भारत सरकार
योजना आयोग
राष्ट्रीय सूचना विज्ञान केन्द्र
पंजाब राज्य एकक
एस सी ओ 69, सैक्टर 17-डी
चण्डीगढ़-160017

सत्यमेव जयते

Government of India
Planning Commission
NATIONAL INFORMATICS CENTRE
Punjab State Unit
SCO 69, Sector 17-D,
Chandigarh-160017

Ref. No. NIC/PSU/93/ May 24, 1993.

# C E R T I F I C A T E

It is certified that Mr. Himnish Narang and Miss. Jagmeet Kaur, final year students of Master in Computer Applications of Thapar Institute Of Engineering & Technology, Patiala, undertook a project entitled "NATURAL LANGUAGE PROCESSING SYSTEM USING PUNJABI (NLPS-PUNJABI)" at N.I.C. Punjab state Unit, Chandigarh from January 15, 1993 to May 24, 1993 and successfully completed the same.

They are Hard Working, Sincere and amiable. We wish them all success in their future endeavors.

P.S. Coordinator

24/5/93

( A K AGGARWAL)
State Informatics Officer

Project Supervisor

24/5/93

( N S ARNEJA)
Systems Analyst

## ACKNOWLEDGEMENT

The formal acknowledgement will hardly be sufficient in expressing our deep sense of gratitude to Mr. Anshul Kumar Aggrawal, State Informatics Officer, National Informatic Centre( Punjab State Unit), Chandigarh, for his valuable guidance and continuous encouragement to accomplish this project.

We owe our sincere gratitude towards our project guide Mr. Narinder Singh Arneja, System Analyst at National Informatic Centre for explaining us all the details of the project and all kind of help provided during the course of our project.

We are highly indebted to Mr. Madhu Sudan Dada, Senior System Analyst at National Informatic Centre, for his valueable discussions and active help throughout all the phases of the system development.

We will be ever grateful to Prof. Radha Krishnan at Technical Teachers Training Institue, sector-26, Chandigarh, for providing us the books and journals and also his timely suggestions and help.

We are also thankful to Mr. Arshinder Singh Chawla of Punjab Engineering College, Chandigarh, for his timely help and valuable suggestions.

We are also thankful to Mr. Sunil Mahajan, student B.E. (Final Year) (Computer Science), Punjab Engineering College , Chandigarh, for his timely help and friendly co-operation to complete the project.

We are highly obliged to the rest of the staff at National Informatic Centre, Punjab State Unit, for their kind co-operation and generous help.

Himnish Narang
(Himnish Narang)
Jagmeet Kaur
(Jagmeet Kaur)

INDEX


| CONTENTS | Page # |
|---|---|
| 1. INTRODUCTION TO ORGANISATION | 1-4 |
| 2. INTRODUCTION TO NATURAL LANGUAGE PROCESSING | 5-21 |
| 3. SYSTEM ANALYSIS | 22-36 |
| a) PROBLEM DEFINITION | 22 |
| b) ANALYSIS OF THE PROBLEM | 25 |
| c) PROPOSED SYSTEM | 35 |
| 4. SYSTEM DESIGN | 37-63 |
| a) ENVIRONMENT | 37 |
| b) SYSTEM DESIGN | 44 |
| c) SYSTEM FLOW CHARTS | 55 |
| 5. IMPLEMENTATION AND TESTING | 64-89 |
| a) DESCRIPTION OF PROCEDURES | 64 |
| b) TESTING | 76 |
| c) PROCESSING LOGIC / PSEUDOCODE | 82 |
| 6. CONCLUSION & REMARKS | |
| a) ENHANCEMENTS | 90 |
| 7. ANNEXURES | 91 |
| 8. BIBLIOGRAPHY | 93 |

# NATIONAL INFORMATICS CENTRE

## A PROFILE

## NATIONAL INFORMATICS CENTRE : A PROFILE

In 1976, under the Department of Electronics , nucleus of experts was formed for the introduction of modern information technology and this was termed as National Informatics Centre (NIC). During the past decade NIC has been in the forefront in field of Information Technology. Its pioneering efforts for the bringing about a change in the operation of management techniques with reference to taxation administration , agriculture , health and all other areas of national economy.

In addition , NIC has set up a most modern satellite based network of computers , starting from district levels to state , regional and central systems interconnections at every level. The system is operational and this opens up an enormous potential for data collection , data processing , analysis for administrative and decision making purposes. NIC has also been able to make a headway in creating and introducing an awareness in government departments for computer based information systems as an effective tool for decision support. It is the information that will keep the vitality of the economy in future. The economy would become an "Information Movement and Management (IM&M) " economy. Today , it is not only the most sought

after costly commodity , but also the mainstay of decision making process in any organisational set up.

## ROLE OF NIC IN GOVERNMENT INFORMATICS

In order to explore and exploit the vast opportunities offered by information technology which is improving and accelerating the planning process and implementation of socio-economic programmes , all-round development growth of the nation , National Informatics Centre , the NODAL organisation of Government Of India to introduce computer based MIS , file-less office concept , electronic mail services and telematic services in the central , state and district govt. departments , has set up a satellite based Computer-Communication Network (NICNET) covering all districts , state capitals and the centre . This is facilitating the development of District Information system at District level (DISNIC) and essential databases for decision making at various levels at state and central govt. departments.

NICNET comprises of

- very large computers (NEC-S1000) at the NIC - regional centres (Delhi, Pune, Bubneshwar and Hyderabad )
- ND-550 or equivalent super mini computers at state capitals , for providing informatics services to the states

- super PC-AT computer systems at each district
- Mother Earth Station at NIC Headquarters ,Delhi and a Micro Earth Station at each of NIC state units
- District Informatics Centres, for exchange of information via satellite (INSAT-1D).

## INFORMATION FLOW

Each District Informatics Centre has facilites to process information for monitoring socio-economic development activities within the district. Each District Informatics Centre has communication link with the NICNET for theflow of information between any two nodes of NICNET centres.

The State Informatics Centre supports and coordinates all the activities of district centres and in addition, supports the state government departments. It also provides interactive facility to all the state govt. departments and district administrations. The satellite network has high reliability and is available all the time for efficient information flow.

The State, Regional and National Centres have facilities to share loads and also serve as back up to one another. The Regional Centres provide technical support to the states in the respective regions. They provide

specialised peripherals and high processing power for high level modelling and simulation studies. It also serves as a repository for compilation of information on the state systems, if required. In addition, the Centre has sophisticated research and development projects to develop relevant software tools to support district, state and regional level requirements.

# INTRODUCTION
# TO
# NATURAL LANGUAGE PROCESSING

## INTRODUCTION TO NATURAL LANGUAGE PROCESSING

Before we go into the detailed description of our system we would like to explain what is meant by natural language proceesing and other related topics such as syntactic analysis , lexical analysis,semantic analysis etc.

Natural Language Processing,written in short as NLP, tries to make the computer capable of understanding commands written in standard human language. Developing programs to understand natural language is important because a natural form of communication with system is essential for user acceptance. We say that a program understands a natural language if it behaves by taking a correct or acceptable action in response to the input. For example,we say a person demonstrates understandig if he responds with correct answers to a question. The action taken need not be an external response.It may simply be the creation of some internal data structure as would occur in learning some new facts.

## OVERVIEW OF LINGUISTIC

An understanding of lingustic is not essential to understand natural language understanding , but we must be familiar with the basics of grammar,which is certainly important. We must understand how words and sentences are combined to produce meaningful word strings before we can expect to design successful language understanding systems. A sentence is made up of words which express a complete thought. To express a complete thought a sentence must have a subject and a predicate. The subject is what sentence is about and the predicate says something about subject.

Sentences are classified by structure and usage . A simple sentence has one independent clause comprised of a subject and a predicate. A compound sentence consists of two or more independent clauses connected by a conjuction or a semicolon ' . A complex sentence consists of an independent clause and one or more dependent clauses. Sentences are used to assert , query and describe.

A word functions in a sentence as a part of speech. Parts of speech for the English language are nouns, pronouns ,verbs , adjectives , adverbs , prepositions , conjuctions and injections.

## SYNTACTIC PROCESSING

Syntactic Processing is the step in which a flat input sentence is converted into a hierarchical structure that corresponds to the units of meaning in the sentence. This process is called parsing. This step plays an important role in many natural language understanding systems for two reasons :

- Semantic processing must operate on sentence constituents. If there is no syntactic parsing step, then the semantics must decide on its own constituents. If parsing is done, on the other hand, it constrains the number of constituents that semantics can consider. Syntactic parsing is computationally less expensive than is semantic processing. Thus it can play a significant role in reducing overall complexity.

- Althogh it is often possible to construct the meaning of a sentence without using grammatical facts, it is not always possible to do so. Consider, for example, the sentences

The satellite orbited Mars.

Mars orbited the satellite.

In the second sentence facts demand an interpretation in which a planet (Mars) revolves around a satellite, despite the improbability of such a scenario.

There are many ways to produce a parse, almost all systems that are actually used have two main components :

- A declarative representation called, a grammar, of syntactic facts about the language.

- A procedure, called a parser, that compares the grammar against input sentences to produce parsed structure.

## GRAMMARS

The common way to represent grammars is -as a set of production rules. Although details of the forms that are allowed in the rules vary, the basic idea remains the same. This basic idea is illustrated in the following figure, which shows a simple cotext free, phrase structure grammar for English.

S ----> <NP> <VP>

NP ----> the <NP1>

NP ----> <PRO>

NP ----> <PN>

NP ----> <NP1>

NP ----> <ADJ> <N>

VP ----> <V>

VP ----> <V> <NP>

```
 N  ----> file/printer
 PN ----> Bill
PRO ----> I
ADJ ----> short/long/fast
  V ----> printed/created/want
```

We read the first rule as " A setence is composed of a noun phrase followed by the verb phrase ." In this grammar the vertical bar represents an "or". Symbols that are further expanded by rules are called non-terminal symbols. Symbols that correspond directly to strings that must be found in an input sentence are called terminal symbols.

## PARSING AND SEARCHING

Before the meaning of a sentence can be determined , the meaning of its constituent parts must be established. This requires the knowledge of the structure of a sentence , the individual words and how the words modify each other. The process of determining the syntactical structure of a sentence is known as parsing.

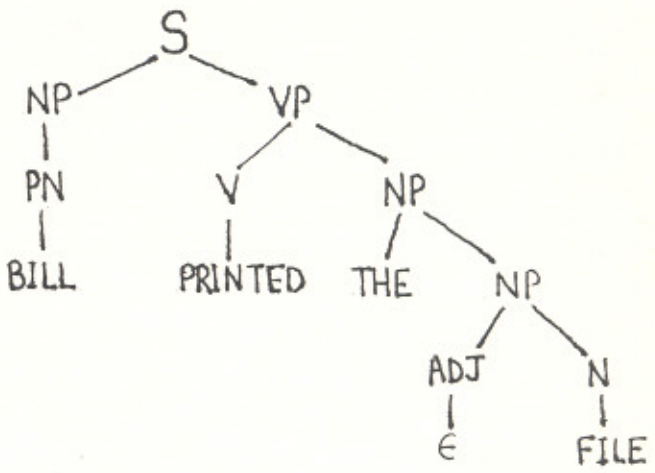
Parsing is the process of analyzing a sentence by taking it apart word by word and determining its structure from its constituent parts and subparts. The parsing process

is basically the inverse of the sentence generation process since it involves finding a grammatical sentence structure from an input string ,the lexical parts or root word must be identified by type and then the role they play in a sentence must be determined. These parts can then be combined successively into larger units until a complete tree structure has been completed.

