters.

```
>
>Foreign access
>Data base name : help
   The name of the data base that is going to be accessed.
   This name will be checked against the already known data
   bases. In the case that the data base is a known one
   the system programmer has the option to specify whether
   the existing information can be used as default or not.
>Data Base name : datatrieve
   Acquiring information on the communication line.
>Protocol (default: 1): help
   Protocol to be used after next card reset:
         1 = Asynchronous Protocol
         2 = Data Link Protocol
>Protocol (default : 1) : 1
>Interrupt mask (default : 0) : 0
```

```
>Control block mask (default : 0) : 0
>Modem line interrupt mask (default : 0) : 0
>Connection timeout (default : 0) :
>No activity timeout (default : 0) :
>Lost carrier timeout (default : 40) :
>Transmit timeout (default : 10) :
>Transmit speed (default : 7) : help
  Transmit speed (baud rate).
  Acceptable values:
      0 = external clock 8 =   600
      1 =  50            9 =  1200
      2 =  75           10 =  1800
      3 = 110           11 =  2400
      4 = 134.5         12 =  3600
      5 = 150           13 =  4800
      6 = 200           14 =  9600
      7 = 300           15 = 19200
>Transmit speed (default : 7) : 7
>Receive speed (default : 7) :
>Protocol handshake (default : 0) : help
  Protocol Handshake that is to be used.
      0 = Protocol handshake disabled
      1 = ENQ/ACK with desktop computer as the host
      2 = ENQ/ACK, desktop computer as terminal
      3 = DC1/DC3, desktop computer as host
      4 = DC1/DC3, desktop computer as a terminal
      5 = DC1/DC3, desktop computer both as host and terminal
>Protocol handshake (default : 0) : 0
>Hardware handshake (default : 1) : help
  Hardware handshake that is to be used.
      0 = Handshake OFF, non-modem connection.
      1 = FULL-DUPLEX modem connection
      2 = HALF-DUPLEX modem connection
      3 = Handshake ON, non-modem connection
>Hardware handshake (default : 1) : 1
>Control character mask (default : 6) :
>First protocol character (default : 6) :
>Second protocol character (default : 5) :
>Length of End-of-line sequence (default : 2) :
>First End-of-line character (default : 13) :
>Second End-of-line character (default : 10) :
>Length of Prompt sequence (default : 0) :
>First Prompt character (default : 17) :
```

```
>Second Prompt character (default : 0) :
>Bits per character (default : 2) : help
  The number of bits per character is set as follows:
      0 = 5 bits/character
      1 = 6 bits/character
      2 = 7 bits/character
      3 = 8 bits/character
  When 8 bits/character, parity must be NONE, ODD, or EVEN.
>Bits per character (default : 2) :
>Stop bits per character (default : 0) :
>Parity (default : 2) : help
  The Parity is set as follows:
    0 = NONE; no parity bit is included with any characters.
    1 = ODD;  parity bit is SET if there is an even number
           of "1"s in the character body.
    2 = EVEN; parity bit is OFF if there is an odd number
           of "1"s in the character body.
    3 = ZERO; parity bit is always zero, but parity is not checked.
    4 = ONE;  parity bit is always one, but parity is not checked.
  When 8 bits/character, parity must be NONE, ODD, or EVEN.
>Parity (default : 2) : 2
>Inter-character time gap (default : 0) : 2
>Break time (default : 4) : 8
>Phone number : help
  If the communication with the foreign computer is going to take
  place through the HP 13265A modem with an autodial capability,
  the sequence of the numbers to be dialed should be typed in.
  A ">" in the beginning of the string denotes fast dial mode;
  a "@" causes a delay of approximately 1 sec between dialing
  of two consecutive numbers. Any other character in the string
  is considered as non-existent; i.e., it is ignored.
  e.g.,
   >@ 9{outgoing} @-wait for dial tone-(213)-4494048 CIT TIMEVAX
>Phone number : >9 @ 449 4048 /Timevax - Caltech
  Enter information on how to log in
>Action : help
  The action to be taken. This can be either
  one of the following:
         send
         sendln
         expect
         try
         end try
```

```
        end login
  Then, the computer will ask about the string and the
