Storage Strategies

Dr. Munesh Singh

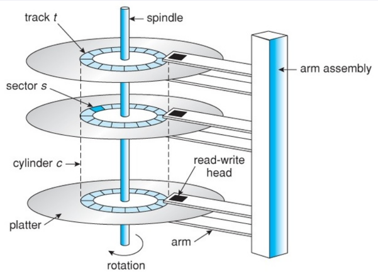
Indian Institute of Information Technology
Design and Manufacturing,
Kancheepuram
Chennai-600127

April 8, 2019

Magnetic Disk

Information Storage Medium

- The primary medium for long term on-line storage of data
- Disk storage is refereed to as direct-access storage, because it is possible to read from the disk randomly
- Disk storage survive power failure and system crash



Magnetic Disk

Information Storage Medium

- Data on a hard disk is arranged in concentric rings or tracks on or more platter
- Read-Write head is positioned very close to platter surface
- Over 16,000 tracks per platter are on typical hard disk
- Each track is divided into arc-shaped sectors
- A sector is the smallest unit of data that can be read or written
- Sector size typically 512 byte
- Data is written to an read from disk block by block.
- The block size is a set to a multiple of the sector size
- Typical disk block size are 4KB or 8KB

Performance Measures of Disk

Information Storage Medium

- **Access time** is the time from when a read or write request is issued to when data transfer begin
- To access data on a given sector of disk, the arm first must move so that it is positioned over the correct track
- **rotational latency time** Later wait for sector to come under the read/write head
- **Data transfer rate** is the rate at which data can be retrieved from or store to the disk
- **Access time** is the sum of seek time, rotational delay, and transfer time

RAID

Information Storage Medium

- A variety of disk organization techniques collectively called as redundant arrays of independent disk
- RAID have been proposed to achieved improved performance and reliability
-
- **Data transfer rate** is the rate at which data can be retrieved from or store to the disk
- **Access time** is the sum of seek time, rotational delay, and transfer time

RAID

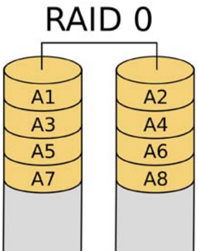
Information Storage Medium

- The solution to the problem of reliability is to introduce redundancy, which can be done using the technique of mirroring or shadowing
- There are a number of variates (RAID-Levels) differing w.r.t the following characteristics:
- (a) Striping unit (data interleaving)
  - how to scatter or split (primary) data across disk?
  - fine bit level stripping or coarse block level striping?
- (b) How to compute and distribute redundant information?
  - what kind of redundant information(parity, ECC)?
  - where to allocate redundant information(separate/few disk/all disk array)?

RAID Level 0

Information Storage Medium

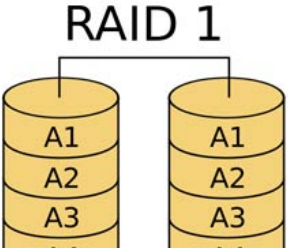
- In this level, there is no redundancy, just stripping.
- The five disks are taken as one unit and the data are scattered over all the disks
- Advantage/Disadvantage of this storage strategy
    - least storage overhead
    - no extra effort for write-access
    - not the best read performance



RAID Level 1

Information Storage Medium

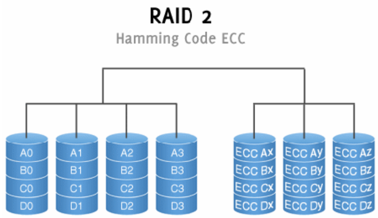
- In this level, there is mirroring.
- It requires redundant information to be stored in double storage space
- Advantage/Disadvantage of this storage strategy
  - double necessary storage space
  - double write access
  - optimized read-performance due to alternative I/O path



RAID Level 2

Information Storage Medium

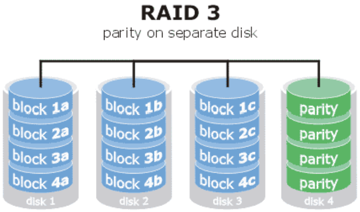
- In this level, there is memory-style error correcting code (ECC)
- It requires additional disk space to store the ECC
- Advantage/Disadvantage of this storage strategy
  - Computer error-correcting codes for data of n disks
  - failure recovery: determine lost disk by using the n-1 extra disk
  - correct its contents from 1 of those