Regardless of the theoratical basis of the grammar, the parsing process takes the rules of the grammar and compares them against the input sentence. Each rule that matches adds something to the complete structure that is being build for the sentence. The simplest structure to build in a parse tree, which simply records the rules and how they are matched. Following figure shows the parse tree that would be produced for the sentence

"Bill printed the file "

using the above grammar :

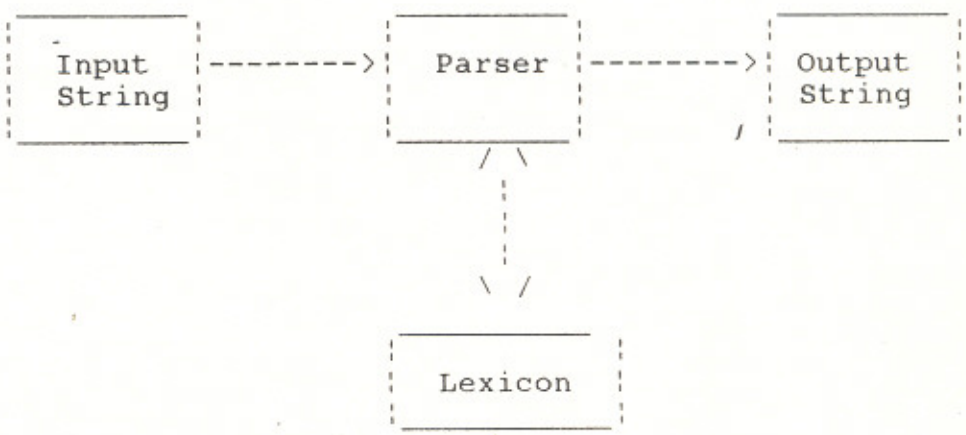



Every node of the parse tree corresponds either to an input word or to a non-terminal in our grammar. Each level in the parse that corresponds to the application of one grammar rule. As a result, it is clear that a grammar specifies two things about a laguage :

- Its weak generative capacity, by which we mean the set of sentences that are contained with in the language. This set, called the set of grammatical sentences, is made up of precisely those sentences that can be completely matched by a series of rules in the grammar.
- Its strong generative caoacity, by which we mean the structure to be assigned to each grammatical sentence of the language.

## LEXICON

To determine the meaning of a word , a parser must have access to a lexicon. When a parser selects a word from the input stream it locates the word in lexicon and obtains word's possible functions and other features , including semantic information. This information is then used in building a tree or other representation structure. The general parsing process is illustrated in the following figure :




A lexicon is a dictionary of words, where each word contains some syntactic,semantic and possibly some pragmatic information.The information in lexicon is needed to help determine the function and meaning of the words in the sentence . Each entry in a lexicon will

contain a root word called the head. Different derivatives of the word , if any, will also be given and the roles it can play in a sentence. A lexicon may also be organised to contain some entries for words with more than one function by giving their separate identities.

## A Simple Parsing problem

Consider a very simple problem ; namely how we set about parsing the english sentence ' John climbed the tree '. The problem is simple for two reasons :

1. The example sentence contains none of the phenomenon that make the parsing written English less than straight forward.
2. By choosing an example from written English , we get a head start , as the datum comes to us prtially preparsed.

## Top-Down Parsing Versus Bottom-Up Parsing

Parsers may be designed to process a sentence using either a top-down or bottom-up approach. A top-down parser begins by hypothesizing a sentence (the symbol S) and successively predicating lower level constituents until individual preterminal symbols are written.

These are then replaced by the input sentence words which match the terminal categories. For example , a possible top-down parse of the sentence

" John climbed the tree "

could be given by

S ----> <NP> <VP>

----> NAME <VP>

----> John <VP>

----> John <V> <NP>

----> John climbed <NP>

----> John climbed <ART> <NP>

----> John climbed the <NP>

----> John climbed the tree

Here

<NP> Noun Phrase

<VP> Verb Phrase

<V> Verb

<ART> Article

A bottom-up parser , on the other hand , begins with the acual words appearing in the sentence and is ,therefore, data driven. A possible bottom-up parse of the same sentence might proceed as follows :

----> John climbed the tree

----> NAME climbed the tree

----> NAME <V> the tree

----> NAME <V> <ART> tree

----> NAME <V> <ART> <N>

----> <NP> <V> <ART> <NP>

----> <NP> <V> <NP>

----> <NP> <VP>

----> S

Words in the input sentence are replaced with their syntactic categories and those in turn are replaced by constituents of the same or smaller size until S has been rewritten or until failure occurs.

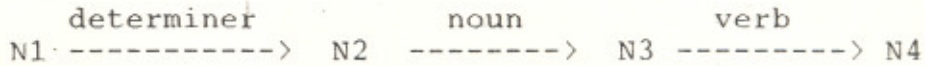
## Breadth-first And Depth-first search

All parsers presented so far have been depth-first search parsers. A depth-first strategy is one in which each hypothesis is pushed as far as it can be before entering upon the explorattion of another hypothesis. Thus , our bottom-up parser went as far as it could with the hypothesis that the verb 'jumped' was the sole content of a dominating verb phrase before being forced to abondon this idea and explore instead the hypothesis of putting the verbs together with the following noun phrase. A

depth first strategy then is a sequential one. By contrast , a breadth first search maintains a number of hypothesis at the same time , advancing each in turn by a single step. Ideally , as time goes by , hypothesis fail and the breadth first device is left with a smaller space hypothesis to consider.

## TRANSITION NETWORKS

Transition networks are another popular method used to represent formal and natural language structures. They are based on the application of directed graphs and finite state automata. A transition network consists of a number of nodes and labeled arcs . The nodes represent different status in traversing a sentence , and the arcs represent the rules or test conditions required to make the transition from one state to the next . A path through a transition network corresponds to a permissible sequence of word types for a given grammar. Thus , if a transition network can be successfully traversed , it will recognise a permissible sentence structure. For example , a network used to recognise a sentence consisting of a determinant , a noun and a verb would be represented by the three node graph as follows :

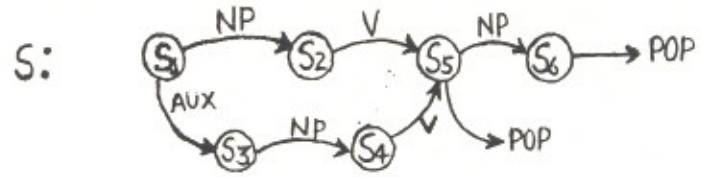


Starting at node N1 , the transition from node N1 to N2 will made if a determiner is the first input word found. If successful , state N2 is entered . The transition from state N2 to N3 can be made if noun is found next.

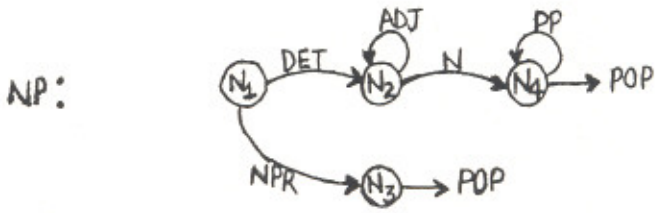
The final transition from N3 to N4 will be made if the last word is a verb. If the three word category sequence is not found , the parse fails. Clearly this type of network is very limited since it will not recognise simple sentences of the form DET N V. The utility of anetwork such as this could be increased if more than a single choice were permitted at some of the nodes.

## RECURSIVE TRANSITION NETWORKS

A Recursive Transition Network (RTN) is a transition network which permits arc labels to refer to other networks ( including the network's own name ), and they in turn may refer back to the referring network rather than just permitting word categories used previously. For example, an RTN is illustrated in the following figure; here the main network calls two subnetworks. The NP and PP networks are illustrated in second and third figures




(i) TOP LEVEL RTN




(ii) NOUN PHRASE SUBNETWORK

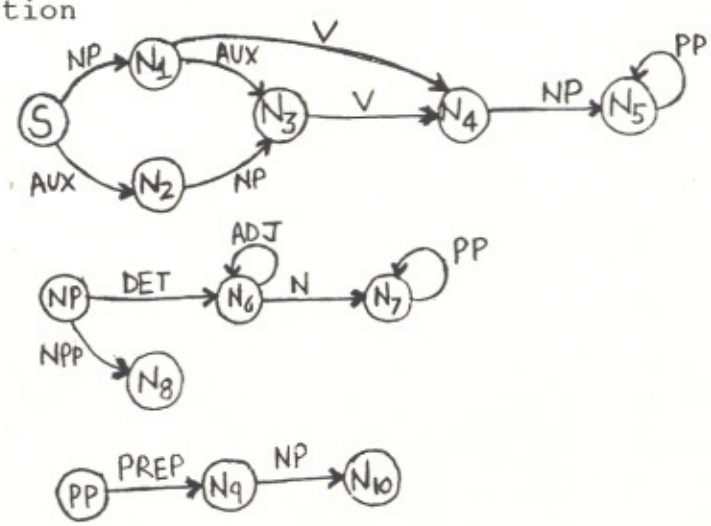

(iii) PREPOSITIONAL PHRASE NETWORK

## AUGMENTED TRANSITION NETWORKS

An Augmeted Transition Network (ATN) is a top-down parsing procedure that allows various kinds of knowledge to be incorporated into the parsing system so that it can operate efficiently. The ATN is similar to finite state machine in which class of labels that can be attached to the arcs that define transition between states has been augmented. Arcs may be labeled with an arbitrary combination ef the following :

- Specific words such as "in".
- Word categories such as "noun".
- Pushes to other networks that recognize significant components of a sentence. For example, a network designed to recognize a propositional phrase (PP) may include an arc that asks for a noun phrase (NP).
- Procedures that perform arbitrary tests on both the current input and on sentence. Components that have already been identified.
- Procedures that build structures that will form part of the final parse.

The figure below shows an example of an ATN in graphical notation




To see how an ATN works, let us trace the execution of this ATN as it parses the following sentence

"The long file has printed"

This execution proceeds as follows :

1. Begin in state S.

2. Push to NP.

3. Do a category test to see if "the" is a determiner.

4. This test succeeds, so get the DETERMINER register to DEFINITE and go to state N6.

5. Do a category test to see if "long" is an adjective.

6. This test succeeds, so append "long" to the list contained in the ADJ register. Stay in state N6.

7. Do a category test to see if "file" is an adjective. This test fails.

8. Do a category test if "file" is a noun. This test succeeds, so set the NOUN register to "file" and go to state N7.

9. Push to PP.

10. Do a category test to see if "has" is a preposition. This test fails, so pop and signal failure.

11. There is nothing else that can be done from state N7. So pop.

12. Do a category test if "has" is a verb. This test succeeds, so set the AUX register to NIL and V register to "has". Go to state N4.

13. Push to state NP. Since the next word, "printed", is not a determiner or proper noun, NP will pop and return failure.

14. The only other thing to do is, state N4 is to halt. But more input remains, so a complete parse has not been formed. Backtracking is now required.

15. The last choice point was at N1, so return there. The registers AUX and V must be unset.

16. Do a category test to see if "has" is an auxillary. This test succeeds, so set the AUX register to "has" and go to state N3.

17. Do a category test to see if "printed" is a verb. This test succeeds, so set the V register to "printed".

18. Now, since the input is exhausted, N4 is an acceptable final state. POP and return the final structure.

## SEMANTIC ANALYSIS

Producing a syntactic parse of a sentence is only the first step towards understanding it. We must still produce a representation of the meaning of the sentence. Because understanding is a mapping process, we must first define the language into which we are trying to map. There is no single, definite language in which all sentence meanings can be described.

# SYSTEM ANALYSIS

## PROBLEM DEFINITION

The main problem that the system is to deal with - is to reply the query asked in the Punjabi language by retreiving the relevent information from the database. The format of the queries asked by the user is not fixed, rather the user can ask in his/her own language in any way he/she likes. So the additional problem is to handle all types of queries .

The main functional subsystem of the Natural Language Interface to the database system are :

1. Accept the queries in Punjabi
2. Perform the lexical analysis of the query to see whether the sentence is valid as per language grammar or not.
3. Perform semantic analysis of the query .(Semantic analysis is undertaken because the same word may represnt different meaning in different situations and the same query can be put in many ways.)
4. Information retreival from the database to reply the query
5. Display reply to the query.

