Hashing And File Structure

Unit#5

Marwadi University

Department of Computer Engineering

Data Structure
01CE0301 / 3130702

Ravikumar Natarajan

Highlights

- Concepts of fields, records and files,
- Sequential, Indexed and Relative/Random File Organization,
- Indexing structure for index files

Introduction

- Most Applications collect huge amount of Data, [Bank Accounts, College Admissions, Scientific Experiments etc.]
- Files stored on computer are a good alternative to store.
- Primary Storage vs. Secondary Storage.
- Less Time and Less Space vs. More Time and More Space.

File???

- A file is a collection of data stored on mass storage (eg disk or tape)
- The data is subdivided into records (e.g student information)
- Each record contains number of fields (e.g roll number, name)
- One or more fields considered as the key field. (e.g roll number). Key is an attribute that uniquely identifies the records of a file. It contains unique values to which can be used to distinguish one record from another in a file.
- Page: A file is loaded in the main memory to perform operations like insertion, modification, deletion, etc., on it. If the file is too large in size, it is decomposed into equal size pages, which is the unit of exchange between the disk and the main memory.

Organization of Records in File

- Fixed-Length Records
- All the records in a file of fixed-length record are of same length. In a file of fixed-length records, every record consists of same number of fields and size of each field is fixed for every record. It ensures easy location of field values, as their positions are predetermined.
- Since each record occupies equal memory, as shown in Figure 9.1, identifying start and end of record is relatively simple.
- A major drawback of fixed-length records is that a lot of memory space is wasted.

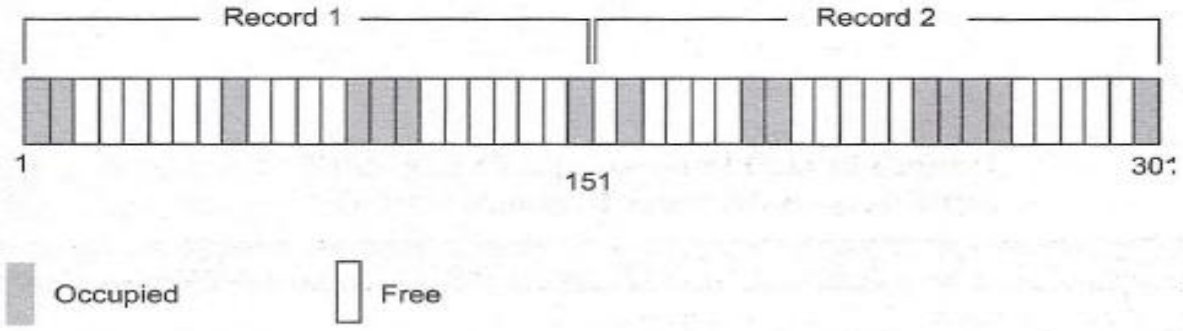


Figure 9.1 Location of Fixed-Length Records

Organization of Records in File

- Variable-Length Records
- Variable-length records may be used to utilize memory more efficiently. In this approach, the exact length offield is not fixed in advance. Thus, to determine the start and end of each field within the record, special separator characters, which do not appear anywhere within the field value, are required (see Figure 9.3). Locating any field within the record requires scan of record until the field is found.