  the timeout, if expecting a string.
>Action : try
>How many times to try (default : 2) : 20
>Action : sendln
>What to send (default : CR) :
>Action : expect
>What to expect (default : CR) : *
>Timeout (default : 0) : help
  Timeout in seconds when expecting a response
  from the foreign computer. A value of 0
  disables the timeout. This can be a potential
  danger, though, since it can lead to an infinite
  wait time if something goes wrong.
>Timeout (default : 0) : 10
>Action : end try
>Action : sendln
>What to send (default : CR) : 32
>Action : expect
>What to expect (default : CR) : START
>Timeout (default : 0) : 20
>Action : expect
>What to expect (default : CR) :
>Timeout (default : 0) : 20
>Action : try
>How many times to try (default : 2) : 100
>Action : sendln
>What to send (default : CR) :
>Action : expect
>What to expect (default : CR) : Username:
>Timeout (default : 0) : 15
>Action : end try
>Action : sendln
>What to send (default : CR) : AHP
>Action : expect
>What to expect (default : CR) : Password:
>Timeout (default : 0) : 30
>Action : sendln
>What to send (default : CR) : xxx
>Action : expect
>What to expect (default : CR) : $
>Timeout (default : 0) : 300
```

```
>Action : sendln
>What to send (default : CR) : dtr
>Action : expect
>What to expect (default : CR) : DTR>
>Timeout (default : 0) : 40
>Action : sendln
>What to send (default : CR) : set dictionary cdd$top.dtr$lib.demo
>Action : expect
>What to expect (default : CR) : DTR>
>Timeout (default : 0) : 40
>Action : end login
 Enter information on how to log out
>Action : sendln
>What to send (default : CR) :  exit
>Action : expect
>What to expect (default : CR) : $
>Timeout (default : 0) : 40
>Action : sendln
>What to send (default : CR) : logout
>Action : expect
>What to expect (default : CR) : AHP
>Timeout (default : 0) : 40
>Action : end logout
>Access word :
 The Datacomm board configuration, login,
 and logout information for datatrieve
 access has been stored for future use.
>
>
>Foreign access
>Data base name : datatrieve
 DATATRIEVE is a known data base.
 Information regarding the login, logout, and Datacomm board
 configuration exists. Should that information be used as a
 default?
>Yes or No : yes
>Access word : help
 To set up the transfer of data from a foreign Data Base System,
 you need to enter the ASK words you wish to use in referring to the
 corresponding foreign data. Enter one such word here. You will
 then be prompted for the ASK category of this word and for the
 way to retrieve the associated data out of the foreign system. Once
 this information has been entered for the given word, you will be
```

```
  prompted for another word, etc. until you return a nil (carriage
  return only) response.
>Access word: yacht
>ASK word category: help
  To specify the ASK category of the word you have just entered, you
  need to enter whether it is a class, text, attribute or relation
  word.
>ASK word category: class
  Enter information on how to retrieve the data
  out of the foreign computer.
>Action : help
  The action to be taken. This can be any
  one of the following:
          send
          sendln
          expect
          collect
          end collect
          try
          end try
          end data retrieval
  Then, the computer will ask about the string and the
  the timeout, if it is expecting a string.
>Action : sendln
>What to send (default : CR) : ready yachts
>Action : expect
>What to expect (default : CR) : DTR>
>Timeout (default : 0) : 100
>Action : sendln
>What to send (default : CR) : report yachts
>Action : expect
>What to expect (default : CR) : RW>
>Timeout (default : 0) : 100
>Action : sendln
>What to send (default : CR) : print manufacturer
>Action : expect
>What to expect (default : CR) : RW>
>Timeout (default : 0) : 100
>Action : collect
>Action : sendln
>What to send (default : CR) : end-report
>Action : expect
>What to expect (default : CR) : DTR>
```

```
>Timeout (default : 0) : 9000
>Action : end collect
>Action : end data retrieval
  Enter information on the structure
  of the output of the foreign computer.
>[ 1:0 ]Record type : help
  The system has to read the output of foreign computers.