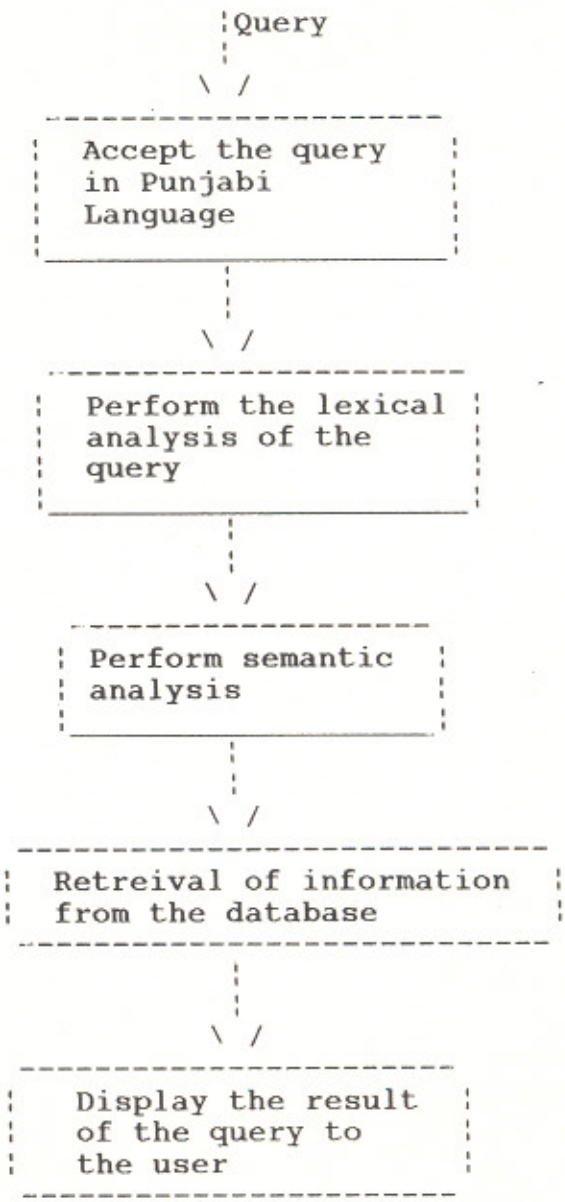
The additional tasks required in working this system are :

1. Creation and maintenence of databases.
2. Creation and maitanence of data dictionary.
3. Making a user interface for the ease of user.

Thus in brief the project problem can be defined as "Design and development of a natural language understanding interface to a database for easy retreival of information by the user."

Diagrammatically the problem can be shown as

(Turn Please)


Query
Accept the query in Punjabi Language
Perform the lexical analysis of the query
Perform semantic analysis
Retreival of information from the database
Display the result of the query to the user

## ANALYSIS OF THE PROBLEM

As described in the definition of the problem the problem was to reply the queries asked by the user in his own language. So, the problem was analysed as follows

Firstly we write the queries, the type of queries that can be asked about a database. As an example we took the village database that contains the information about LUDHIANA district.

After writing the queries that can be there for any database we scanned all the queries so as to know what are the words that are common to all the queries and what is the interrelationship between these words. Then we thought of the words that can be substituted in place of a word in any query by the different users. Consider, for example

ਪਿੰਡ ਅਾਲਮਗੀਰ ਦੀ ਜਨਸੰਖਿਆ ਦੱਸੋ।

ਪਿੰਡ ਅਾਲਮਗੀਰ ਦੀ ਅਾਬਾਦੀ ਦੱਸੋ।

Both the sentences differ in two words only. Syntactically and symantically both the queries are the same. After analysing the words and their aliases in a query the next need was to store the words and the required information about that word. So the solution to the problem was to store the important words in a dictionary.

Now there was the problem of parsing the sentence bacause the core of any NLP system is the parser. The parser is a section of code that reads each sentence, word by word, to decide what is what.

To parse a sentence, we have three methods available.

1. The State machine parser
2. The context free recursive descent parser
3. The Noise disposal parser

Each parser views a sentence differently and has its own specific applications.There are two opposite approaches to NLP. One approach attempts to use all of the information in a sentence, just as a human would. The goal of this approach is to make the computer capable of carrying on conversation. However, this is quite complex and difficult to accomplish. The other approach tries to allow the computer to accept natural language commands, but only to extract the information that is essential to that command- a much easier task to program . The only parser that reaches the first goal is the context free parser and the second goal can be achieved by any of the three parsers.

Each method has been described below. But for our problem we will be using the Noise Disposal Parser.

## 1. STATE MACHINE PARSER

The state machine parser uses the current state of the sentence to predict what type of word may legally follow i.e. in this parser we restrict our language or we may say that we are restricting our grammar because a language os formed by using the grammar. For example, the grammar rule may say that a noun can be followed by a verb or a preposition. In this case we have restricted our grammar. By implementing this parser in 'C' we can use it to break sentences down into their components. We can also use it to determine whethetr the sentence is correctly constructed as per the rules of the grammar or not.

### Disadvantages of the State Machine Parser

1. The worst problem with the state machine parser is its complexity. Even for the simple grammar we need several separate conditional statements to determine the legality of a state transition.
2. Another problem with state machine parser was that it does not know how it got to any particular state i.e. if its current state is noun and a noun can be preceded by a verb or pronoun then it does not know which route it has followed i.e. whether it has come from pronoun or verb to this state.

## 2.THE CONTEXT FREE RECURSIVE DESCENT PARSER

To understand the context free parser, one must look at the construction of the sentence in a completely different way from the way that one looks at the state machine model. For the context free parser, think of the sentence as being composed of various items which themselves are composed of different items and so on until we break the sentence down to its atomic elements, noun, verb, adjectives and so on. As we know the grammar is represented as a set of production rules a context free parser uses these rules to analyze a sentence.

Production rules for some grammars can be written as

```
      S ---> <NP>   <VP>
   <NP> ---> <DET>  <N>
   <NP> ---> <PREP> <NP>
   <VP> ---> <V>    <NP>
   <VP> ---> <V>    <ADV>  <NP>
```

In this grammar the noun phrase is recursive when a prepositional phrase is encountered and the verb phrase is indirectly recursive because it may invoke a noun phrase.

We thought of solving the problem in following thrre ways

1. By using structures

The problem can be solved in 'C' by using structures. By this we mean that we can retrieve information from the database by using structures for each of the file described above i.e. we will be declaring two structures for each of the two files.

**Advantage :**

The only advantage of using this method is that we can retrieve the information by relating the two files easily.

**Disadvantages :**

1. Since we are using a file which contains all the information about a village such as population of the village, number of primary schools or number of high schools, number of wells, number of hospitals, any ITI is there or not etc. etc. So in all there are 113 fields on which a user can put his query. So to declare such a large structure in our program is very cumbersome task to do.

2. Since size of the structure is very large so memory consumption is also very large.

3. This method is not the general one i.e. it is not portable for other systems. If we want to operate this program for other database then we will have to declare the structures for that database, only then we can ask any queries about that database.

## II. User Interface system

This is the second way by which we can reply to the queries asked by the user.In this user friendly system we provide four options to the user :

Whether he wants to get the information at

i) Tehsil Level or

ii) Block Level or

iii) Village Level or

iv) he wants to quit.

If he chooses the option (i) then a list is displayed having the list of all the tehsils. Then we offer the user to select any of the tehsils on which he/she wants to have the information. When he/she has selected the tehsil then we provide the list of fields on which the information is available. By seeing the list of the fields he/she can put in the query and then the reply to that query is displayed.

Similar is the case with other two choices i.e. Block and village.

## Advantages

1. The main advantage of this system is that it is very user friendly i.e. it prompts the user to what to do next.

2. Being user friendly the user finds it very interesting to work on this.

## Disadvantages

1. The disadvantage is that - that it is not dynamic in nature i.e. we can't port this program to other systems.

2. It look likes all other user friendly system, so user might think that the concept of natural language is not there.

## III. DYNAMIC PROGRAM

In this method we do not have either of the concepts used in two previous methods. This method is used so that it can be used for any databases with minor modifications. When we run this program, at the start we display the message that the information about the villages is available with us. You can ask any query about any village. ( This message is displayed in Punjabi language)

After this the user is asked to enter the query. We, then get that query, process it and display the answer to the query.

Advantages

1. It consumes much less memory as compared to the structures method.

2. It is also user friendly because after replying to each query, it is asked that whether the user want to ask more queries or not. If the answer is yes then the user is prompted to enter another query.

3. Beimg dynamic in nature, we can use it with any of the the database.

### Advantages

1.It is easy to implement.

2.We can use it to deal with a sentence at both the word level and the phrase level.

3.It knows where it is in the sentence all the times. This is different from the state machine parser, which had no idea of where it was in a sentence.

### Disadvantages

The main disadvantage with context free parsers is that they may not be handling the many valid ways that can be used to construct a sentence in Punjabi.

## 3. NOISE DISPOSAL PARSER

Certain applications are concerned with only a few key words that a sentence contains and not with all of the associative words that make up a language. Essentially, these type of applications are interested only in the information that a sentence contains. This idea leads to the variation in the context free parser called the noise disposal parser. This parser treats all the words that it does not require as noise and discards them.

This type of parser is quite common in database applications. Since our problem is also related to the

databases, so we can use this parser in our case. For example, if the user wants to enquire about the population of the village Alamgir he/she can ask this in many ways such as

ਪਿੰਡ ਆਲਮਗੀਰ ਵਿੱਚ ਘੰਦਿਆਂ ਦੀ ਗਿਣਤੀ ਦੱਸੋ।

ਪਿੰਡ ਆਲਮਗੀਰ ਦੀ ਜਨਸੰਖਿਆ ਦੱਸੋ।

ਪਿੰਡ ਆਲਮਗੀਰ ਦੀ ਜਨਸੰਖਿਆ ਕਿੰਨੀ ਹੈ।

ਕੀ ਤੁਸੀ ਮੈਨੂੰ ਪਿੰਡ ਆਲਮਗੀਰ ਦੀ ਜਨਸੰਖਿਆ ਦਸ ਸਕਦੇ ਹੋ।

ਪਿੰਡ ਆਲਮਗੀਰ ਵਿੱਚ ਘੰਦਿਆਂ ਦੀ ਗਿਣਤੀ ਕਿੰਨੀ ਹੈ।

In all of the above sentences we will have some useful words i.e. the words that are required to retreive the information. For example, in the first sentence the useful words are ਪਿੰਡ , ਆਲਮਗੀਰ , ਘੰਦਿਆਂ and ਗਿਣਤੀ . The unuseful words are ਦੀ , ਵਿੱਚ and ਦੱਸੋ .

Similarly useful and unuseful words can be extracted from the other sentences.

**Advantages**

1. The principal advantage of a noise disposal parser is that it is simple to implement, and that it gets to the information in the sentence quickly.
2. When it is used with another type of parser, it can reduce the number of variations that must be accepted.

## PROPOSED SYSTEM

To solve the problem describaed in the problem definition we propose a system with the following characteristics :

1. A dictionary that contains the words that are commonly used in queries and their english correspondings,and the punjabi correspondings of the database fields and the values of these fields that are in english with their punjabi correspondents. Along with the words the dictionary contains some other information (semantic information) regarding these words to identify these words.

2. A parser that will extract each word from the query and will see whether it is a valid word or not. Also it will fetch the semantic information about the query from the dictionary database.

3. A generalised method to use this information so that the system can give the reply to the query asked by the user relating to any database, provided the user appends his own dictionary and changes the files in the database.

4. The facility to the user to append his own dictionary and to change the database files.

5. Also the user can see the different fields on which the information is available on the system. By seeing this the user can ask his own query on any of the fields of the database.

