**CHAPTER 1**

**INTRODUCTION**

* **HISTORY OF FILE STRUCTURES:**

Early in computing history, secondary storage was in the form of magnetic tape and punched cards. Storage was cheap but access was limited to sequential. In 1956, IBM introduced the RAMAC magnetic disk device. It could be leased for $620 month and could store approximately 5 Mb of data. Data could be accessed directly instead of sequentially. Conserving the space on the disk and getting to the data quickly became an area of research. This was the dawn of the study of file structures. A file structure is a combination of data representation on file and the operations for accessing this data. A file may be designed to store a picture, a written message, a video, a computer program, or a wide variety of other kinds of data. Some types of files can store several types of information at once. By using computer programs, a person can open, read, change, and close a file. Files may be reopened, modified, and copied an arbitrary number of times. Typically, files are organized in a file system, which keeps track of where the files are located on disk and enables user access. On most modern operating systems, files are organized into one-dimensional arrays of bytes. The format of a file is defined by its content since a file is solely a container for data, although, on some platforms the format is usually indicated by its filename extension, specifying the rules for how the bytes must be organized and interpreted meaningfully. In physical terms, most computer files are stored on some type of data storage device. For example, most operating systems store files on a hard disk.

* **Problem statement**: Develop a project for HASHING using BUCKETS as a collision resolution on a file of MUSIC objects. This project Implements insert ( ), search ( ), delete ( ) and display ( ).
* **Description of the technique:** Hashing is a technique to convert a range of key values into a range of indexes of an array.A **hash** table is a data structure which implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an index into an array of **buckets** or slots, from which the desired value can be found. Each data value has its own unique index value.
  1. **Requirements Analysis**

The file contains the details of the music albums that is its id, album name, album artists, album release year and album type. Each record should have a unique key value. The user can add new items into the file and these items should be stored in the file with respect to the unique key value by making use of the hashing technique. The user can search and delete an item by entering the primary key value. The user can also view the list of items that are present in the file.

**CHAPTER 2**

**DESIGN**

**2.1 DATA FLOW DIAGRAM:**

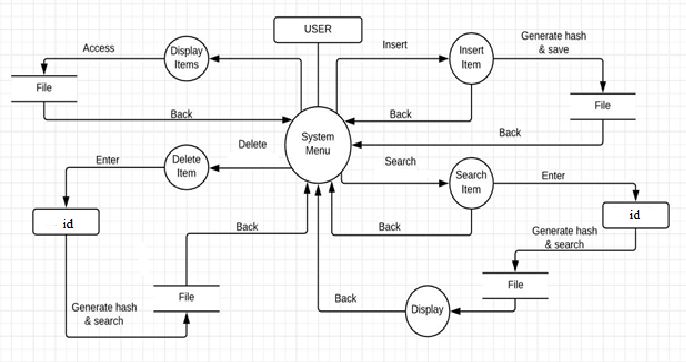
A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.

The Music Records Management system Data Flow Diagram example contains four processes, one external entity and one data store.

Based on the diagram, we know that a User can add an entry into the file. The Insert item process receives the data, generates the hash , store it in the file. System Menu can search Records through the Search Item process, which takes the hash of the ID, searches the file at that position and displays the record if present.

System Menu can also delete a record through the process Delete Item by providing the ID. The process generates the hash for the ID, searches the file for the specified model number, and reclaims the space by deleting the record.

The Display process fetches all data from the file and gives it to the System Menu.

****

**CHAPTER 3**

**SYSTEM REQUIREMENTS**

* 1. **SOFTWARE REQUIRMENTS:**

Operating System : Windows 7 and above.

JDK : Java Development Kit v 8.1

Integrated Development Kit Environment: NetBeans IDE 8.2

Text editor : Notepad ++

* 1. **HARDWARE REQUIRMENTS:**

Processor : Intel i5 6500 processor

Technology : NetBeans.

RAM : 4GB

Hard Disk : Minimum 1GB

* 1. **TOOLS: \\ONLY NET BEANS AND JDK**

**NETBEANS**

NetBeans is an integrated development environment (IDE) for Java. NetBeans allows

applications to be developed from a set of modular software components called modules. NetBeans runs on Microsoft Windows, macOS, Linux and Solaris. In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML, Javadoc, and Javascript. Applications based on NetBeans, including the NetBeans IDE, can be extended by third party developers. The NetBeans Team actively supports the product and seeks feature suggestions from the wider community. Every release is preceded by a time for Community testing and feedback.

The NetBeans Platform is a framework for simplifying the development of Java

Swing desktop applications. The NetBeans IDE bundle for Java SE contains what is

needed to start developing NetBeans plugins and NetBeans Platform based applications; no additional SDK is required. Applications can install modules dynamically. Any application can include the Update Center module to allow users of the application to download digitally signed upgrades and new features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again.

The platform offers reusable services common to desktop applications, allowing

developers to focus on the logic specific to their application. Among the features of the platform are:

* User interface management (e.g. menus and toolbars)
* User settings management
* Storage management (saving and loading any kind of data)
* Window management
* Wizard framework (supports step-by- step dialogs)
* NetBeans Visual Library
* Integrated development tools

These modules are part of the NetBeans IDE.

**CHAPTER 4**

**IMPLEMETATION**

**4.1 LIST OF MODULES USED:-**

The main modules/functionalities that has been used in this project is as follows:-

* INSERT
* SEARCH
* DELETE
* DISPLAY

**INSERT**: This module basically consists of two very important operations namely write and pack functions. The Write function basically takes in all the data from the user under the respective attributes such as Musician’s ID, Name, etc.. The pack function is mainly used to pack all the data together into a file in a particular format by using newlines and delimiters. (no bullets directly start with one tab space)

**SEARCH**: This module is mainly used for searching all the data related to a specific musician. Take in the input of MID from the user, and the information corresponding to this MID(if present) will be displayed. If it is not present the operation fails.

**DELETE**: This module is mainly used for deleting the data related to a specific musician. Search for the required MID that has to be deleted, and the information corresponding to this MID (if present) will be deleted. If it is not present the operation fails.

**DISPLAY**: Here all the records related to the musicians is displayed.

**CHAPTER 5**

**SNAPSHOTS.**

**CHAPTER 6**

**CONCLUSION.**

In this project, we have implemented hashing using buckets on a music object. Both front end and back end is developed using netbeans ide. Front end uses Jframes and back end is a java code.

It takes different attributes like ID, Album Name, Artists Name, Music Director and Release Year and inserts the record into the file. Searches and deletes records in the file and also displays the content if the file.

ADV

* Main advantage is synchronization.
* In many situations, hash tables turn out to be more efficient than search trees or any other table lookup structure. For this reason, they are widely used in many kinds of computer softwares, particularly for associative arrays, database indexing, caches and sets.

DISADV

* Fixed number of buckets M is a problem if the number of records in the file grows or shrinks.
* Ordered access on the hash key is inefficient (requires sorting the records).

APPLICATION

* Hashing is mainly used in Networking Problems
* Interactive multimedia (games, videoconferencing)
* Trading platforms
* High-performance systems/data centers

**CHAPTER 7**

**REFERENCES**

1. Michael J. Folk, Bill Zoellick, Greg Riccardi: File Structures-An Object Oriented Approach with C++, 3rd Edition, Pearson Education, 1998.
2. Java- The complete reference, Herbert Schildt, seventh edition.