Chapter 8

File Organization & Indexing

Database Management System (DBMS)

4th Semester , BEIT & BCE

Compiled by :

Ankit Bhattarai, Assistant Professor
ankitbhattarai@cosmoscollege.edu.np

Cosmos College of Management & Technology

Syllabus

Full marks: 100

Internal: 50
Final: 50

Internal Marks:

Theory: 30
Practical: 20

Unit	Contents	Hours	Remarks
1.	Introduction	4	
2.	Data Models	4	
3.	Relational Model	4	
4.	Relational Database Query Languages	8	Important
5.	Database Constraints and Relational Database Design	8	Important
6.	Security	3	
7.	Query Processing	3	
8.	File Organization & Indexing	4	
9.	Crash Recovery	3	
10.	Transaction Processing & Concurrency Control	4	
11.	Advanced Database Concepts	3	

Chapter 8

Main Topics

(4 hours)

- Memory Hierarchy (Storage Device)
- File Organization
  - ✓ Heap file Organization
  - ✓ Sequential File Organization
  - ✓ Hash File Organization
- Indexing & Hashing
- B+ Tree

File System Introduction

- A collection of data or information that has a name, called the ***filename***.
- In computer system every record has a filename associated with it.
- The basic problem in the **physical database representation** is to select suitable file system to store the desired information.
- The basic file operations that can be performed on file are:

✓ Creation of file

✓ Reading a file

✓ Update of a file

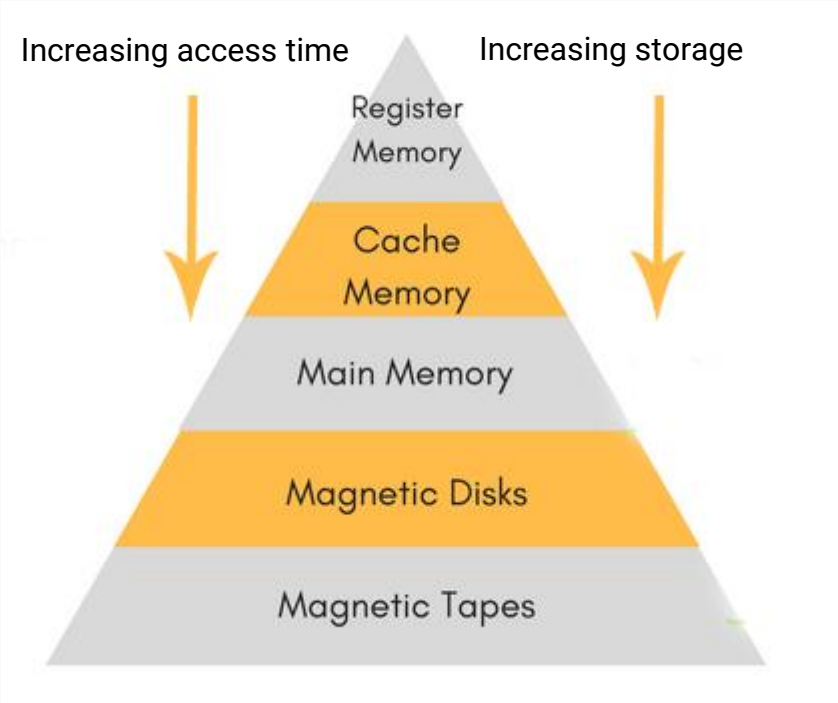
✓ Insertion in a file

✓ Deletion of a file

Memory Hierarchy

- In computer architecture, the memory hierarchy separates computer storage into a hierarchy based on response time
- As we go down the hierarchy, the following occurs.

1. **Decreasing cost per bit**
2. **Increasing capacity**
3. **Increasing access time**
4. **Decreasing frequency of access of the memory by the processor.**



Memory Hierarchy

- Computer memory can be classified in the below given hierarchy:

1. **Internal register**:

- Internal register in a CPU is used for holding variables and temporary results.
- Internal registers have a very small storage; however they can be accessed instantly.
- Accessing data from the internal register is the fastest way to access memory.

Memory Hierarchy

2. Cache:

- Cache is used by the CPU for memory which is being accessed over and over again.
- Instead of pulling it every time from the main memory, it is put in cache for fast access. It is also a smaller memory, however, larger than internal register.

Cache is further classified to L1, L2 and L3:

a. L1 cache: It is accessed without any delay.

b. L2 cache: It takes more clock cycles to access than L1 cache.

c. L3 cache: It takes more clock cycles to access than L2 cache.