# SYSTEM DESIGN

## ENVIRONMENT

For the development of any good computerised system, it is essential to have a good blend of hardware and equally good system software available. NIC being the pioneering national computer organisation has all the latest technology available at its state headquarters. Following is a brief hardware and software used during the course of project development :

### HARDWARE

The whole work is done on HCL PC SUPER-AT and L32QI Dot Matrix Printer whose system configuration is as follows :

### HCL Super Chip PC/AT 386

It is a multiyser system with six dumb terminals attached to it.

| | |
|---|---|
| MAKE | : HCL Super Chip PC/AT 386 |
| MAIN MEMORY | : Base 640 KB Extended 7168 KB |
| SECONDARY MEMORY | : 1 Hard Disk of 300 MB |
| MICROPROCESSOR | : INTEL 80386 running at 20 MHz |
| MATH COPROCESSOR | : INTEL 80387 running at 20 MHz |
| TERMINALS | : 6 HCL make dumb terminals |
| FLOPPY DRIVE | : 5-1/4" High Density Drive |
| CARTRIDGE TAPE DRIVE | : Standard Cartridge Drive supporting 60 MB or higher |

Since our project is to deal with Punjabi, so for doing work in Punjabi we were provided with the GIST (Graphics and Intelligence based Script Technology) terminals. GIST makes it possible to interact with the computer in most Indian languages and a few foreign languages. It adapts available English software to provide a universal script processing environment.

## Compatibility of GIST Terminals

The GIST terminal provides emulation for

(i) IBM ANSI terminal

(ii) DEC VT-52 terminals

(iii) DEC VT-100 terminals

(iv) DEC VT-220 terminals

(v) DEC VT-320 terminals

So computers using any of these terminal types can be upgraded for multi-lingual script interaction by replacing it with a GIST terminals.

Since the terminals attached to the HCL-386 system are having the type DEC VT-100, so the system has been upgraded to have interaction with the GIST terminals.

## PRINTER

The L32QI Printer is a parallel interface dot matrix printer which provides a range of types of high quality printing while maintaining a high standard of performance and reliability. The following features are provided :

- 132 Column printer at 10 cpi
- Bidirectional printing using logic seeking to maximize printing efficiency.
- Standard IBM PC compatible character sets plus eight software selectable alternate international character sets (EPSON FX80 Compatible). Printable draft mode characters may be redefined using FX-80 compatible down loading.
- Correspondence quality printing selectable by push button, microswitch or software command.
- Printing of underline, double width, emphasized, double strike, subscript or superscript characters.
- High resolution Bit Image Graphics. IBM compatible line and Mosaic Graphics in both Draft and CQ modes.
- Switch and software selectable form lengths.
- Various vertical and horizontal tab settings.
- Rear and bottom paper insertion path for fanfold paper, front insertion for cut sheets. Sensor switch detects when paper required.

- Operational control panel with POWER ON and ONLINE indicators and ONLINE, LINE FEED/FORM FEED and Correspondence Quality (CQ) push buttons.
- Integral anti-noise cover with paper tear bar.
- A serial interface option can be implemented by the user at site.

## SYSTEM SOFTWARE

### Operating System (XENIX) :

XENIX is the most popular implementation of the UNIX operating system at the micro level. This syatem is at the heart of more working multi-user business solutions than all other UNIX system versions combined. Today, the SCO XENIX System V offers the most productive, powerful and flexible operating environment available for PCs.

A powerful multi-user, multi-tasking operating system, SCO XENIX System V allows file sharing between independent terminal operators or between application programmers under the control of one or more users. Originally developed by Microsoft Corporation, XENIX has become the standard UNIX System implementation for personal computers today, installed on more than 85% of all Microcomputers running any version of the UNIX System world-wide.

SCO XENIX System V consists of three modules :

(a) Operating System

(b) Development System

(c) Text Processing System

**(a) Operating System :**

It contains the full set of XENIX utilities required to run business applications, administer the system, edit files and communicate with others. , csh(advanced c shell), vsh(visual shell), vi(visual editor), electronic mail, uucp and over 100 other UNIX System commands.
The hard disk can be shared between seperate XENIX and DOS partitions, allowing the user to alternate between operating systems. Utilities are provided for moving files between the XENIX and DOS environments.

**(b) Development System :**

It supplies all the tools needed to write 'C' and assembly language programmes as well as providing a powerful DOS cross development environment to create programs for both DOS and UNIX.

**(c) Text Processing System :**

It contains tools for the preparation of large or complex documents. Included are the nroff, document formatter, macro packages and special formatting tools.

Our system has been developed in 'C' language because it is a powerful tool for system programming.

**'C' Language Characteristics:**

C is a language of functions,data types, assignments and flow control. To program in 'C', a function must be called and most functions return values. The value returned from a function, the value of a data variable , or the value of a constant can be used in an assignment statement to change the value of another variable. With the addition of flow control- if, while, for, do, switch the 'C' language takes on the structure of a high level language, enabling and promoting good programming style.

In 'C', pointer variable can be declared that points to any data type. The address arithmetic of 'C' is sensitive to the properties of the pointer being adjusted. Pointers to functions are also supported.

'C' functions are recursive by default. A function can be coded that does not work in a recursive operation, but the language tends to naturally support recursion and requires little recursion programming effort.

'C' allows to develop a program in multiple source files that are independently compiled. The relocatable object

modules of individual source files are linked into a single executable program.

The 'C' language has no input/ output operations. The compiler compiles a language of functions, and all input and output is done with functions. Because of this feature, a standard library of functions has evolved, and this standard is what gives C its most endearing quality 'C' language code can be portable.

## SYSTEM DESIGN

The main objective of the system is to reply to any type of query, relating to the database and that has some meaning, asked by the user.

The system is based on the relationship of two databases but the user is kept opaque of all these and will feel like working with the single integrated system.

Since our project is related to Punjabi language and the processing on the computer is to be done in English. So to let the system able to process the query, asked in Punjabi, we have maitained a dictionary. This dictionary has been created in **FOXBASE PLUS**. This dictionary is having four fields. The name and description of each field is given below :

| S.NO. | FIELD NAME | DESCRIPTION |
|---|---|---|
| 1. | WORD | Stores the Punjabi word |
| 2. | G_TYPE | Stores the type of the word i.e. whether it is database word or a functiion or a value |
| 3. | E_WORD | Stores the English word correspondig to the Punjabi word stored in the field WORD |

| S.NO. | FIELD NAME | DESCRIPTION |
|---|---|---|
| 4. | USEFUL | Tells whether the word is useful or not |

The user can use this system for any database by just appending his own dictionary and by changing the file names used in the program.

The system will work like this:-

As the user will start the system the following menu will appear

```
**************************************************************
*                                                            *
*       1. ਨਵੀਂ ਡਿਕਸ਼ਨਰੀ ਭਰਨਾ ਚਾਹੁੰਦੇ ਹੋ।                    *
*                                                            *
*       2. ਪ੍ਰਸ਼ਨ  ਪੁੱਛਣਾ ਚਾਹੁੰਦੇ ਹੋ।                       *
*                                                            *
*       3.DATABASE ਦੀ ਜਾਣਕਾਰੀ ਦੇਖਣਾ ਚਾਹੁੰਦੇ ਹੋ।              *
*                                                            *
*       4. ਬਾਹਰ                                             *
*                                                            *
**************************************************************
```

ਆਪਣੀ choice ਭਰੋ।

If the choice is 1

The following line will appear

(a) ਤੁਹਾਨੂੰ ਕੇਵਲ field_name ਜਾਂ ਫੇਰ value ਭਰਨ ਦੀ ਇਜਾਜ਼ਤ ਹੈ।

(b) ਤੁਸੀਂ field_name ਭਰਨਾ ਚਾਹੁੰਦੇ ਹੋ ਜਾਂ value (f/v)

After filling one word, the user is prompted to tell whether he wants to append more words or not. If the user replies affirmatively step (b) appears again.

If choice is 2

the user is prompted to ask the query that he wants to ask.

If choice is 3

The Punjabi correspondent of all the database fields get displayed.

If choice is 4

The user comes out of the system.

As described in the problem definition, we have divided our problem into five parts. So we will be designing our system by solving individually these five parts.

## I. ACCEPTING THE QUERY IN PUNJABI

Accepting any query is very easy in 'C'. Since query is nothing but a string of characters, so we can accept the string using standard get string function (gets()) of 'C'.

## II. LEXICAL AND SEMANTIC ANALYSIS OF THE QUERY

This part consists of two functions

1. Extract one word from the query

2. Retrieve the information stored about that word in the dictionary database .By retrieving the information from the dictionary database, we will get to know whether the word is useful for the retrieval of the query or not. If the word is useful for the retrieval of the query, it is divided on the basis of its property. The useful words can have any of the following properties. It can be

i) A field of the database

For example V_NAME is a field in the village database, that we had taken as an example for the development of this project. When in a query we encounter the word [redacted]

we look for the word in the dictionary database and the following information is stored there

Word = ਪਿੰਡ

TYPE = dw

ENGLISH WORD = VILL_NAME

USE = us

So by retrieving this information, we get to know that the word ਪਿੰਡ corresponds to database field V_NAME.

ii) It can be a value that a database field takes.Since the information will be stored in the data database in English and the same information in the query will be in Punjabi. So there is the need to store in dictionary the values in English and their Punjabi correspondence.

EXAMPLE

Consider the query

ਪਿੰਡ ਕਨੇਚ ਵਿੱਚ ਕਿੰਨੇ ਬੰਦੇ ਹਨ।

By looking at this query, we can see that we have to search for the population in the record where the value of the field V_NAME is KANECH. But in query we encounter the word ਕਨੇਚ . So we can come to know that ਕਨੇਚ correspond to KANECH only if the corresponding information is there in the dictionary.

So the following information regarding the word ਕਨੇਚ will be fetched from the dictionary.

WORD = ਕਨੇਚ

TYPE = v.a

ENGLISH WORD = KANECH

USE = us

3. The third property of the words can be that they tell us about the functions that we are to perform for the retrieval of the query.The examples of these words are count, list etc which correspond to Punjabi words ਕਿੰਨੇ and ਲਿਹੜੇ .

Consider the query

ਪਿੰਡ ਕਨੇਚ ਵਿੱਚ ਚੀਨਿਆਂ ਦੀ ਗਿਣਤੀ ਦੱਸੋ।

The presence of word ਗਿਣਤੀ will tell us the function that we have to perform , when we encounter the word ਗਿਣਤੀ the following information will be retreived from the dictionary database

WORD = ਗਿਣਤੀ

TYPE = fn

E-WORD = COUNT

USE = us

This information signifies, what function we have to perform for the retrieval of the answer to the query.

## III. INFORMATION RETRIEVAL

The next step, after performing the lexical and semantic analysis of the word, is to retrieve the information, required for answering the query. For this purpose, there is the need to store anywhere the useful words of the query and the corresponding information. For this purpose an array of structures is maintained. In this array the fields of a structure are

**1.word**

This field is of type array of characters. In this field the Punjabi word is stored.

**2.eword**

This field is also of type array of characters. In this field the English correspondence of the Punjabi word is stored.

**3. Offset**

This field is of type unsigned character. It stores the offset of the field, if the word corresponds to a field of the database. If we fetch one record of a database in the memory, then the offset of the field tells us at which absolute address, the information for that field will be there.

Absolute address of the field = address at the begining of the record + offset

The offset of a field can be calculated by reading the field description from the database file, sequentially , into a structure whose fields correspond to the description of the field in the database file.

In the database file the description of the field takes 32 bytes, and this information can be read in a structure containing the following components :

**i) field name**

This component of the structure is of type array of characters of length 11.

**ii) field_type**

This component of the structure is of type character and the information regarding the type of field whether it is of type CHARACTER, NUMERIC or DATE gets stored into it.

**iii) garbage1**

This component of the structure is of type long. In the field description of the database file 4 bytes are free. This field is used to store those 4 bytes.