  In order to do that, there must be some information about
  the structure of the output given to the computer. This is
  accomplished through a record structure that has two types
  of records: "normal" and "pick." Whatever is within the body
  of a pick record is picked up. The normal ones are used in
  facilitating the description of the variable length logical
  records that are involved.
>[ 1:0 ]Record type : normal
  Enter information about the beginning of the record.
>[ 1:0 ]Action : help
  The cursors that are used in describing the logical record
  structure of the output of a foreign computer are moved through
  a series of actions. These actions are:
            right       right <number>
            left        left <number>
            up          up <number>
            down        down <number>
            search+     search+<number>
            search-     search-<number>
                        end of string
                        beginning of string
>[ 1:0 ]Action : search+
>[ 1:0 ]String : MANUFACTURER
>[ 1:0 ]Action : down 1
>[ 1:0 ]Action : right 15
>[ 1:0 ]Action :
  Enter restricting conditions for the logical record body.
>[ 1:0 ]Pseudo action : *
>[ 1:0 ]Pseudo action :
>[ 1:0 ]Restricting condition : help
  A condition for a record can be either:
        <condition expression>
        or
        not <condition expression>
  where a condition expression is one of the following:
            match string                blank line
```

```
              blank field                   alphanumeric string
              alphameric string             numeric string
              cursor expression in fieldt
              string in field               string type in field
              true                          false
>[ 1:0 ]Restricting condition : true
>[ 1:0 ]Restricting condition :
>[ 2:1 ]Nested record type : pick
  Enter information about the beginning of the nested record
>[ 2:1 ]Action : down
>[ 2:1 ]Action : left 15
>[ 2:1 ]Action :
  Enter restricting conditions for the logical record body.
>[ 2:1 ]Pseudo action : *
>[ 2:1 ]Pseudo action :
>[ 2:1 ]Restricting condition : string type in field
>[ 2:1 ]String type : help
  A string type can be any one of the following:
                    alphanumeric
                    alphameric
                    numeric
>[ 2:1 ]String type : alphanumeric
>[ 2:1 ]Field type : help
  A field type is any one of the following:
                    column
                    line
                    window
  and it is defined through two cursor positions,
  in any sequence of cursor positions.
>[ 2:1 ]Field type : line
  Enter information about the first argument of the field.
>[ 2:1 ]Pseudo action : *
>[ 2:1 ]Pseudo action :
  Enter information about the second argument of the field.
>[ 2:1 ]Pseudo action : right 15
>[ 2:1 ]Pseudo action :
>[ 2:1 ]Restricting condition :
>[ 3:2 ]Nested record type :
  Enter the end of logical record actions/conditions
>[ 2:1 ]Action : right 15
>[ 2:1 ]Action :
>[ 2:1 ]End of logical record condition : string type in field
>[ 2:1 ]String type : alphanumeric
>[ 2:1 ]Field type : line
  Enter information about the first argument of the field.
>[ 2:1 ]Pseudo action : down
>[ 2:1 ]Pseudo action : left 15
>[ 2:1 ]Pseudo action :
  Enter information about the second argument of the field.
>[ 2:1 ]Pseudo action : down
>[ 2:1 ]Pseudo action :
>[ 2:1 ]End of logical record condition :
>[ 3:1 ]Same level record type :
  Enter the end of logical record actions/conditions.
>[ 1:0 ]Action : *
>[ 1:0 ]Action :
>[ 1:0 ]End of logical record condition : true
>[ 1:0 ]End of logical record condition :
>[ 3:0 ]Same level record type :
>Access word :
  This Foreign access has been set up.
>
>
>Foreign access
>Data base name : datatrieve
  DATATRIEVE is a known data base.
  Information regarding the login, logout, and Datacomm board
  configuration exists. Should that information be used as a
  default?
>Yes or No : yes
>Access word : rig
>ASK word category: relation
  Enter information on how to retrieve the data
  out of the foreign computer.
>Action : sendln
>What to send (default : CR) : ready yachts
>Action : expect