Memory Hierarchy

3. Main memory or RAM (Random Access Memory):

- It is a type of the computer memory and is a hardware component.
- It can be increased provided the operating system can handle it.
- It is accessed slowly as compared to cache. They are volatile i.e. content of memory are usually loss in case of power failure or crash.

4. Hard disk:

- A hard disk is a hardware component in a computer.
- Data is kept permanently in this memory. Memory from hard disk is not directly accessed by the CPU, hence it is slower.
- As compared with RAM, hard disk is cheaper per bit. Typically, the entire database is stored on disk. Data must be moved from disk to main memory in order for data to be processed.

Memory Hierarchy

5. **Magnetic tape:**

- Magnetic tape memory is usually used for backing up large data. When the system needs to access a tape,
- it is first mounted to access the data. When the data is accessed, it is then unmounted. The memory
- access time is slower in magnetic tape and it usually takes few minutes to access a tape.

File Organization

- It is a method of arranging the records in a file when the file is stored on disk.

  A relation is typically stored as a file of records. Mainly two type of records:

  1. Fixed Length Records
  2. Variable Length Records

- **Fixed Length Records**

✓ each field is of fixed length

✓ In these type of records, the record number can be used to locate a specific record

✓ Number of records, the length of each field is available in file header

File Organization

- **Variable Length record type:**

✓ arise due to missing fields, repeating fields, variable length fields

✓ special separator symbols are used to indicate the field boundaries and record boundaries

✓ the number of records, the separator symbols used are recorded in the file header

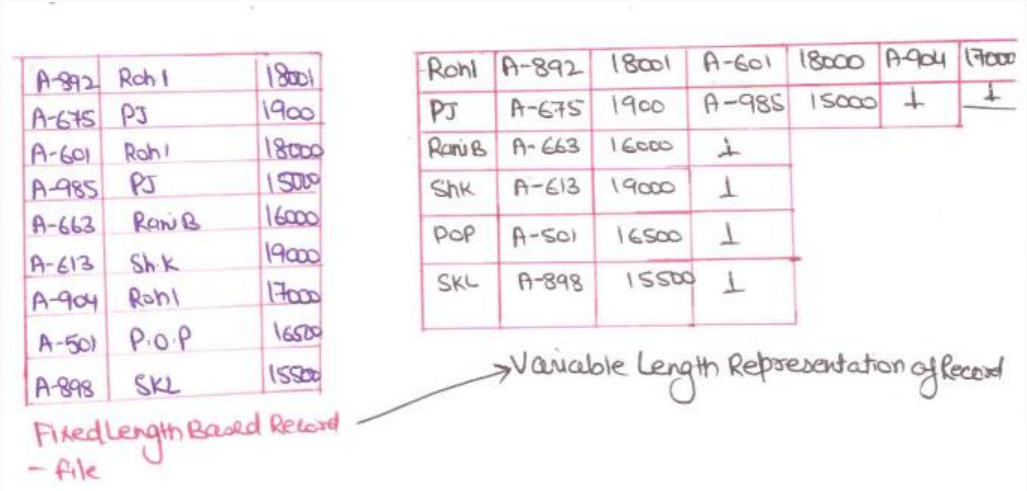


Fig. Type of records in File

Organization of Records in a File

- So far, we have studied how records are represented in a file structure. A relation is a set of records. Given a set of records; the next question is how to organize them in a file. Several of the possible ways of organizing records in files are:

1. Heap File Organization: Any record can be placed anywhere in the file where there is space for a record. There is no ordering of records. Typically, there is a single file for each relation.

2. Hashing File Organization: A hash function is computed on some other attribute of each record. The result of the hash function specifies in which block of the file the record should be placed.

3. Sequential File Organization: Records are stored in sequential order, according to the value of a “search key” of each record.

Organization of Records in a File

- So far, we have studied how records are represented in a file structure. A relation is a set of records. Given a set of records; the next question is how to organize them in a file. Several of the possible ways of organizing records in files are:

1. Heap File Organization: Any record can be placed anywhere in the file where there is space for a record. There is no ordering of records. Typically, there is a single file for each relation.

2. Hashing File Organization: A hash function is computed on some other attribute of each record. The result of the hash function specifies in which block of the file the record should be placed.

3. Sequential File Organization: Records are stored in sequential order, according to the value of a “search key” of each record.