RAID Level 3

Information Storage Medium

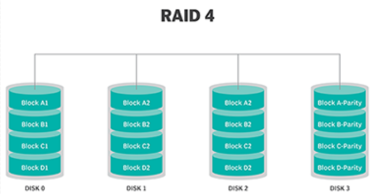
- In this level, there is bit interleaved parity
- Advantage/Disadvantage of this storage strategy
  - one parity disk suffices, since controller can easily identify faulty disk
  - distribute data bitwise onto data disk
  - read and write access goes to all disk, no inter-I/O parallelism, but high bandwidth



RAID Level 4

Information Storage Medium

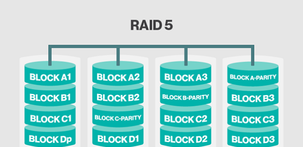
- In this level, there is block-interleaved parity.
- Block are seen as various arcs on the disk
- Large read requests can be made here
- Advantage/Disadvantage of this storage strategy
  - Like RAID 3, but distribute data block-wise (variable block size)
  - small read I/O goes to only one disk
  - bottleneck: all write I/Os go to the one parity disk



RAID Level 5

Information Storage Medium

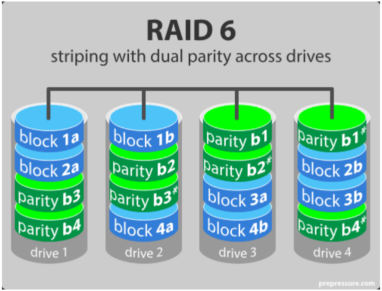
- In this level, there is block-interleaved striped parity.
- This level is an improvement over level 4, where parity block are distributed uniformly over all disk
- Several write request can be processed in parallel
- Advantage/Disadvantage of this storage strategy
  - Like RAID 4, but distribute parity data block across all disk load balancing
  - best performance for small and large read as well as large write I/Os
  - variants w.r.t distribution of block



RAID Level 6

Information Storage Medium

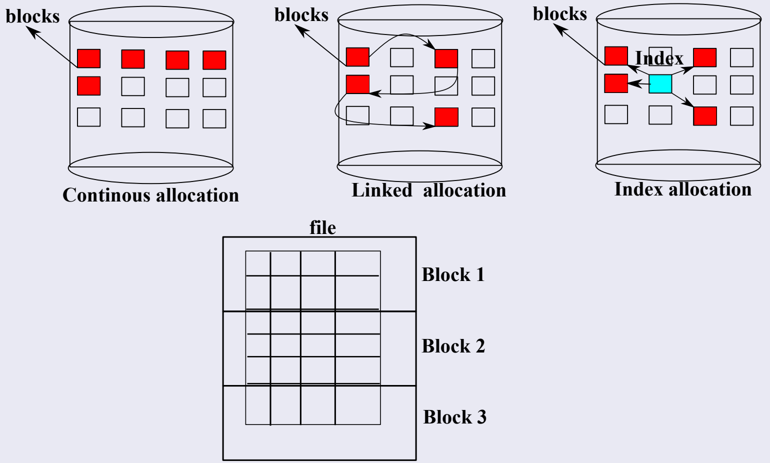
- In this level, there is block-interleaved variable striped parity.



File Structures

- A file is a collection of records that may resides on several pages
- These records are mapped onto disk blocks.
- We need to consider ways of representing logical data models in terms of files
- Al-through blocks are of a fixed size determined by the physical properties of the disk and by the operating system
- Types of block allocation
    - Continuous Memory Allocation
    - Linked Memory Allocation

File Structures



File Structures

Sorted file

- Can be sorted only according to one attribute (search key)
- Searching fast
- Insertion and deletion will be difficult