**iv) field length**

This component of the structure is of type character. This field is used to store the length of the field.

v) field decimal

This component is used to store the decimal length of the field.

vi) garbage2

The next 14 bytes of the field description in the database file are free. This component is used to store those 14 bytes.

Hence, by reading the field discription sequentially and by summing up the field length of each field, till we reach the field corresponding to the word, we get the offset of the field.

4.c_type

This component of the structure, of type character, is used to store the information regarding the word whether it is a field name or the value of the field or it is a function. If the word corresponds to the field of the database then 'd' gets stored in it. If the word is a function then 'f' gets stored into it. If the word is the value of the field then 'v' gets stored into it.

5. type

This component of the structure, of type character, is used to store the type of the word like CHARACTER or NUMERIC if the word corresponds to the field of the

database. If the field is of type CHARACTER "C" gets stored into this component. If the field is of type NUMERIC "N" gets stored into this component.

6.length

This component of the structure is of type character and is used to store the length of the field, if the word corresponds to a database field.

7.dep

A limitation has been imposed on the system that the value of the field, must follow the word corresponding to that field in the query. So as soon as we encounter a value, in this component of the previous element of array 'i' gets inserted. With this information, later on we come to know that we have to reach a record where he field with dep='i' has the value stored in the next element of the array.

Now, once the information regarding the useful words in the query gets stored into the array, our next job is how to use this information to retrieve the required information.For this purpose we scan the array until we reach the element of the array with c_type='f'.If the value of the field ewr of that element of the array is

'count' then we count the values in the field for which the query was made.

## IV DISPLAYING THE ANSWER

The next step is that the answer gets displayed.

## V DICTIONARY UPDATION

The next step is the dictionary uptation that is required if the user wants to port this system to some other database.The user can append only field names or the values of the fields that are stored in ENGLISH in the database.The user is asked whether he wants to append in dictionary the value of the field or the field_name. After getting the answer the user is asked to append field_name or value and their Punjabi correspondents. If the user wants to append in dictionary the field then the concatented information

Punjabi_word va English_word us

is appended in the dictionary.

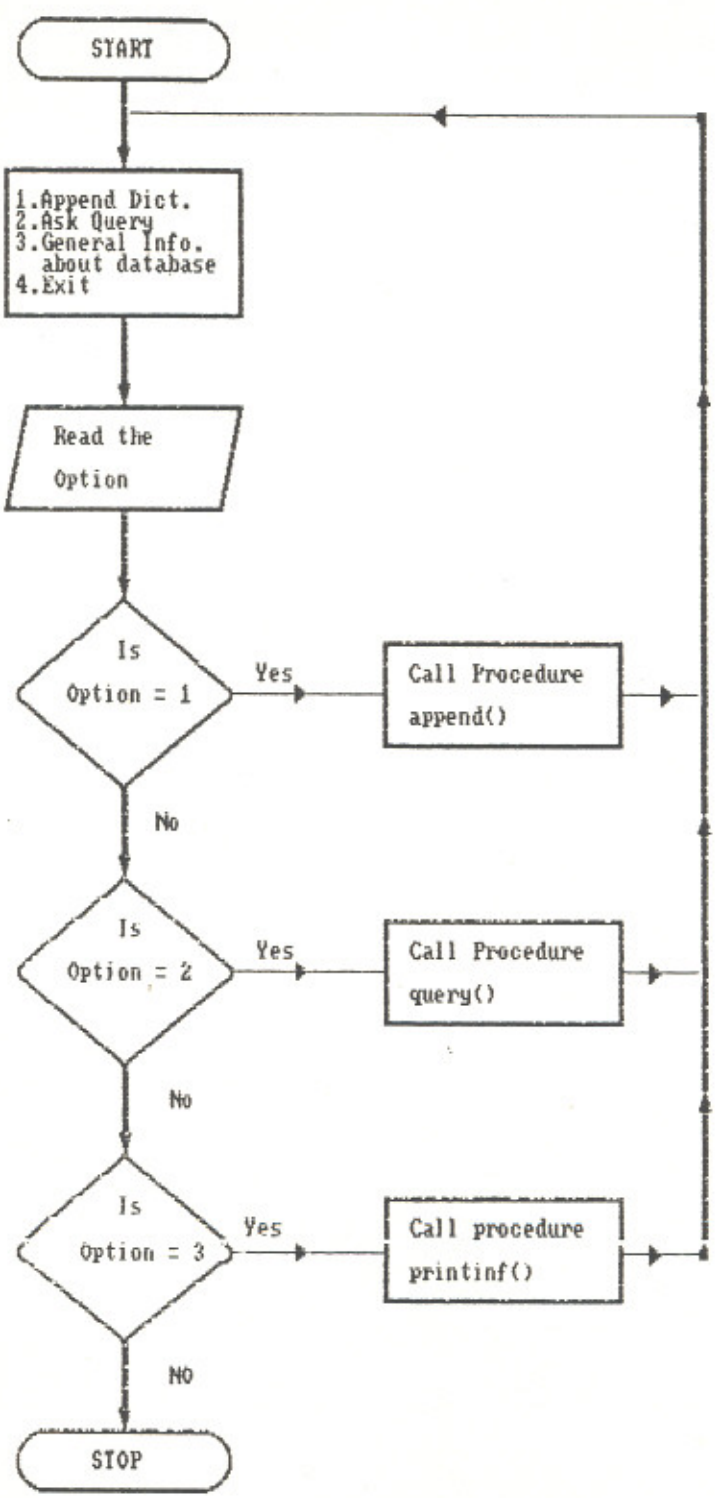
1F the user wants to append fieldname the information

Punjabi_word dw English_word us

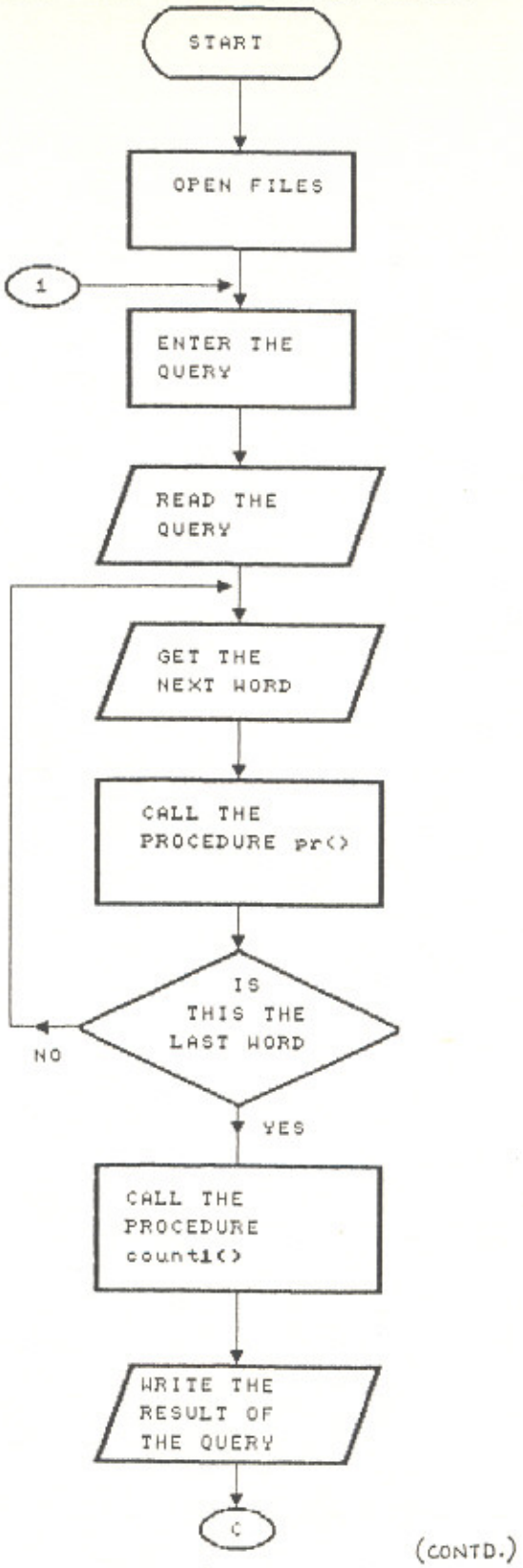
is stored in the dictionary.

# SYSTEM FLOW CHARTS

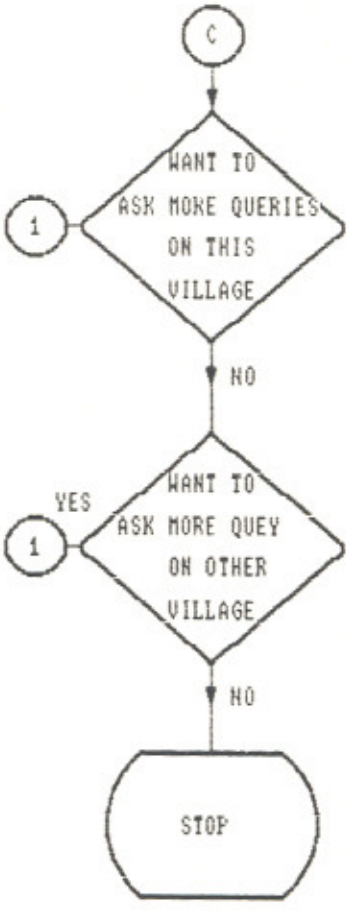
FLOW CHART FOR MAIN PROGRAM



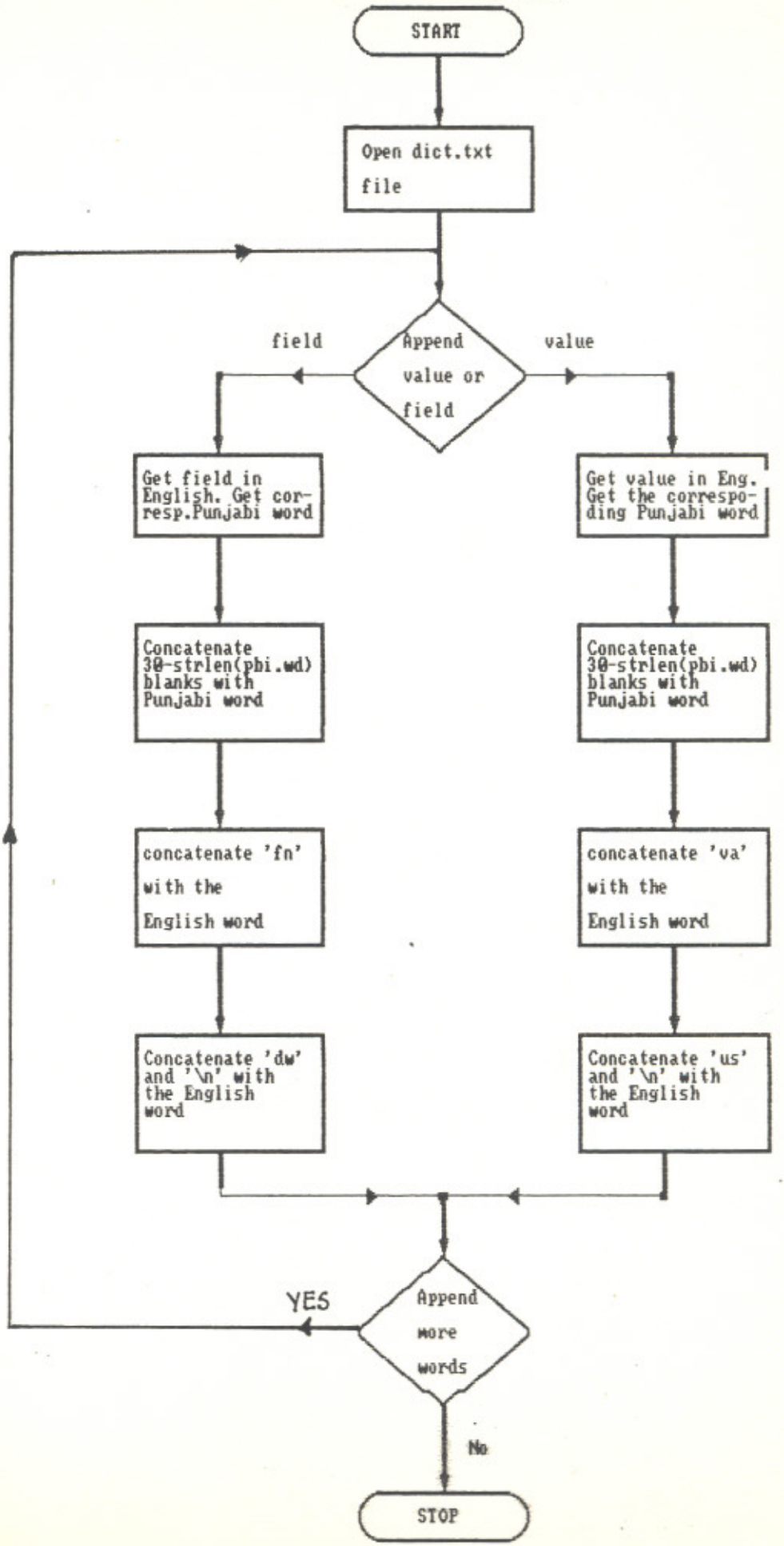
FLOW CHART FOR PROCEDURE query()