Organization of Records in a File

- So far, we have studied how records are represented in a file structure. A relation is a set of records. Given a set of records; the next question is how to organize them in a file. Several of the possible ways of organizing records in files are:

1. Heap File Organization: Any record can be placed anywhere in the file where there is space for a record. There is no ordering of records. Typically, there is a single file for each relation.

2. Hashing File Organization: A hash function is computed on some other attribute of each record. The result of the hash function specifies in which block of the file the record should be placed.

3. Sequential File Organization: Records are stored in sequential order, according to the value of a “search key” of each record.

Sequential File Organization

- Designed for efficient processing of records in sorted order based on search key.
- Suitable for applications that require sequential processing of entire file.
- The records in the file are ordered by a search key. A search key is any attributes or set of attributes.
- For fast retrieval of records in search key order: records are linked together by pointer.

A-217	Brighton	750	
A-101	Downtown	500	
A-110	Downtown	600	
A-215	Mianus	700	
A-102	Perryridge	400	
A-201	Perryridge	900	
A-218	Perryridge	700	
A-222	Redwood	700	
A-305	Round Hill	350	

Indexing and Hashing

- An index or database index is a data structure which is used to quickly locate and access the data in a database table.
- Indexing is a way to optimize performance of a database by minimizing the number of disk accesses required when a query is processed.
- Indexing is defined based on its indexing attributes. Indexing are of following types:

✓ **Primary Index** – Primary index is defined on an ordered data file. The data file is ordered on a key field. The key field is generally the primary key of the relation.

✓ **Secondary Index** – Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.

✓ **Clustering Index** – Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.

Indexing and Hashing

- Primary index is of two types:

  Dense Index

✓ For every search key value in the data file, there is an index record.

✓ This record contains the search key and also a reference to the first data record with that search key value.

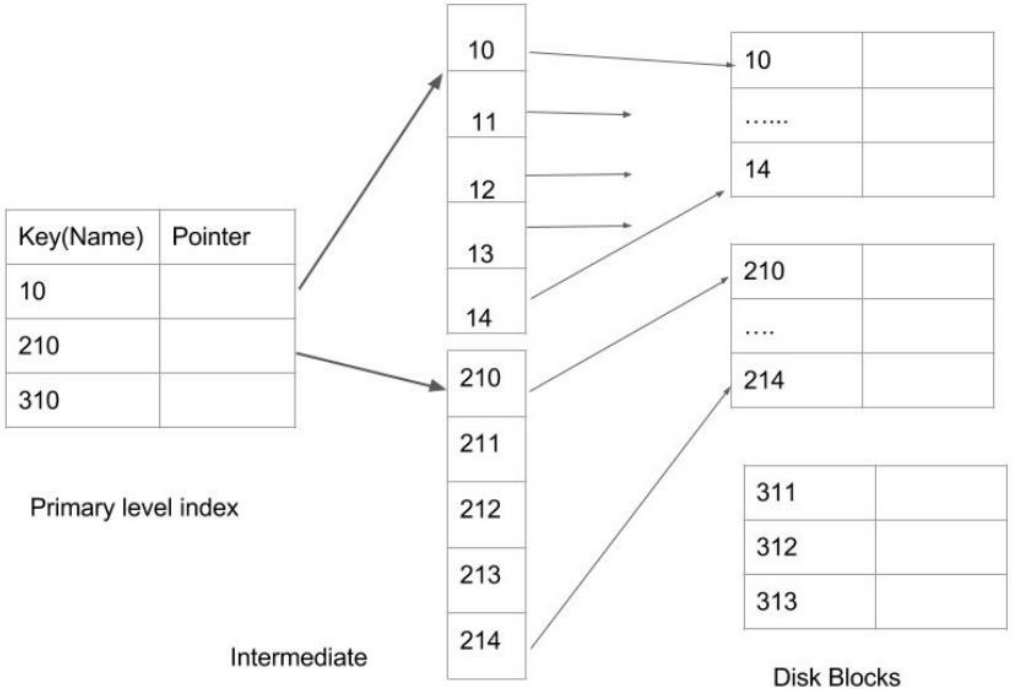
Sparse Index

- The index record appears only for a few items in the data file. Each item points to a block as shown.
- To locate a record, we find the index record with the largest search key value less than or equal to the search key value we are looking for.
- We start at that record pointed to by the index record, and proceed along the pointers in the file (that is, sequentially) until we find the desired record.