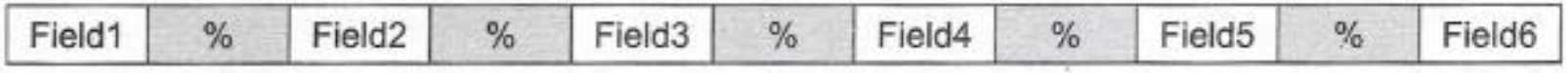


Figure 9.3 Organization of Variable-Length Records with '%' Delimiter

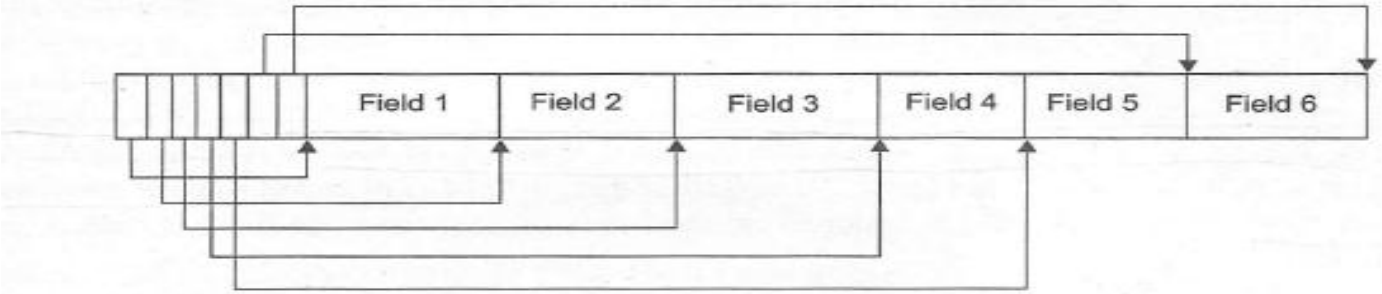


Figure 9.4 Variable-Length Record Organization Using an Array of Field Offsets

File structure and Data structure

- A data structure could be present both in RAM and DISK.
- Technically the file structures are more standardized, especially if one wants to allow a third party app or a newer version of the same app to open the file.
- Data structures can change between app versions and there is no need for compatibility.
- In most (binary) file structures, the basic organizing elements are Pages and Physical Links.
- Page (computer memory) map to file system chunks that are (hopefully) guaranteed to be stored contiguously and can be read and written in a single operation.

Data Hierarchy

- Every file contains data which can be organized in hierarchy for systematic organization.
  - Data Field
  - Record
  - File
  - Directory

File Attributes

- Every file is stored in directory in computer system.
- Each file is having list of attributes associated with it.
- They state how to use file to any software program looks up the directory. Just like hidden files are not displayed in DOS when we apply DIR command.

- File name – string of characters vary by OS
- File position – position where the next read/write operation can be performed
- File structure – whether text file or binary file
- File access method – how to access records, sequentially or randomly

File Attributes

- Attributes Flag – a single byte is used to specify 6 flags attached to a file

Attribute	Byte
Read-Only	0000 0001
Hidden	0000 0010
System	0000 0100
Volume Label	0000 1000
Directory	0001 0000
Archive	0010 0000

File Attributes

- Read-only – can't get deleted or modified. Get message 'access denied' if you do so.
- Hidden – Not displayed in directory listing.
- System – Important as used by system and must not be altered or removed. More than Read-only.
- Volume label – Every disk volume is assigned for identification, mainly at the time of formatting.
- Directory – Extra bit to differentiate directory from file. In directory listing it is used to specify sub directories. Theoretically by changing this bit, we can convert a file to directory but practically results in failure.
- Archive – Backup program uses it by setting 1 to 0 and 0 to 1 to have the latest copy.

Text files vs. Binary files

- Basically every file is just a series of bytes one after the other.
- In general every file is a binary file, but if the data in it contains only text like letter, numbers and other symbols one would use in writing, and if it consists of lines, then we consider it a text file.

Text files vs. Binary files

- A binary file is basically any file that is not "line-oriented".
- Any file where besides the actual written characters and newlines there are other symbols as well.
- A Microsoft word file is a binary file as besides the actual text, it also contains various characters representing font size and color.
- Program written in the C programming language is a text file, but after you compiled it, the compiled version is binary.

File Organization

- File is a collection of related records so main issue in file management is the way records are organized inside the file.