Unsorted file

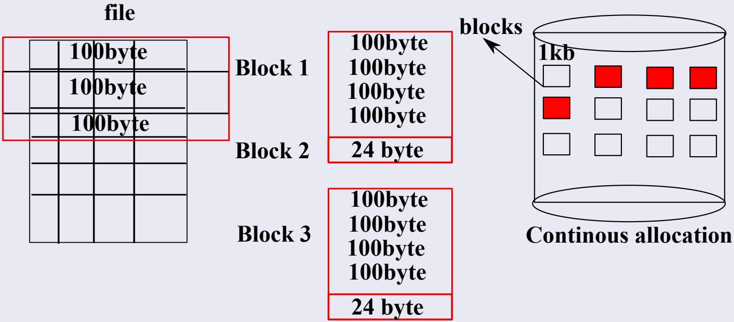
- Random order, any record can be placed any where
- Searching will be slow
- Insertion and deletion will be easy

File Structures

Spanned/Unspanned Mapping

Spanned mapping: Prevent the memeory waste,but increase the search time of the particular record

Unspanned mapping: It does not prevent memory waste, but reduce the block search



Indexing

Lets say blocks are sorted

Binary Search

record =9456

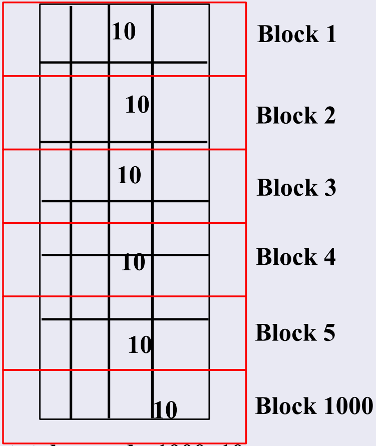
$\log_2 1000 = 10$

Maximum block

Lets say block are unsorted

how may block access required

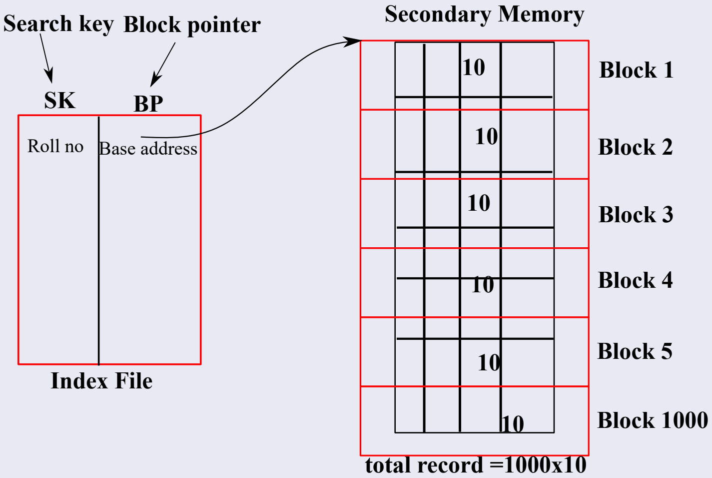
Secondary Memory



total record =1000x10

Indexing

Sparse Indexing :only keep one value per block in index SK (main file sorted)
Dense Indexing :keep all value per block in index SK (main file unsorted)



Types of Indexing

- **Primary Indexing**
- **Clustered indexing**
- **Secondary indexing**
- **Multi-level Tree**

Primary Indexing

Primary

- Main file should be sorted
- Primary key is used as the anchor attributes
- Example of sparse indexing
- No of entries in index file = No of block acquired by the main file
- no of access required = $1og_2 n + 1$

Primary Indexing

For main file

- Let say we have 30000 records in a file and each record takes 100 bytes of space, and in index file each record will take 15 byte
- The given block size of the secondary memory is 1024 bytes
- Find the blocking factor=?
- If spanned and unspanned mapping is not given in the problem, then always takes unspanned allocation.

Primary Indexing

For main file

- Let say we have 30000 records in a file and each record takes 100 bytes of space, and in index file each record will take 15 byte
- The given block size of the secondary memory is 1024 bytes
- Find the blocking factor=?
- If spanned and unspanned mapping is not given in the problem, then always takes unspanned allocation.
  - Blocking factor= $\frac{Block\ size}{record\ size}$ no of record per block
- Number of blocks required for main files

Primary Indexing

For main file

- Let say we have 30000 records in a file and each record takes 100 bytes of space, and in index file each record will take 15 byte
- The given block size of the secondary memory is 1024 bytes
- Find the blocking factor=?
- If spanned and unspanned mapping is not given in the problem, then always takes unspanned allocation.
  - Blocking factor= $\frac{Block\ size}{record\ size}$ no of record per block
- Number of blocks required for main files
  - Blocking factor= $\frac{No\ of\ records\ in\ main\ file}{blocking\ factor}$
- No of block access required