Secondary Indices

- An index whose search key specifies an order different from the sequential order of the file. Also called non clustering index.
- It is used to optimize query processing and access records in a database with some information other than the usual search key (primary key).
- In this two levels of indexing are used in order to **reduce the mapping size** of the first level and in general.
- Initially, for the first level, a large range of numbers is selected so that the mapping size is small. Further, each range is divided into further sub ranges.
- In order for quick memory access, first level is stored in the **primary memory**. Actual physical location of the data is determined by the second mapping level.

Indexing and Hashing



Secondary level Indexing

Hashing

- Hash File organization method is the one where **data is stored at the data blocks** whose address is generated by using hash function.
- The memory location where these records are stored is called as data block or data bucket. This data bucket is capable of storing one or more records.
- **Hash function***:* A hash function h is a mapping function **that maps all the set of search keys k to the address** where the actual records are placed. It is a function from search keys to bucket address.

Hashing

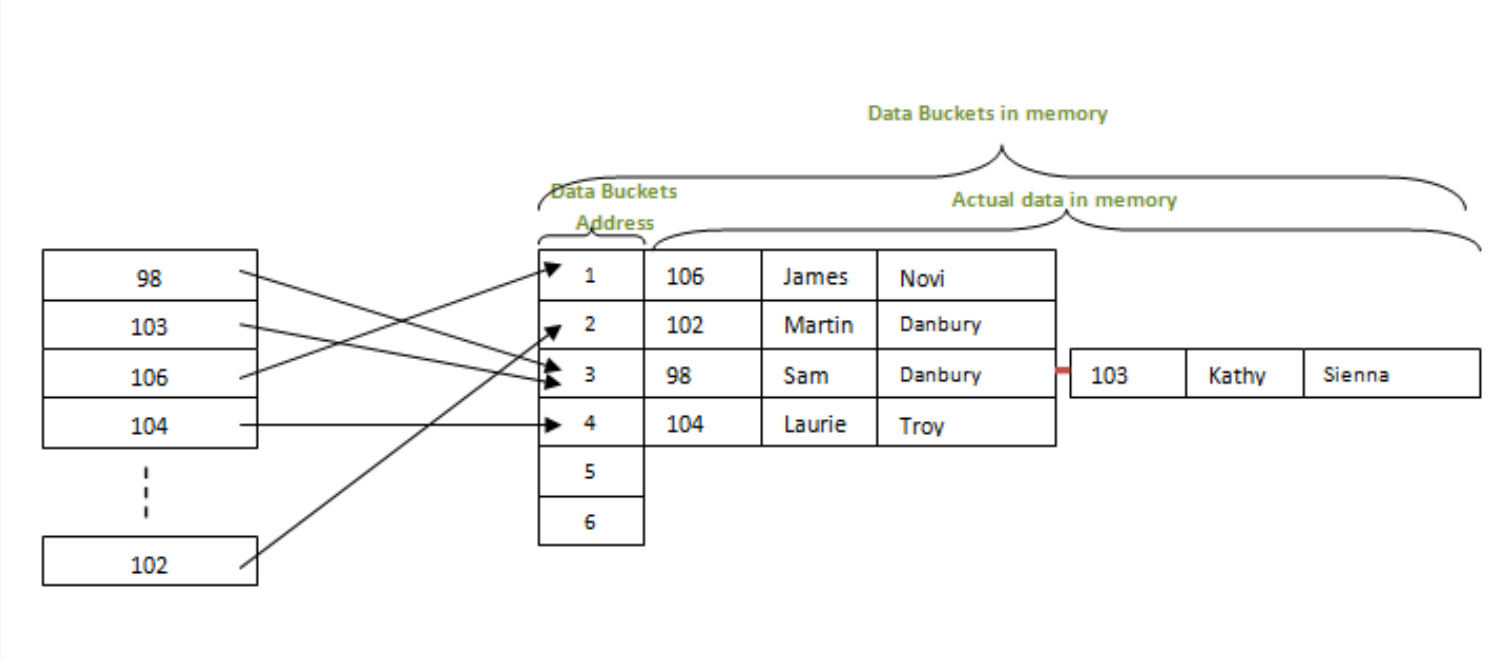


Fig. Example of Hash Index

Hashing

- A hash function, h(K), is a mapping function that maps all the set of search-keys K to the bucket where keys and its associated pointers are placed. The hash function in the above function is **H(K)= Key values % 5**
- Bucket Overflow can occur because of

✓ Insufficient buckets

✓ Skew : A bucket may overflow even when buckets still have space, the situation is called skew.

- Bucket Overflow can be handled:

✓ Overflow chaining: the overflow buckets of a given bucket are chained together in a linked list called closed hashing.