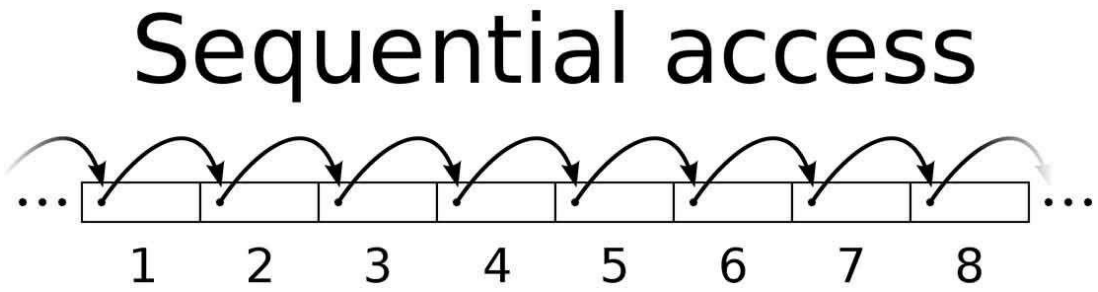
- How to select file organization method:
  - Rapid access to one or more records
  - Ease of insert/update/delete one or more records w/o affecting speed of accessing records
  - Efficient storage of records

File Organization

- Sequential Organization
- Random/Direct/Relative File Organization
- Indexed Sequential File Organization
- Multi-key File Organization and Access Methods

Sequential Organization

- In a sequential file, we place records in a block one after another until there is no room for another complete record in the block.
- If we wish to access all the records, one after another, this is a highly efficient file organization.
- Suppose we only wish to find one record. We may have to search the entire file, reading one block at a time to find the record.



Sequential Organization

- Advantages
  - Simple to program and easy to design
  - Best in terms of storage space
- Disadvantages
  - Not possible to add record in middle of file without rewriting the file
  - Time consuming because random search not possible
  - High data redundancy

Relative File Organization

- In contrast to SEQ files, records of a RELATIVE file can be accessed by specifying the record sequence number/relative key and without needing to read all the previous records.
- All the records are stored in Direct Access Storage Device (DASD) such as hard disk. The records are randomly placed.
- Records need not be placed in sequence. Direct update and rewrite in same location possible.
- Record with sequence number 16 is located just after the 15th record.
- Range from 0 to n-1.
- This is also called as hashing.

Relative File Organization

- Address of ith record =
  base_add + (i-1)*record_length
- Address of 5th record = 1000+(4)*20 = 1080
- Possible only if records are of equal lengths.

Relative Record Number	Records in Memory
0	Record 0
1	Record 1
2	Free
3	Free
…….	…….
98	Free
99	Record 99

Relative File Organization

- This can be used as Relative as well as sequential
- Random access, ease of processing, low overhead, add/delete based on relative key
- Only disk devices uses this, limited to fixed length records, relative record number must be known in advance to access any record

Relative File Organization

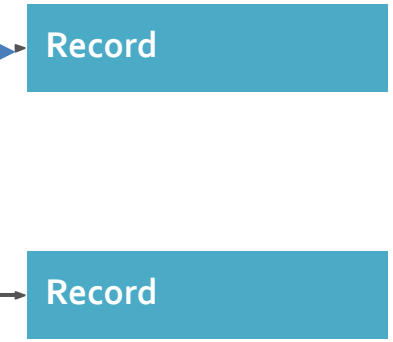
- Advantages
  - Direct access file helps in Online Transaction Processing System (OLTP) like railway reservation system.
  - Sorting not required
  - Updates files quickly.
- Disadvantages
  - No backup facility
  - Expensive
  - Less storage space then sequential file.

Indexed Sequential File Organization

- Combines both sequential & direct file organization.
- Records stored randomly in disk using primary key.
- Provides fast data retrieval
- Mostly records are of fixed length
- Index table stores address of each record
- Sequential and random access is possible
- Index table is read sequentially to find the address of the desired record
- The index is stored in a file and read into memory when the file is opened.

Indexed Sequential File Organization

Record Number	Address of Record
1	1269
2	Null
……..	……..
99	6351



Indexed Sequential File Organization

Book_Id	Book_title	Category	Page_count	
354	Ransack	Novel	200	Record 1
556	Differential Calculus	Textbook	450	Record 2
489	C++	Textbook	800	Record 3
678	Call Away	Novel	200	Record 4
456	Introduction to German Language	Language Book	200	Record 5
887	Learning French Language	Language Book	500	Record 6