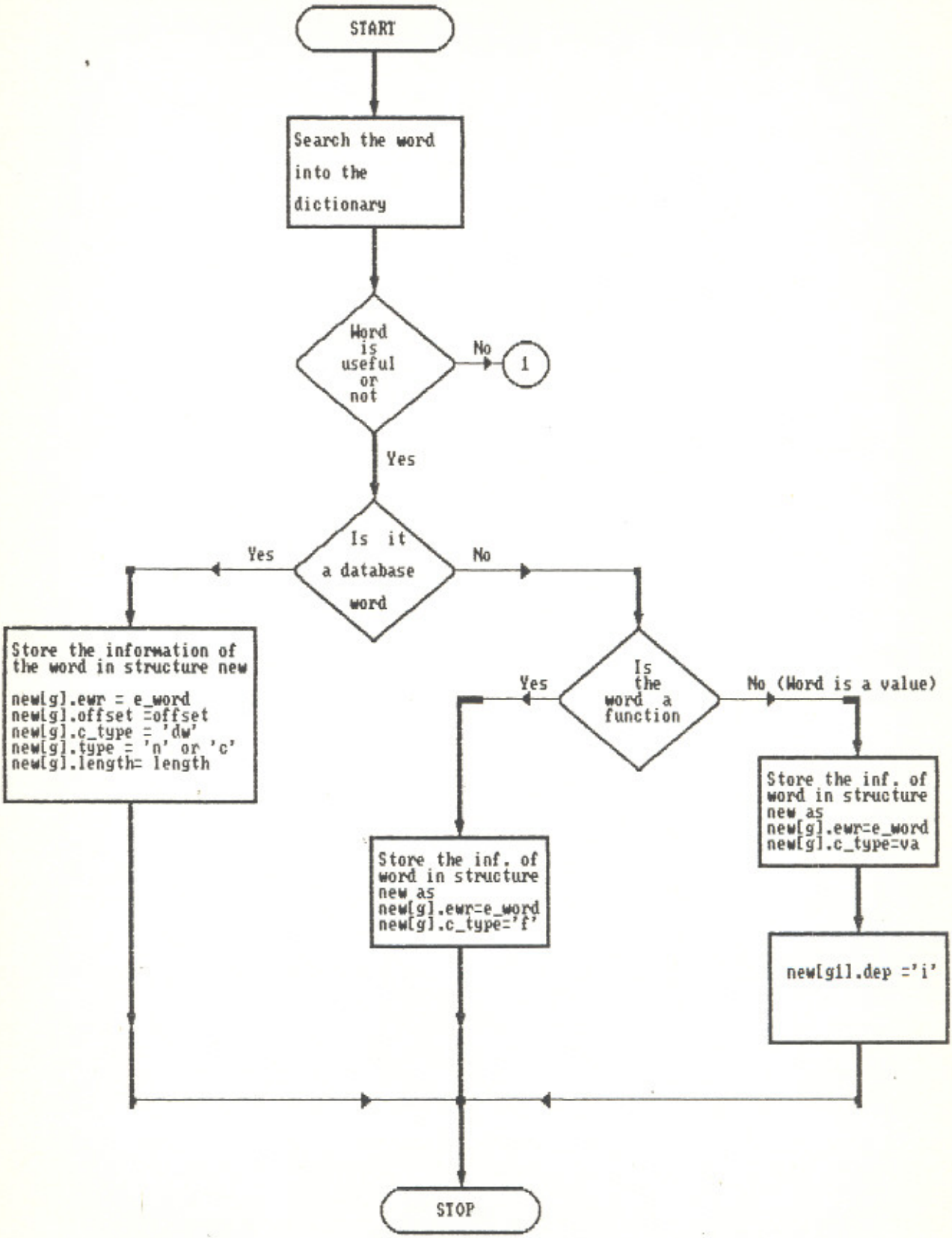

(CONTD.)


C
WANT TO
ASK MORE QUERIES
ON THIS
VILLAGE
1
NO
WANT TO
ASK MORE QUEY
ON OTHER
VILLAGE
YES
1
NO
STOP

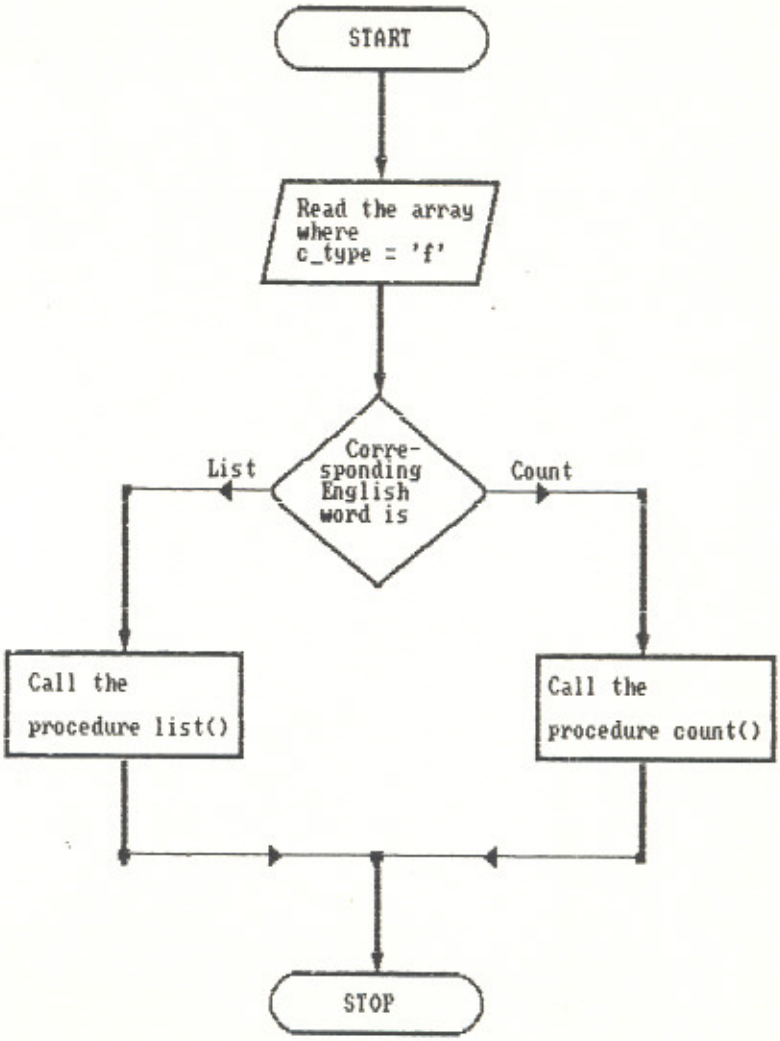
FLOW CHART FOR PROCEDURE append()



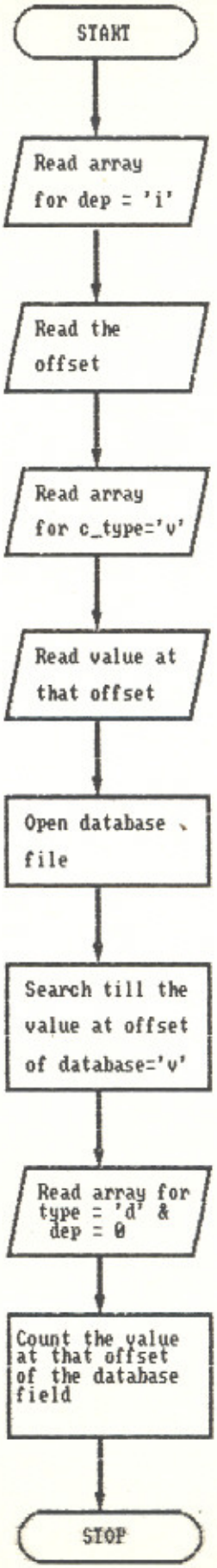
FLOW CHART FOR PROCEDURE pr()



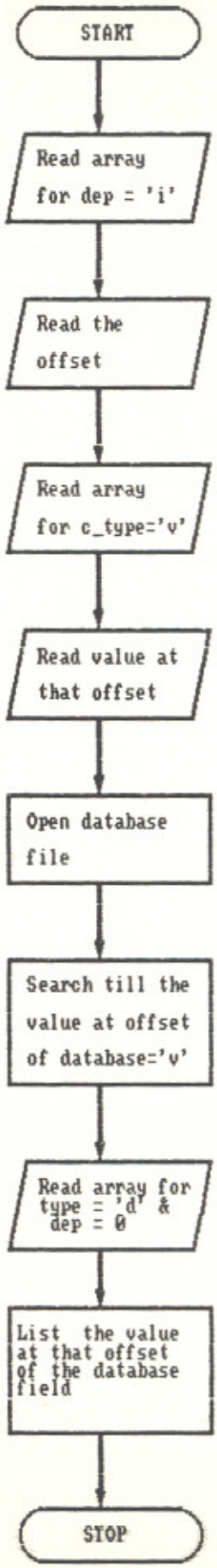
FLOW CHART FOR PROCEDURE count1()



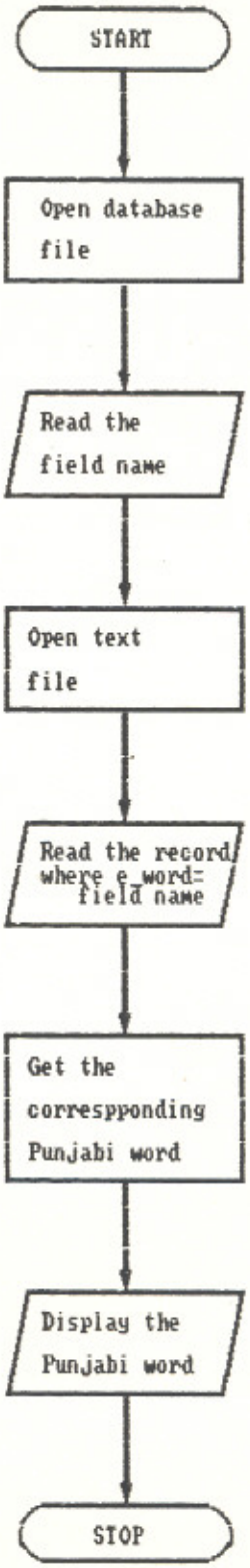
FLOW CHART FOR PROCEDURE count()



FLOW CHART FOR PROCEDURE list()



FLOW CHART FOR PROCEDURE printinf()



# TESTING

## DESCRIPTION OF PROCEDURES

PROCEDURE NAME : pr

PART OF : Natural Language Processing(NLP)

CALLED BY : query()

PROCEDURES CALLED : append()

PURPOSE

Retrieves information from the dictionary of the words. For the words the information is like, whether the word is a database word, or a function or the value. After retrieving the information about each WORD, it calls the procedure to append the information about the word in an array of structures, from where the information is retrieved later on for query purpose.