>What to expect (default : CR) : DTR>
>Timeout (default : 0) : 100
>Action : sendln
>What to send (default : CR) : find yachts
>Action : expect
>What to expect (default : CR) : DTR>
>Timeout (default : 0) : 100
>Action : sendln
>What to send (default : CR) : report yachts
```

```
>Action : expect
>What to expect (default : CR) : RW>
>Timeout (default : 0) : 100
>Action : sendln
>What to send (default : CR) : print manufacturer, rig
>Action : expect
>What to expect (default : CR) : RW>
>Timeout (default : 0) : 100
>Action : collect
>Action : sendln
>What to send (default : CR) : end-report
>Action : expect
>What to expect (default : CR) : DTR>
>Timeout (default : 0) : 9000
>Action : end collect
>Action : end data retrieval
 Enter information on the structure
 of the output of the foreign computer.
>[ 1:0 ]Record type : normal
 Enter information about the beginning of the record.
>[ 1:0 ](*)Action : search+
>[ 1:0 ](*)String : MANUFACTURER
>[ 1:0 ](*)Action : down 1
>[ 1:0 ](*)Action : right 14
>[ 1:0 ](*)Action :
>[ 1:0 ](#)Action : search+
>[ 1:0 ](#)String : RIG
>[ 1:0 ](#)Action : down 1
>[ 1:0 ](#)Action : right 5
>[ 1:0 ](#)Action :
 Enter restricting conditions for the logical record body.
>[ 1:0 ](*)Pseudo action : *
>[ 1:0 ](*)Pseudo action :
>[ 1:0 ](#)Pseudo action : #
>[ 1:0 ](#)Pseudo action :
>[ 1:0 ](*)Restricting condition : true
>[ 1:0 ](*)Restricting condition :
>[ 1:0 ](#)Restricting condition : true
>[ 1:0 ](#)Restricting condition :
>[ 2:1 ]Nested record type : pick
 Enter information about the beginning of the nested record.
>[ 2:1 ](*)Action : down
>[ 2:1 ](*)Action : left 14
>[ 2:1 ](*)Action :
>[ 2:1 ](#)Action : down
>[ 2:1 ](#)Action : left 10
>[ 2:1 ](#)Action :
 Enter restricting conditions for the logical record body.
>[ 2:1 ](*)Pseudo action : *
>[ 2:1 ](*)Pseudo action :
>[ 2:1 ](#)Pseudo action : #
>[ 2:1 ](#)Pseudo action :
>[ 2:1 ](*)Restricting condition : string type in field
>[ 2:1 ](*)String type : alphanumeric
>[ 2:1 ](*)Field type : line
 Enter information about the first argument of the field.
>[ 2:1 ](*)Pseudo action : *
>[ 2:1 ](*)Pseudo action :
 Enter information about the second argument of the field.
>[ 2:1 ](*)Pseudo action : right 14
>[ 2:1 ](*)Pseudo action :
>[ 2:1 ](*)Restricting condition :
>[ 2:1 ](#)Restricting condition : string type in field
>[ 2:1 ](#)String type : alphanumeric
>[ 2:1 ](#)Field type : line
 Enter information about the first argument of the field.
>[ 2:1 ](#)Pseudo action : #
>[ 2:1 ](#)Pseudo action :
 Enter information about the second argument of the field.
>[ 2:1 ](#)Pseudo action : right 10
>[ 2:1 ](#)Pseudo action :
>[ 2:1 ](#)Restricting condition :
>[ 3:2 ]Nested record type :
 Enter the end of logical record actions/conditions.
>[ 2:1 ](*)Action : right 14
>[ 2:1 ](*)Action :
>[ 2:1 ](#)Action : right 10
>[ 2:1 ](#)Action :
>[ 2:1 ](*)End of logical record condition : string type in field
>[ 2:1 ](*)String type : alphanumeric
>[ 2:1 ](*)Field type : line
 Enter information about the first argument of the field.
>[ 2:1 ](*)Pseudo action : down
>[ 2:1 ](*)Pseudo action : left 14
>[ 2:1 ](*)Pseudo action :
 Enter information about the second argument of the field.
```

```
>[ 2:1 ](*)Pseudo action : **down**
>[ 2:1 ](*)Pseudo action :
>[ 2:1 ](*)End of logical record condition :
>[ 2:1 ](#)End of logical record condition : string type in field
>[ 2:1 ](#)String type : alphanumeric
>[ 2:1 ](#)Field type : line
 Enter information about the first argument of the field.
>[ 2:1 ](#)Pseudo action : down
>[ 2:1 ](#)Pseudo action : left 10
>[ 2:1 ](#)Pseudo action :
 Enter information about the second argument of the field.
>[ 2:1 ](#)Pseudo action : down
>[ 2:1 ](#)Pseudo action :
>[ 2:1 ](#)End of logical record condition :
>[ 3:1 ]Same level record type :
 Enter the end of logical record actions/conditions.
>[ 1:0 ](*)Action : *
>[ 1:0 ](*)Action :
>[ 1:0 ](#)Action : #
>[ 1:0 ](#)Action :
>[ 1:0 ](*)End of logical record condition : true
>[ 1:0 ](*)End of logical record condition :
>[ 1:0 ](#)End of logical record condition : true
>[ 1:0 ](#)End of logical record condition :
>[ 3:0 ]Same level record type :
>Access word :
 This Foreign access has been set up.
>
```

