INTRODUCTION TO THE DESIGN AND SPECIFICATION OF THE FILE STRUCTURES

1.1 The Heart of the file structure Design

File : A data structure on secondary storage which acts as a non-volatile container for data. File is a name given to any kind of document stored in any type of storage device which can be read by the computer. A file is identified by a name followed by a filename extension.

File Structure : A pattern for arranging data in a file. It is a combination of representations for data in files and of operations for accessing the data.

Primary Goals for Design of File Structures and Algorithms

1) Minimize the number of disk accesses.

- If possible, transfer all information needed in one access.
- Group related information physically so it can be accessed together.

2) Maximize the space utilization

- Use compression techniques wherever possible
- Apply defragmentation procedures
- Avoid data redundancy

Problems and Concerns

- File data is frequently dynamic - that is, it changes from time to time.
- Designing file structures for changes adds complexity.
- Typical file sizes are growing.
- Solutions which work for small files may be inadequate for large files.
- File structures, algorithms, and data structures must work together.

1.2 A Short History of File Structure Design

- Earlier, the file access was sequential, and the cost of access grew in direct proportion to the size of the file. So, Indexes were added to files.
- Indexes made it possible to keep a list of keys and pointers in a smaller file that could be searched more quickly.
- Simple indexes became difficult to manage for dynamic files in which the set of keys changes. Hence tree structures were introduced.
- Trees grew unevenly as records were added and deleted, resulting in long searches requiring multiple disk accesses to find a record. Hence an elegant, self-adjusting binary tree structure called an AVL tree was developed for data in memory.

- Even with a balanced binary tree, dozens of accesses were required to find a record in moderate-sized files.
- A method was needed to keep a tree balanced when each node of the tree was not a single record, as in a binary tree, but a file block containing hundreds of records. Hence, B-Trees were introduced.
- AVL trees grow from top down as records are added, B-Trees grow from the bottom up.
- B-Trees provided excellent access performance but, a file could not be accessed sequentially with efficiency.
- The above problem was solved using B+ tree which is a combination of a B-Tree and a sequential linked list added at the bottom level of the B-Tree.
- To further reduce the number of disk accesses, hashing was introduced for files that do not change size greatly over time.
- Extendible, dynamic hashing was introduced for volatile, dynamic files which change.

2. Fundamental File Processing Operations

2.1 Physical Files and Logical Files

Physical file

A file as seen by the operating system, and which actually exists on secondary storage.

Logical file

A file as seen by a program.

- Programs read and write data from logical files.
- Before a logical file can be used, it must be associated with a physical file.
- This act of connection is called "opening" the file.
- Data in a physical file is persistent.
- Data in a logical file is temporary.
- A logical file is identified (within the program) by a program variable or constant.
- The name and form of the physical file are dependent on the operating system, not on the programming language.

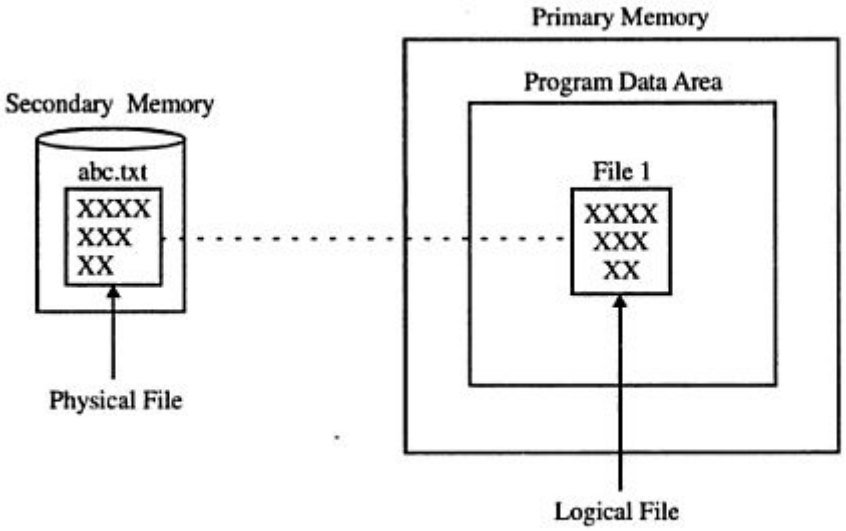

Figure 1.1 Relationship between Physical File and Logical File

Source : http://elearningatria.files.wordpress.com/2013/10/ise-vi-file-structures-10is63-notes.pdf