Figure 9.6 A Sample of Books File

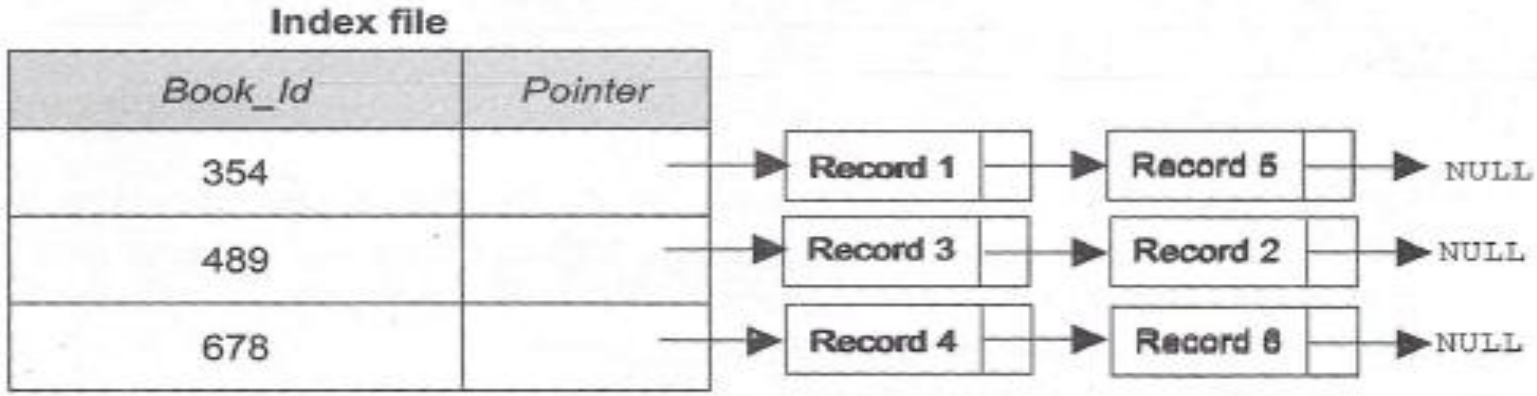


Figure 9.7 Indexed Sequential File Organization

Indexed Sequential File Organization

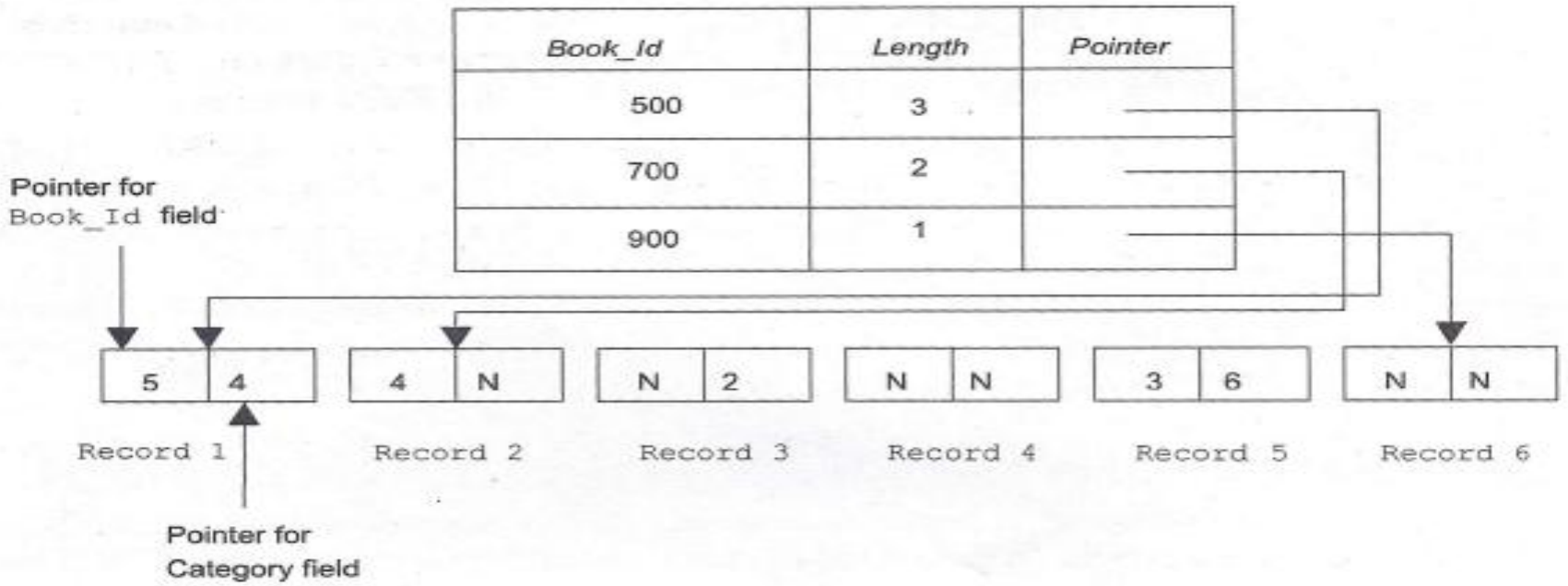
- Advantages
  - Indexes are small and searched quickly
  - Reduces degree of sequential search
- Disadvantages
  - Requires unique keys
  - Extra storage space required to store index
  - Expensive and special software required