APPENDIX B: EXAMPLE FOR FOREIGN DATA BASE ACCESS

The following example demonstrates the use of the access words defined during the dialogues of Appendix A. This example is the recording of an ASK system session. Initially, the whole session is shown as it is experienced by the user. Then, the same example is repeated including comments that show the various phases of the foreign access that were going on behind the scenes. In the repeated example, the comments are enclosed in square brackets. User's queries are shown in boldface characters, whereas the queries sent by the local system to the foreign data base are in italic characters.

- User's experience:

```
 Welcome to ASK.  Please identify yourself.
>MASTER
>Password?
 MASTER is in COMMAND.
>enter BASE
 You are in BASE.  Proceed.
>What are yachts?
 There are 62 lines in your answer.
>Which lines do you want? all
 ALBERG
 ALBIN
 AMERICAN
 BAYFIELD
 BLOCK I.
 BOMBAY
 BUCCANEER
 C&C
```

CABOT
CAL
CAPE DORY
CAPITAL
CARIBBEAN
CHALLENGER
CHRIS-CRAF
COLUMBIA
DOUGLAS
DOWN EAST
DUFOUR
EASTWARD
ENCHILADA
ENDEAVOUR
ERICSON
FISHER
FJORD
GRAMPIAN
GULFSTAR
HUNTER
I. TRADER
IRWIN
ISLANDER
LINDSEY
MARIEHOLD
METALMAST
MOODY
NAUTOR
NEWPORT
NICHOLSON
NORTHERN
O'DAY
OLYMPIC
ONTARIO
PACESHIP
PEARSON
RANGER
RHODES
ROBERTS
ROGGER FD
RYDER
S2
SABRE

```
  SALT
  SAN JUAN
  SCAMPI
  SOLNA CORP
  TA CHIAO
  TANZER
  VENTURE
  WESTERLY
  WESTSAIL
  WINDPOWER
  WRIGHT
>What are rigs?
  KETCH
  SLOOP
  MS
>exit
  You have returned to COMMAND.
>
```

- Specially monitored example:

```
  Welcome to ASK.  Please identify yourself.
>MASTER
>Password?
  MASTER is in COMMAND.
>enter BASE
  You are in BASE.  Proceed.
>What are yachts?
[--> Foreign Access starts ]
[--> The communication board is set up]
[--> The phone number of foreign data base computer is dialed]
*
SERVICE 32 START

CSS Time-sharing VAX version V3.4
 18+0 jobs; load  1.56  2.16  2.22


Username: AHP
Password:
      Welcome to VAX/VMS version V3.4 on node TIMEVX
```

```
$ dtr

VAX-11 Datatrieve V2.0
DEC Query and Report System
Type HELP for help
DTR> set dictionary cdd$top.dtr$lib.demo
[--> Login completed; Access to the data starts]
DTR> ready yachts
DTR> report yachts
RW> print manufacturer
RW> end-report


                              27-Nov-1983
                              Page 1


           MANUFACTURER

            ALBERG
            ALBIN
            ALBIN
            ALBIN
            AMERICAN
            AMERICAN
            BAYFIELD
            BLOCK I.
            BOMBAY
            BUCCANEER
            BUCCANEER
            C&C
            CABOT
            CAL
            CAL
            CAL
            CAL
            CAL
            CAPE DORY
            CAPE DORY
            CAPE DORY
            CAPITAL
            CARIBBEAN
            CHALLENGER
```