GLOBALS

str2: This value of this variable is taken from the main. It stores the word extracted from the query.

s[3]: This varible is of type array of characters of length 3. The information regarding the word whether it is useful or not gets stored into it.

ty[3]: This variable is of type array of characters of length 3. The information regarding the useful words, whether it is a database field,or function , or a value gets stored into it.

## DESCRIPTION OF PROCEDURES

PROCEDURE NAME : pr

PART OF : Natural Language Processing(NLP)

CALLED BY : query()

PROCEDURES CALLED : append()

PURPOSE

Retrieves information from the dictionary of the words. For the words the information is like, whether the word is a database word, or a function or the value. After retrieving the information about each WORD, it calls the procedure to append the information about the word in an array of structures, from where the information is retrieved later on for query purpose.

GLOBALS

str2: This value of this variable is taken from the main. It stores the word extracted from the query.

s[3]: This varible is of type array of characters of length 3. The information regarding the word whether it is useful or not gets stored into it.

ty[3]: This variable is of type array of characters of length 3. The information regarding the useful words, whether it is a database field,or function , or a value gets stored into it.

cty[3]: This variable is of type array of characters of length 3. This variable is used to store the type of the database field. It stores whether the variable is of type character, or numeric or date. Later on this information gets stored in the array of structures.

len[3] : This variable is of type array of characters of length 3. It stores the length of the field if the world corresponds to database field.

name : This field is of type structure here. This is used to read the information from the database file.

header: This variable is of type of structure head. This is used to read the information from the dictionary database.

LOCALS

ch[30] : This variable is used to store the Punjabi words when the dictionary is read sequentially .

PROCEDURE NAME : appe

PART OF : N.L.P

CALLED BY : pr()

PROCEDURE CALLED : NONE

PURPOSE

This procedure inserts in an array of structrures of type list, the information that the procedure pr() retrieves about the words that are useful for the purpose of the query. The constituents of the structure list are

1. char ewr[30]
2. unsigned char offset
3. char c_type
4. char type
5. char length

Since the internal processing is in English so in the field ewr of the array the corresponding English word of the Punjabi word is stored. If the word is a database word, then its offset calculated by the procedure pr() is stored in the field offset of the structure. In field c_type the information regarding the type of word, whether it is a database word, or a function, or a value is stored. In the field type the information 'C' is stored, if the word is a database word and is of type character, and the value 'n' is stored if the type is numeric, and the information 'd' is stored if the type is date.

GLOBALS ty[3],cty[3],s[3],len[3]

This procedure reads the values of these variables from the procedure pr().

new[g] This variable is of type array of structures and it is used to store the information regarding each word.

PROCEDURE NAME : coun

PART OF : N.L.P

CALLED BY : query()

PROCEDURE CALLED : NONE

PURPOSE

This procedure retreives the information that is asked in the query .It scans the array and stops at the element that corresponds to the function. If the function is count' then it calls the procedure count ,if the function is list then it calls the procedure list.

GLOBALS

new[g]: The values stored in this array are used to get the information regarding the words.

LOCALS

b : This variable is used to sequentially scan the array.

mn[13] : This variable is of type array of characters of

length 13. This is used to store the english word of the array of structures with c_type='f';

*ind : This variable is used to store the value of ewr of the element of array of structures.

PROCEDURE NAME : printinf()

PART OF : N.L.P

CALLED BY : main(),query();

PROCEDURE CALLED : NONE

PARAMETERS : NONE

PURPOSE

This procedure works first of all procedures when the user wants to get informatin from the database. This procedure shows the list of punjabi correspondents of all the fields that are there in the database.

GLOBALS

name: This variable is of type structure here.It is used to read the header of the database file.

field: This variable is of type structure fld ,it is used to store the information of fields sequentially.

header: This variable is used to store the information from the dictionary file.

fd: This variable is used as a handle of dictionary database file.

fg: This variable is used as a handle of dictionary database.

LOCALS

rs[12]: This variable of type array of characters of length 12. This is used to store the fieldnames while reading the database file's fields description sequentially.

ew[30]: This variable is of type array of characters of length 30. It is used to store the punjabi correspondings of field name while reading the dictioary database.

PROCEDURE NAME : append

PART OF : N.L.P

CALLED BY : main()

CALLING : NONE

PARAMETERS : NONE

PURPOSE

This procedure helps the user to append his own dictionary.The user is asked whether he wants to append field name or value. Then the punjabi words and the punjabi correspondings are got from the user. The information regarding the words that is needed later on by the user is attached itself by the system and is appended to the dictionary database.

GLOBALS

fg: This variable is of type integer and is used as handle to the dictionary database.

LOCALS

ck[47]: This variable is of type character of arrays, of length equal to the record length.

ci[30]: This variable is of type array of characters of length 30. It is used to get the Punjabi word from the user.

cr[30] :This variable is used to store the remaining blanks to the length of 30 to the array ci.

cl[12]: This variable is used to store the english word.

cn[2]: This variable is used to append "va" or "fn" to the word.

cs[2]: This variable is used to append "us" to the word.

PROCEDURE NAME : count()

PART OF : N.L.P

CALLED BY : coun()

CALLING : NONE

PURPOSE

This procedure is called by coun(). If the function in the query is count. It counts the values of that field for which the query is asked.If the field is of type character,it summs up 1 whenever record matching information is found. If the field is of type numeric it adds up the values.

GLOBALS

*ind This variable is used to take the value from the information stored in the array .

LOCALS

ch1[30] This variable is used to store the retreived information. cn This variable is used to store the summed up value.

PROCEDURE NAME : list()

PART OF : N.L.P

CALLED BY : coun()

CALLING : NONE

PURPOSE

This procedure is called by coun(). If the function in the query is count. It counts the values of that field for which the query is asked.If the field is of type character,it summs up 1 whenever record matching information is found. If the field is of type numeric it adds up the values.

GLOBALS

*ind This variable is used to take the value from the information stored in the array .

LOCALS

ch1[30] This variable is used to store the retreived information. cn This variable is used to store the summed up value.

PROCEDURE :query()

CALLED BY :main()

CALLING :printinf(),pr(),count()

PURPOSE

This procedure prints the fields of the datagets the query ,and gives the reply. GLOBALS :same as the globals in other procedures. st This variable is used to store each word of the query. LOCALS :str1[100] This variable is used to get the query. str2 this variable is used to store each word.

## TESTING

The designed system has been tested against the VILLAGE DATABASE. This database has been created in FOXBASE PLUS. This database contains all the information about a particular village. The fields on which a user can put in the query and the description of each field is being given below :

| S.NO. | FIELD NAME | DESCRIPTION |
|---|---|---|
| 1. | TEHSIL_COD | Tehsil Code |
| 2. | BLOCK_COD | Block Code |
| 3. | V_NAME | Name of Village |
| 4. | HD_BS_NO | Had Bast Number |
| 5. | PTVAR_AR | Patwar Halqa |
| 6. | KANUN_AR | Kanungo Halqa |
| 7. | POP_1971 | Population in 1971 |
| 8. | POP_1981 | Population in 1981 |
| 9. | FAMLY_NOS | Number of Families |
| 10. | SC_ST_NOS | Number of SC/ST's |
| 11. | WORKR_NOS | Number of Workers |
| 12. | LITRAT_NOS | Number of Literates |
| 13. | V_AREA | Total Area of Village |
| 14. | NET_SWON_A | Nett Sown Area |
| 15. | GR_SWON_A | Gross Sown Area |

| S.NO. | FIELD NAME | DESCRIPTION |
|---|---|---|
| 16. | NET_IRRI_A | Nett Irrigated Area |
| 17. | GR_IRRI_A | Gross Irrigated Area |
| 18. | THUR_SEM_A | Area under Thur/Sem |
| 19. | IR_WEL_NOS | Number of Irrigation Wells |
| 20. | EL_MOTOR | Number of Electric Motors |
| 21. | DL_ENGINE | Number of Diesel Engines |
| 22. | ADDL_BORES | Number of Additional Bores |
| 23. | EL_DL_BOTH | Tubewells Driven By Elect. & Diesel |
| 24. | TRACTORS | Number of Tractors |
| 25. | THRESHERS | Number of Threshers |
| 26. | HARV_COMBN | Number of Harvester Combines |
| 27. | SCH_PRIMAR | Number of Primary Schools |
| 28. | SCH_MIDDLE | Number of Middle Schools |
| 29. | SCH_MID_KM | Distance from Middle School in KMs |
| 30. | SCH_HIGH | Number of High Schools in village |
| 31. | SCH_HIG_KM | Distance from High/Sr.Sec School(KM) |
| 32. | ITI | I.T.I.(Y/N) |
| 33. | ITI_KM | Distance from I.T.I in KMs |
| 34. | P_TECH_KM | Distance from PolyTechnic in KMs |
| 35. | DEG_COL_KM | Distance from Degree College in KM |

| | | |
|---|---|---|
| 36. | GIRL_C_KM | Distance From Girls' College in KM |
| 37. | ADULT_ED_C | Adult Education Centre (Y/N) |
| 38. | DIS_ALO | Allopathic Dispensary (Y/N) |
| 39. | DIS_ALO_KM | Distance From Allopathic Dispensary |
| 40. | DIS_AUR | Ayurvedic Dispensary (Y/N) |
| 41. | DIS_ROM | Unani Dispensary (Y/N) |
| 42. | DIS_HOM | Homeopathic Dispensary (Y/N) |
| 43. | DIS_SUB | SubCentre (Y/N) |
| 44. | DIS_SUB_KM | Distance from Subcentre in KMs |
| 45. | HOS25BED | 25 Bed Hospital (Y/N) |
| 46. | HOS25BEDKM | Distance from 25 bed Hospital in KM |
| 47. | DENT_CL | Dental Clinic (Y/N) |
| 48. | HOS_OTHERS | Other Hospital (Y/N) |
| 49. | PHC | Primary Health Centre (Y/N) |
| 50. | PHC_KM | Distance from Pri. Health Centre |
| 51. | VET_HOSP | Veterinary Hospital (Y/N) |
| 52. | VET_HOSPKM | Distance from Veterinary Hospital |
| 53. | VET_CEM | Semen Centre (Y/N) |
| 54. | VET_CEM_KM | Distance from Semen Centre in KMs |

| S.NO. | FIELD NAME | DESCRIPTION |
|---|---|---|
| 55. | VET_DIS | Veterinary Dispensary (Y/N) |
| 56. | VET_DIS_KM | Distance from Veterinary Dispensary |
| 57. | MILK_C_CEN | Milk Collection Centre (Y/N) |
| 58. | COP_AGRI | No. of Co-Operative Societies(Agri.) |
| 59. | COP_OTHER | No. of Co-Op. Socities(non-agri.) |
| 60. | COPAGRI_MB | Members of Co-Op. Societies(Agri.) |
| 61. | COPOTHR_MB | Members of Other Co-Op.Soci(Non-agri) |
| 62. | MAIN_ROAD | Main Road |
| 63. | LINK_ROAD | Link Road (Y/N) |
| 64. | LINK_RD_KM | Length of Link Road |
| 65. | BUS_SRV | Bus Stand (Y/N) |
| 66. | BUS_SRV_KM | Distance from Bus Stand |
| 67. | RAIL_ST | Railway Station (Y/N) |
| 68. | RAIL_ST_KM | Distance from Railway Station |
| 69. | RAIL_S_NAM | Name of Nearest Railway Station |
| 70. | WATER_SPLY | Water Supply (Y/N) |
| 71. | TELEPHN | Telephone (Y/N) |
| 72. | TELEPHN_KM | Distance from Telephone in KMs |
| 73. | LIBRARY | Library (Y/N) |