Multi-key File Organization and Access Methods

- So far we have discussed the file organization methods that allow -records to be accessed based on a single key. There might be a situation where it is desirable or even necessary to access the records on anyone of the number of keys.
- For example, consider Book file shown in Figure 9.6. Different users may need to access the records of this file in different way. Some users may need accessing the record based on the field Book_Id, others may need accessing the record based on the field Category.
- To implement such searches, the idea of indexing can be generalized and a similar index may be defined on any field of resulting in a multi-key file organization.
- There are two main techniques used to implement multi-key file organization, namely, *multi-lists and inverted-lists*.

Multi-list File Organization and Access Methods

- So far we have discussed the file organization methods that allow -records to be accessed based on a single key. There might be a situation where it is desirable or even necessary to access the records on anyone of the number of keys.
- For example, consider Book file shown in Figure 9.6. Different users may need to access the records of this file in different way. Some users may need accessing the record based on the field Book _Id, others may need accessing the record based on the field Category.
- To implement such searches, .the idea of indexing can be generalized and a similar index may be defined on any field of resulting in a multi-key file organization.
- There are two main techniques used to implement multi-key file organization, namely, *multi-lists and inverted-lists*.



Category	Length	Pointer
Language Book	2	Rec_5
Novel	2	Rec_1
Textbook	2	Rec_3

Figure 9.8 A Multi list on `Book_Id` and `Category`

Inverted File Organization and Access Methods

- Like multi-lists structure, inverted list structures can also maintain multiple indexes on the file.
- The only difference is that instead of maintaining pointers in each record as in multi-lists, indexes in the inverted file maintain multiple pointers to point to the records.
- Indexes on Book_ Id and Category field for inverted file are shown in Figure 9.9.

Category	*Pointer*
Language Book	Rec_5, Rec_6
Novel	Rec_1, Rec_4
Textbook	Rec_2, Rec_3

Book_Id	*Pointer*
500	Rec_1, Rec_3, Rec_5
700	Rec_2, Rec_4
900	Rec_6

Figure 9.9 An Index on `Book_Id` and `Category` for Inverted File

Indexing

- Index to file is like a catalogue to library
- Indexed sequential file organization is efficient to use but in practical where a file may have millions of records, method fails
- Based on access time, access type, insertion time, deletion time, we have two categories,
    - Ordered Indices
    - Hash Indices

Ordered Indices

- Indexes are used to provide fast and random access to records
- Primary Index – Index whose search key specifies the sequential order of file.
- Example, STUDENT file has sequential order starting from roll no. 1 to 70. So roll number is primary index. Call roll no 20 and you get record.

Ordered Indices

- Secondary Index, while primary index provides sequential order only. Here the search key which provides random access or different from sequential order will work as secondary index.
- Example, in STUDENT file use student name as index which never provides sequential order, as they are sorted roll number wise.

Hash Indices

- Based on Hashing Techniques

Up Next

- Hashing Concepts and methods.
- Hash Table Methods - Introduction, Hashing Functions.
- Collusion and its understanding.
- Discuss different Collision-Resolution Techniques with examples.

Thank You.