CHALLENGER
CHALLENGER
CHRIS-CRAF
COLUMBIA
COLUMBIA
COLUMBIA
DOUGLAS
DOWN EAST
DOWN EAST
DUFOUR
EASTWARD
ENCHILADA
ENDEAVOUR
ERICSON
ERICSON
FISHER
FISHER
FJORD
GRAMPIAN
GRAMPIAN
GRAMPIAN
GRAMPIAN
GRAMPIAN
GULFSTAR
HUNTER
HUNTER
I. TRADER
IRWIN
IRWIN
IRWIN

MANUFACTURER

IRWIN
ISLANDER
ISLANDER
ISLANDER
ISLANDER
ISLANDER

LINDSEY
MARIEHOLD
METALMAST
MOODY
NAUTOR
NEWPORT
NEWPORT
NEWPORT
NICHOLSON
NORTHERN
NORTHERN
O'DAY
O'DAY
OLYMPIC
ONTARIO
ONTARIO
PACESHIP
PEARSON
PEARSON
PEARSON
PEARSON
PEARSON
PEARSON
PEARSON
PEARSON
PEARSON
PEARSON
RANGER
RANGER
RANGER
RANGER
RHODES
ROBERTS
ROBERTS
ROGGER FD
RYDER
S2
S2
SABRE
SALT
SAN JUAN
SAN JUAN
SCAMPI

```
                SOLNA CORP
                TA CHIAO
                TANZER
                TANZER
                VENTURE

                            27-Nov-1983
                            Page 3


               MANUFACTURER

                VENTURE
                WESTERLY
                WESTSAIL
                WINDPOWER
                WRIGHT

DTR>
[--> Foreign Access ends]
 There are 62 lines in your answer.
>Which lines do you want? all
 ALBERG
 ALBIN
 AMERICAN
 BAYFIELD
 BLOCK I.
 BOMBAY
 BUCCANEER
 C&C
 CABOT
 CAL
 CAPE DORY
 CAPITAL
 CARIBBEAN
 CHALLENGER
 CHRIS-CRAF
 COLUMBIA
 DOUGLAS
 DOWN EAST
 DUFOUR
 EASTWARD
 ENCHILADA
 ENDEAVOUR
 ERICSON
 FISHER
 FJORD
 GRAMPIAN
 GULFSTAR
 HUNTER
 I. TRADER
 IRWIN
 ISLANDER
 LINDSEY
 MARIEHOLD
 METALMAST
 MOODY
 NAUTOR
 NEWPORT
 NICHOLSON
 NORTHERN
 O'DAY
 OLYMPIC
 ONTARIO
 PACESHIP
 PEARSON
 RANGER
 RHODES
 ROBERTS
 ROGGER FD
 RYDER
 S2
 SABRE
 SALT
 SAN JUAN
 SCAMPI
 SOLNA CORP
 TA CHIAO
 TANZER
 VENTURE
 WESTERLY
 WESTSAIL
 WINDPOWER
 WRIGHT
```

>What are rigs?