| S.NO. | FIELD NAME | DESCRIPTION |
|---|---|---|
| 74. | THANA | Police Station/Chowki |
| 75. | THANA_KM | Distance from Police Station/Chowk |
| 76. | THANA_NAM | Name of Nearest Police Station |
| 77. | POSTOF | Post Office (Y/N) |
| 78. | POSTOF_KM | Distance from Post Office |
| 79. | POSTOF_NAM | Name of Nearest Post Office |
| 80. | PANCH_TV | Panchayati T.V.(Y/N) |
| 81. | PANCH | Panchayat (Y/N) |
| 82. | PANCH_SCST | No.of Panchayat Members (SC/ST) |
| 83. | PANCH_OTHR | No. of Panchayat Members (others) |
| 84. | J_GHAR_SC | Janj-Ghar for SC/ST |
| 85. | J_GHAR_OTH | Janj Ghar for others |
| 86. | BANK | No. of Banks |
| 87. | BANK_KM | Distance from Bank in KMs |
| 88. | BANK_V_NAM | Name of Nearest Vill.Where Bank Exists |
| 89. | MKT_CMTY | Market Committee (Y/N) |
| 90. | MKT_C_KM | Distance from Market Committee |
| 91. | FOCALPT | Focal Point (Y/N) |
| 92. | FOCALPT_KM | Distance from Focal POint |
| 93. | REST_H | Rest House (Y/N) |

| S.NO. | FIELD NAME | DESCRIPTION |
|---|---|---|
| 94. | CTRL_SP | Fair Price Shop (Y/N) |
| 95. | CTRL_SP_KM | Distance from Fair Price Shop |
| 96. | COLD_ST | Cold Store (Y/N) |
| 97. | COLD_ST_KM | Distance from Cold Store |
| 98. | GODOWN | Godown (Y/N) |
| 99. | GODOWN_KM | Distance from Godown |
| 100. | AGRO_SR | Agro Service Centre (Y/N) |
| 101. | AGRO_SR_KM | Distance from Agro Service Centre |
| 102. | IRDP_UP_88 | No.of IRDP Beneficiaries upto Mar,88 |
| 103. | IRDP_8889 | No.of IRDP Beneficiaries in 1988 |
| 104. | HOUS_UP_88 | Houses to Houseless upto Mar,1988 |
| 105. | HOUSE_8889 | Houses Given to Houseless in 1988-89 |
| 106. | Y_CARD_SC | Yellow Card Holders (SC) |
| 107. | Y_CARD_OTH | Yellow Card Holders (Others) |
| 108. | YC_SC_LOAN | Loan to Y.C.Holders (S.C.) |
| 109. | YC_OT_LOAN | Loan to Yellow Card Holders (Others) |
| 110. | COM_BIOGAS | No.of Community Biogas Plants |
| 111. | BIOGAS_PVT | No.of Private Biogas Plants |

The types of queries that a user can ask are given in the annexure 1

PROCESSING LOGIC FOR MAIN PROGRAM

(i) Choose Option

Want to

1. Append the dictionary
2. Ask the query
3. Have the general information about the database
4. Exit

(ii) Read the option

(iii) IF Option = 1

THEN Call the procedure append()

go to step (i)

ELSE IF Option = 2

THEN Call the procedure query()

go to step (i)

ELSE IF Option = 3

THEN Call the procedure printinf()

go to step (i)

ELSE Exit

ENDIF

ENDIF

ENDIF

PROCESSING LOGIC FOR PROCEDURE QUERY()

1. Open files dict.dbf and villdata.dbf
2. Prompt the user to ask the query
3. Read the query
4. Get the next word of the query.
5. Call the procedure pr().
6. IF the word which has been read is not the last word
   THEN go to step 4
   ELSE continue
   ENDIF
7. Call the procedure count1().
8. Write the result of the query.
9. IF user wants to ask more queries on this village
   THEN go to step 2
   ELSE go to step 10
   ENDIF
10. IF user wants to ask more queries on any other village
   THEN go to step 2
   ELSE go to step 11
   ENDIF

PROCESSING LOGIC FOR PROCEDURE pr()

1. Search the word from the dictionary.
2. IF the word is useful

THEN IF the word is a database word

THEN store the information of the word in structure new as

new[g].ewr = corresponding english word

new[g].offset = offset

new[g].c_type = 'd'

new[g].type = 'number' or 'character'

new[g].length = length

g = g+1

ELSE IF the word is a function

THEN store the information of the word in structure new as

new[g].ewr = corresponding english word

new[g].c_type = 'f'

g = g+1

ELSE (the word is a value)

store the information of the word in structure new as

```
                    new[g].ewr = corresponding english
                                 word
                 new[g].c_type = va
                  new[g-1].dep = 'i'
                             g = g+1
            ENDIF
        ENDIF
    ELSE go to step 4 of procedure QUERY()
    ENDIF
```

**PROCESSING LOGIC FOR PROCEDURE COUNT1()**

1. Read the array where c_type = 'f'
2. IF the English word corresponding to this c_type = list

   THEN call procedure list()

   ELSE call procedure count()

   ENDIF

PROCESSING LOGIC FOR PROCEDURE COUNT()

1. Read array for dep = 'i'
2. Read the offset
3. Read array for c_type = 'v'
4. Open the database file
5. Search till the value at offset of the database = 'v'
6. Read array for type = 'd' and dep = 0
7. Count the value at that offset of the database field

PROCESSING LOGIC FOR PROCEDURE LIST()

1. Read array for dep = 'i'
2. Read the offset
3. Read array for c_type = 'v'
4. Open the database file
5. Search till the value at offset of the database = 'v'
6. Read array for type = 'd' and dep = 0
7. List the value at that offset of the database field

PROCESSING LOGIC FOR PROCEDURE PRINTINF()

1. Open the database file
2. Read the field name
3. Open text file
4. Read the record where English word = field name
5. Get the corresponding Punjabi word
6. Display the Punjabi word

PROCESSING LOGIC FOR PROCEDURE APPEND()

1. Open text file dict.txt

2. Choose option

   (i) Append value

   (ii) Append field

3. IF option = (i)

   THEN read value in English

   get the corresponding Punjabi word

   concatenate ( 30 - strlen(Punjabi word)) blanks with Punjabi word

   concatenate 'va' with Punjabi word

   concatenate corresponding English word with the Punjabi word

   concatenate (12 - strlen(English word)) blanks with the concatenated word

   concatenate 'us' and '\n' with the word

   write concatenated word in the file

   go to step 2

   ELSE

   read value in English

   get the corresponding Punjabi word

   concatenate ( 30 - strlen(Punjabi word)) blanks with Punjabi word

   concatenate 'dw' with Punjabi word

   concatenate corresponding English word with the Punjabi word

   concatenate (12 - strlen(English word)) blanks with the concatenated word

```
            concatenate 'us' and '\n' with the word
            write concatenated word in the file

            go to step 2

ENDIF
```

# CONCLUSION & REMARKS

## ENHANCEMENTS

Our system study took a lot of time, so we could not devote much time on system design. Following are the some of the enhancements of our system that can be implemented by some modifications :

1.The limitation that was imposed on the system, that the value must follow the field name, can be removed with little modification.

2. The linear search has been implemented to get the information from the databases, thus slowing down the speed. The system can be made faster by implementing some other search techniques like quick sort and binary search.

3.The system can be made more user friendly if it gives the answers in the same way, the user had asked the question.

4.In the system the grammar of the sentence was ignored. The system can be made to recognize the grammatical formation of the sentence also. For this purpose the concepts of recursive transition network and augmented transition network can be introduced.

5. The system can not reply to the queries involving comparison. It can be made to reply this type of queries by little modifications.

# ANNEXURES

# ANNEXURE-1

1. ਪਿੰਡ ਆਲਮਗੀਰ ਵਿੱਚ ਕਿੰਨੇ ਚੌਦੇ ਹਨ।

2. ਪਿੰਡ ਆਲਮਗੀਰ ਵਿੱਚ ਚੌਦਿਆਂ ਦੀ ਗਿਣਤੀ ਦੱਸੋ।

3. ਇਸ ਪਿੰਡ ਦੀ ਆਈ ਟੀ ਆਈ ਤੋਂ ਦੂਰੀ ਦੱਸੋ।

4. ਕੀ ਪਿੰਡ ਕਨੇਚ ਵਿੱਚ ਆਈ ਟੀ ਆਈ ਹੈ।

5. ਅੰਹਮੀਲ ਚੋਡ 1 ਵਿੱਚ ਕਿਹੜੇ ਪਿੰਡ ਹਨ।

6. ਪਿੰਡ ਕਟਾਈ ਖੁਰਦ ਵਿੱਚ ਕਿੰਨੇ ਕਾਰੀਗਰ ਹਨ।

7. ਪਿੰਡ ਕਟਾਈ ਕਲਾਂ ਵਿੱਚ ਕਿੰਨੇ ਪਰਿਵਾਰ ਹਨ।

8. ਪਿੰਡ ਹਸਨਪੁਰ ਦਾ ਖੇਤਰਫਲ ਦੱਸੋ।

9. ਪਿੰਡ ਹਸਨਪੁਰ ਵਿੱਚ ਕਿੰਨੇ ਟਰੈਕਟਰ ਹਨ।

10. ਕੀ ਪਿੰਡ ਹੇਰਾਂ ਵਿੱਚ ਪ੍ਰਾਈਮਰੀ ਸਕੂਲ ਹੈ।

11. ਪਿੰਡ ਆਲਮਗੀਰ ਦੀ ਹਾਈ ਸਕੂਲ ਤੋਂ ਦੂਰੀ ਦੱਸੋ।

12. ਪਿੰਡ ਚੋਗਾੜਾ ਦੀ ਡਿਗਰੀ ਕਾਲਜ ਤੋਂ ਦੂਰੀ ਦੱਸੋ।

13. ਕੀ ਪਿੰਡ ਹੇਰਾਂ ਵਿੱਚ ਡਿਸਪੈਂਸਰੀ ਹੈ।

14. ਕੀ ਪਿੰਡ ਕਨੇਚ ਵਿੱਚ ਐੱਲੋਪੈਥਿਕ ਡਿਸਪੈਂਸਰੀ ਹੈ।

15. ਕੀ ਪਿੰਡ ਹਸਨਪੁਰ ਵਿੱਚ 25 ਬੈਡਵਾਲਾ ਹਸਪਤਾਲ ਹੈ।

16. ਪਿੰਡ ਹਸਨਪੁਰ ਦੀ ਹਸਪਤਾਲ ਤੋਂ ਦੂਰੀ ਦੱਸੋ।

17. ਕੀ ਤੁਸੀ ਮੈਨੂੰ ਪਿੰਡ ਚੋਗਾੜਾ ਦੀ ਹਸਪਤਾਲ ਤੋਂ ਦੂਰੀ ਦੱਸ ਸਕਦੇ ਹੋ।

18. ਪਿੰਡ ਚੋਗਾੜਾ ਤੋਂ ਖਾਨਦਰਾਂ ਦਾ ਹਸਪਤਾਲ ਕਿੰਨੀ ਦੂਰ ਹੈ।

19. ਕੀ ਪਿੰਡ ਹਸਨਪੁਰ ਵਿੱਚ ਕੋਆਪਰੇਟਿਵ ਸੋਸਾਇਟੀ ਹੈ।

20. ਕੀ ਪਿੰਡ ਹਸਨਪੁਰ ਵਿੱਚ ਧਰਮ-ਸਟੈਂਡ ਹੈ।

# BIBLIOGRAPHY

## BIBLIOGRAPHY


1. Artificial Intelligence by Rich and Knight

2. Artificial Intelligence by Herbert Schildt

3. Artificial Intelligence using Pascal by Herbert Schildt

4. Artificial Intelligence using 'C' by Herbert Schildt

5. Natural Language Procesing in Prolog by Gerald Gazdar and Chris Mellish

6. Computational Semantics by E.Charniak and Y.Wilks