Primary Indexing

For main file

- Let say we have 30000 records in a file and each record takes 100 bytes of space, and in index file each record will take 15 byte
- The given block size of the secondary memory is 1024 bytes
- Find the blocking factor=?
- If spanned and unspanned mapping is not given in the problem, then always takes unspanned allocation.
  - Blocking factor= $\frac{Block\ \ size}{record\ \ size}$ no of record per block
- Number of blocks required for main files
  - Blocking factor= $\frac{No\ \ of\ \ records\ \ in\ \ main\ \ file}{blocking\ \ factor}$
- No of block access required
  - $log_2 n$

Primary Indexing

For Index file

- Calculate the blocking factor for index file
- Calculate the no of blocks for index files
- No of block access required

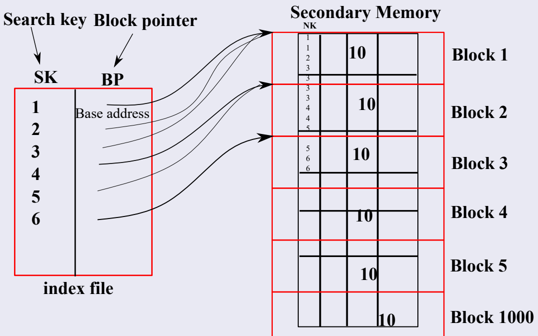
Primary Indexing

For Index file

- Calculate the blocking factor for index file
- Calculate the no of blocks for index files
- No of block access required
- $log_2 n + 1$

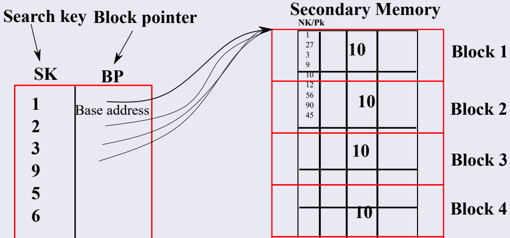
Cluster Indexing

- Main file is sorted (on some non-key attributes)
- There will be one entry for each unique value of the non-key attribute
- if number of block acquired by index file is n, then block access required will be $>= log_2 n + 1$



Secondary Indexing

- Main file unsorted
- Can be done on key as well as on non-key attributes
- Called secondary because normally one indexing is already done
- Example of dense indexing
- no of entry in index file = no of entry in main file
- no of block access=$log_2 n + 1$



Secondary Indexing

For main file

- Let say we have 30000 records in a file and each record takes 100 bytes of space, and in index file each record will take 15 byte
- The given block size of the secondary memory is 1024 bytes
- Find the blocking factor=?
- If spanned and unspanned mapping is not given in the problem, then always takes unspanned allocation.

Secondary Indexing

Storage Strategies

Dr. Munesh Singh

For main file

- Let say we have 30000 records in a file and each record takes 100 bytes of space, and in index file each record will take 15 byte
- The given block size of the secondary memory is 1024 bytes
- Find the blocking factor=?
- If spanned and unspanned mapping is not given in the problem, then always takes unspanned allocation.
  - Blocking factor= $\frac{Block\ size}{record\ size}$ no of record per block
- Number of blocks required for main files

Secondary Indexing

For main file

- Let say we have 30000 records in a file and each record takes 100 bytes of space, and in index file each record will take 15 byte
- The given block size of the secondary memory is 1024 bytes
- Find the blocking factor=?
- If spanned and unspanned mapping is not given in the problem, then always takes unspanned allocation.
  - Blocking factor= $\frac{Block\ \ size}{record\ \ size}$ no of record per block
- Number of blocks required for main files
  - Blocking factor= $\frac{No\ \ of\ \ records\ \ in\ \ main\ \ file}{blocking\ \ factor}$
- No of block access required

Secondary Indexing

For main file

- Let say we have 30000 records in a file and each record takes 100 bytes of space, and in index file each record will take 15 byte
- The given block size of the secondary memory is 1024 bytes
- Find the blocking factor=?
- If spanned and unspanned mapping is not given in the problem, then always takes unspanned allocation.
  - Blocking factor= $\frac{Block\ size}{record\ size}$ no of record per block
- Number of blocks required for main files
  - Blocking factor= $\frac{No\ of\ records\ in\ main\ file}{blocking\ factor}$
- No of block access required
  - n