[--> Access to the data starts; no login is necessary]

```
DTR> ready yachts
DTR> find yachts

[113 records found]
DTR> report yachts
RW> print manufacturer, rig
RW> end-report

                          27-Nov-1983
                          Page 1

 MANUFACTURER                   RIG

  ALBERG                       KETCH
  ALBIN                        SLOOP
  ALBIN                        SLOOP
  ALBIN                        SLOOP
  AMERICAN                     SLOOP
  AMERICAN                     MS
  BAYFIELD                     SLOOP
  BLOCK I.                     SLOOP
  BOMBAY                       SLOOP
  BUCCANEER                    SLOOP
  BUCCANEER                    SLOOP
  C&C                          SLOOP
  CABOT                        SLOOP
  CAL                          SLOOP
  CAL                          SLOOP
  CAL                          SLOOP
  CAL                          SLOOP
  CAL                          SLOOP
  CAPE DORY                    SLOOP
  CAPE DORY                    SLOOP
  CAPE DORY                    SLOOP
  CAPITAL                      SLOOP
  CARIBBEAN                    SLOOP
  CHALLENGER                   SLOOP
  CHALLENGER                   SLOOP
  CHALLENGER                   KETCH
  CHRIS-CRAF                   SLOOP
  COLUMBIA                     SLOOP
```

COLUMBIA	SLOOP
COLUMBIA	SLOOP
DOUGLAS	SLOOP
DOWN EAST	SLOOP
DOWN EAST	SLOOP
DUFOUR	SLOOP
EASTWARD	MS
ENCHILADA	SLOOP
ENDEAVOUR	SLOOP
ERICSON	SLOOP
ERICSON	SLOOP
FISHER	KETCH
FISHER	KETCH
FJORD	MS
GRAMPIAN	SLOOP
GRAMPIAN	SLOOP
GRAMPIAN	SLOOP
GRAMPIAN	SLOOP
GRAMPIAN	KETCH
GULFSTAR	KETCH
HUNTER	SLOOP
HUNTER	SLOOP
I. TRADER	KETCH
IRWIN	SLOOP
IRWIN	SLOOP
IRWIN	KETCH

27-Nov-1983
Page 2

MANUFACTURER	RIG
IRWIN	SLOOP
ISLANDER	SLOOP
ISLANDER	SLOOP
ISLANDER	SLOOP
ISLANDER	SLOOP
ISLANDER	KETCH
LINDSEY	MS
MARIEHOLD	SLOOP
METALMAST	SLOOP
MOODY	SLOOP

NAUTOR	SLOOP
NEWPORT	SLOOP
NEWPORT	SLOOP
NEWPORT	SLOOP
NICHOLSON	SLOOP
NORTHERN	SLOOP
NORTHERN	KETCH
O'DAY	SLOOP
O'DAY	SLOOP
OLYMPIC	KETCH
ONTARIO	SLOOP
ONTARIO	SLOOP
PACESHIP	SLOOP
PEARSON	SLOOP
PEARSON	SLOOP
PEARSON	SLOOP
PEARSON	SLOOP
PEARSON	SLOOP
PEARSON	SLOOP
PEARSON	SLOOP
PEARSON	KETCH
PEARSON	SLOOP
PEARSON	KETCH
RANGER	SLOOP
RANGER	SLOOP
RANGER	SLOOP
RANGER	SLOOP
RHODES	SLOOP
ROBERTS	SLOOP
ROBERTS	SLOOP
ROGGER FD	MS
RYDER	SLOOP
S2	SLOOP
S2	SLOOP
SABRE	SLOOP
SALT	SLOOP
SAN JUAN	SLOOP
SAN JUAN	SLOOP
SCAMPI	SLOOP
SOLNA CORP	SLOOP
TA CHIAO	SLOOP
TANZER	SLOOP
TANZER	SLOOP

```
 VENTURE                              SLOOP

                                  27-Nov-1983
                                  Page 3


 MANUFACTURER                          RIG

  VENTURE                             SLOOP
  WESTERLY                            SLOOP
  WESTSAIL                            SLOOP
  WINDPOWER                           SLOOP
  WRIGHT                              SLOOP

DTR>
[--> Foreign Access ends ]
  KETCH
  SLOOP
  MS
>
>exit
[--> Logout actions start ]
exit
$ logout
  AHP     logged out at 27-NOV-1983 19:21:45.05
[--> Logout actions end ]
[--> Phone line is disconnected]
  You have returned to COMMAND.
>
```