B+ Tree

- Primary disadvantage of index-sequential file **organization is that performance degrades as the file grows.** This can be remedied by costly re-organizations.
- A B+ tree is a balanced binary search tree that follows a **multilevel index format**. The leaf nodes of a B+ tree denote actual data pointers. B+ tree ensures that all leaf nodes remain at the same height, thus balanced.
- Additionally **leaf nodes are linked using a linked list** therefore a B+ tree can support random access as well as sequential access.
- B+ tree indices are **alternative to indexed sequential files**.

B+ Tree

- **Advantages:**

✓ Automatically reorganizes itself with small, local, changes, in the face of insertions and deletions.

✓ Reorganization of entire file is not required to maintain performance.

- **Disadvantage of B+ trees**: Extra insertion and deletion overhead, space overhead

B+ Tree

- B+ tree structure consists of two parts:

1. **Index set**

✓ Provide indexes for fast access of data

2. **Sequence set**

✓ Consist of leaf nodes that contain data pointers.

✓ Provides efficient sequential access of data.

B+ Tree

- Insertion in B+ Tree

Step 1: Insert the new node as a leaf node

Step 2: If the leaf doesn't have required space, split the node and copy the middle node to the next index node.

Step 3: If the index node doesn't have required space, split the node and copy the middle element to the next index page.

B+ Tree

- Deletion in B+ Tree

Step 1: Delete the key and data from the leaves.

Step 2: if the leaf node contains less than minimum number of elements, merge down the node with its sibling and delete the key in between them.

Step 3: if the index node contains less than minimum number of elements, merge the node with the sibling and move down the key in between them.

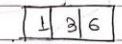
Question: Create a B+ tree of order 4 with following data:

4, 9, 16, 25, 1, 20, 13, 15, 10, 11, 12

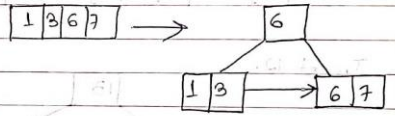
B+ Tree

(Q) Construct a B+tree for the following key values (1,3, 6,7,11,17,19, 23, 30, 32) of order 4.
Assume that tree is initially empty & values are added in ascending order.
Also, show the formation of tree after the insertion of 9.

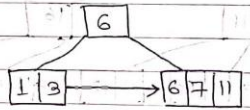
Insert 1,3,6



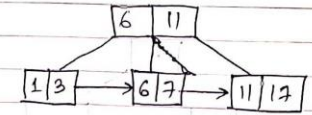
Insert 7



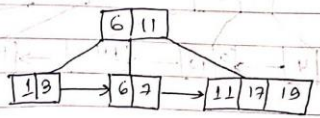
Insert 11



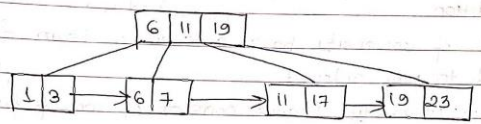
Insert 17.



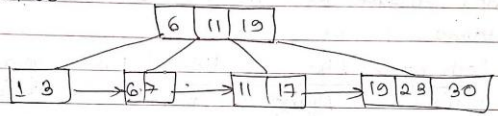
Insert 19.



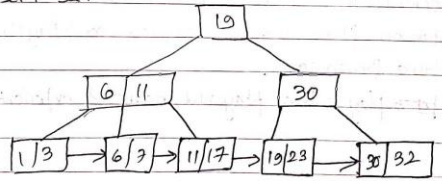
Insert 23



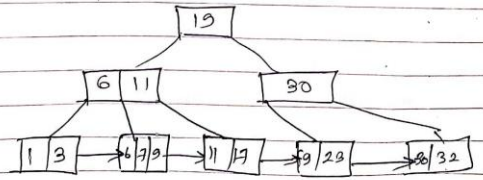
Insert 30



Insert 32.



Again, inserting 9 we get



Questions

1. How the file is organized using hash function and hashing? Explain.

2. Explain the hierarchy of storage.

3. Compare the advantages and disadvantages of heap file organization and sequential file organization. If records are to be processed randomly which one do you prefer among those two organizations?

4. Construct B+-tree for the following set of key values: (1, 3, 6, 7, 11, 19, 23, 30, 32). Assume that the tree is initially empty and values are added in ascending order. Construct B+-trees for the case where the number of pointers that will fit in one node is Four. Also show the form of the tree after insertion of 9.

THANK YOU

Any Queries ?