Secondary Indexing

For Index file

- Calculate the blocking factor for index file
- Calculate the no of blocks for index files
- No of block access required

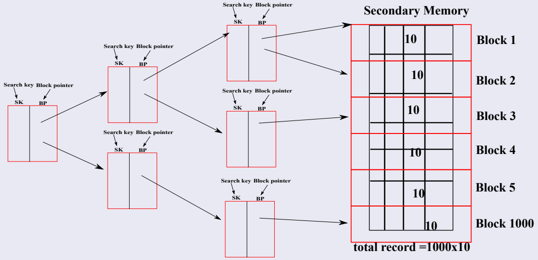
Secondary Indexing

For Index file

- Calculate the blocking factor for index file
- Calculate the no of blocks for index files
- No of block access required
- $log_2 n + 1$

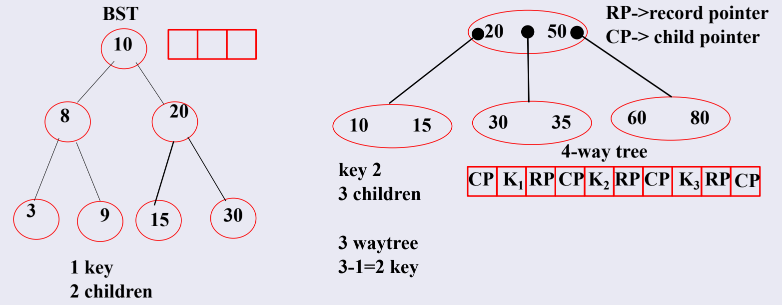
Multi-Level Indexing

- In multilevel indexing we divided the big index file into smaller index file (dense indexing case).
- Multilevel indexing permits the faster access of record
- Multilevel indexing concept derived from the m-way search tree



What is m-way tree

BST vs m-way tree



Multi-Level Indexing

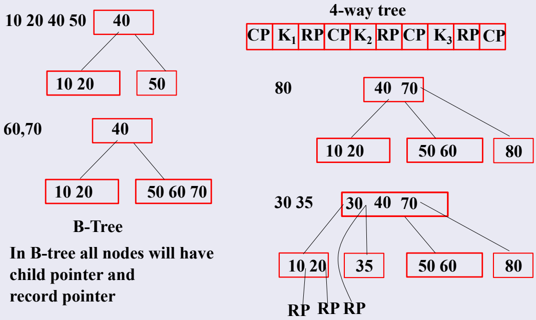
m-way search tree

- m-way search tree has no control on reducing the height of the tree
- To resolve the issues of m-way search tree, B-tree is introduced.
- B-tree set some rules for insertion/deletion of keys (it also a kind of m-way search tree with rule)
  - $\lfloor \frac{m}{2} \rfloor$ children
  - Root node can have minimum 2 children
  - All leaf node must be at same level
  - Bottom-up approach

Multi-Level Indexing Example

B-tree

- Lets say m=4, then m-1 key should be there
- Given keys 10,20,40,50;



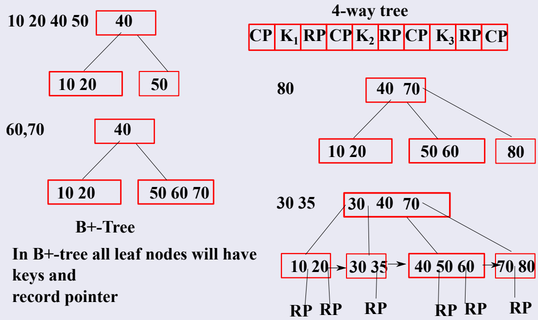
B-Tree

In B-tree all nodes will have child pointer and record pointer

Multi-Level Indexing Example

B+-tree

- In B+-tree all the leaf node will have record pointer
- Every key also present in the leaf node
- And the leaf nodes are connected to another leaf node



B+-Tree

In B+-tree all leaf nodes will have keys and record pointer

Multi-Level Indexing Example

Deletion on B-tree

- Deletion of 2,21,10,3,4