# Chapter 2

# SYSTEM ANALYSIS

## 2.1 Analysis of the Project

Voter ID Management System is used in any Election Commission of India to store voter details. The system is initially used to add the details with all its specifications entered correctly into a file corresponding to its Voter ID, which is different for each and every voter. The system can be used to search, delete, modify or display existing records of all the voters.

## 2.2 Structure used to Store the Fields and Records

### a) Storing Fields

Fixing the Length of Fields:

In the Voter ID Management System, the Voter ID field is a character array that can hold an integer value of fixed size. Here other fields are character arrays, here the Voter ID is the primary key. By using primary key the entire records can be easily obtained. We preserve the identity of fields by separating them with delimiters. We have chosen the vertical bar character (|), as the delimiter here.

### b) Storing Records

Making Records a Predictable Number of Fields:

In this system, we have a variable number of fields, each with a maximum length, that combine to make a data record. Varying the number of fields in a record does not imply that the size of fields in the record is fixed. The records are used as containers to hold a mix of fixed and variable-length fields within a record.

Using an Index to Keep Track of Addresses:

We use secondary indexes to keep byte offsets for each record in the original file. Secondary indexes can built on any field of data file or on combination of fields. Secondary indexes will typically have multiple locations for a single key. Our choice of a record delimiter for the data files is the end-of-line (new-line) character (\n).

## 2.3 Operations Performed on a File

**a) INSERTION:** The system is initially used to add voter records containing the voter attributes like Voter ID, name of the voter, address, contact number and sex. Records with duplicate Voter ID fields are not allowed to be inserted. If we try insert a duplicate Voter ID it displays a message Voter ID is already present.

**b) DISPLAY:** The system can then be used to display existing records of all voters. The records are displayed based on which we have entered the voter details which is not in sorted order.

**c) SEARCH:** The system can then be used to search for existing records of all the voters. The user is prompted for a Voter ID, which is used as the key in searching for records in the simple index. If we select secondary indexing it asks for the voter name which is the secondary key in searching for the records in the secondary index files. The secondary key is searched to obtain the desired primary key of the voter, which is then used to seek to the desired data record in any of the voter files. The details of the requested record, if found, are displayed, with suitable headings on the user's screen. If absent, a "record not found" message is displayed to the user.

**d) DELETE:** The system can then be used to delete existing records from all voter details. The reference to a deleted record is removed from index file and $ is placed in data file for a particular record which is deleted. The requested record, if found, is marked for deletion, a "record deleted" message is displayed. If absent, a "record not found" message is displayed to the user.

**e) MODIFY:** If selected for modify option, the System will ask to enter the particular Voter ID which is to be modified and thereby after entering the Voter ID, the corresponding details of the voter are also displayed and the option to modify any of the field excluding the primary key which is Voter ID here is asked to enter. The user can give all related information and then press enter. And the updated or modified values will reflect back into the file.

## 2.4 Indexing Used

### B -TREE

A B -TREE of simple indexes on the primary key is used to provide direct access to data records. Each node in the B -TREE consists of a primary key with the reference to record.

The primary key field is the V_ID field while the reference field is the starting byte offset of the matching record in the data file. Each B-TREE node can have max of 2 child node.

The V_ID is stored in the B -TREE, and hence written to an index file. On retrieval the matched V_ID is used, before it is used to seek to a data record, as in the case of requests for a search, delete, modify operation. As records are added, nodes of the B-TREE undergo splitting (on detection of overflow), merging (on detection of underflow) or redistribution (to improve storage utilization), in order to maintain a balanced tree structure. The data files are entry-sequenced, that is, records occur in the order they are entered into the file. The contents of the index files are loaded into their respective B -TREEs, prior to use of the system, each time. Each B-TREE is updated as requests for the available operations are performed. Finally, the B -TREEs are written back to their index files, after every insert, delete and modify operation.

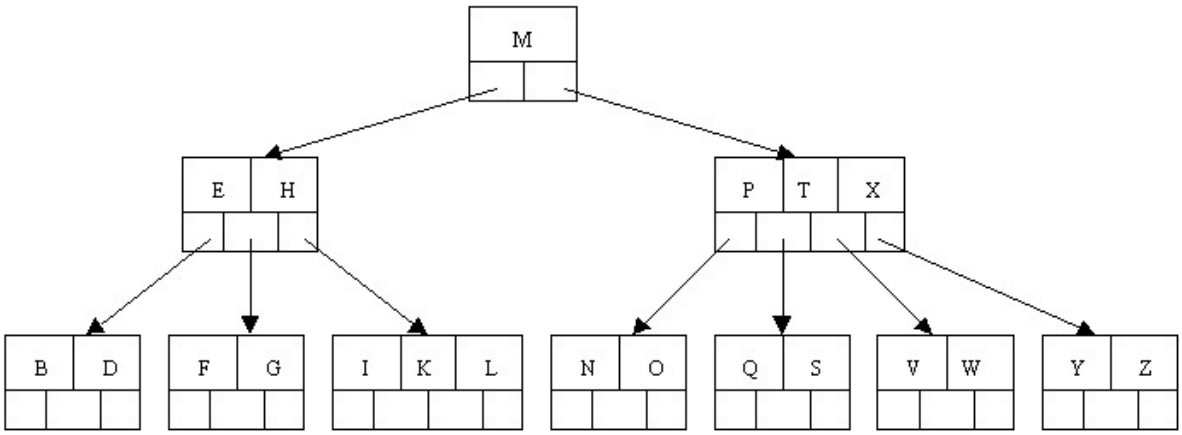


**Figure 2.1 A B-Tree Implementation**